How resource abundance and stochasticity affect animals' spatial needs

Appendix 1: Simulations

Stefano Mezzini[1] Adam T. Ford[1] Jacob R. Goheen[2]
E. Patrícia Medici[3,4,5] Michael J. Noonan[1]

[1] The Irving K. Barber Faculty of Science, The University of British Columbia, Okanagan Campus, Kelowna, Canada.
[2] Department of Zoology and Physiology, University of Wyoming, Laramie, Wyoming, United States of America
[3] Lowland Tapir Conservation Initiative (LTCI), Instituto de Pesquisas Ecológicas (IPÊ), Rodovia Dom Pedro I, km 47, Nazaré Paulista, São Paulo 12960-000, Brazil.
[4] IUCN SSC Tapir Specialist Group (TSG), Campo Grande, Brazil.
[5] Escola Superior de Conservação Ambiental E Sustentabilidade (ESCAS/IPÊ), Rodovia Dom Pedro I, km 47, Nazaré Paulista, São Paulo 12960-000, Brazil.

Contents

To do

- fix captions (incomplete, HR not V(pos)); take captions from manuscript

1 Overview

This appendix illustrates all the steps necessary to produce the simulation figures in the main manuscript. For ease of reference, we also include the figures here (figures 1 and 2). To achieve full transparency while minimizing computational times, the code illustrated in this pdf is not executed during the knitting of the document. Instead, the R Markdown document (`writing/appendix-1-simulations.Rmd`) contains code chunks that import the `RDS` files saved by the scripts used during the analysis via `R` code that is not printed in the pdf file. Although one can replicate the analyses by running the code in this pdf, we suggest only using this document for illustrative purposes and as a general guide. We suggest using the `R` scripts to replicate the simulations, instead.

