Case Study: guillemot Isle of May

Table of Contents

[Overview 1](#_Toc200348816)

[Input data 4](#_Toc200348817)

[IBM configuration 4](#_Toc200348818)

[Driver information - specifying the environment 6](#_Toc200348819)

[Species information - properties that inform individual agents 8](#_Toc200348820)

[States Profile 8](#_Toc200348821)

[Driver Responses 10](#_Toc200348822)

[Create the <Species> object 11](#_Toc200348823)

[Setting up and running the IBM 11](#_Toc200348824)

[Initialisation 12](#_Toc200348825)

[Running the simulation 12](#_Toc200348826)

[Digesting the results 14](#_Toc200348827)

[Comparing scenarios 16](#_Toc200348828)

[Bodymass traces 18](#_Toc200348829)

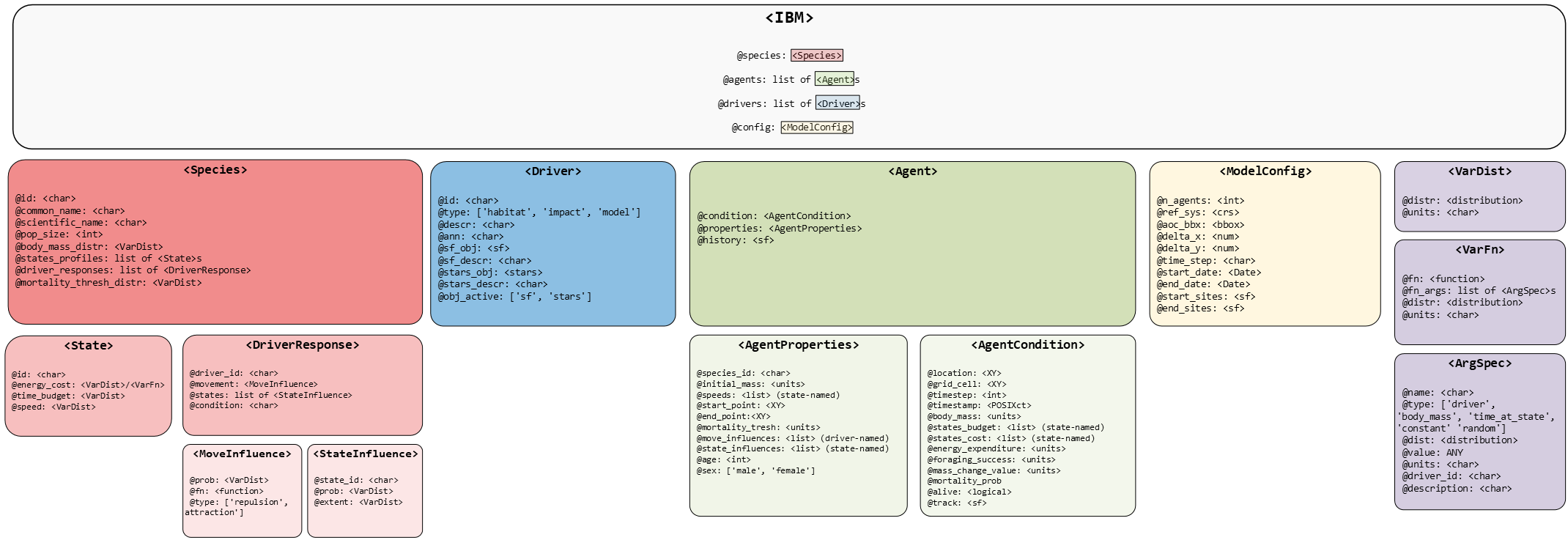
[Agent tracks 19](#_Toc200348830)

[Use of counterfactals 21](#_Toc200348831)

[References 22](#_Toc200348832)

# Overview

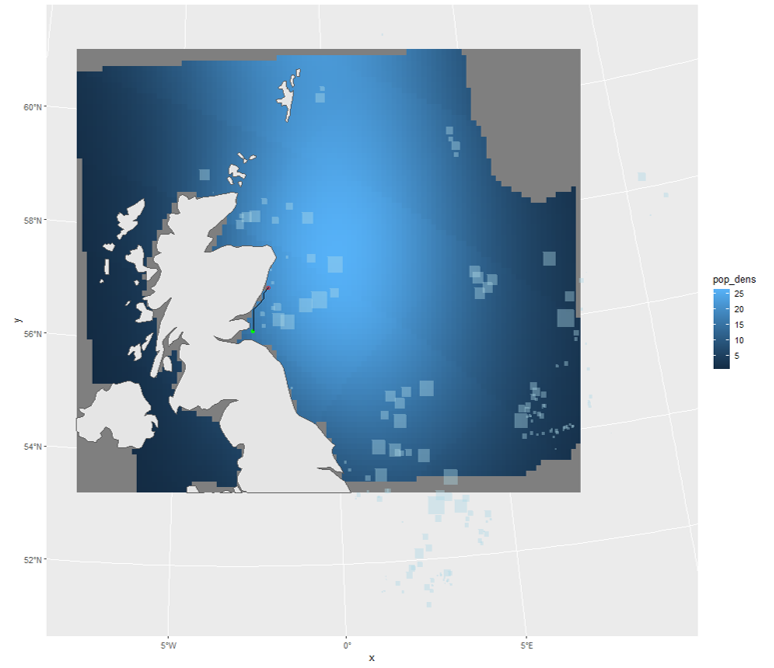
Here we use {roamR} to simulate the movement and energetics for a population of common guillemot (*Uria aalge*) on the Isle of May. For broad-scale details on the {roamR} package, refer to the general guide, with the general architecture repeated here:



