```
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#  Sampling design: scheduling
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Zhenming Su, IFR, ANN ARBOR

# Created: March 15, 2010
# Revised: Aug 1, 2021 for inland creel evaluation
# Revised: Aug 7, 2021
#   Count times are generated using uniform distribution
# New: Aug 18, added incompleted trip counting and interviewing
# New: Aug 20, added individual incompleted trip interviewing
#      Aug 25, fixed most errors
# New: Aug 27, created bus-route schedule functions

# Days population: all days in a month available
f_gen_days_pop <- function(is_simu_pop, trip_pop)
{
  # all available days for sample
  days_pop <- sort(unique(trip_pop$DAY))
  days_pop
}

f_sample_days_shifts <- function(is_simu_pop, trip_pop, days_pop, ndays_s, n_shifts, s
trat_by_wkday)
{
    if (!strat_by_wkday)
    {
        # Select days randomly within a month without replacement
        # no stratification of the days by daytype
        days_s <- sort(sample(days_pop, size = ndays_s))
    }
    else
    {
        # Select days randomly by daytype in a month
        yr_simu  <- trip_pop$YEAR[1]  # only consider one year
        mon_simu <- trip_pop$MONTH[1] # only have one month
        dates <- paste(yr_simu, "-", mon_simu, "-", days_pop, sep="")
        wkdys <- c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
        wkday <- match(weekdays(as.Date(dates), abbreviate = T), wkdys)

        #date_str <- paste(mon_simu, "/", days_pop,"/", yr_simu, sep="")
        #wkday <- weekdays(date_str)

        Days_Shifts <- data.frame(DAY = days_pop, DATE=dates, DayType = wkday)

        # Select all weekend days when ndays_s > 10
        wedays_s <- Days_Shifts$DAY[Days_Shifts$DayType > 5]
        n_wedays <- length(wedays_s)
        if (ndays_s <= 10)
        {
            # Number of weekend days sampled is half of that of the sample size in a m
onth
            n_wedays <- ceiling(ndays_s * 1/2)
            # Weekend day sample
            wedays_s <- sort(sample(wedays_s, n_wedays))
        }

        # Weekdays sampled
        n_wkdays <- ndays_s - n_wedays
        wkdays <- Days_Shifts$DAY[Days_Shifts$DayType <= 5]
        wkdays_s <- sort(sample(wkdays, n_wkdays))

        days_s <- sort(c(wkdays_s, wedays_s))
    }
```

```
  # to-do
    # (1) by week, three work days and two weekend days / week
    # (2) sample two shifts for three shifts day

    # sampling shifts in a sampled day
    #  No subsampling
    switch (as.character(n_shifts),
        "1"={#n_shifts==1
          sh <- c("O")
        },
        # n_shifts==2
        "2"={
          sh <- c("A", "B")
        },
        # n_shifts==3
        "3"={
          sh <- c("AB", "BC", "AC")
        },
        stop(paste("Number of shifts, ", n_shifts, ", is not considered.", sep =""))
    )

    # Select shifts
    if (n_shifts==1)
    {
      shifts_s <- rep("O", ndays_s)
    }
    else
    {
      shifts_s <- sample(sh, size = ndays_s, replace = T)
    }

    if (!is_simu_pop)
    {
      # Use observed data
      #  ET -- end time of a trip
      maxEt <- tapply(trip_pop$ET, trip_pop$DAY, max)
      #  Trips observed in the "A" shift had end times < 15
      #  ???? 15
      shifts_s <- ifelse(maxEt < 15, "A", "B")
    }

    data.frame(days_s = days_s, shifts = shifts_s)
}

# sample count times
#  Fishing day: [4 to 22]
    # A shift: 4 to 12
    # B shift: 12 to 22
f_sample_instant_count_times <- function(is_simu_pop = TRUE, ndays_s = 10, ncnts = 1,
n_shifts = 2, shifts_s = "O", shift_times = c(4, 22), shifts_non_overlapping = TRUE)
{
    # Instant count times: can bias the boat-hours estimates if the time interval is l
arge
    #  This happened when using "Sample" to generate discrete times
    #     sample(seq(from = SStart, to = SEnd, by = 0.5)
    # Inv_time_sampling = 0.005   July to Aug 6, 2021 changed the interval from 0.5 h
to small intervals
    #     sort(sample(seq(from = SStart, to = SEnd, by = Inv_time_sampling), size = nc
nts))
    # Changed to use uniform distribution, runif, to sample instant continuous times,
Aug 6, 2021
    #  treat time as continuous values

    count_times <- matrix(NA, nrow = ndays_s, ncol = ncnts, byrow =T)
    if (n_shifts == 1)
    {
        #shift_times = c(4,22)
        SStart <- shift_times[1] #4
        SEnd   <- shift_times[2] #22
    }

    if (n_shifts ==2)
    {
        if (shifts_non_overlapping)
        {
            #shift_times = c(4, 13, 22)
            AStart <- shift_times[1] #4
            AEnd   <- shift_times[2] #13
            BStart <- AEnd #shift_times[2] #13
            BEnd   <- shift_times[3] #2
        }else{
            # Two overlapping shifts
            AStart <- shift_times[1] #4
            AEnd   <- shift_times[2] #16
            BStart <- shift_times[3] #14
            BEnd   <- shift_times[4] #22
        }
    }
    if (n_shifts ==3)
    {
        #shift_times = c(4, 10, 16, 22)
        AStart <- shift_times[1] #4
        AEnd   <- shift_times[2] #10
        BStart <- AEnd #10
        BEnd   <- shift_times[3] #16
        CStart <- BEnd #16
        CEnd   <- shift_times[4] #22
    }

    i <- 1
    for (d in 1:ndays_s){
        switch(n_shifts,
            # one shift
            {
                count_times[i, ] <- sort(runif(ncnts, SStart, SEnd))
            },
            # two shifts
            {
                if (shifts_s[i] == "A")
                {
                    count_times[i, ] <- sort(runif(ncnts, AStart, AEnd))
                }

                if (shifts_s[i] == "B")
                {
                    count_times[i, ] <- sort(runif(ncnts, BStart, BEnd))
                }
            },
            # three shifts
            {
                if (shifts_s[i] == "AB")
                {
                    count_times[i, ] <- sort(runif(ncnts, AStart, BEnd))
                }

                if (shifts_s[i] == "BC")
                {
                    count_times[i, ] <- sort(runif(ncnts, BStart, CEnd))
                }
                if (shifts_s[i] == "AC")
                {
                    AorC = sample(1:2, size = 1)
                    if (ncnts %% 2 >0)
                    {