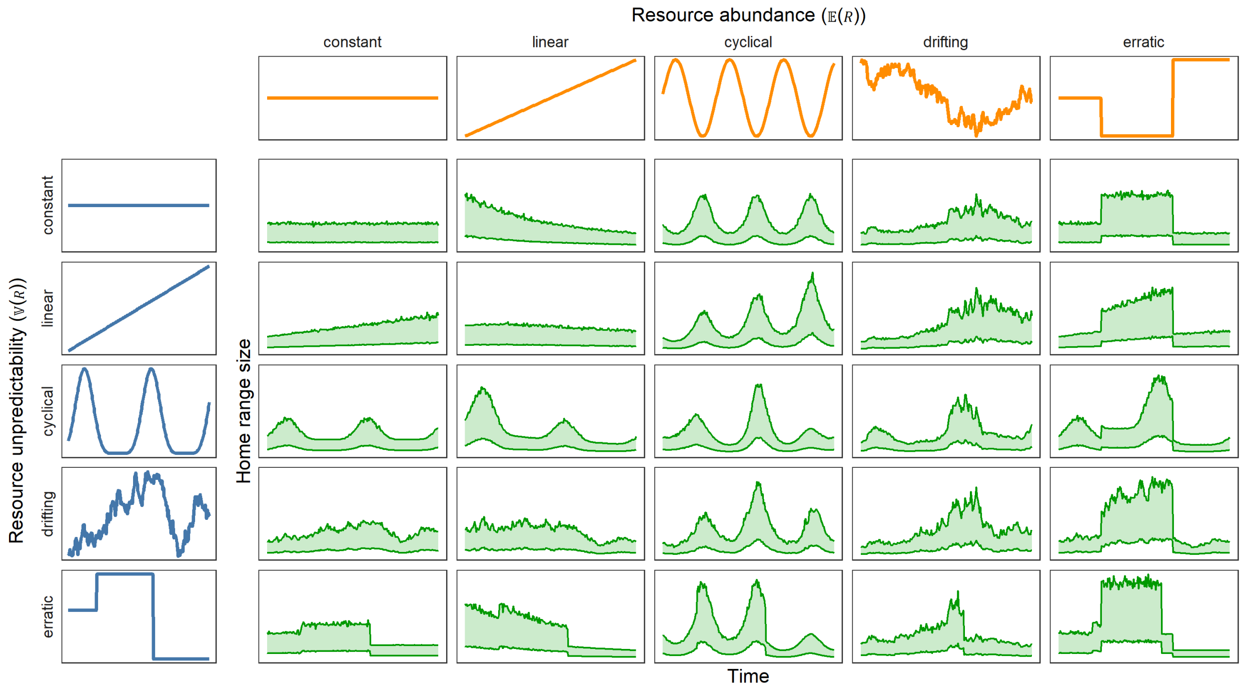

Figure 1: Simulated spatial requirements for animals living in habitats where the mean and variance in resource abundance (R) are constant, linearly increasing, cyclical, drifting, or erratic over time. The lower green line indicates the animal's core home range (50% quantile), while the top line indicates the 0.95 utilization quantile. Note how both quantiles decrease nonlinearly as $\mathbb{E}(R)$ increases, and they increase nonlinearly as $\mathbb{V}(R)$ increases. Additionally, the variance in both quantiles is higher when $\mathbb{V}(R)$ is higher, and changes in $\mathbb{V}(R)$ have greater impacts when $\mathbb{E}(R)$ is low. Simulations were run such that animals followed the same 1000 tracks at each time point starting from the location $\langle 0, 0 \rangle$ until they reach satiety, after which they returned to $\langle 0, 0 \rangle$ over the same amount of time required to reach satiety. The animal's home range was then estimated using an Ornstein-Uhlenbeck Foraging (OUF) model via the `ctmm` package (Fleming and Calabrese 2021).

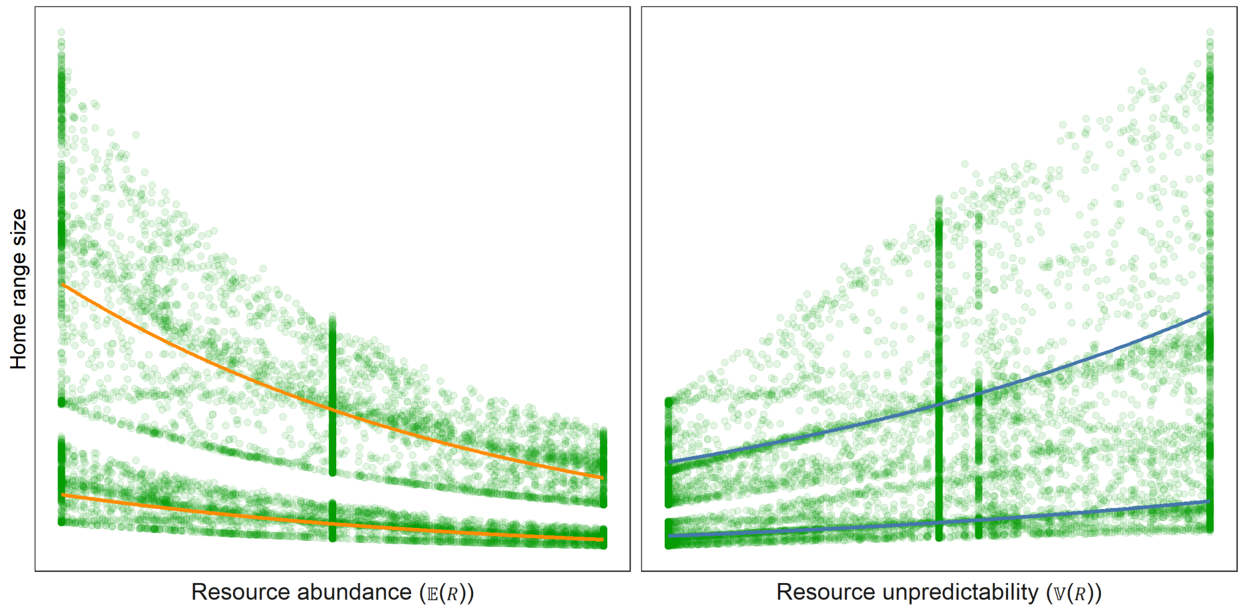

Figure 2: Effects of $\mathbb{E}(R)$ and $\mathbb{V}(R)$ on simulated spatial requirements. The relationships were estimated using a Generalized Linear Model with a Gamma family of distributions that accounted for the effects of both $\mathbb{E}(R)$ and $\mathbb{V}(R)$ as well as differences between the two quantiles. The bottom line indicates the relationships with the animal's core home range (0.5 quantile), while the top line indicates the relationship with the 0.95 utilization quantile. Note the nonlinear decrease in both utilization quantiles as $\mathbb{E}(R)$ increases and the nonlinear increase in both utilization quantiles as $\mathbb{V}(R)$ increases.