**Figure 1: {roamR} arhitecture**

The simulations here cover the non-breeding season (some 9 months, from July to March) under two broad scenarios:

* An environment free of Off-Shore Windfarms (OWFs) - nominally the status quo
* An environment with many synthetic OWF developments



**Figure 2: Synthetic windfarms within North Sea, against a predicted guillemot density surface. Density is clipped to the AoC, windfarms are the light blue boxes**

The overall intention is to quantify the effects of potential displacement from these developments, based on counter-factual comparisons of animal’s condition under these scenarios. In keeping with the architecture of the {roamR} package, the following main components will be populated:

* **IBM general settings** (<ModelConfig>) sundry high-level controls for the simulation, such as the number of agents, broad spatial boundaries, spatial projections, time-steps, start & finish dates.
* **Species-level information** (<Species>) such as distributions governing initial bodymass, what behavioural states are possible, movement parameters, and how agents respond to their environment e.g. avoidance of windfarms or landmass, costs associated with activity.
* **Drivers** (<Driver>) descriptions/data that define the environment that agents may interact with or respond to e.g. sea surface temperatures, locations of windfarms, coastlines, prey fields etc.

The extent to which these can be populated will obviously vary, with the guillemot chosen here as a population that is relatively well studied. {roamR} is intentionally general and has substantial functionality that will not be used in a simulation. We will populate various elements of the simulation in turn, before turning to running the simulation and post-processing the results.

# Input data

As a relatively well-studied species/population, there are many {roamR} elements to populate. Mainly these are:

* Density maps
* SST
* Activity data
* Energetics
* Bodyweight
* Bodymass conversion

which will be a) described in detail and b) specified in {roamR} forms in the following sections. {roamR} can be run with very little data (an example of a more data sparse species is given in the alternative test case of red-throated divers), with the results being correspondingly less informative and more stochastic.

## IBM configuration

The broad configuration of the IBM is specified via the function ModelConfig().In the interest of speed, the example simulation here will not be large in terms of the number of agents to run - we’ll opt for few agents (n\_agents), whereas a typical run would be 1000s. The non-breeding season for these animals runs from approximately start of July for 9 months (start\_date, end\_date). We’ll opt for a uniform 1km^2 spatial resolution (x\_delta, y\_delta) and have everything operate on the UTM 30N coordinate system (ref\_sys). This will be the basis of ingesting and aligning the general spatial inputs. Note: the package currently requires all spatial inputs (e.g. density maps) to be provided in a common CRS projection [[1]](#footnote-1). The {sf} package is generally used for dealing with spatial data (and the interlinked {stars} package for spatio-temporal).

# Set UTM zone 30N  
utm30 <- st\_crs(32630)

Agent start and end locations (start\_sites, end\_sites) must be supplied assf objects containing two required columns: id (a unique site identifier) and prop (the proportion of agents assigned to each site).

In this example, we use the Isle of May as the sole starting location, specified by its geographic coordinates (longitude/latitude). Note that end\_sites is not utilised in this the simulation, meaning the movement model assumes agents remain at their final locations once the simulation ends. As noted earlier, the site must be re-projected to the common UTM Zone 30N coordinate reference system.

# location of colony in long/lat degrees - start/finish locations  
isle\_may <- st\_sf(  
 id = "Isle of May",  
 prop = 1,  
 geom = st\_sfc(st\_point(c(-2.5667, 56.1833))),  
 crs = 4326)   
  
# re-project to UTM 30N  
isle\_may <- st\_transform(isle\_may, crs = utm30)

In terms of bounding the entire simulation spatially, we’ve opted for a semi-arbitrary bounding box around the Isle of May that encompasses a large swathe of the North Sea and far to the west of the UK, which covers the bulk of the guillemot density maps. Note this a hard boundary in terms of simulations - data outside this will have no influence.

AoC <- st\_bbox(c(xmin = 178831, ymin = 5906535, xmax = 1174762, ymax = 6783609), crs = st\_crs(utm30))

Passing the above configuration inputs to ModelConfig() creates a <ModelConfig> object, which we assigned to guill\_ibm\_config:

# IBM Settings - assume fixed for these simulations  
  
guill\_ibm\_config <- ModelConfig(  
 n\_agents = 4,  
 ref\_sys = utm30,  
 aoc\_bbx = AoC,   
 delta\_x = 1000,  
 delta\_y = 1000,  
 delta\_time = "1 day",  
 start\_date = date("2025-07-01"),  
 end\_date = date("2025-07-01") + 270,   
 start\_sites = isle\_may  
)  
  
class(guill\_ibm\_config)  
#> [1] "ModelConfig"  
#> attr(,"package")  
#> [1] "roamR"

## Driver information - specifying the environment

Drivers define the environment that agents may interact with or respond, with each driver being specified via the Driver() function. The scope of these will be defined by the level of knowledge for the species in hand. Any environmental component that can/will be functionally linked to the animals behaviour (activity states or movement) or their energetics, must be included in the definition of the environment via the drivers.

In this application to the common guillemot, the following drivers are required - all provided as spatio-temporal datacubes (2D rasters giving values for locations over times) so the agents can query their environment at any :

* Monthly species density surfaces, for both baseline and impacted scenarios.
* Monthly energy intake maps (kJ/h), for both baseline and impacted scenarios.
* Monthly average Sea Surface Temperature (SST) maps.

{roamR} enforces a strict requirement that all input variables be accompanied by appropriate measurement units. This ensures that all computations performed during simulation are unit-aware, allowing for accurate operations and conversions.

We begin by uploading these datacubes, assigning measurement units where they are missing.