                        n1 = ifelse(AorC==1, (ncnts %/% 2) + 1, (ncnts %/% 2))

                    }else
                    {
                        n1 = ncnts %/% 2
                    }
                    n2 = ncnts-n1

                    count_times[i, ] <- c(sort(runif(n1, AStart, AEnd)), sort(runif(n2
, CStart, CEnd)))
                }
            }
        )
        i <- i+1
    }

    count_times
}

# sample count times
#  Fishing day: [4 to 22]
    # A shift: 4 to 12
    # B shift: 12 to 22
f_sample_progressive_count_times <- function(is_simu_pop = TRUE, ndays_s = 10, count_d
uration = 0.5, ncnts = 1, n_shifts = 2, shifts_s = "O", shift_times = c(4, 22), shifts
_non_overlapping = TRUE)
{
    # Progressive count time sampling: can bias the boat-hours estimates if an uniform
 distribution is
    #   used to generate sample in the interval (0, T-tau)
    # The correct procedure is to sample a block randomly from the non-overlapping blo
cks that divide (0, T)
    #     sample(seq(from = SStart, to = SEnd, by = 0.5)
    # Aug 11, 2021

    count_times <- matrix(NA, nrow = ndays_s, ncol = ncnts, byrow =T)
    if (n_shifts == 1)
    {
        #shift_times = c(4,22)
        SStart <- shift_times[1] #4
        SEnd   <- shift_times[2] #22
    }

    if (n_shifts ==2)
    {
        if (shifts_non_overlapping)
        {
            #shift_times = c(4, 13, 22)
            AStart <- shift_times[1] #4
            AEnd   <- shift_times[2] #13
            BStart <- AEnd #shift_times[2] #13
            BEnd   <- shift_times[3] #2
        }else{
            # Two overlapping shifts
            AStart <- shift_times[1] #4
            AEnd   <- shift_times[2] #16
            BStart <- shift_times[3] #14
            BEnd   <- shift_times[4] #22
        }
    }
    if (n_shifts ==3)
    {
        #shift_times = c(4, 10, 16, 22)
        AStart <- shift_times[1] #4
        AEnd   <- shift_times[2] #10
        BStart <- AEnd #10
        BEnd   <- shift_times[3] #16
        CStart <- BEnd #16
        CEnd   <- shift_times[4] #22
    }

    i <- 1
    for (d in 1:ndays_s){
        # Start times of counts to be made. Each count will take 'count_duration' to f
inish
        switch(n_shifts,
            # one shift
            {
                H <- SEnd-SStart # duration of shift
                k <- trunc(H/count_duration) # max. number of counts that can be made
in the shift
                count_times[i, ] <- sort(sample(seq(from = SStart, to = SEnd-count_dur
ation, by = count_duration), size = ncnts))
            },
            # two shifts
            {
                if (shifts_s[i] == "A")
                {
                    count_times[i, ] <- sort(sample(seq(from = AStart, to = AEnd-count
_duration, by = count_duration), size = ncnts))
                }

                if (shifts_s[i] == "B")
                {
                    count_times[i, ] <- sort(sample(seq(from = BStart, to = BEnd-count
_duration, by = count_duration), size = ncnts))
                }
            },
            # three shifts
            {
                if (shifts_s[i] == "AB")
                {
                    count_times[i, ] <- sort(sample(seq(from = AStart, to = BEnd-count
_duration, by = count_duration), size = ncnts))
                }

                if (shifts_s[i] == "BC")
                {
                    count_times[i, ] <- sort(sample(seq(from = BStart, to = CEnd-count
_duration, by = count_duration), size = ncnts))
                }

                if (shifts_s[i] == "AC")
                {
                    AC_time_pop = c(seq(from = AStart, to = AEnd-count_duration, by =
count_duration),
                                     seq(from = CStart, to = CEnd-count_duration, by =
 count_duration))
                    count_times[i, ] <- sort(sample(AC_time_pop, size = ncnts))
                }
            }
        )
        i <- i+1
    }
    count_times
}

#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#  Sampling: counts and interviews
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# make counts
# Fishing day: [4:00 to 22:00], 18 hours
# A shift: 4:00 to 12:00
# B shift: 12:00 to 22:00

trips_intercepted_in_roving_count <- function(start_time_sess, duration, trips_pop_day
)
{
  # the time at which the clerk intercepts an angler party
  #  when roving along a predefined count or interview route
  #  start_time_sess, end_time and duration are all numerical values between 0-24

  end_time <- start_time_sess + duration

  # all available trips that can be intercepted during this specific roving duration
  trips_pop_cnts <- trips_pop_day#[(start_time_sess <= trips_pop_day$ET) & (trips_pop_
day$ST < end_time),]

  if (nrow(trips_pop_cnts) == 0){
    count_sample <- trips_pop_cnts
  } else {
      trips_pop_cnts$intercepted = FALSE
      trips_pop_cnts$t_clerk = -1

      # clerk travel speed: total travel distance (the length of route) is scale to 1
      roving_speed <- 1/duration  # per hour

      # random start location of the clerk relative to a reference start location (0)
      start_loc <- runif(1, 0, 1)
      direction <- sample(1:2, size = 1)

      # positions of parties relative to start_loc
      if (direction==1) {
        positions_rel <- trips_pop_cnts$position - start_loc
        positions_rel <- ifelse(positions_rel>=0, positions_rel, 1-abs(positions_rel))
      } else {
        positions_rel <- trips_pop_cnts$position - start_loc
        positions_rel <- ifelse(positions_rel<=0, positions_rel, -(start_loc + 1 - tri
ps_pop_cnts$position))
        positions_rel <- abs(positions_rel)
      }

      # the time needed by the clerk to intercept a party
      trips_pop_cnts$positions_rel <- positions_rel

      # order the trips in trips_pop_cnts by their position
      trips_pop_cnts <- trips_pop_cnts[order(trips_pop_cnts$positions_rel), ]
      n_trips <- nrow(trips_pop_cnts)

      # Simulate the roving count process
      # compute the clerk's two positions at the start-time and end-time of each party
      #  and check if the position contain the party's position then the party can be
intercepted for counting
      for (i in 1:n_trips) {
        # the clerk's two positions at the start-time and end-time of each party