2 Simulating the movement tracks

To reduce sampling variance between simulations, we use the same set of simulated tracks for each time point in each panel (figure 3). In the `analysis/simulations/tracks.R` script, we generate $2^{10} = 1024$ tracks to check how many tracks are necessary to obtain stable home range estimates in the best- and worst-case scenarios. Most intermediate and diagnostic checks included in the `R` scripts are not included in this document for the sake of brevity and simplicity. In this script, we use the `ctmm` package (version 1.1.0, Fleming and Calabrese 2021) for movement modeling, the `raster` package (version 3.6-3, Hijmans 2022) to work with the simulated resource rasters, the `dplyr` (version 1.0.10, Wickham et al. 2022), `purrr` (version 0.3.5, Henry and Wickham 2022), and `tidyr` (version 1.2.1, Wickham and Girlich 2022) packages for data wrangling, and the the `ggplot2` (version 3.4.0, Wickham 2016) and `cowplot` (version 1.1.1, Wilke 2020) packages for plotting.

```r
# NOTE: change working directory to "hr-resource-stoch" or modify file paths
setwd('H:/GitHub/hr-resource-stoch')

# attach all necessary packages
library('ctmm')    # for generating movement models and movement modeling
library('raster') # for working with raster data
library('dplyr')  # for data wrangling
library('purrr')  # for functional programming
library('tidyr')   # for data wrangling (e.g., nested tibbles)
library('ggplot2') # for fancy plots
```

