Package 'stochLAB'

December 22, 2022

Type Package

Title Stochastic Collision Risk Model

Version 1.1.2

Description Collision Risk Models for avian fauna (seabird and migratory birds) at offshore wind farms. The base deterministic model is derived from Band (2012) <https://tethys.pnnl.gov/publications/using-collision-risk-model-assess-bird-collision-risks-offshore-wind-farms>. This was further expanded on by Masden (2015) <doi:10.7489/1659-1> and code used here is heavily derived from this work with input from Dr A. Cook at the British Trust for Ornithology. These collision risk models are useful for marine ornithologists who are working in the offshore wind industry, particularly in UK waters. However, many of the species included in the stochastic collision risk models can also be found in the North Atlantic in the United States and Canada, and could be applied there.

License GPL (>= 3)

Depends R (>= 4.0)

Imports cli, dplyr, glue, logr, magrittr, msm, pracma, purrr, rlang, stats, tibble, tidyr

Suggests rmarkdown, knitr, spelling, testthat (>= 3.0.0), covr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

LazyDataCompression bzip2

URL https://github.com/HiDef-Aerial-Surveying/stochLAB, https://hidef-aerial-surveying.github.io/stochLAB/

BugReports https://github.com/HiDef-Aerial-Surveying/stochLAB/issues

RoxygenNote 7.2.2

VignetteBuilder knitr

NeedsCompilation no

Author Grant Humphries [aut, cre] (<https://orcid.org/0000-0001-5783-9892>),
Bruno Caneco [aut],
Aonghais Cook [aut],
Elizabeth Masden [aut],
Marine Scotland [fnd, cph],
HiDef [cph],
DMPstats [cph],
Kelly Street [rev] (@kstreet13)

Maintainer Grant Humphries <grwhumphries@blackbawks.net>

Repository CRAN

Date/Publication 2022-12-22 08:50:07 UTC

R topics documented:

---

band_crm *Collision risk model, for a single species and one turbine scenario*

Description

Core function of the Collision Risk Model (CRM). Calculates the expected number of in-flight collisions per month of a seabird species on a given offshore windfarm, for a choice of model options.

Calculations are equivalent to those performed on the original CRM spreadsheet, as per Band (2012), providing backward compatibility with the original outputs.

Usage

```
band_crm(
  model_options = c("1", "2", "3", "4"),
  flight_speed,
  body_lt,
  wing_span,
  flight_type,
  avoid_rt_basic = NULL,
  avoid_rt_ext = NULL,
  noct_activity,
  prop_crh_surv = NULL,
  dens_month,
  prop_upwind,
  site_fhd = NULL,
  gen_fhd = NULL,
  rotor_speed,
  rotor_radius,
  blade_width,
  blade_pitch,
  n_blades,
  hub_height,
  chord_prof = chord_prof_5MW,
  n_turbines,
  turb_oper_month,
```

```
  wf_width,
  wf_latitude,
  tidal_offset,
  lrg_arr_corr = TRUE,
  xinc = 0.05,
  yinc = 0.05,
  ...
)
```

Arguments

`model_options`	Character vector, the model options for calculating collision risk (see **Details** section below).
`flight_speed`	Numeric value. The bird flying speed (v), in metres/sec.
`body_lt`	Numeric value. The length of the bird (L), in metres.
`wing_span`	Numeric value. The wingspan of the bird (W), in metres.
`flight_type`	A character string, either 'flapping' or 'gliding', indicating the species' characteristic flight type.
`avoid_rt_basic, avoid_rt_ext`	Numeric values. The avoidance rate for, respectively, the basic model (i.e. required for model Options 1 and 2) and the extended model (i.e. required for Options 3 and 4). Avoidance rate expresses the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision.
`noct_activity`	A numeric value. The nocturnal flight activity level, expressed as a proportion of daytime activity levels (f_night).
`prop_crh_surv`	The proportion of flights at collision risk height derived from site survey (Q_2R). Only required for model Option 1.
`dens_month`	Data frame, containing estimates of daytime in-flight bird densities per month within the windfarm footprint, in birds/km^2. It must contain the following named columns: • `month`, the month names. • `dens`, the number of birds in flight at any height per square kilometre in each month.
`prop_upwind`	Numeric value between 0-1 giving the proportion of flights upwind - defaults to 0.5.
`gen_fhd, site_fhd`	Data frame objects, with flight height distributions (fhd) of the species - the relative frequency distribution of bird flights at 1-metre height intervals from sea surface. Specifically: • `gen_fhd`, Data frame with the species' generic fhd derived from combining wider survey data. Only required for model Options 2 and 3 • `site_fhd`, Data frame with the species' site-specific fhd derived from local survey data. Only required for model Option 4 Data frames must contain the following named columns:

- height, integers representing height bands from sea surface, in metres. Function expects 0 as the first value, representing the 0-1m band.
- prop, the proportion of flights at each height band.

rotor_speed — Numeric value. The operational rotation speed, in revolutions/min.

rotor_radius — Numeric value. The radius of the rotor (R), in metres.

blade_width — Numeric value, giving the maximum blade width, in metres.

blade_pitch — Numeric value. The average blade pitch angle, the angle between the blade surface and the rotor plane (γ), in radians.

n_blades — An integer, the number of blades in rotor (b).

hub_height — A numeric value, the height of the rotor hub (H), given by the sum of rotor radius and minimum blade clearance above the highest astronomical tide (HAT), in metres.

chord_prof — A data frame with the chord taper profile of the rotor blade. Function expects two named columns:

- pp_radius, equidistant intervals of radius at bird passage point, as a proportion of rotor_radius, within the range $[0, 1]$.
- chord, the chord width at pp_radius, as a proportion of blade_width.