# driver spatial surfaces  
  
spec\_map <- readRDS("data/bioss\_spec\_map.rds") |>   
 mutate(density = units::set\_units(density, "counts"))  
  
spec\_imp\_map <- readRDS("data/bioss\_spec\_imp\_map.rds") |>   
 mutate(density = units::set\_units(density, "counts"))  
  
intake\_map <- readRDS("data/guill\_energy\_intake\_map.rds")  
  
imp\_intake\_map <- readRDS("data/guill\_impacted\_energy\_intake\_map.rds")  
  
sst\_map <- readRDS("data/bioss\_sst\_stars.rds") |>   
 mutate(sst = units::set\_units(sst, "degree\_Celsius")) |>   
 stars::st\_warp(crs = sf::st\_crs(spec\_map), threshold = 20028)

Next we specify the corresponding drivers, and stored the as a list of <Driver> objects. Collectively these define the environment the agents will move through. This is very flexible, and might have consisted of coastal polygons, OWF footprints, prey-fields etc. Here the feasible locations are defined by the density surfaces (so coast is implicit), and OWF are implicit in the “impact” density surfaces.

# Set up IBM drivers   
  
dens\_drv <- Driver(  
 id = "dens",  
 type = "resource",  
 descr = "species dens map",  
 stars\_obj = spec\_map,  
 obj\_active = "stars"  
)  
  
dens\_imp\_drv <- Driver(  
 id = "dens\_imp",  
 type = "resource",  
 descr = "species redist map",  
 stars\_obj = spec\_imp\_map,  
 obj\_active = "stars"  
)  
  
energy\_drv <- Driver(  
 id = "energy",  
 type = "resource",  
 descr = "energy map",  
 stars\_obj = intake\_map,  
 obj\_active = "stars"  
)  
  
imp\_energy\_drv <- Driver(  
 id = "imp\_energy",  
 type = "resource",  
 descr = "energy impact map",  
 stars\_obj = imp\_intake\_map,  
 obj\_active = "stars"  
)  
  
  
sst\_drv <- Driver(  
 id = "sst",  
 type = "habitat",  
 descr = "Sea Surface Temperature",  
 stars\_obj = sst\_map,  
 obj\_active = "stars"  
)  
  
  
# store as list for initialisation  
guill\_drivers <- list(  
 dens = dens\_drv,  
 imp\_dens = dens\_imp\_drv,  
 energy = energy\_drv,  
 imp\_energy = imp\_energy\_drv,  
 sst = sst\_drv  
)

## Species information - properties that inform individual agents

Species-level information is defined using the Species() function, which depends on a set of input objects. For clarity and modularity, these inputs should be prepared and specified beforehand.

### States Profile

We begin by defining the behavioural states to include in the model. Each state represents a specific activity, characterized by parameters such as energy expenditure, time allocation, and movement speed.

States are created using the State() function. {roamR} supports flexible specification of state properties, allowing the incorporation of stochastic variation at both the population and individual (agent) level.

Here we include 4 states:

* flying
* diving
* active on water (i.e. swimming)
* inactive on water (i.e. resting)

We start with the ‘flight’ state. For the current simulation, we assume the energetic cost of flying for each agent varies throughout the simulation, following a Normal distribution with mean 507.6 kJ/h and standard deviation of237.6 kJ/h. The source for these figures are Elliott et al. (2013).

Stochasticity can enter in various ways, here we specify the average speed of each agent to be fixed over the simulation (e.g. we’re implying relatively fast/slow animals), with agents speeds drawn from a uniform distribution, as specified below.

# user-defined function returning the energy cost of flying  
flight\_cost\_fn <- function(mean, sd){  
 e <- rnorm(1, mean, sd)  
 (max(e, 1)) |>  
 units::set\_units("kJ/h")  
}  
  
flight <- State(  
 id = "flight",   
 energy\_cost = VarFn(  
 flight\_cost\_fn,   
 args\_spec = list(mean = 507.6, sd = 237.6),   
 units = "kJ/hour"  
 ),   
 time\_budget = VarDist(0.056, "hours/day"),   
 speed = VarDist(dist\_uniform(10, 20), "m/s")  
)

The state representing the ‘diving’ activity (Elliott et al. 2013) as energy output contingent on the amount of diving. Here we are performing day-level calculations, meaning we are far from simulating at the dive level, and can use a mean dive-length without loss of generality. This is t\_dive and populated later from tag information.

Where is the dive length of dive in minutes.

# define costing function  
dive\_cost\_fn <- function(t\_dive, alpha\_mean, alpha\_sd){  
 alpha <- rnorm(1, alpha\_mean, alpha\_sd)  
 (max(alpha\*sum(1-exp(-t\_dive/1.23))/sum(t\_dive)\*60, 1)) |>  
 units::set\_units("kJ/h")  
}  
  
  
# Construct <State> object  
dive <- State(  
 id = "diving",   
 energy\_cost = VarFn(  
 dive\_cost\_fn,   
 args\_spec = list(t\_dive = 1.05, alpha\_mean = 3.71, alpha\_sd = 1.3),   
 units = "kJ/hour"  
 ),   
 time\_budget = VarDist(3.11, "hours/day"),   
 speed = VarDist(dist\_uniform(0, 1), "m/s")  
)

State representing ‘active on water’ (Buckingham et al. 2023) is a linear function in SST:

where has a mean of 113 and SD of 22. is a constant of 2.75.

active\_water\_cost\_fn <- function(sst, int\_mean, int\_sd){  
 int <- rnorm(1, int\_mean, int\_sd)  
 (max(int-(2.75\*sst), 1)) |>  
 units::set\_units("kJ/h")  
}  
  