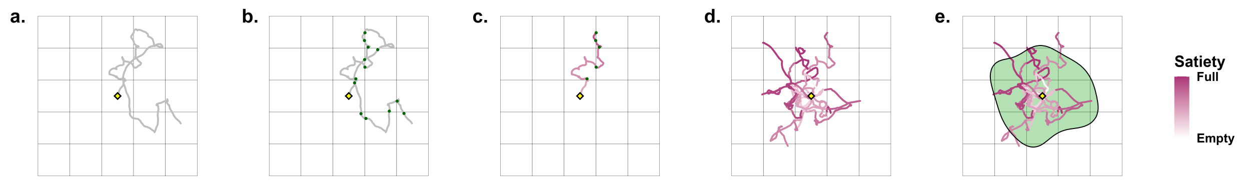

Figure 3: Overview of how animals' spatial needs were simulated. (a.) Animal tracks were simulated using an Integrated Ornstein-Uhlenbeck model (IOU; model, an infinitely diffusive and continous-velocity movement mode), starting from the point $\langle 0,0 \rangle$ (black and yellow square). (b.) Each time the track crossed into a new cell (green dots), the animal collected a random amount of resources that followed a Gamma distribution with common mean $\mu(t)$ and variance $\sigma^2(t)$. (c.) Each time the animal collected more resources, its satiety (purple) increased. Once the animal collected sufficient resources, the animal stopped moving (i.e., the track was truncated). (d.) The process was repeated $2^{10} = 1024$ times (13 tracks pictured in this panel). (e.) The final set of (truncated) tracks was then modeled using Ornstein-Uhlenbeck Foraging models to estimate the 95% and 50% home range estimates using Autocorrelated Kernel Density Estimates.

```r
source('functions/rgamma2.R') # rgamma() parameterized by mean and variance
source('analysis/figures/mean-variance-trends-panel-data.R') # mu and sigma2
source('analysis/movement-model.R') # for consistency across scripts
source('functions/get_hr.R') # for extracting gaussian home range
source('functions/label_visits.R') # decides when animal encounters food

DELTA_T <- 60 # sampling interval in seconds
SAMPLES <- seq(0, 60 * 60 * 12, by = DELTA_T) # 12 hours by DELTA_T seconds

# projected raster of resources
PROJECTION <- '+proj=aeqd +lon_0=0 +lat_0=0 +datum=WGS84'
HABITAT <- matrix(data = 1, nrow = 500, ncol = 500) %>%
  raster(xmx = 1e3, xmn = -1e3, ymx = 1e3, ymn = -1e3, crs = PROJECTION)

# infinitely diffusive movement model
model <- ctmm(tau = c(Inf, 1e3), sigma = 0.1, mu = c(0, 0))

N_DAYS <- 2^10 # number of "days" (i.e., tracks with different seeds)

# extracts tracks from a ctmm movement model for given sample times
get_tracks <- function(day, times = SAMPLES) {
  simulate(model, # ctmm movement model
           t = times, # sampling times in seconds
           seed = day, # for a consistent track each day
           complete = TRUE, # add lat, long, and timestamp to telemetry
           crs = PROJECTION) # CRS projection string
}

# generate simulated tracks (will be truncated at satiety later)
tracks <- tibble(day = 1:N_DAYS, # a simulation for each day
                 tel = map(.x = day, # set a seed for consistent results
                               .f = get_tracks)) # function to generate tracks
tracks
```

```
# A tibble: 1,024 x 2
     day tel
   <int> <list>
 1     1 <telemtry[,8]>
 2     2 <telemtry[,8]>
 3     3 <telemtry[,8]>
 4     4 <telemtry[,8]>
 5     5 <telemtry[,8]>
 6     6 <telemtry[,8]>
 7     7 <telemtry[,8]>
 8     8 <telemtry[,8]>
 9     9 <telemtry[,8]>
10    10 <telemtry[,8]>
# i 1,014 more rows
```

```r
# find patch visits and calories consumed from the tracks
tracks <-
  transmute(tracks, # drop tel column
            day, # keep day column
            track = map(.x = tel, # add a column of full tracks
                        .f = \(x) {
                          label_visits(.tel = x, .habitat = HABITAT)
                        }))

# make a single, large tibble
tracks <- tidyr::unnest(tracks, track)
tracks
```

```
# A tibble: 738,304 x 11
     day     t       x      y         vx      vy   longitude    latitude
   <int> <dbl>   <dbl>  <dbl>      <dbl>   <dbl>       <dbl>       <dbl>
 1     1     0  0      0       0         0         0           0
 2     1    60  0.0363 0.0593 -0.000529 0.00271   0.000000326 0.000000536
 3     1   120  0.0722 0.395   0.000595 0.00755   0.000000648 0.00000358
 4     1   180  0.0276 0.967  -0.00517  0.0102    0.000000248 0.00000875
 5     1   240 -0.273  1.66   -0.00323  0.0130   -0.00000245  0.0000150
 6     1   300 -0.351  2.42    0.000977 0.00894  -0.00000315  0.0000219
 7     1   360 -0.221  2.92    0.00258  0.00534  -0.00000199  0.0000264
 8     1   420 -0.126  3.39    0.00182  0.00871  -0.00000113  0.0000307
 9     1   480  0.0258 3.74    0.00272  0.00354   0.000000231 0.0000338
10     1   540  0.138  4.07    0.00133  0.00783   0.00000124  0.0000368
# i 738,294 more rows
# i 3 more variables: timestamp <dttm>, cell_id <dbl>, new_cell <lgl>
```

After generating the tracks, we performed the following tests to ensure the number and length of the tracks were large enough for results to be stable. For the sake of conciseness, the code for each of the checks is not presented in this appendix, but it is available in the `R` scripts referenced in each section.

2.1 Checking whether adding return trips is necessary

Script: `analysis/figures/simulations/return-sensitivity.R`

Adding return trips to $\langle 0, 0 \rangle$ after an animal reached satiety doubled computational times without appreciable improvements on the home range estimates (including the 95% CIs).

2.2 Checking whether the sampling interval is sufficiently small

Script: `analysis/figures/simulations/delta-t-sensitivity.R`

Using three tracks generated using three arbitrary seeds (1, 2, and 3), we explored the effects of sampling interval on the number of encounters with food (i.e., movements to new cells) detected. From each of the four checks, we created an exploratory plot that we present in figure 4.