Defaults to a generic profile for a typical modern 5MW turbine. See chord_prof_5MW() for details.

n_turbines — Integer, the number of turbines on the wind farm (T).

turb_oper_month

Data frame, holding the proportion of time during which turbines are operational per month. The following named column are expected:

- month, the month names.
- prop_oper, the proportion of time operating, per month.

wf_width — Numeric value, the approximate longitudinal width of the wind farm, in kilometres (w).

wf_latitude — A decimal value. The latitude of the centroid of the windfarm, in degrees.

tidal_offset — A numeric value, the tidal offset, the difference between HAT and mean sea level, in metres.

lrg_arr_corr — Boolean value. If TRUE, the large array correction will be applied. This is a correction factor to account for the decay in bird density at later rows in wind farms with a large array of turbines.

yinc, xinc — numeric values, the increments along the y-axis and x-axis for numerical integration across segments of the rotor circle. Chosen values express proportion of rotor radius. By default these are set to 0.05, i.e. integration will be performed at a resolution of one twentieth of the rotor radius.

... — Additional arguments to pass on when function is called in stoch_crm(), namely rotor_grids and wf_daynight_hrs_month.

Details

Collision risk can be calculated under 4 options, specified by `model_options`:

- **Option 1** - Basic model with proportion at collision risk height derived from site survey (`prop_crh_surv`).
- **Option 2** - Basic model with proportion at collision risk height derived from a generic flight height distribution (`gen_fhd`).
- **Option 3** - Extended model using a generic flight height distribution (`gen_fhd`).
- **Option 4** - Extended model using a site-specific flight height distribution (`site_fhd`).

Where,

- Basic model - assumes a uniform distribution of bird flights at collision risk height (i.e. above the minimum and below the maximum height of the rotor blade).
- Extended model - takes into account the distribution of bird flight heights at collision risk height.

Value

Returns the expected number of bird collisions per month, for each of the chosen CRM Options. Returned months are those shared between `dens_month` and `turb_oper_month`. Output format is specific to how the function is called:

- data frame object, if called as a stand-alone function.
- list object, if called inside `stoch_crm()`.

Validation and pre-processing of inputs

`band_crm()` requirements and behaviour are dependent on how it is called:

As a stand-alone function
- All arguments are expected to be specified as describe above
- Input validation and pre-processing are carried out.

Inside `stoch_crm()`
- Assumes inputs have already been pre-processed and validated, and thence it skips those steps.
- Additional arguments `rotor_grids` and `wf_daynight_hrs_month` need to be passed to the function. Under the stochastic context, these quantities can be calculated ahead of the simulation loop to maximize performance.
- Furthermore, `gen_fhd`, `site_fhd`, `dens_month` and `turb_oper_month` can be provided as numeric vectors

Code revision and optimization

Core code performing Band calculations in Masden (2015) implementation was substantially re-factored, re-structured and streamlined to conform with conventional R packages requirements.

Furthermore, previous code underpinning key calculations for the extended model was replaced by an alternative approach, resulting in significant gains in computational speed over Masden's approach. This optimization is particularly beneficial under a stochastic context, when Band calculations are called repeatedly, potentially thousands of times. See `generate_rotor_grids()` for further details.

Examples

```
# ------------------------------------------------------
# Run with arbitrary parameter values, for illustration
# ------------------------------------------------------

# Setting a dataframe of parameters to draw from
params <- data.frame(
  flight_speed = 13.1,         # Flight speed in m/s
  body_lt = 0.85,              # Body length in m
  wing_span = 1.01,            # Wing span in m
  flight_type = "flapping",    # flapping or gliding flight
  avoid_rt_basic = 0.989,      # avoidance rate for option 1 and 2
  avoid_rt_ext = 0.981,        # extended avoidance rate for option 3 and 4
  noct_activity = 0.5,         # proportion of day birds are inactive
 prop_crh_surv = 0.13,      # proportion of birds at collision risk height (option 1 only)
  prop_upwind = 0.5,           # proportion of flights that are upwind
  rotor_speed = 15,            # rotor speed in m/s
  rotor_radius = 120,          # radius of turbine in m
  blade_width = 5,             # width of turbine blades at thickest point in m
  blade_pitch = 15,            # mean radius pitch in Radians
  n_blades = 3,                # total number of blades per turbine
  hub_height = 150,            # height of hub in m above HAT
  n_turbines = 100,            # number of turbines in the wind farm
  wf_width = 52,               # width across longest section of wind farm
  wf_latitude = 56,            # latitude of centroid of wind farm
  tidal_offset = 2.5,          # mean tidal offset from HAT of the wind farm
  lrg_arr_corr = TRUE          # apply a large array correction?
)

# Monthly bird densities
b_dens <- data.frame(
  month = month.abb,
  dens = runif(12, 0.8, 1.5)
)

# flight height distribution from Johnston et al
gen_fhd_dat <- Johnston_Flight_heights_SOSS %>%
  dplyr::filter(variable=="Gannet.est") %>%
  dplyr::select(height,prop)


# monthly operational time of the wind farm
turb_oper <- data.frame(
  month = month.abb,
  prop_oper = runif(12,0.5,0.8)
)


band_crm(
  model_options = c(1,2,3),
  flight_speed = params$flight_speed,
  body_lt = params$body_lt,
```

```
  wing_span = params$wing_span,
  flight_type = params$flight_type,
  avoid_rt_basic = params$avoid_rt_basic,
  avoid_rt_ext = params$avoid_rt_ext,
  noct_activity = params$noct_activity,
  prop_crh_surv = params$prop_crh_surv,
  dens_month = b_dens,
  prop_upwind = params$prop_upwind,
  gen_fhd = gen_fhd_dat,
  site_fhd = NULL,  # Option 4 only
  rotor_speed = params$rotor_speed,
  rotor_radius = params$rotor_radius,
  blade_width = params$blade_width,
  blade_pitch = params$blade_pitch,
  n_blades = params$n_blades,
  hub_height = params$hub_height,
  chord_prof = chord_prof_5MW,
  n_turbines = params$n_turbines,
  turb_oper_month = turb_oper,
  wf_width = params$wf_width,
  wf_latitude = params$wf_latitude,
  tidal_offset = params$tidal_offset,
  lrg_arr_corr = params$lrg_arr_corr
  )
```