  
# Construct <State> object  
active <- State(  
 id = "active\_on\_water",   
 energy\_cost = VarFn(  
 active\_water\_cost\_fn,   
 args\_spec = list(sst = "driver", int\_mean = 113, int\_sd = 22),   
 units = "kJ/hour"  
 ),   
 time\_budget = VarDist(10.5, "hours/day"),   
 speed = VarDist(dist\_uniform(0, 1), "m/s")  
)

State for ‘inactive on water’ (Buckingham et al. 2023), follows the same linear function in SST, but where has a mean of 72.2 and SD of 22. is similarly constant at 2.75.

inactive\_water\_cost\_fn <- function(sst, int\_mean, int\_sd){  
 int <- rnorm(1, int\_mean, int\_sd)  
 (max(int-(2.75\*sst), 1)) |>  
 units::set\_units("kJ/h")  
}  
  
  
inactive <- State(  
 id = "inactive\_on\_water",   
 energy\_cost = VarFn(  
 active\_water\_cost\_fn,   
 args\_spec = list(sst = "driver", int\_mean = 72.2, int\_sd = 22),   
 units = "kJ/hour"  
 ),   
 time\_budget = VarDist(10.3, "hours/day"),   
 speed = VarDist(dist\_uniform(0, 1), "m/s")  
)

These are combined to give a list covering all states: guill\_states:

guill\_states <- list(  
 flight = flight,  
 dive = dive,  
 active = active,  
 inactive = inactive  
)

### Driver Responses

In this section, we define species-level, agent-specific responses to the environmental drivers introduced and defined earlier. For the guillemot model, we assign the density drivers - identified as "dens" (baseline) and "dens\_imp" (impacted) - as the primary determinants of agent movement (density maps as per Buckingham et al. (2022)). For each scenario, we also specify the probability that an agent is influenced by the respective driver. In the baseline case, all agents “respond” to the density map for their movement. For driver "dens\_imp", this probability reflects how likely an agent is to respond to a OWF installation (Peschko et al. 2024) - hence their influence map differs.

resp\_dens <- DriverResponse(  
 driver\_id = "dens",  
 movement = MoveInfluence(  
 prob = VarDist(distributional::dist\_degenerate(1)),  
 type = "attraction",  
 mode = "cell-value",  
 sim\_stage = "bsln"  
 )  
)  
  
resp\_imp\_dens <- DriverResponse(  
 driver\_id = "dens\_imp",  
 movement = MoveInfluence(  
 prob = VarDist(distributional::dist\_normal(0.67, sd = 0.061)),  
 type = "attraction",  
 mode = "cell-value",  
 sim\_stage = "imp"  
 )  
)

### Create the <Species> object

In addition to the parameters defined above, we set the remaining species-level properties, including the body mass distribution (used to initialise each agent’s body mass) and the energy-to-mass conversion rate (Dunn et al. 2022), which is assumed constant across agents and simulated time steps. The distribution of bodymass at start of breeding season was drawn/inferred from Harris and Wanless (1988) (with additional advice from F. Daunt, *pers. comm.*, 2025) .

guill <- Species(  
 id = "guill",  
 common\_name = "guillemot",  
 scientific\_name = "Uria Aalge",  
 body\_mass\_distr = VarDist(dist\_normal(mean = 929, sd = 56), "g"),  
 energy\_to\_mass\_distr = VarDist(0.072, "g/kJ"),  
 states\_profile = guill\_states,  
 driver\_responses = list(resp\_dens, resp\_imp\_dens)  
)

# Setting up and running the IBM

Now the key components of the IBM have been specified, {roamR} can be used for the initialisation and running of the simulations.

## Initialisation

The intialisation stage performs two main tasks prior to running:

* The checking of inputs for conformity, some adjustments (e.g. clipping to the AoC) and derivation of of vector fields where needed.
* The generation the n\_agents as indicated in the model config object.

set.seed(1009)  
  
guill\_ibm <- xfun::cache\_rds({  
 rmr\_initiate(  
 model\_config = guill\_ibm\_config,  
 species = guill,  
 drivers = guill\_drivers  
)  
  
})  
#> ℹ Validating inputs✔ Validating inputs [386ms]  
#> ℹ Processing the AOC✔ Processing the AOC [10.8s]  
#> ℹ Cropping drivers to AOC✔ Cropping drivers to AOC [369ms]  
#> ℹ Handling movement-influencing drivers✔ Handling movement-influencing drivers [336ms]  
#> ℹ Calculate vector fields for drivers "aoc".✔ Calculate vector fields for drivers "aoc". [5.3s]  
#> ℹ Processing Activity States✔ Processing Activity States [329ms]  
#> ℹ Initialize Agents✔ Initialize Agents [1.3s]  
#> ℹ Initialize <IBM> object✔ Initialize <IBM> object [283ms]  
#> ✔ Initialization Done! 🚀

## Running the simulation

The running of the simulation involves moving each of the initialised agents through the defined environment, with monitoring of their condition through time and determining their responses to these. Parallelisation is dealt with (and assumed to be generally used) such that individual agents are piped out to independent threads of calculation. The furrr package handles the parallelisation, and usually the number of workers is slightly less than those available to allow scope for other tasks without saturating the computer.