Exploratory plot 4a. The amount of time between encounters ranged from 1 second (the minimum sampling interval) to 24 minutes and 10 seconds. Approximately 93% of the encounters (500/536, excluding the first 3 events of each track) occurred with 30 or more seconds between events.

Exploratory plot 4b. Halving the sampling interval had little to no effect on the total number of encounters for $\Delta t \lessapprox 60\,s$.

Exploratory plot 4c. A sampling interval of $\Delta t = 30$ seconds was small enough to capture fine-scale movement in the tracks while avoiding excessive amounts of data and an inflated amount of encounters with resources while an animal was near cell boundaries.

Exploratory plot 4d. The three tracks used for these exploratory plots are sufficiently different that we considered them to be a representative sample of movement tracks simulated by the OUF model. All three tracks have a reasonable amount of tortuous movement and directed movement.

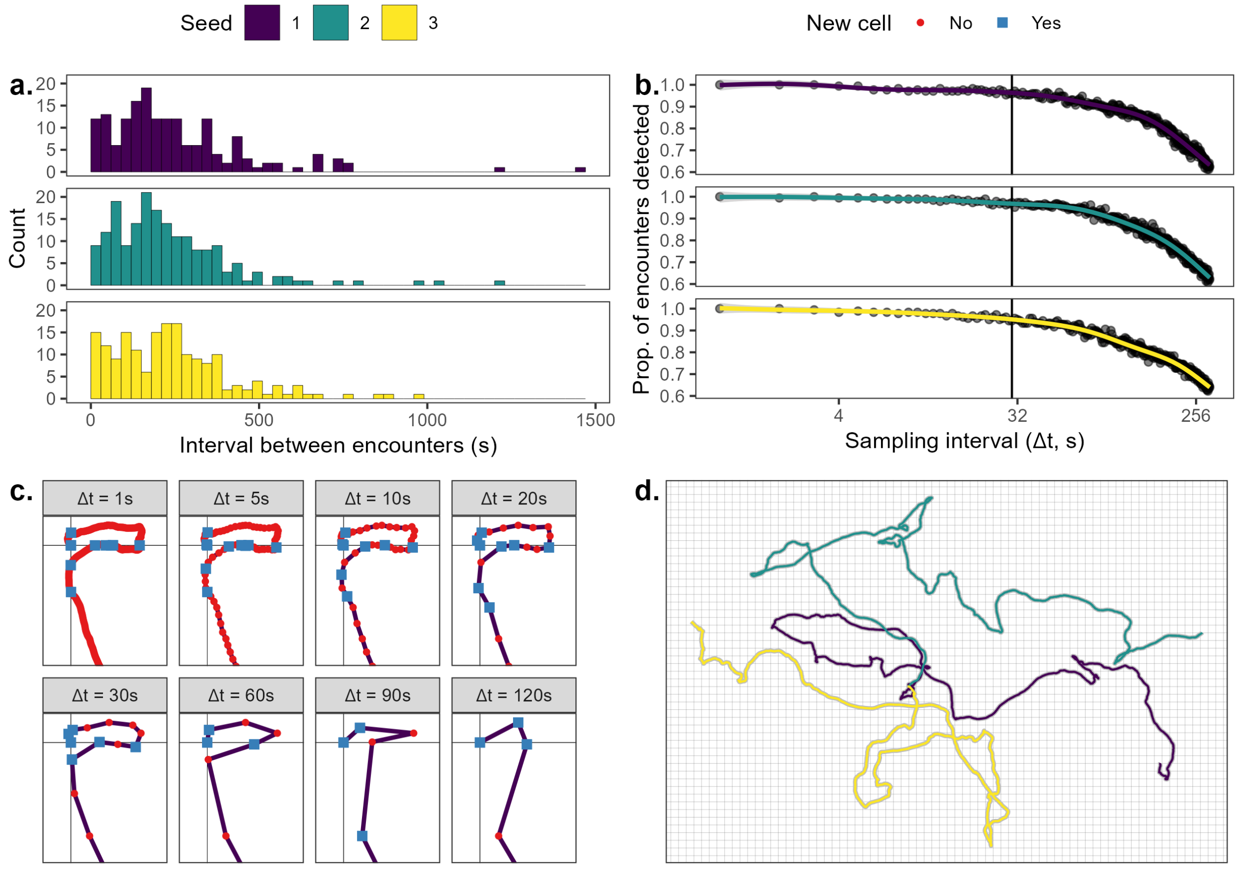

Figure 4: Exploratory plots used to decide an appropriate sampling interval. (a.) Histograms of the number of encounters as a function of the interval between encounters, with a binwith of 30 seconds. Although some encounters occur with less than 30 seconds between them, most of them occur at least 60 seconds apart. (b.) Number of encounters with food detected as a function of sampling interval. The colored lines indicate the estimated relationship based on a Generalized Additive Model fit using the `geom_smooth` function from the `ggplot2` package. Although the number of encounters detected decreases as sampling interval doubles, the loss at 30 seconds is negligible. (c.) Beginning of the track generated with seed "1" (purple line) with the location of each sample for different sampling intervals. Red dots indicate samples where the animal remained in the same cell, while the blue squares indicate when an animal was in a new cell, and thus encountered food. While the number of encounters detected decreases as the sampling interval increases, most of the encounters lost at $\Delta t = 30$s occured because the animal remained almost adjacent to the borders between cells. (d.) The three tracks used in these tests over the raster used to determine when the animals encountered food.