`band_spreadsheet_dt` *Parameter values and outputs from Band's Collision Risk spreadsheet ("Final_Report_SOSS02_BandSpreadSheetWorkedExampl1.xlsm")*

Description

A list object comprising parameter values and outputs from Band's worked example for testing the consistency of package functions with the original model and its implementation.

Usage

```
band_spreadsheet_dt
```

Format

A list object containing:

flight_speed Bird flight speed, in m/s

rotor_radius rotor radius, in meters ...

band_spreadsheet_dt_2 *Parameter values and outputs from Band's Collision Risk spreadsheet, example nr. 2*

Description

A list object comprising parameter values and outputs from Band's worked example for testing the consistency of package functions with the original model and its implementation.

Usage

```
band_spreadsheet_dt_2
```

Format

A list object containing:

flight_speed Bird flight speed, in m/s

rotor_radius rotor radius, in meters ...

bird_pars_wide_example *Example of bird parameters stored in wide format*

Description

A data frame of (fake) data on biological parameters of three seabird species.

Usage

```
bird_pars_wide_example
```

Format

A 3 x 16 data frame, with the biological parameters (columns) for each of the 3 species (rows). Columns include:

Species Species name

AvoidanceBasic Mean of basic avoidance rate

AvoidanceBasicSD SD of basic avoidance rate

AvoidanceExtended Mean of extended avoidance rate

AvoidanceExtendedSD SD of extended avoidance rate

Body_Length Mean body length

Body_LengthSD SD of body length ...

Details

Intended to illustrate the application of `stoch_scrm()` to a multiple scenario setting, where parameter data is available from tables in wide format.

---

chord_prof_5MW *Rotor blade chord profile*

Description

A data.frame giving the blade's chord width profile, i.e. the chord width along the length of the blade, provided as a proportion of its maximum width.

Usage

```
chord_prof_5MW
```

Format

A dataframe

pp_radius radius at bird passage point, as a proportion of rotor radius (R)

chord chord width at pp_radius, as a proportion of the maximum chord width ...

Details

This is a generic profile for a typical modern 5MW turbine used for offshore generation. Due to commercial sensitivities by blade manufacturers, some of this detailed information may not be readily available for each make/model of blade and hence generic information may have to be used.

Note

`"chord_prof_5MW"` is numerically identical to `"coverC"`. `"coverC"` should become deprecated in future versions of the code

---

crm_opt1 *Number of collisions under model option 1*

Description

Wrapper function to run CRM calculations under option 1:

- Basic model, i.e. flights across collision risk height are uniformly distributed.
- Proportion at collision risk height derived from site survey.

Usage

```
crm_opt1(
  flux_factor,
  prop_crh_surv,
  avg_prob_coll,
  mth_prop_oper,
  avoidance_rate,
  lac_factor
)
```

Arguments

flux_factor	a vector containing the flux factor for each month
prop_crh_surv	The proportion of flights at collision risk height derived from site survey (Q_2R). Only required for model Option 1.
avg_prob_coll	A numeric value, the average probability of collision for a single bird transit through a rotor, assuming no avoidance action ($p_{average}$).
mth_prop_oper	A numeric vector, the proportion of time during which turbines are operational per month.
avoidance_rate	A numeric value within the interval $[0, 1]$. The avoidance rate, expressing the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision.
lac_factor	A numerical value, the large array correction factor. Defaults to 1, meaning large array correction is not applicable.