Much of the parameterisiation and data from previous sections are encapsulated within the ibm object (guill\_ibm) passed to the simulation. Several additional parameters as per documentation can be entered here as needed. Here we specify:

* What state(s) are feeding states or resting states - the state-balancing calculations (refer main documentation) will increase/decrease these as required
* feed\_avg\_net\_energy the average net energy per unit feeding - here a tuning parameter, caclcuable directly as the energy required to balance the energetics equations for an average agent.
* target\_energy the objective of the state rebalancing in terms of daily energy. Here a modest increase is attempted, based on figures indicating a modest gain over the non-breeding season.
* smooth\_body\_mass a use-defined function to convert energy time-series to mass. Here mass deposition occurs as a function of the preceding 7 days energy intake.

plan(multisession, workers = 2)  
  
guill\_results <- xfun::cache\_rds({  
 run\_disnbs(  
 ibm = guill\_ibm,  
 run\_scen = "baseline-and-impact",   
 dens\_id = "dens",   
 intake\_id = "energy",   
 imp\_dens\_id = "dens\_imp",   
 imp\_intake\_id = "imp\_energy",   
 feed\_state\_id = "diving",   
 roost\_state\_id = "inactive\_on\_water",   
 feed\_avg\_net\_energy = units::set\_units(422, "kJ/h"),   
 target\_energy = units::set\_units(1, "kJ"),   
 smooth\_body\_mass = bm\_smooth\_opts(time\_bw = "7 days"),   
 waypnts\_res = 1000,   
 seed = 1990  
)  
  
})  
#>   
#> ── Running the DisNBS Individual-Based Model ───────────────────────────────────  
#> ℹ Performing validation checks on inputs and underlying data.✔ Performing validation checks on inputs and underlying data. [308ms]  
#> ℹ Preparing and configuring data for simulation.✔ Preparing and configuring data for simulation. [8.9s]  
#> ℹ Simulating agents' journeys under the baseline-case scenarioℹ Simulating baseline scenario ■■■■■■■■■■■■■■■■ 2/4 Agents | El… ℹ Simulating agents' journeys under the baseline-case scenario✔ Simulating agents' journeys under the baseline-case scenario [2m 7.9s]  
#> ℹ Simulating agents' journeys under the impact-case scenarioℹ Simulating impact scenario ■■■■■■■■■■■■■■■■ 2/4 Agents | Elap… ℹ Simulating agents' journeys under the impact-case scenario✔ Simulating agents' journeys under the impact-case scenario [2m 5.9s]  
#> ✔ Model simulation finished! 🛬  
  
plan(sequential)

# Digesting the results

{roamR} records an extensive amount of information from its running and monitoring of the agents. The primary output is a list, with one element for each agent. The stored agents consist of three main components (each their own class, as per the package schema):

* properties - were drawn/set at the intialisation of the simulation from the species definition, remain constant throughout
* condition - the specific condition of the agent at any point in the simulation. This will be the final condition at when the simulation completes.
* history - a detailed record of elements of the agent’s condition throughout the simulation. Spatiotemporally stamped, and includes post-processed energy-to-mass conversions.

These form the basis of any downstream calculations based on the simulation outputs. When there are impact scenarios in play, there will be more than one such list, with agents paired over the impact scenarios e.g. baseline versus impact.

We can examine the agent’s simulation histories directly - here we have two scenarios, each containing a number of agents:

# two scenarios  
 names(guill\_results)  
#> [1] "agents\_bsln" "agents\_imp"  
  
# several agents within each  
 length(guill\_results$agents\_bsln)  
#> [1] 4  
  
# one agents history  
guill\_results$agents\_bsln[[1]]@history  
#> Simple feature collection with 272 features and 14 fields  
#> Geometry type: POINT  
#> Dimension: XY  
#> Bounding box: xmin: 526895.8 ymin: 5958868 xmax: 973581 ymax: 6668645  
#> Projected CRS: WGS 84 / UTM zone 30N  
#> First 10 features:  
#> timestep timestamp track\_id body\_mass body\_mass\_smooth  
#> 1 0 <NA> 0 892.7471 [g] NA [g]  
#> 2 1 2025-07-01 1 843.4863 [g] 841.0222 [g]  
#> 3 2 2025-07-02 1 824.1961 [g] 841.3407 [g]  
#> 4 3 2025-07-03 1 860.4239 [g] 841.6555 [g]  
#> 5 4 2025-07-04 1 786.7908 [g] 841.9606 [g]  
#> 6 5 2025-07-05 1 899.6028 [g] 842.2460 [g]  
#> 7 6 2025-07-06 1 851.8596 [g] 842.5069 [g]  
#> 8 7 2025-07-07 1 797.9183 [g] 842.7435 [g]  
#> 9 8 2025-07-08 1 860.3766 [g] 842.9635 [g]  
#> 10 9 2025-07-09 1 844.7294 [g] 843.1735 [g]  
#> states\_budget.flight states\_budget.diving states\_budget.active\_on\_water  
#> 1 0.002336644 0.12976717 0.4381207  
#> 2 0.002503428 0.06765173 0.4693928  
#> 3 0.002432399 0.09410517 0.4560748  
#> 4 0.002565795 0.04442471 0.4810865  
#> 5 0.002294667 0.14540035 0.4302501  
#> 6 0.002685079 0.00000000 0.5034522  
#> 7 0.002534260 0.05616915 0.4751738  
#> 8 0.002335640 0.13014076 0.4379326  
#> 9 0.002565621 0.04448955 0.4810539  
#> 10 0.002508006 0.06594702 0.4702511  
#> states\_budget.inactive\_on\_water states\_unit\_cost.flight  
#> 1 0.4297755 0.0000 [kJ/h]  
#> 2 0.4604520 -326.0301 [kJ/h]  
#> 3 0.4473876 -580.4143 [kJ/h]  
#> 4 0.4719230 -267.6850 [kJ/h]  
#> 5 0.4220549 -380.0569 [kJ/h]  
#> 6 0.4938627 -540.9809 [kJ/h]  
#> 7 0.4661228 -458.9594 [kJ/h]  
#> 8 0.4295910 -452.3054 [kJ/h]  
#> 9 0.4718909 -173.1052 [kJ/h]  
#> 10 0.4612939 -818.3772 [kJ/h]  
#> states\_unit\_cost.diving states\_unit\_cost.active\_on\_water  
#> 1 0.00000 [kJ/h] 0.00000 [kJ/h]  
#> 2 -176.78357 [kJ/h] -108.17686 [kJ/h]  
#> 3 -210.44304 [kJ/h] -100.59187 [kJ/h]  
#> 4 -138.40550 [kJ/h] -76.24058 [kJ/h]  
#> 5 -87.91532 [kJ/h] -118.91570 [kJ/h]  
#> 6 -90.63622 [kJ/h] -132.35904 [kJ/h]  
#> 7 -87.63698 [kJ/h] -34.04614 [kJ/h]  
#> 8 -66.87361 [kJ/h] -123.85927 [kJ/h]  
#> 9 -178.81558 [kJ/h] -91.25478 [kJ/h]  
#> 10 -142.40802 [kJ/h] -84.17964 [kJ/h]  
#> states\_unit\_cost.inactive\_on\_water energy\_expenditure  
#> 1 0.000000 [kJ/h] 0.00000 [kJ]  
#> 2 -64.856872 [kJ/h] -684.17668 [kJ]  
#> 3 -29.316059 [kJ/h] -952.09712 [kJ]  
#> 4 -47.738181 [kJ/h] -448.93350 [kJ]  
#> 5 -49.479257 [kJ/h] -1471.61472 [kJ]  
#> 6 -8.571322 [kJ/h] 95.21804 [kJ]  
#> 7 -10.708986 [kJ/h] -567.88115 [kJ]  
#> 8 -46.383505 [kJ/h] -1317.06559 [kJ]  
#> 9 -59.964954 [kJ/h] -449.59019 [kJ]  
#> 10 -6.390341 [kJ/h] -666.91142 [kJ]  
#> geometry  
#> 1 POINT (526895.8 6226565)  
#> 2 POINT (543716.2 6248777)  
#> 3 POINT (543716.2 6248777)  
#> 4 POINT (543716.2 6248777)  
#> 5 POINT (543716.2 6248777)  
#> 6 POINT (543716.2 6248777)  
#> 7 POINT (543716.2 6248777)  
#> 8 POINT (543716.2 6248777)  
#> 9 POINT (543716.2 6248777)  
#> 10 POINT (543716.2 6248777)