        pos_st <- (trips_pop_cnts[i,]$ST-start_time_sess) * roving_speed
        pos_et <- (trips_pop_cnts[i,]$ET-start_time_sess) * roving_speed

        intercepted <- ifelse((pos_st <= trips_pop_cnts$positions_rel[i]) & (trips_pop
```

```
_cnts$positions_rel[i] <= pos_et), TRUE, FALSE)
        trips_pop_cnts[i,]$intercepted <- intercepted

        if (intercepted) {
          trips_pop_cnts[i,]$t_clerk <-  trips_pop_cnts$positions_rel[i]/roving_speed
        }
      }

      count_sample <- trips_pop_cnts[trips_pop_cnts$intercepted, ]
  }

count_sample
}

# Count of angler parties intercepted when a clerk roves through the fishing area
#   during the interval 'count_duration'
f_make_progressive_counts <- function(simu_trip_pop, days_s, count_duration, count_tim
es)
{
    nrec <- length(days_s) * ncol(count_times)

    counts <- data.frame(
            Waterbody = (rep("Inland", nrec)),
            DOW = rep(0, nrec),
            DATE =(rep("", nrec)),
            FishSite = rep("001", nrec),
            YEAR = rep(0, nrec),
            MONTH = rep(0, nrec),
            DAY = rep(0, nrec),
            MODE = (rep("", nrec)),
            ORD = rep(1, nrec),
            ORDNM = rep("", nrec),
            StartHour = rep(0, nrec),
            CNTTYPE = rep(2, nrec),
            DUR = rep(0, nrec),
            COUNT = rep(0, nrec),
            stringsAsFactors = FALSE)

    r <- 1
    id <- 1
    for (d in days_s)
    {
        ord <- 1
        for (ct in count_times[id, ])
        {
          # Progressive counts: count of all boats visible in the time interval 'count
_duration'
          #  when roving along a predefined count route

          # ST, ET and ct are all numerical values between 0-24
          # all the trips in the population on day d
          trips_pop_day <- simu_trip_pop[simu_trip_pop$DAY == d,]

          # count of trips intercepted by the clerk during the roving count
          cnt_sample <- trips_intercepted_in_roving_count(ct, count_duration, trips_po
p_day)
          cnt <- nrow(cnt_sample)

          if (cnt ==0) cnt_sample <- trips_pop_day[1,]

          counts$Waterbody[r] = cnt_sample$Waterbody[1]
          counts$DOW[r] = cnt_sample$DOW[1]
          counts$DATE[r] = cnt_sample$date[1]
          counts$FishSite[r] = cnt_sample$FishSite[1]
          counts$YEAR[r] = cnt_sample$YEAR[1]
          counts$MONTH[r] = cnt_sample$MONTH[1]
          counts$DAY[r] = cnt_sample$DAY[1]
          counts$MODE[r] = cnt_sample$MODE[1]
          counts$ORD[r] = ord
          counts$ORDNM[r] = paste("Count", ord, sep="")
          counts$StartHour[r] = ct
          counts$CNTTYPE[r] = 2
          counts$DUR[r] = 0
          counts$COUNT[r] = cnt

          ord <- ord + 1
          r = r + 1
        } # count times
        id = id + 1
    } #day
    counts <- counts[1:(r-1), ]

    counts
}

f_make_instant_counts <- function(simu_trip_pop, days_s, count_times)
{
    nrec <- length(days_s) * ncol(count_times)

    counts <- data.frame(
            Waterbody = (rep("Inland", nrec)),
            DOW = rep(0, nrec),
            DATE =(rep("", nrec)),
            FishSite = rep("001", nrec),
            YEAR = rep(0, nrec),
            MONTH = rep(0, nrec),
            DAY = rep(0, nrec),
            MODE = (rep("", nrec)),
            ORD = rep(1, nrec),
            ORDNM = rep("", nrec),
            StartHour = rep(0, nrec),
            CNTTYPE = rep(2, nrec),
            DUR = rep(0, nrec),
            COUNT = rep(0, nrec),
            stringsAsFactors = FALSE)

    r <- 1
    id <- 1
    for (d in days_s)
    {
        ord <- 1
        for (ct in count_times[id, ])
        {
          # Instantaneous counts: count of all boats visible at time point ct
          #  i.e., number of trips that are observed at time ct
          # ST, ET and ct are all numerical values between 0-24
          # all trips in day d
          trips_pop_day <- simu_trip_pop[simu_trip_pop$DAY == d,]

          # trips that will be counted at time ct
          cnt_sample <- trips_pop_day[(trips_pop_day$ST <= ct) & (ct < trips_pop_day$E
T),]
          cnt <-  nrow(cnt_sample)
          if (cnt > 0)
          {
                counts$Waterbody[r] = cnt_sample$Waterbody[1]
                counts$DOW[r] = cnt_sample$DOW[1]
                counts$DATE[r] = cnt_sample$date[1]
                counts$FishSite[r] = cnt_sample$FishSite[1]
                counts$YEAR[r] = cnt_sample$YEAR[1]
                counts$MONTH[r] = cnt_sample$MONTH[1]
                counts$DAY[r] = cnt_sample$DAY[1]
                counts$MODE[r] = cnt_sample$MODE[1]
                counts$ORD[r] = ord
                counts$ORDNM[r] = paste("Count", ord, sep="")
                counts$StartHour[r] = ct
                counts$CNTTYPE[r] = 2
                counts$DUR[r] = 0
                counts$COUNT[r] = cnt

                ord <- ord + 1
                r = r + 1
           }
           else
           {
                counts$Waterbody[r] = trips_pop_day$Waterbody[1]
                counts$DOW[r] = trips_pop_day$DOW[1]
                counts$DATE[r] = trips_pop_day$date[1]
                counts$FishSite[r] = trips_pop_day$FishSite[1]
                counts$YEAR[r] = trips_pop_day$YEAR[1]
                counts$MONTH[r] = trips_pop_day$MONTH[1]
                counts$DAY[r] = trips_pop_day$DAY[1]
                counts$MODE[r] = trips_pop_day$MODE[1]
                counts$ORD[r] = ord
                counts$ORDNM[r] = paste("Count", ord, sep="")
                counts$StartHour[r] = ct
                counts$CNTTYPE[r] = 2
                counts$DUR[r] = 0
                counts$COUNT[r] = 0
                ord <- ord + 1
                r = r + 1
           }
        } # count times
        id = id + 1
    } #day
    counts <- counts[1:(r-1), ]
    counts
}

make_roving_interviews <- function(start_time_shift, end_time_shift, dur_route, interv
iew_time, trips_pop_ints)
{
  #  The clerk roves along a predefined interview route, and interview parties that ar
e encountered
  #  Each interview takes 'interview_time' minutes
  #  dur_route: hours used to finish one round of interviews for the entire fishing ar
ea
  #  start_time_shift, end_time_shift: the times at which the clerk starts and ends th
e roving session