Value

A numeric vector, the expected number of collisions per month based on model option 1

See Also

get_flux_factor() for calculating the flux factor

Examples

```
 flux_fct <- get_flux_factor(
      n_turbines = 100,
      rotor_radius = 120,
      flight_speed = 13.1,
      bird_dens = c(1.19,0.85,1.05,1.45,1.41,1.45,1.12,1.45,0.93,0.902,1.06,1.23),
      daynight_hrs = Day_Length(52),
      noct_activity = 0.5
      )

turb_oper <- data.frame(
   month = month.abb,
   prop_oper = runif(12,0.5,0.8)
```

```
   )
turb_oper_month <- turb_oper$prop_oper

crm_opt1(
 flux_factor = flux_fct,
 prop_crh_surv = 0.13,
 avg_prob_coll = 0.1494609,
 mth_prop_oper = turb_oper_month,
 avoidance_rate = 0.989,
 lac_factor = 0.9998287)
```

`crm_opt2` *Number of collisions under model option 2*

Description

Wrapper function to run CRM calculations under option 2, i.e.:

- Basic model, i.e. flights across collision risk height are uniformly distributed
- Proportion at collision risk height derived from a flight height distribution (Q'_2R)

Usage

```
crm_opt2(
  d_y,
  flux_factor,
  avg_prob_coll,
  mth_prop_oper,
  avoidance_rate,
  lac_factor
)
```

Arguments

`d_y`	a vector with the proportion of bird flights at height bands within the rotor disc
`flux_factor`	a vector containing the flux factor for each month
`avg_prob_coll`	A numeric value, the average probability of collision for a single bird transit through a rotor, assuming no avoidance action ($p_{average}$).
`mth_prop_oper`	A numeric vector, the proportion of time during which turbines are operational per month.
`avoidance_rate`	A numeric value within the interval $[0, 1]$. The avoidance rate, expressing the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision.
`lac_factor`	A numerical value, the large array correction factor. Defaults to 1, meaning large array correction is not applicable.

Value

A numeric vector, the expected number of collisions per month based on model Option 2.

See Also

`get_fhd_rotor()`, `get_flux_factor()`

Examples

```
avg_collision_risk <-
 get_avg_prob_collision(
   flight_speed = 13.1,
   body_lt = 0.85,
   wing_span = 1.01,
   prop_upwind = 0.5,
   flap_glide = 1,
   rotor_speed = 15,
   rotor_radius = 120,
   blade_width = 5,
   blade_pitch = 15,
   n_blades = 3,
   chord_prof = chord_prof_5MW
 )


 gen_fhd_dat <- Johnston_Flight_heights_SOSS %>%
      dplyr::filter(variable=="Gannet.est") %>%
      dplyr::select(height,prop)

 gen_fhd <- gen_fhd_dat$prop

 gen_fhd_at_rotor <-
    get_fhd_rotor(
      hub_height = 150,
      fhd = gen_fhd,
      rotor_radius = 120,
      tidal_offset = 2.5,
      yinc = 0.05)


 flux_fct <- get_flux_factor(
      n_turbines = 100,
      rotor_radius = 120,
      flight_speed = 13.1,
      bird_dens = c(1.19,0.85,1.05,1.45,1.41,1.45,1.12,1.45,0.93,0.902,1.06,1.23),
      daynight_hrs = Day_Length(52),
      noct_activity = 0.5
      )

turb_oper <- data.frame(
   month = month.abb,
```

```
   prop_oper = runif(12,0.5,0.8)
   )
turb_oper_month <- turb_oper$prop_oper

crm_opt2(
    flux_factor = flux_fct,
    d_y = gen_fhd_at_rotor,
    avg_prob_coll = avg_collision_risk,
    mth_prop_oper = turb_oper_month,
    avoidance_rate = 0.989,
   lac_factor = 0.9998299)
```

`crm_opt3` *Number of collisions under model Option 3*

Description

Wrapper function to run CRM calculations under option 3, i.e.:

- Using the extended model, which takes into account the distribution of bird flight heights at risk height (above the minimum and below the maximum height of the rotor blade)
- Using generic flight height distribution data

Usage

```
crm_opt3(
  rotor_grids,
  d_y_gen,
  rotor_radius,
  blade_width,
  rotor_speed,
  blade_pitch,
  flight_type,
  wing_span,
  flight_speed,
  body_lt,
  n_blades,
  prop_upwind,
  avoidance_rate,
  flux_factor,
  mth_prop_oper,
  lac_factor
)
```

Arguments

rotor_grids	A list object containing geometric attributes of the rotor at equidistant points within its unit circle. This object should be built via the function generate_rotor_grids().
d_y_gen	a vector with the proportion of birds at height bands within the rotor disc, from a generic flight height distribution
rotor_radius	Numeric value. The radius of the rotor (R), in metres.
blade_width	Numeric value, giving the maximum blade width, in metres.
rotor_speed	Numeric value. The operational rotation speed, in revolutions/min.
blade_pitch	Numeric value. The average blade pitch angle, the angle between the blade surface and the rotor plane (γ), in radians.
flight_type	A character string, either 'flapping' or 'gliding', indicating the species' characteristic flight type.
wing_span	Numeric value. The wingspan of the bird (W), in metres.
flight_speed	Numeric value. The bird flying speed (v), in metres/sec.
body_lt	Numeric value. The length of the bird (L), in metres.
n_blades	An integer, the number of blades in rotor (b).
prop_upwind	Numeric value between 0-1 giving the proportion of flights upwind - defaults to 0.5.
avoidance_rate	a numeric value within the interval $[0, 1]$. The avoidance rate, expressing the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision.
flux_factor	a vector containing the flux factor for each month
mth_prop_oper	A numeric vector, the proportion of time during which turbines are operational per month.
lac_factor	A numerical value, the large array correction factor. Defaults to 1, meaning large array correction is not applicable.