## Comparing scenarios

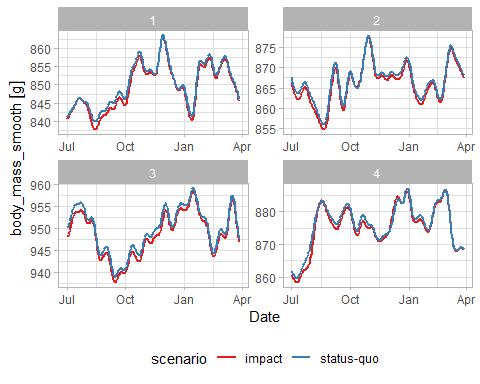
Here we extract the history from agents under the two scenarios for comparison. All agents over both scenarios are combined into one dataset.

# gather history from all agents, under the 2 scenarios, into one data frame  
guill\_history <- map(guill\_results, function(scn){  
 map(scn, ~.x@history) |>   
 setNames(1:guill\_ibm\_config@n\_agents) |>   
 list\_rbind(names\_to = "agent")   
}) |>   
 setNames(c("status-quo", "impact")) |>   
 list\_rbind(names\_to = "scenario") |>   
 mutate(  
 Date = as.Date(timestamp),   
 agent = as.numeric(agent),  
 month = lubridate::month(timestamp)  
 )  
  
# add information on whether agents are susceptible to be influenced by the impact  
infl <- tibble(  
 agent = 1:guill\_ibm\_config@n\_agents,  
 suscep = purrr::map\_lgl(guill\_ibm@agents, ~.x@properties@move\_influences$dens\_imp$infl) == TRUE  
)  
  
guill\_history <- left\_join(guill\_history, infl, by = "agent") |> st\_as\_sf()  
  