  #  end_time_shift <- start_time_shift + dur_route

  # all available trips on day d that can be potentially intercepted during this speci
fic roving session

  # Aug 19, 2021

  n_trips <- nrow(trips_pop_ints)
  trips_pop_ints$intercepted = FALSE

  # clerk travel speed: total travel distance (the length of route) is scale to 1
  # count roving speed is assumed the max speed the clerk can travel
  # if moving with the max speed, the clerk can make several rounds through the fishin
g area
```

```
  roving_speed <- 1/dur_route  # per hour

  # random start location of the clerk relative to a reference start location (0)
  #  at the start of the shift
  start_loc <- runif(1, 0, 1)
  direction <- sample(1:2, size = 1)
  # start time
  t_clerk <- start_time_shift
  # store interviews
  interviewed_trips <- NULL
  n_ints <- 0
  last_interviews <- ""
  while (t_clerk < end_time_shift) {
      # positions of parties relative to start_loc
      if (direction==1) {
            positions_rel <- trips_pop_ints$position - start_loc
            positions_rel <- ifelse(positions_rel>=0, positions_rel, 1-abs(positions_r
el))
      } else {
            positions_rel <- trips_pop_ints$position - start_loc
            positions_rel <- ifelse(positions_rel<=0, positions_rel, -(start_loc + 1 -
 trips_pop_ints$position))
            positions_rel <- abs(positions_rel)
      }
      trips_pop_ints$positions_rel <- positions_rel

      # At the start of the session, find the first party that can be reached and inte
rviewed
      #      when the clerk starts from start_loc

      # order the trips in trips_pop_ints by their start_times ST and positions_rel
      trips_pop_ints <- trips_pop_ints[order(trips_pop_ints$ST, trips_pop_ints$positio
ns_rel), ]

      # cal the time at which the clerk would reach each party's position
      #  if the clerk moved in constant speed with no interruption
      times_to_parties <- t_clerk + trips_pop_ints$positions_rel/roving_speed
      trips_pop_ints$times_to_parties <- times_to_parties

      # check if they can be intercepted
      intercepted <- ifelse((trips_pop_ints$ST <= times_to_parties) & (trips_pop_ints$
ET >= times_to_parties), TRUE, FALSE)
      trips_pop_ints$intercepted <- intercepted

      # Move on to other party: skip last interviewed trip
      id_last_int <- which(row.names(trips_pop_ints) == last_interviews)
      trips_pop_ints$intercepted[id_last_int] <- FALSE

      # Find the nearest party that can be intercepted for interview
      trip_for_interview <- trips_pop_ints[trips_pop_ints$intercepted,]
      if (nrow(trip_for_interview)==0) break

      trip_for_interview$interview_time <- 0
      trip_for_interview$smb_incomp <- 0

      # Find the nearest party that can be intercepted for interview
      id_interviewed_trip <- which(trip_for_interview$times_to_parties == min(trip_for
_interview$times_to_parties))

    if (id_interviewed_trip > 0)
      {
      n_ints <- n_ints + 1
        last_interviews <- row.names(trip_for_interview[id_interviewed_trip, ])

      if (n_ints == 1)
```

```
     interviewed_trips <- trip_for_interview[id_interviewed_trip, ]
   else
   {
     interviewed_trips <- rbind(interviewed_trips, trip_for_interview[id_intervie
wed_trip, ])
   }

   interviewed_trips$interview_time[n_ints] <- interviewed_trips$times_to_parties[n
_ints]
   trip_len_in <- interviewed_trips$interview_time[n_ints]-interviewed_trips$ST[n_i
nts]
   trip_len <- interviewed_trips$ET[n_ints]-interviewed_trips$ST[n_ints]
   interviewed_trips$smb_incomp[n_ints] <- round(interviewed_trips$smb[n_ints] * tr
ip_len_in/trip_len)

   # track the time position of the clerk
   t_clerk <- interviewed_trips$interview_time[n_ints]

   # add the interview time
   t_clerk <- t_clerk + interview_time

   # set start_loc to the position of current party
   start_loc <- interviewed_trips$position[n_ints]
   }
 }

 if (is.null(interviewed_trips))
 {
  cat("No interview sampled on day ", trips_pop_ints$DAY[1],  "\n")
  interviewed_angler_trips <- NULL
  return(interviewed_angler_trips)
 }
 # clean the data
 # drop some variables
 #interviewed_trips[,c("X", "intercepted","positions_rel","times_to_parties")] <- lis
t(NULL)

 # renames variables for incompleted trips
 names(interviewed_trips)[names(interviewed_trips) == 'smb'] <- 'smb_true'
 names(interviewed_trips)[names(interviewed_trips) == 'smb_incomp'] <- 'smb_party'

 idx_smb_c <- which(names(interviewed_trips)=="smb_true")
 idx_smb <- which(names(interviewed_trips)=="smb_party")
 interviewed_trips <- interviewed_trips[, c(1:idx_smb_c, idx_smb, (idx_smb_c+1):(idx_
smb-1))]

 interviewed_trips$triplen <- interviewed_trips$interview_time-interviewed_trips$ST

 # set interview times as endTime of the trip
 names(interviewed_trips)[names(interviewed_trips) == "EndTime"] <- "EndTime_true"
 #names(interviewed_trips)[names(interviewed_trips) == "interview_time"] <- "EndTime"

 # convert decimal hours to "hh:mm" formats
 hours <- trunc(interviewed_trips$interview_time)
 minutes <- round((interviewed_trips$interview_time-hours)*60)
 hours[minutes>=60] <- hours[minutes>=60] + 1
 minutes[minutes>=60] <- minutes[minutes>=60]-60

 interviewed_trips$EndTime <- paste(hours,":",minutes, sep="")

 interviewed_trips$comptrip <- "2"
 interviewed_trips$individualInterview <- "Yes"

 n_incomp_party_trips <- nrow(interviewed_trips)
```

```
  # create individual interviews
  numang <- interviewed_trips$NumAnglers
  idx <- rep(1:nrow(interviewed_trips), numang)
  interviewed_angler_trips <- interviewed_trips[idx, ]

  seq2 <- Vectorize(seq.default, vectorize.args = c("from", "to"))

  order_in_party <-  as.vector(unlist(seq2(from = rep(1,length(numang)), to = numang,
by = 1)))

  if (class(order_in_party)[1]=="matrix")
  {
    cat("order_in_party is a matrix",  "\n")
  }

  interviewed_angler_trips$order_in_party <- order_in_party

  interviewed_angler_trips$party_no <- idx
  #interviewed_angler_trips$NumAnglers <- ifelse(interviewed_angler_trips$order_in_par
ty==1, interviewed_angler_trips$NumAnglers, NA)
  interviewed_angler_trips$NumAnglers0 <- ifelse(interviewed_angler_trips$order_in_par
ty==1, interviewed_angler_trips$NumAnglers, "")