Value

A numeric vector, the expected number of collisions per month based on model option 3

Examples

```
rotor_grids <- generate_rotor_grids(yinc = 0.05, xinc = 0.05, chord_prof_5MW)

gen_fhd_dat <- Johnston_Flight_heights_SOSS %>%
     dplyr::filter(variable=="Gannet.est") %>%
     dplyr::select(height,prop)

gen_fhd <- gen_fhd_dat$prop

gen_fhd_at_rotor <-
   get_fhd_rotor(
     hub_height = 150,
```

```
      fhd = gen_fhd,
      rotor_radius = 120,
      tidal_offset = 2.5,
      yinc = 0.05)


 flux_fct <- get_flux_factor(
      n_turbines = 100,
      rotor_radius = 120,
      flight_speed = 13.1,
      bird_dens = c(1.19,0.85,1.05,1.45,1.41,1.45,1.12,1.45,0.93,0.902,1.06,1.23),
      daynight_hrs = Day_Length(52),
      noct_activity = 0.5
      )

turb_oper <- data.frame(
   month = month.abb,
   prop_oper = runif(12,0.5,0.8)
   )
turb_oper_month <- turb_oper$prop_oper

crm_opt3(
  rotor_grids = rotor_grids,
  d_y_gen = gen_fhd_at_rotor,
  rotor_radius = 120,
  blade_width = 5,
  rotor_speed = 15,
  blade_pitch = 15,
  flight_type = "flapping",
  wing_span = 1.01,
  flight_speed = 13.1,
  body_lt = 0.85,
  n_blades = 3,
  prop_upwind = 0.5,
  avoidance_rate = 0.981,
  flux_factor = flux_fct,
  mth_prop_oper = turb_oper_month,
  lac_factor = 0.9998299
  )
```

`crm_opt4` *Number of collisions under model Option 4*

Description

Wrapper function to run the CRM calculations under option 4, i.e.:

- Using the extended model, which takes into account the distribution of bird flight heights at risk height (above the minimum and below the maximum height of the rotor blade)
- Using site-specific flight height distribution of the species obtained from site survey data

Usage

```
crm_opt4(
  rotor_grids,
  d_y_surv,
  rotor_radius,
  blade_width,
  rotor_speed,
  blade_pitch,
  flight_type,
  wing_span,
  flight_speed,
  body_lt,
  n_blades,
  prop_upwind,
  avoidance_rate,
  flux_factor,
  mth_prop_oper,
  lac_factor
)
```

Arguments

rotor_grids	A list object containing geometric attributes of the rotor at equidistant points within its unit circle. This object should be built via the function generate_rotor_grids().
d_y_surv	a vector with the proportion of birds at height bands within the rotor disc, from a site-specific flight height distribution
rotor_radius	Numeric value. The radius of the rotor (R), in metres.
blade_width	Numeric value, giving the maximum blade width, in metres.
rotor_speed	Numeric value. The operational rotation speed, in revolutions/min.
blade_pitch	Numeric value. The average blade pitch angle, the angle between the blade surface and the rotor plane (γ), in radians.
flight_type	A character string, either 'flapping' or 'gliding', indicating the species' characteristic flight type.
wing_span	Numeric value. The wingspan of the bird (W), in metres.
flight_speed	Numeric value. The bird flying speed (v), in metres/sec.
body_lt	Numeric value. The length of the bird (L), in metres.
n_blades	An integer, the number of blades in rotor (b).
prop_upwind	Numeric value between 0-1 giving the proportion of flights upwind - defaults to 0.5.
avoidance_rate	a numeric value within the interval $[0, 1]$. The avoidance rate, expressing the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision.
flux_factor	a vector containing the flux factor for each month

mth_prop_oper	A numeric vector, the proportion of time during which turbines are operational per month.
lac_factor	A numerical value, the large array correction factor. Defaults to 1, meaning large array correction is not applicable.

Value

A numeric vector, the expected number of collisions per month based on model option 4

Examples

```
 rotor_grids <- generate_rotor_grids(yinc = 0.05, xinc = 0.05, chord_prof_5MW)

 site_fhd_dat <- Johnston_Flight_heights_SOSS %>%
      dplyr::filter(variable=="Gannet.est") %>%
      dplyr::select(height,prop)

 site_fhd <- site_fhd_dat$prop

 surv_fhd_at_rotor <-
    get_fhd_rotor(
      hub_height = 150,
      fhd = site_fhd,
      rotor_radius = 120,
      tidal_offset = 2.5,
      yinc = 0.05)


 flux_fct <- get_flux_factor(
      n_turbines = 100,
      rotor_radius = 120,
      flight_speed = 13.1,
      bird_dens = c(1.19,0.85,1.05,1.45,1.41,1.45,1.12,1.45,0.93,0.902,1.06,1.23),
      daynight_hrs = Day_Length(52),
      noct_activity = 0.5
      )

turb_oper <- data.frame(
   month = month.abb,
   prop_oper = runif(12,0.5,0.8)
   )
turb_oper_month <- turb_oper$prop_oper

crm_opt4(
  rotor_grids = rotor_grids,
  d_y_surv = surv_fhd_at_rotor,
  rotor_radius = 120,
  blade_width = 5,
  rotor_speed = 15,
  blade_pitch = 15,
  flight_type = "flapping",
```

```
wing_span = 1.01,
flight_speed = 13.1,
body_lt = 0.85,
n_blades = 3,
prop_upwind = 0.5,
avoidance_rate = 0.981,
flux_factor = flux_fct,
mth_prop_oper = turb_oper_month,
lac_factor = 0.9998299
)
```