2.3 Checking how many tracks were necessary

Script: `analysis/hr-simulation-extreme-scenarios.R`

In this script, we check how many tracks are necessary to produce stable and accurate estimates of the spatial needs. We do so by estimating the home ranges of animals in the best-case scenario (highest $\mathbb{E}(R)$ and lowest $\mathbb{V}(R)$) and worst-case scenario (lowest $\mathbb{E}(R)$ and highest $\mathbb{V}(R)$).

```r
set.seed(1) # for consistent results

tels <- readRDS('simulations/tracks.rds') # list of telemetry tracks
tracks <- readRDS('simulations/labelled-tracks.rds') # tibble of tracks
MAX_T <- max(tracks$t) # maximum amount of exploration time

WORST <- filter(d55, mu == min(mu)) %>% # lowest mean resources
  filter(sigma2 == max(sigma2)) %>% # with highest variance
  slice(1) # take the first row only
BEST <- filter(d55, mu == max(mu)) %>% # highest mean resources
  filter(sigma2 == min(sigma2)) %>% # with lowest variance
  slice(1) # take the first row only

days <-
  # modify WORST and BEST to follow the syntax in 'offspring-simulations.R'
  transmute(bind_rows(WORST, BEST),
            animal,
            mu,
            sigma2,
            d = list(tracks),
            scenario = c('Worst case', 'Best case')) %>%
  unnest(d) %>% # unnest the datasets so we have a single, large tibble
  select(-timestamp) %>%
  # generate the food for each row from a gamma distribution
  mutate(food = rgamma2(mu = mu, sigma2 = sigma2, N = n()),
         # the animal finds food if it visits a new cell, otherwise not
         food = if_else(new_cell, food, 0)) %>%
  # end the movement once the animal has reached satiety
  group_by(day, animal, scenario) %>%
  # calculate the total visits, total calories, and if animal is full
  mutate(satiety = cumsum(food), # for diagnostics if animals don't get full
         full = satiety >= REQUIRED) %>% # did the animal reach its needs?
  filter(cumsum(full) <= 1) %>% # full only once
  ungroup()

if(FALSE) {
  # check if the ends of each day are correct and make sense
  days_end <-
    days %>%
    group_by(scenario, day) %>%
    filter(full, ! duplicated(full)) %>% # take 1st row where animal is full
    rename(t_expl = t) %>% # to avoid duplicated colnames with tracks
    # remove unneded columns (also avoids duplicated colnames with tracks)
    dplyr::select(-c(x, y, vx, vy, longitude, latitude, food))

  # check max fraction of time used (should be < 1)
  max(days_end$t_expl) / MAX_T

  # are animals are full only once/day? (should be == 1)
  sum(days_end$full) / (max(days$day) * 2)

  # plot of satiety over time by animal
  ggplot(days, aes(t, satiety, group = day)) +
    facet_wrap(~ scenario) +
    geom_line(alpha = 0.05) +
    geom_point(aes(t_expl), days_end, alpha = 0.1) +
    geom_hline(yintercept = REQUIRED, color = 'red') +
    geom_vline(xintercept = MAX_T, color = 'blue')

  # check distribution of animals
  ggplot(days_end, aes(scenario, t_expl)) +
    geom_hline(yintercept = MAX_T, color = 'red') +
    geom_violin(fill = 'forestgreen', alpha = 0.3) +
    geom_boxplot(fill = NA) +
    labs(x = '', y = 'Exploration time')

  # check home ranges of animals
  ggplot(days) +
    facet_grid(. ~ scenario) +
    coord_equal() +
    geom_hex(aes(longitude, latitude)) +
    scale_fill_distiller('Count', type = 'seq', na.value = 'transparent') +
    theme(legend.position = 'top')
}

# single estimates that eventually converge to the asymptote ----
days_summarized <-