  stopifnot(max(numang) <= 6)

  prob_angl = list(pcA1 = 1, pcA2 = c(0.695, 0.305), pcA3 = c(0.652,0.177,0.172), pcA4
 = c(0.6, 0.14, 0.13, 0.13), pcA5 = c(0.6, 0.1, 0.1, 0.1, 0.1), pcA6 = c(0.6, 0.08, 0.
08, 0.08, 0.08, 0.08))

  c_angl <- NULL
  for (i in 1:n_incomp_party_trips){
    size <- interviewed_trips$smb_party[i]
    c_ind <- as.vector(rmultinom(n = 1, size = size, prob = prob_angl[[numang[i]]]))
    c_angl <- c(c_angl, c_ind)
  }
  interviewed_angler_trips$smb <- c_angl

  interviewed_angler_trips <- interviewed_angler_trips[interviewed_angler_trips$triple
n>0,]

  interviewed_angler_trips
}

f_get_incomplete_trip_interviews <- function(opt_int_sampling, simu_trip_pop, max_nint
s, days_s, n_shifts, shifts_s, shift_times = c(4, 13, 22))
{
    # Roving interviews
    # Rove through the fishery and interview parties that are encountered
    interview_time <- 5/60  # each interview takes 5 min

    id <- 1
    for (d in days_s)
    {
        # Trips in the population on day d
        int_trips_d <- simu_trip_pop[simu_trip_pop$DAY == d,]
        switch(n_shifts,
            # one shift
            {
                # Cover the fishery during the entire fishing period on day d
                #shift_times = c(4,22)
                SStart <- shift_times[1] #4
                SEnd   <- shift_times[2] #22
                dur_shift <- SEnd - SStart
                int_trips <- make_roving_interviews(SStart, SEnd, trunc(dur_shift/3),
interview_time, int_trips_d)
```

```
            },
            # two shifts
            {
              if (shifts_non_overlapping){
                #shift_times = c(4, 13, 22)
                if (shifts_s[id] == "A")
                {
                    # Only consider trips completed and available for interviewing bef
ore the shift endtime
                    AStart <- shift_times[1] #4
                    AEnd   <- shift_times[2] #13
                    dur_shift <- AEnd - AStart

                    int_trips <- int_trips_d[(int_trips_d$ET) <= AEnd, ]
                    int_trips <- make_roving_interviews(AStart, dur_shift, interview_t
ime, int_trips)
                }
                else
                {
                    BStart <- shift_times[2] #13
                    BEnd   <- shift_times[3] #2
                    dur_shift <- BEnd - BStart
                    # B shift
                    # Get all trips completed and available for interviewing after the
 last shift endtime
                    int_trips <- int_trips_d[(int_trips_d$ET) > BStart,]
                    int_trips <- make_roving_interviews(BStart, dur_shift, interview_t
ime, int_trips)
                }

              }else{
                # Overlapping shift times
                #shift_times = c(8, 16, 14, 22)
                AStart <- shift_times[1] #4
                AEnd   <- shift_times[2] #16
                BStart <- shift_times[3] #14
                BEnd   <- shift_times[4] #22
                if (shifts_s[id] == "A")
                {
                    dur_shift <- AEnd - AStart

                    int_trips <- int_trips_d[(int_trips_d$ET) <= AEnd, ]
                    int_trips <- make_roving_interviews(AStart, dur_shift, interview_t
ime, int_trips)
                }
                else
                {
                    # B shift
                    dur_shift <- BEnd - BStart
                    # B shift
                    # Get all trips completed and available for interviewing after the
 last shift endtime
                    int_trips <- int_trips_d[(int_trips_d$ET) > BStart,]
                    int_trips <- make_roving_interviews(BStart, dur_shift, interview_t
ime, int_trips)
                }

              }
            },
            # Three shifts
            {
                ## shift_times = c(4, 10, 16, 22)
                if (shifts_s[id] == "AB")
                {
                    # 2 shifts A and B
                    AStart <- shift_times[1] #4
                    BEnd   <- shift_times[3] #13
                    dur_shift <- BEnd - AStart

                    int_trips <- int_trips_d[(int_trips_d$ET) <= BEnd, ]
                    int_trips <- make_roving_interviews(AStart, dur_shift, interview_t
ime, int_trips)
                }

                if (shifts_s[id] == "BC")
                {
                    #    |------| 10
                    BStart <- shift_times[2] #4
                    CEnd   <- shift_times[4] #13
                    dur_shift <- CEnd - BStart

                    int_trips <- int_trips_d[(int_trips_d$ET) > BStart, ]
                    int_trips <- make_roving_interviews(BStart, dur_shift, interview_t
ime, int_trips)

                }

                if (shifts_s[id] == "AC")
                {
                    AStart <- shift_times[1] #4
                    AEnd   <- shift_times[2] #13
                    dur_shift <- AEnd - AStart

                    int_trips_A <- int_trips_d[(int_trips_d$ET) <= AEnd, ]
                    int_trips_A <- make_roving_interviews(AStart, dur_shift, interview
_time, int_trips_A)

                    CStart <- shift_times[3] #4
                    CEnd   <- shift_times[4] #13
                    dur_shift <- CEnd - CStart
                    int_trips_C <- int_trips_d[(int_trips_d$ET) > CStart, ]
                    int_trips_C <- make_roving_interviews(CStart, dur_shift, interview
_time, int_trips_C)

                    int_trips <- rbind(int_trips_A, int_trips_C)
                }

            },
            {
                stop("shifts must be less than 4")
            }
        )


    if (id == 1)
    {
        interviews <- int_trips
    }
    else
    {
        interviews <- rbind(interviews, int_trips)
    }

    id = id + 1
    }

    interviews
}

f_get_completed_trip_interviews <- function(opt_int_sampling, simu_trip_pop, max_nints
, days_s, n_shifts, shifts_s, shift_times = c(4, 13, 22))
{
    # Get a sample of angling trips each day
    # Only trips completed and available for interviewing before the shift endtime
    sample_available <- TRUE
    id <- 1
    for (d in days_s)
    {
        int_trips_d <- subset(simu_trip_pop, DAY == d)

        switch(n_shifts,
            # one shift