`Day_Length` *Total day and night hours per month*

Description

Taken from Forsythe et al.(1995) A model comparison for daylength as a function of latitude and day of year. Ecological Modelling. 80: 87 - 95

Usage

```
Day_Length(wf_latitude)
```

Arguments

`wf_latitude`	A decimal value. The latitude of the centroid of the windfarm, in degrees.

Value

data frame with total number of daylight hours and night hours in each month at the specified latitude.

Examples

```
x <- Day_Length(54.6)
```

dens_tnorm_wide_example

Example of Truncated Normal parameters for monthly estimates of bird density

Description

A data frame of (fake) monthly bird density parameters for three seabird species.

Usage

```
dens_tnorm_wide_example
```

Format

A 3 x 25 data frame, with the monthly density parameters (columns) for each of the 3 species (rows). Columns include:

Species Species name

Jan January mean density

JanSD SD of density in January

Feb February mean density

FebSD SD of density in February ...

Details

Intended to illustrate the application of `stoch_scrm()` to a multiple scenario setting, where parameter data is available from tables in wide format.

format_months

Format any month name to three letter code

Description

Format any month name to three letter code

Usage

```
format_months(months)
```

Arguments

`months`	character string or vector. The name of the month

Value

a character string. The three letter name of the month

Examples

```
format_months("January")
```

---

generate_rotor_grids *Geometric attributes at equidistant points within the rotor's unit circle*

Description

Generates a list of grids containing geometric attributes of the rotor disk at equidistant locations, taking the center of the rotor as the reference point and overlaying the left-half of the rotor disk. The size of grid cells are determined by `xinc` and `yinc`, and their values map properties of the rotor at the cell's location.

Usage

```
generate_rotor_grids(yinc = 0.05, xinc = 0.05, chord_prof)
```

Arguments

`yinc, xinc`	numeric values, the grid increments along the y-axis and x-axis (i.e. grid cell dimensions)
`chord_prof`	A data frame with the chord taper profile of the rotor blade. Function expects two named columns: • `pp_radius`, equidistant intervals of radius at bird passage point, as a proportion of `rotor_radius`, within the range $[0, 1]$. • `chord`, the chord width at `pp_radius`, as a proportion of `blade_width`. Defaults to a generic profile for a typical modern 5MW turbine. See `chord_prof_5MW()` for details.

Details

These grids are required for an alternative approach to the calculation of probability of collision under the extended model (i.e. non-uniform flight distribution at risk height).

Functions `xrisksum2`, `pcollxy` and `pcoll` used in Masden & Cook implementation (`PCollFunctions.r` script) were based on Visual Basic computations available in the original Band worksheet. This Visual Basic code was devised under a deterministic context, i.e. for one single set of collision calculations for one given species and turbine scenario. However, in a stochastic context, where potentially thousands of collision calculations are performed per species and turbine scenario, it became clear that `xrisksum2` and associated functions were highly inefficient for the task at hand.

The alternative approach streamlines computations by calculating (relative) rotor geometric attributes outside the stochastic sampling loop, which remain constant over iterations. These elements, calculated via `generate_rotor_grids`, are then applied to sampled parameters via vectorized operations. The number of calculations per iteration are thence substantially reduced, leading to significant gains in computational speed over Masden's implementation for a 1000 iterations run.

Value

A list with the following elements, taking the center of the rotor as the origin in the rotor's plane:

- `x_grid`, a 2D array of horizontal distances from the rotor's horizontal axis, as proportion of rotor radius, at each grid point
- `y_grid`, a 2D array of vertical distances from the rotor's vertical axis, as proportion of rotor radius, at each grid point
- `r_grid`, a 2D array of radial distances from rotor center, as proportion of rotor radius, at each grid point
- `phi_grid`, a 2D array of angles, relative to vertical, at each grid point
- `chord_grid`, a 2D array of blade chord width at each grid point

All elements are representative of the left-half of the rotor circle

See Also

`get_x_grid()`, `get_y_grid()`, `get_phi_grid()`

Examples

```
rotor_grids <- generate_rotor_grids(yinc = 0.05, xinc = 0.05, chord_prof_5MW)
```

---

generic_fhd_bootstraps

Bootstrap samples of generic FHDs of 25 seabird species

Description

A list object comprising bootstrap samples of the generic flight height distribution (FHD) of 25 seabird species. FHD is expressed as the proportion of bird flights at 1 metre height intervals.