  days %>%
  # find how long it took to reach satiety
  group_by(scenario, day) %>%
  nest(tel_day = -c(scenario, day)) %>%
```

```r
  mutate(t_expl = map_dbl(tel_day, \(d) max(d$t))) %>%
  # add days sequentially
  group_by(scenario) %>%
  mutate(t_start = lag(2 * t_expl), # add the return time before next "day"
         t_start = if_else(is.na(t_start), 0, t_start), # start at 0, not NA
         t_start = cumsum(t_start), # make start times comsecutive
         tel_day = map2(day, t_expl,
                        \(i, te) tels$tel[[i]] %>% # extract tel for the day
                          data.frame() %>% # for filtering
                          filter(t <= te))) %>% # end tracks at satiety
  unnest(tel_day) %>% # make one big dataset
  mutate(t = t + t_start, # make times consecutive
         individual.local.identifier = scenario, # ctmm identifier
         timestamp = as.POSIXct(t, origin = '2000-01-01')) %>% # use new times
  ungroup() # remove grouping by scenario

if(FALSE) {
  # check times are adding up correctly
  # best case should require less time
  # blue and red lines should have same length (within the pair)
  # black lines should be horizontal
  days_summarized %>%
    filter(day <= 10) %>%
    ggplot(aes(day, timestamp)) +
    facet_wrap(~ scenario, scales = 'free_y') +
    geom_line() +
    geom_line(aes(group = day), color = 'blue', lwd = 30) +
    geom_line(aes(day, timestamp + t_expl, group = day), color = 'red',
              lwd = 30)

  days_summarized %>%
    filter(day <= 10) %>%
    ggplot() +
    facet_wrap(~ scenario) +
    coord_equal() +
    geom_path(aes(x, y, group = day), alpha = 0.5) +
    geom_point(aes(0, 0)) +
    geom_point(aes(x, y), filter(days_summarized, day <= 10, t == 0),
               color = 'red')
}

# estimate saturation curve of home range size over number of days
saturation_days <-
  expand_grid(n_days = (2^seq(1, log2(1e3), by = 0.2)) %>%
                round() %>%
                unique(),
              case = unique(days_summarized$scenario)) %>%
  mutate(data = map2(n_days, case,
                     \(.n, .case) filter(days_summarized,
                                         day <= .n,
                                         scenario == .case)),
         tel = map(data, as.telemetry), # convert to telemetry for modeling
         theta = map(tel, \(x) ctmm.guess(data = x, interactive = FALSE)),
         m = map(1:n(), \(i) {
           cat('Fitting model', i, '\n')
           ctmm.fit(tel[[i]], theta[[i]])
           }), # fit movement model
         sigma = map_dbl(m, \(.m) ctmm:::area.covm(.m$sigma)), # var(pos)
         hr = get_hr(.sigma = sigma, quantile = 0.95)) # Gaussian home range

saveRDS(saturation_days, 'simulations/hr-saturation-days.rds')

saturation_days %>%
  select(case, n_days, sigma, hr) %>%
  readr::write_csv('simulations/hr-saturation-days.csv')

ggplot(saturation_days, aes(n_days, hr)) +
  facet_wrap(~ case, nrow = 1) +
  geom_vline(xintercept = 100, color = 'darkorange') +
  geom_smooth(method = 'gam', color = 'black',
              formula = y ~ s(x, bs = 'cs', k = 10),
              method.args = list(family = Gamma(link = 'log'))) +
  geom_point(alpha = 0.3) +
  scale_x_continuous(expression(Number~of~days~sampled~(log[2]~scale)),
                     trans = 'log2', breaks = c(2, 16, 128, 1024),
                     limits = c(2, 1100)) +
  scale_y_log10(expression(Estimated~home~range~(log[10]~scale)))
```