            {
                # Get all interviews on day d
                int_trips <- int_trips_d
            },
            # two shifts
            {
              if (shifts_non_overlapping){
                #shift_times = c(4, 13, 22)
                if (shifts_s[id] == "A")
                {
                    # Trips with the middle trip time (ST+(ET-ST)/2) <= the middle day
 time AEnd (e.g. 13)
                    #  including trips intercept AEnd but with the middle trip time
                    #  less than AEnd, 8/2/2021
                    #  |-----|    13
                    #    |------| 13
                    #       |---o-13-|
                    #        |---o13--|
                    if (!sample_available)
                        # Get all trips that completed within the shift
                        #   and some with their middle trip times passed the shift end
time
                        int_trips <- subset(int_trips_d, ((ST+(ET-ST)/2) < shift_times
[2]))
                    else
                        # Only trips completed and available for interviewing before t
he shift endtime
                        int_trips <- subset(int_trips_d, (ET) <= shift_times[2])
                }
                else
                {
                    # B shift
                    if (!sample_available)
                        # Get trips with the middle trip times or endtimes passed the
shift starttime
                        int_trips <- subset(int_trips_d, ((ST+(ET-ST)/2) >= shift_time
s[2]))
                    else
                        # Get all trips completed and available for interviewing after
 the last shift endtime
                        int_trips <- subset(int_trips_d, (ET) > shift_times[2])
                }

              }else{
                #shift_times = c(8, 16, 14, 22)
                AStart <- shift_times[1] #4
                AEnd   <- shift_times[2] #16
                BStart <- shift_times[3] #14
                BEnd   <- shift_times[4] #22
                if (shifts_s[id] == "A")
                {
                    # Trips with the middle trip time (ST+(ET-ST)/2) <= the middle day
 time AEnd (e.g. 13)
```

```
                    #  including trips intercept AEnd but with the middle trip time
                    #  less than AEnd, 8/2/2021
                    #  |-----|    13
                    #    |------| 13
                    #       |---o-13-|
                    #        |---o13--|
                    if (!sample_available)
                        int_trips <- subset(int_trips_d, ((ST+(ET-ST)/2) < AEnd))
                    else
                        int_trips <- subset(int_trips_d, (ET) <= AEnd)
                }
                else
                {
                    # B shift
                    if (!sample_available)
                        int_trips <- subset(int_trips_d, ((ST+(ET-ST)/2) >= BStart))
                    else
                        int_trips <- subset(int_trips_d, (ET) > BStart)
                }

              }
            },
            # Three shifts
            {
                ## shift_times = c(4, 10, 16, 22)
                if (shifts_s[id] == "AB")
                {
                    # 2 shifts A and B
                    # Trips to the left of shift_times[2]: ET <= shift_times[2]
                    #  |-----|    10
                    #    |------| 10
                    if (!sample_available)
                        int_trips <- subset(int_trips_d, ((ST+(ET-ST)/2) < shift_times
[3]))
                    else
                        int_trips <- subset(int_trips_d, (ET) <= shift_times[3]) #BEnd
                }

                if (shifts_s[id] == "BC")
                {
                    if (!sample_available)
                        int_trips <-subset(int_trips_d, ((ST+(ET-ST)/2) >= shift_times
[2]))
                    else
                        int_trips <-subset(int_trips_d, (ET) > shift_times[2]) #AEnd
                }

                if (shifts_s[id] == "AC")
                {
                    if (!sample_available)
                    {
                        ##int_trips <- with(simu_trip_pop, simu_trip_pop[DAY == d & (S
T >= 16), ])
                        int_trips_A <- subset(int_trips_d, ((ST+(ET-ST)/2) < shift_tim
es[2]))
                        int_trips_C <- subset(int_trips_d, ((ST+(ET-ST)/2) >= shift_ti
mes[3]))
                        int_trips <- rbind(int_trips_A, int_trips_C)
                    }
                    else
                    {
                        int_trips_A <- subset(int_trips_d, (ET) <= shift_times[2]) #AE
nd
                        int_trips_C <- subset(int_trips_d, (ET) > shift_times[3]) #BEn
d
```

```
                        int_trips <- rbind(int_trips_A, int_trips_C)
                    }
                }

            },
            {
                stop("shifts must be less than 4")
            }
        )

    # random sample a proportion of trips (<= max_nints)
    # in each day

    if (opt_int_sampling == "fix_prop")
    {
        # Can only sample a fixed proportion of trips from each day
        # due to time limits
        prop <- 0.5
        ssize <- round(prop * nrow(int_trips))
        ri <- sample(1:nrow(int_trips), size = ssize)

        int_trip_sub <- int_trips[ri, ]
    }
    else if (opt_int_sampling == "threshhold")
    {
      # Can only sample at most max_nints trips from each day
      # due to time limits
      if (nrow(int_trips) > max_nints)
      {
        ri <- sample(1:nrow(int_trips), size = max_nints)

        int_trip_sub <- int_trips[ri, ]
      }
      else
      {
        # get all trips for light fishing days
        int_trip_sub <- int_trips
      }
    } else
    {
        # get all trips for light fishing days
        int_trip_sub <- int_trips
    }

    if (id == 1)
    {
        interviews <- int_trip_sub
    }
    else
    {
        interviews <- rbind(interviews, int_trip_sub)
    }

    id = id + 1
    }
    interviews$individualInterview <- "No"
    n_row <- nrow(interviews)
    interviews$party_no <- 1:n_row

    interviews
}


f_gen_bus_route_schedule <- function(start_of_shift, duration_route, route_proto)
{
```