Usage

```
generic_fhd_bootstraps
```

Format

A list object with 25 elements, one for each species, containing a data frame with 500 bootstrap samples of the distribution of bird flights with height. Each nested data frame contains:

height Height above sea level, in metres. First element represents the 0-1 meters height band, and height interval is 1 metre.

bootId_1 First bootstrap sample of the proportion of birds flights within each height interval ...

bootId_200 200th bootstrap sample of the proportion of birds flights within each height interval

Source

https://besjournals.onlinelibrary.wiley.com/doi/full/10.1111/1365-2664.12191

`get_avg_prob_collision` *Average single transit collision risk with no avoidance*

Description

Calculate the average probability of collision for a single bird transit at any point across the rotor, assuming no avoidance action. Required for the Basic model calculations, where flights at collision risk height are assumed to be uniformly distributed.

Usage

```
get_avg_prob_collision(
  flight_speed,
  body_lt,
  wing_span,
  prop_upwind = 0.5,
  flap_glide,
  rotor_speed,
  rotor_radius,
  blade_width,
  blade_pitch,
  n_blades,
  chord_prof = chord_prof_5MW
)
```

Arguments

`flight_speed`	Numeric value. The bird flying speed (v), in metres/sec.
`body_lt`	Numeric value. The length of the bird (L), in metres.
`wing_span`	Numeric value. The wingspan of the bird (W), in metres.
`prop_upwind`	Numeric value between 0-1 giving the proportion of flights upwind - defaults to 0.5.
`flap_glide`	Numeric value representing the correction for flapping or gliding birds (F).
`rotor_speed`	Numeric value. The operational rotation speed, in revolutions/min.
`rotor_radius`	Numeric value. The radius of the rotor (R), in metres.
`blade_width`	Numeric value, giving the maximum blade width, in metres.
`blade_pitch`	Numeric value. The average blade pitch angle, the angle between the blade surface and the rotor plane (γ), in radians.
`n_blades`	An integer, the number of blades in rotor (b).
`chord_prof`	A data frame with the chord taper profile of the rotor blade. Function expects two named columns: • `pp_radius`, equidistant intervals of radius at bird passage point, as a proportion of `rotor_radius`, within the range $[0, 1]$.

- chord, the chord width at pp_radius, as a proportion of blade_width.

Defaults to a generic profile for a typical modern 5MW turbine. See chord_prof_5MW() for details.

Details

Methodology and assumptions underpinning get_avg_prob_collision are described in "Stage C" of Band (2012).

Value

A numeric value. The average collision probability (risk) for a bird flying through any point of the rotor circle area.

Examples

```
get_avg_prob_collision(
    flight_speed = 13.1,
    body_lt = 0.85,
    wing_span = 1.01,
    prop_upwind = 0.5,
    flap_glide = 1,
    rotor_speed = 15,
    rotor_radius = 120,
    blade_width = 5,
    blade_pitch = 15,
    n_blades = 3,
    chord_prof = chord_prof_5MW
    )
```

---

get_collisions_basic *Get expected collisions based on the basic model*

Description

Provides the number of collisions expected to occur by month. The basic model assumes a uniform distribution of bird flights at risk height (i.e. between min and max rotor height).

Usage

```
get_collisions_basic(
  n_transits,
  avg_prob_coll,
  mth_prop_oper,
  avoidance_rate,
  lac_factor = 1
)
```

Arguments

n_transits	A numeric vector, the estimated number of bird flights crossing the rotors of the wind farm at each time period.
avg_prob_coll	A numeric value, the average probability of collision for a single bird transit through a rotor, assuming no avoidance action ($p_{average}$).
mth_prop_oper	A numeric vector, the proportion of time during which turbines are operational per month.
avoidance_rate	A numeric value within the interval $[0, 1]$. The avoidance rate, expressing the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision.
lac_factor	A numerical value, the large array correction factor. Defaults to 1, meaning large array correction is not applicable.

Value

A numeric vector. The expected number of collisions at each time periods

Examples

```
turb_oper <- data.frame(
 month = month.abb,
 prop_oper = runif(12,0.5,0.8)
 )
 mth_oper_month <- turb_oper$prop_oper

 flux_factor <- get_flux_factor(
   n_turbines = 100,
   rotor_radius = 120,
   flight_speed = 13.1,
   bird_dens = c(1.19,0.85,1.05,1.45,1.41,1.45,1.12,1.45,0.93,0.902,1.06,1.23),
   daynight_hrs = Day_Length(52),
   noct_activity = 0.5
 )

 prop_crh_surv <- 0.13

 n_transits_opt1 <- flux_factor * prop_crh_surv

 get_collisions_basic(
   n_transits = n_transits_opt1,
   avg_prob_coll = 0.1494609,
   mth_prop_oper = mth_oper_month,
   avoidance_rate = 0.989,
   lac_factor = 0.9998287
 )
```

get_collisions_extended

Number of collisions based on the extended model

Description

Calculates the expected number of collisions under the extended model, i.e. taking into account the distribution of bird flight heights at risk height (above the minimum and below the maximum height of the rotor blade)

Usage

```
get_collisions_extended(
  rotor_grids,
  d_y,
  rotor_radius,
  blade_width,
  rotor_speed,
  blade_pitch,
  flight_type,
  wing_span,
  flight_speed,
  body_lt,
  n_blades,
  prop_upwind,
  avoidance_rate,
  flux_factor,
  mth_prop_oper,
  lac_factor
)
```