guill\_history  
#> Simple feature collection with 2176 features and 19 fields  
#> Geometry type: POINT  
#> Dimension: XY  
#> Bounding box: xmin: 363772.8 ymin: 5958868 xmax: 1143527 ymax: 6746621  
#> Projected CRS: WGS 84 / UTM zone 30N  
#> First 10 features:  
#> scenario agent timestep timestamp track\_id body\_mass body\_mass\_smooth  
#> 1 status-quo 1 0 <NA> 0 892.7471 [g] NA [g]  
#> 2 status-quo 1 1 2025-07-01 1 843.4863 [g] 841.0222 [g]  
#> 3 status-quo 1 2 2025-07-02 1 824.1961 [g] 841.3407 [g]  
#> 4 status-quo 1 3 2025-07-03 1 860.4239 [g] 841.6555 [g]  
#> 5 status-quo 1 4 2025-07-04 1 786.7908 [g] 841.9606 [g]  
#> 6 status-quo 1 5 2025-07-05 1 899.6028 [g] 842.2460 [g]  
#> 7 status-quo 1 6 2025-07-06 1 851.8596 [g] 842.5069 [g]  
#> 8 status-quo 1 7 2025-07-07 1 797.9183 [g] 842.7435 [g]  
#> 9 status-quo 1 8 2025-07-08 1 860.3766 [g] 842.9635 [g]  
#> 10 status-quo 1 9 2025-07-09 1 844.7294 [g] 843.1735 [g]  
#> states\_budget.flight states\_budget.diving states\_budget.active\_on\_water  
#> 1 0.002336644 0.12976717 0.4381207  
#> 2 0.002503428 0.06765173 0.4693928  
#> 3 0.002432399 0.09410517 0.4560748  
#> 4 0.002565795 0.04442471 0.4810865  
#> 5 0.002294667 0.14540035 0.4302501  
#> 6 0.002685079 0.00000000 0.5034522  
#> 7 0.002534260 0.05616915 0.4751738  
#> 8 0.002335640 0.13014076 0.4379326  
#> 9 0.002565621 0.04448955 0.4810539  
#> 10 0.002508006 0.06594702 0.4702511  
#> states\_budget.inactive\_on\_water states\_unit\_cost.flight  
#> 1 0.4297755 0.0000 [kJ/h]  
#> 2 0.4604520 -326.0301 [kJ/h]  
#> 3 0.4473876 -580.4143 [kJ/h]  
#> 4 0.4719230 -267.6850 [kJ/h]  
#> 5 0.4220549 -380.0569 [kJ/h]  
#> 6 0.4938627 -540.9809 [kJ/h]  
#> 7 0.4661228 -458.9594 [kJ/h]  
#> 8 0.4295910 -452.3054 [kJ/h]  
#> 9 0.4718909 -173.1052 [kJ/h]  
#> 10 0.4612939 -818.3772 [kJ/h]  
#> states\_unit\_cost.diving states\_unit\_cost.active\_on\_water  
#> 1 0.00000 [kJ/h] 0.00000 [kJ/h]  
#> 2 -176.78357 [kJ/h] -108.17686 [kJ/h]  
#> 3 -210.44304 [kJ/h] -100.59187 [kJ/h]  
#> 4 -138.40550 [kJ/h] -76.24058 [kJ/h]  
#> 5 -87.91532 [kJ/h] -118.91570 [kJ/h]  
#> 6 -90.63622 [kJ/h] -132.35904 [kJ/h]  
#> 7 -87.63698 [kJ/h] -34.04614 [kJ/h]  
#> 8 -66.87361 [kJ/h] -123.85927 [kJ/h]  
#> 9 -178.81558 [kJ/h] -91.25478 [kJ/h]  
#> 10 -142.40802 [kJ/h] -84.17964 [kJ/h]  
#> states\_unit\_cost.inactive\_on\_water energy\_expenditure Date month  
#> 1 0.000000 [kJ/h] 0.00000 [kJ] <NA> NA  
#> 2 -64.856872 [kJ/h] -684.17668 [kJ] 2025-07-01 7  
#> 3 -29.316059 [kJ/h] -952.09712 [kJ] 2025-07-02 7  
#> 4 -47.738181 [kJ/h] -448.93350 [kJ] 2025-07-03 7  
#> 5 -49.479257 [kJ/h] -1471.61472 [kJ] 2025-07-04 7  
#> 6 -8.571322 [kJ/h] 95.21804 [kJ] 2025-07-05 7  
#> 7 -10.708986 [kJ/h] -567.88115 [kJ] 2025-07-06 7  
#> 8 -46.383505 [kJ/h] -1317.06559 [kJ] 2025-07-07 7  
#> 9 -59.964954 [kJ/h] -449.59019 [kJ] 2025-07-08 7  
#> 10 -6.390341 [kJ/h] -666.91142 [kJ] 2025-07-09 7  
#> suscep geometry  
#> 1 FALSE POINT (526895.8 6226565)  
#> 2 FALSE POINT (543716.2 6248777)  
#> 3 FALSE POINT (543716.2 6248777)  
#> 4 FALSE POINT (543716.2 6248777)  
#> 5 FALSE POINT (543716.2 6248777)  
#> 6 FALSE POINT (543716.2 6248777)  
#> 7 FALSE POINT (543716.2 6248777)  
#> 8 FALSE POINT (543716.2 6248777)  
#> 9 FALSE POINT (543716.2 6248777)  
#> 10 FALSE POINT (543716.2 6248777)

### Bodymass traces

We can examine a small number of agents graphically - here their bodymass histories as implied by the simulated energetics. Note {roamR} at its heart simulates energetics, conversion functions from energy to mass are specified by the user.

p\_bdm <- guill\_history |>   
 ggplot() +  
 geom\_line(aes(x = Date, y = body\_mass\_smooth, col = scenario), linewidth = 1) +  
 scale\_color\_brewer(palette = "Set1") +  
 theme(legend.position = "bottom") +  
 facet\_wrap(~agent, ncol = 2, scales = "free")  
   
p\_bdm

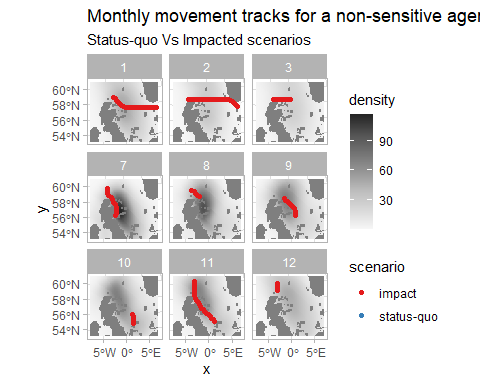


ggsave("images/bodymass.png", p\_bdm, width = 12, height = 12)