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# For each sampled day, generate a creel bus-route schedule
#  with random start and direction based on
#  a Prototype route in data.frame
#  SiteNum          1   2   3   4   5   6
#  SiteName     Northport   M-22    Clinch  Bowers  Elmwood Suttons
#  CreelTime_Min    80  80  80  80  40  30
#  TravelTime_Min   30  25  20  30  25  20
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

n_sites <- nrow(route_proto)
duration_route <- duration_route * 60  # route_proto$travel_end_time[n_sites] #minutes
direct_travel <- sample(1:2, size = 1)

if (direct_travel == 1){
  route_random <- route_proto
} else {
  # counter-clockwise
  route_random <- route_proto[c(1,n_sites:2),]
}

# enlarge the data to facilitate cal
#  make the data like a ring
route_random <- rbind(route_random, route_random)

# Calculate route creel and travel times
route_random$creel_start_time <- rep(0, 2*n_sites)
route_random$creel_end_time   <- rep(0, 2*n_sites)
route_random$travel_end_time  <- rep(0, 2*n_sites)

route_random$creel_start_time[1] <- 0
route_random$creel_end_time[1]   <- route_random$creel_start_time[1] + route_random$Cr
eelTime_Min[1]
route_random$travel_end_time[1]  <- route_random$creel_end_time[1] + route_random$Trav
elTime_Min[1]

for (i in 2:(2*n_sites)){
  route_random$creel_start_time[i] <- route_random$travel_end_time[i-1]
  route_random$creel_end_time[i]   <- route_random$creel_start_time[i] + route_random$
CreelTime_Min[i]
  route_random$travel_end_time[i]  <- route_random$creel_end_time[i] + route_random$Tr
avelTime_Min[i]
}

start_time_proto <- sample(1:(duration_route-1), size = 1)

route_random[, c("creel_start_time", "creel_end_time", "travel_end_time")] <- route_ra
ndom[, c("creel_start_time", "creel_end_time", "travel_end_time")] - start_time_proto

# times of day (fraction of 1)
route_random[, c("creelStart", "creelEnd", "travelEnd")] <- (start_of_shift + route_ra
ndom[, c("creel_start_time", "creel_end_time", "travel_end_time")]/60)/24

start_in_creel = which((route_random$creel_start_time <= 0) & (0 < route_random$creel_
end_time))
start_in_travel = which((route_random$creel_end_time <= 0) & (0 < route_random$travel_
end_time))

# First site and its starttime:
# (1) if the random start fell in the creel time of the start site, then
#     Starttime(1st site) = ShiftStartTime
# (2) if the random start fell in the travel time before getting to the start site, th
en
#     Starttime(1st site) = ShiftStartTime + TAfterS
```

```
# The endtime of the first site:
# (1) if the random start fell in the creel time of the start site, then
#     endtime(1st site) = StartTime1 + TAfterS
# (2) if the random start fell in the travel time before getting to the start site, th
en
#     endtime(1st site) = StartTime1 + CreelTime(1)
shift_start_info <- NULL
if (length(start_in_creel) > 0){
  # The random shift start time is during the prototype creel duration of the start si
te
  shift_start_info$site <- start_in_creel
  shift_start_info$creel <- TRUE
  shift_start_info$time_before_start <- 0-route_random$creel_start_time[start_in_creel
]
  shift_start_info$time_after_start  <- route_random$creel_end_time[start_in_creel] -
0
  route_random$creelStart[shift_start_info$site] <- start_of_shift/24
} else {
  shift_start_info$site <- start_in_travel+1
  shift_start_info$creel <- FALSE
  shift_start_info$time_before_start <- 0-route_random$creel_end_time[start_in_travel]
  shift_start_info$time_after_start  <- route_random$travel_end_time[start_in_travel]
- 0
  route_random$creelStart[shift_start_info$site] <- (start_of_shift+shift_start_info$t
ime_after_start/60)/24
}

end_in_creel = which((route_random$creel_start_time <= duration_route) & (duration_rou
te < route_random$creel_end_time))
end_in_travel = which((route_random$creel_end_time <= duration_route) & (duration_rout
e < route_random$travel_end_time))
shift_end_info <- NULL
if (length(end_in_creel) > 0){
  # The random shift start time is during the prototype creel duration of the start si
te
  shift_end_info$site <- end_in_creel
  shift_end_info$creel <- TRUE
  shift_end_info$time_before_end <- duration_route-route_random$creel_start_time[end_i
n_creel]
  shift_end_info$time_after_end  <- route_random$creel_end_time[end_in_creel] - durati
on_route
  route_random$creelEnd[shift_end_info$site] <- (start_of_shift + duration_route/60)/2
4
  route_random$travelEnd[shift_end_info$site] <- NA
} else {
  shift_end_info$site <- end_in_travel
  shift_end_info$creel <- FALSE
  shift_end_info$time_before_end <- duration_route-route_random$creel_end_time[end_in_
travel]
  shift_end_info$time_after_end  <- route_random$travel_end_time[end_in_travel] - dura
tion_route
  route_random$travelEnd[shift_end_info$site] <- NA
}

#cat(paste("start_time_proto = ", start_time_proto, sep =""), "\n")
#cat(paste("shift_start_info:  ", sep =" "), "\n")
#print(shift_start_info)
#cat(paste("shift_end_info: ", sep =" "), "\n")
#print(shift_end_info)
#
#print(route_random)

bus_route_schedule <- route_random[(shift_start_info$site):(shift_end_info$site),c("si
te_num", "SiteName", "creelStart", "creelEnd", "travelEnd")]
```

```
bus_route_schedule[,c("creelStart", "creelEnd", "travelEnd")] <- bus_route_schedule[,c
("creelStart", "creelEnd", "travelEnd")]*24

list(direct_travel = direct_travel, start_time_proto = start_time_proto, start_in_cree
l = shift_start_info$creel, bus_route_schedule =  bus_route_schedule)
}

make_daily_bus_route_interviews <- function(start_time_shift, duration_route, route_pr
oto, trips_pop_day, opt_int_sampling, max_nints)
{