3 Main scripts (to be run in the following order)

1. analysis/simulations/hr − mean − variance − simulations − days.R
2. analysis/simulations/hr − mean − variance − simulations − days − summarized.R
3. analysis/simulations/hr − mean − variance − simulations − modeling.R
4. analysis/simulations/hr − mean − variance − simulations − hrs.R
5. analysis/simulations/modeling − R − and − hr.R

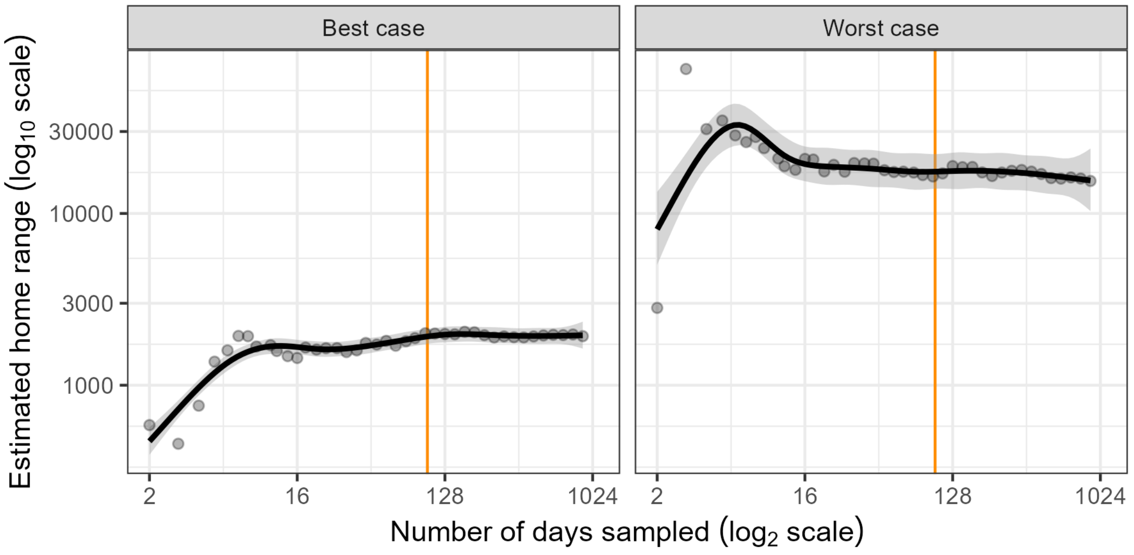

Figure 5: Estimated spatial needs as a function of the number of days sampled for an animal in a habitat with the highest $\mathbb{E}(R)$ and lowest $\mathbb{V}(R)$ (left) and an animal with the lowest $\mathbb{E}(R)$ and highest $\mathbb{V}(R)$ (right). In both cases, 100 days are sufficient to produce stable estimates of spatial needs.

References

Fleming, C. H., and J. M. Calabrese. 2021. Ctmm: Continuous-Time Movement Modeling.

Henry, L., and H. Wickham. 2022. Purrr: Functional Programming Tools.

Hijmans, R. J. 2022. Raster: Geographic Data Analysis and Modeling.

Wickham, H. 2016. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York.

Wickham, H., R. François, L. Henry, and K. Müller. 2022. Dplyr: A Grammar of Data Manipulation.

Wickham, H., and M. Girlich. 2022. Tidyr: Tidy Messy Data.

Wilke, C. O. 2020. Cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'.