### Agent tracks

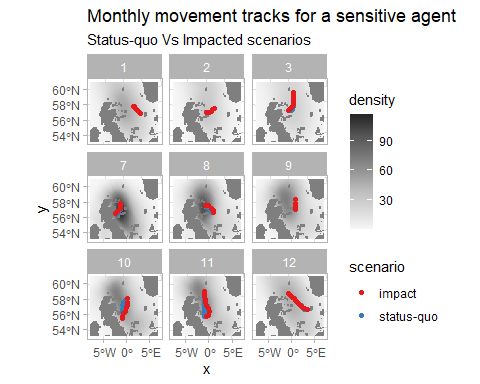
Similarly movement tracks for select agents over time can be visualised, under the two impact scenarios. Here we separate by a property of the agent - those that were assigned susceptibility to OWF in the agent initialisation - recalling this was a stochastic specification at the species level.

p\_tracks <- guill\_history |>   
 drop\_na(timestamp) |>   
 filter(suscep == FALSE) |>   
 filter(agent == last(agent)) |>   
 ggplot() +  
 stars::geom\_stars(data = spec\_imp\_map) +  
 geom\_sf(aes(col = scenario)) +  
 scale\_color\_brewer(palette = "Set1") +  
 scale\_fill\_distiller(palette = "Greys", direction = 1) +  
 facet\_wrap(~month, ncol = 3) +  
 labs(title = "Monthly movement tracks for a non-sensitive agent", subtitle = "Status-quo Vs Impacted scenarios")  
   
p\_tracks



ggsave("images/tracks\_non\_susceptile\_agent.png", p\_tracks, width = 15, height = 15)

p\_tracks <- guill\_history |>   
 drop\_na(timestamp) |>   
 filter(suscep == TRUE) |>   
 filter(agent == last(agent)) |>   
 ggplot() +  
 stars::geom\_stars(data = spec\_imp\_map) +  
 geom\_sf(aes(col = scenario)) +  
 scale\_color\_brewer(palette = "Set1") +  
 scale\_fill\_distiller(palette = "Greys", direction = 1) +  
 facet\_wrap(~month, ncol = 3) +  
 labs(title = "Monthly movement tracks for a sensitive agent", subtitle = "Status-quo Vs Impacted scenarios")  
  
p\_tracks



ggsave("images/tracks\_susceptile\_agent.png", p\_tracks, width = 15, height = 15)

## Use of counterfactals

Quantification of the impact of the perturbation can be done on any of the defined agent condition (and/or history), but for consenting the utility is through EIAs that likely use:

* Cumulative net energy
* Season-end body mass or mass change
* Distributions of activity/behavioural states
* Minimum body-mass over the season
* (relatedly) Mortality

These are not single values, but distributions representing the variability in the simulated populations. While being directly informative at a population level (e.g. the mean %-age of the population lost), the distributions are tangible for down-stream calculations. The most obvious application being in Population Viability Analyses (PVAs) that are frequently required in EIAs for consenting. There the counterfactuals may use:

* Increases in mortality/proportional reductions in population size
* Relationships between body mass and reproductive success, to alter PVA demographic parameters

For example, Natural England provide an R PVA toolset here [nepva]<https://github.com/naturalengland/Seabird_PVA_Tool>, frequently used in consenting, where the distributions of productivity can be specified under differing scenarios. The PVAs provide monte-carlo simulations of projected population sizes, which are matched (impact-to-baseline) to give population-level counterfactuals of impacts. The DisNBS simulations can be used to estimate these productivities.

# References

Buckingham, Lila, Maria I. Bogdanova, Jonathan A. Green, Ruth E. Dunn, Sarah Wanless, Sophie Bennett, Richard M. Bevan, et al. 2022. “Interspecific Variation in Non-Breeding Aggregation: A Multi-Colony Tracking Study of Two Sympatric Seabirds.” *Marine Ecology Progress Series* 684: 181–97. <https://doi.org/10.3354/meps13960>.

Buckingham, Lila, Francis Daunt, Maria I. Bogdanova, Robert W. Furness, Sophie Bennett, James Duckworth, Ruth E. Dunn, et al. 2023. “Energetic Synchrony Throughout the Non-Breeding Season in Common Guillemots from Four Colonies.” *Journal of Avian Biology* 2023 (January). <https://doi.org/10.1111/jav.03018>.

Dunn, Ruth E., Jonathan A. Green, Sarah Wanless, Mike P. Harris, Mark A Newell, Maria I. Bogdanova, Catharine Horswill, Francis Daunt, and Jason Matthiopoulos. 2022. “Modelling and Mapping How Common Guillemots Balance Their Energy Budgets over a Full Annual Cycle.” *Functional Ecology* 36 (July): 1612–26. <https://doi.org/10.1111/1365-2435.14059>.

Elliott, Kyle H., Robert E. Ricklefs, Anthony J. Gaston, Scott A. Hatch, John R. Speakman, and Gail K. Davoren. 2013. “High Flight Costs, but Low Dive Costs, in Auks Support the Biomechanical Hypothesis for Flightlessness in Penguins.” *Proceedings of the National Academy of Sciences of the United States of America* 110 (June): 9380–84. <https://doi.org/10.1073/pnas.1304838110>.

Harris, MP, and S Wanless. 1988. “Measurements and Seasonal Changes in Weight of Guillemots Uria Aalge at a Breeding Colony.” *Ringing & Migration* 9 (1): 32–36.

Peschko, Verena, Henriette Schwemmer, Moritz Mercker, Nele Markones, Kai Borkenhagen, and Stefan Garthe. 2024. “Cumulative Effects of Offshore Wind Farms on Common Guillemots (Uria Aalge) in the Southern North Sea - Climate Versus Biodiversity?” *Biodiversity and Conservation* 33 (March): 949–70. <https://doi.org/10.1007/s10531-023-02759-9>.

1. functionality to homogenise CRSs across spatial inputs during model initialization is expected to be implemented in {roamR} in the near future [↑](#footnote-ref-1)