  # Obtain completed trip interviews at each bus-route creel site for a day
  # Generate a random creel and travel route schedule
  # start_time_shift -- start time of the current shift
  bus_route <- f_gen_bus_route_schedule(start_time_shift, duration_route, route_proto)
  bus_route_schedule <- bus_route$bus_route_schedule
  n_sites <- nrow(bus_route_schedule)

  trips_interviewed <- NULL
  for (i in 1:n_sites) {
    # interview all fishing parties completed fishing and returned at site i
    #   during the entire creel time period at this site
    # trips returned to i in the population on day d
    trips_at_site <- trips_pop_day[trips_pop_day$AccessSite == bus_route_schedule$Site
Name[i], ]
    # trips returned during the creel time window at i
    trips_at_site <- trips_at_site[(bus_route_schedule$creelStart[i] <= trips_at_site$
ET) & (trips_at_site$ET < bus_route_schedule$creelEnd[i]),]
    trips_interviewed <- rbind(trips_interviewed, trips_at_site)
  }

  if (nrow(trips_interviewed) > 0){
    Count_Exiting_parties <- aggregate(list(count = trips_interviewed$AccessSite), lis
t(AccessSite=trips_interviewed$AccessSite), length)
    Count_Exiting_parties <- cbind(Day = rep(trips_interviewed$DAY[1], nrow(Count_Exit
ing_parties)), Count_Exiting_parties)

    if (opt_int_sampling == "threshhold")
    {
        # Can only sample at most max_nints trips from each day
        # due to time limits
        if (nrow(trips_interviewed) > max_nints)
        {
          ri <- sample(1:nrow(trips_interviewed), size = max_nints)

          int_trip_sub <- trips_interviewed[ri, ]
        }
        else
        {
          # get all trips for light fishing days
          int_trip_sub <- trips_interviewed
        }


    } else {
          # get all trips for light fishing days
          int_trip_sub <- trips_interviewed
    }
  } else {
    n_sites <- nrow(route_proto)
    Count_Exiting_parties <- data.frame(Day = rep(trips_pop_day$DAY[1],n_sites), Acces
sSite = route_proto$SiteName, count = rep(0, n_sites))
    int_trip_sub <- NULL
    cat("No bus-route interviews sampled on day=", trips_pop_day$DAY[1], "\n")
  }
  list(interviews = int_trip_sub, counts = Count_Exiting_parties)
}


f_get_bus_route_trip_interviews <- function(route_proto, duration_route, simu_trip_pop
, days_s, n_shifts, shifts_s, shift_times, opt_int_sampling, max_nints)
{
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    # Obtain bus-route completed trip interviews
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    id <- 1
    for (d in days_s)
    {
        #cat("day = ", d, "\n")

        # Trips in the population on day d
        int_trips_d <- simu_trip_pop[simu_trip_pop$DAY == d,]
        switch(n_shifts,
            # one shift
            {
                # Cover the fishery during the entire fishing period on day d
                #shift_times = c(4,22)
                SStart <- shift_times[1] #4
                #SEnd   <- shift_times[2] #22
                #dur_shift <- SEnd - SStart
                int_trips <- make_daily_bus_route_interviews(SStart, duration_route, r
oute_proto, int_trips_d, opt_int_sampling, max_nints)
            },
            # two shifts
            {
              if (shifts_non_overlapping){
                #shift_times = c(4, 13, 22)
                AStart <- shift_times[1] #4
                AEnd   <- shift_times[2] #13
                BStart <- AEnd #13
                BEnd   <- shift_times[3] #22

                if (shifts_s[id] == "A")
                {
                    int_trips <- make_daily_bus_route_interviews(AStart, duration_rout
e, route_proto, int_trips_d, opt_int_sampling, max_nints)
                }
                else
                {
                    # B shift
                    int_trips <- make_daily_bus_route_interviews(BStart, duration_rout
e, route_proto, int_trips_d, opt_int_sampling, max_nints)
                }
              }else{
                # Overlapping shift times
                #shift_times = c(8, 16, 14, 22)
                AStart <- shift_times[1] #4
                AEnd   <- shift_times[2] #16
                BStart <- shift_times[3] #14
                BEnd   <- shift_times[4] #22
                if (shifts_s[id] == "A")
                {
                    int_trips <- make_daily_bus_route_interviews(AStart, duration_rout
e, route_proto, int_trips_d, opt_int_sampling, max_nints)
                }
                else
                {
                    int_trips <- make_daily_bus_route_interviews(BStart, duration_rout
e, route_proto, int_trips_d, opt_int_sampling, max_nints)
```

```
                }

              }
            },
            {
                stop("shifts must be less than 3")
            }
        )


      if (id == 1)
      {
        interviews <- int_trips$interviews
        counts <- int_trips$counts
      }
      else
      {
        interviews <- rbind(interviews, int_trips$interviews)
        counts <- rbind(counts, int_trips$counts)
      }

      id = id + 1
    }

    switch(n_shifts,
    # one shift
    {
      interviews$prob_sampling = 1
    },
    # two shifts
    {
      if (shifts_non_overlapping){
         interviews$prob_sampling = 1/2
      }else{
        # Overlapping shift times
        #shift_times = c(8, 16, 14, 22)
        AStart <- shift_times[1] #4
        AEnd   <- shift_times[2] #16
        BStart <- shift_times[3] #14
        BEnd   <- shift_times[4] #22

        interviews$prob_sampling = ifelse((interviews$ET >= BStart) & (interviews$ET <
 AEnd), 1, 1/2)
      }
    }
    )

    list(interviews = interviews, counts = counts)
}

make_sample_bus_route <- function(is_simu_pop, angler_trip_pop, route_proto, duration_
route, ndays_s = 20,
n_shifts = 2, shift_times, shifts_non_overlapping = FALSE, strat_by_wkday, opt_int_sam
pling, max_nints)
{
    # Days contained in the population data
    days_pop <- f_gen_days_pop(is_simu_pop, angler_trip_pop)

    if (!is_simu_pop)
    {
        # use observed data as the true population
        ndays_s <- length(days_pop)
        n_shifts <- 2
    }
    # sampled days and shifts
    days_shifts <- f_sample_days_shifts(is_simu_pop, angler_trip_pop, days_pop, ndays_
s, n_shifts, strat_by_wkday)

    # Conduct interviews
    interviews_counts <- f_get_bus_route_trip_interviews(route_proto, duration_route,
angler_trip_pop, days_shifts$days_s, n_shifts, days_shifts$shifts, shift_times, opt_in
t_sampling, max_nints)

    list(interviews = interviews_counts$interviews, counts = interviews_counts$counts)
}
```