Arguments

rotor_grids	A list object containing geometric attributes of the rotor at equidistant points within its unit circle. This object should be built via the function generate_rotor_grids().
d_y	a vector with the proportion of birds at height bands within the rotor disc
rotor_radius	Numeric value. The radius of the rotor (R), in metres.
blade_width	Numeric value, giving the maximum blade width, in metres.
rotor_speed	Numeric value. The operational rotation speed, in revolutions/min.
blade_pitch	Numeric value. The average blade pitch angle, the angle between the blade surface and the rotor plane (γ), in radians.
flight_type	A character string, either 'flapping' or 'gliding', indicating the species' characteristic flight type.
wing_span	Numeric value. The wingspan of the bird (W), in metres.

flight_speed	Numeric value. The bird flying speed (v), in metres/sec.
body_lt	Numeric value. The length of the bird (L), in metres.
n_blades	An integer, the number of blades in rotor (b).
prop_upwind	Numeric value between 0-1 giving the proportion of flights upwind - defaults to 0.5.
avoidance_rate	a numeric value within the interval $[0, 1]$. The avoidance rate, expressing the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision.
flux_factor	a vector containing the flux factor for each month
mth_prop_oper	A numeric vector, the proportion of time during which turbines are operational per month.
lac_factor	A numerical value, the large array correction factor. Defaults to 1, meaning large array correction is not applicable.

Value

Numeric vector with the expected number of collisions per month based on the extended model

Note

The original Masden code for extendend model (Option 3) included the calculation of the "Flux Integral" and "Average collision risk for single rotor transit". These quantities are not required for the calculation of collisions under the extended model, nor are used anywhere else in the package. Therefore, those calculations were dropped from the current implementation to keep computations to a minimum required.

See Also

generate_rotor_grids()

Examples

```
rotor_grids <- generate_rotor_grids(yinc = 0.05, xinc = 0.05, chord_prof_5MW)

gen_fhd_dat <- Johnston_Flight_heights_SOSS %>%
     dplyr::filter(variable=="Gannet.est") %>%
     dplyr::select(height,prop)

gen_fhd <- gen_fhd_dat$prop

gen_fhd_at_rotor <-
   get_fhd_rotor(
     hub_height = 150,
     fhd = gen_fhd,
     rotor_radius = 120,
     tidal_offset = 2.5,
     yinc = 0.05)
```

```
 flux_fct <- get_flux_factor(
      n_turbines = 100,
      rotor_radius = 120,
      flight_speed = 13.1,
      bird_dens = c(1.19,0.85,1.05,1.45,1.41,1.45,1.12,1.45,0.93,0.902,1.06,1.23),
      daynight_hrs = Day_Length(52),
      noct_activity = 0.5
      )

turb_oper <- data.frame(
   month = month.abb,
   prop_oper = runif(12,0.5,0.8)
   )
turb_oper_month <- turb_oper$prop_oper

get_collisions_extended(
  rotor_grids = rotor_grids,
  d_y = gen_fhd_at_rotor, # ... with generic FHD = Option 3, with site FHD = Option 4
  rotor_radius = 120,
  blade_width = 5,
  rotor_speed = 15,
  blade_pitch = 15,
  flight_type = "flapping",
  wing_span = 1.01,
  flight_speed = 13.1,
  body_lt = 0.85,
  n_blades = 3,
  prop_upwind = 0.5,
  avoidance_rate = 0.981,
  flux_factor = flux_fct,
  mth_prop_oper = turb_oper_month,
  lac_factor = 0.9998299
  )
```

`get_fhd_rotor` *Returns the proportion of birds at height bands along the rotor*

Description

Derive the expected proportion of bird flights at equidistant height bands between the lowest and highest points of the rotor. In other words, returns the flight height distribution at collision risk ($d(y)$).

Usage

```
get_fhd_rotor(hub_height, fhd, rotor_radius, tidal_offset, yinc = 0.05)
```

Arguments

hub_height	A numeric value, the height of the rotor hub (H), given by the sum of rotor radius and minimum blade clearance above the highest astronomical tide (HAT), in metres.
fhd	A numeric vector, the proportion of bird flights per-1 metre height bands from sea surface, starting from 0-1 metre band.
rotor_radius	A numeric value, the radius of the rotor (R), in metres.
tidal_offset	A numeric value, the tidal offset, the difference between HAT and mean sea level, in metres.
yinc	A numeric value, the increment for height bands between the lowest and highest points of the rotor, expressed as a proportion of rotor radius (defaults is 0.05).

Value

A numeric vector, the flight height distribution at collision risk, i.e. between minimum and maximum rotor height.

Examples

```
gen_fhd_dat <- Johnston_Flight_heights_SOSS %>%
      dplyr::filter(variable=="Gannet.est") %>%
      dplyr::select(height,prop)
if(is.data.frame(gen_fhd_dat)) gen_fhd <- gen_fhd_dat$prop

get_fhd_rotor(
  hub_height = 150,
  fhd = gen_fhd,
  rotor_radius = 120,
  tidal_offset = 2.5,
  yinc = 0.05)
```

---

get_flux_factor *Flux factor*

Description

Returns the flux factor, expressing the total number of bird flights through the rotors of the wind farm per month if all flights occurred within the rotor's circle area of all turbines, i.e. before the proportion at risk height and avoidance level are taken into account.

Usage

```
get_flux_factor(
  n_turbines,
  rotor_radius,
  flight_speed,
  bird_dens,
  daynight_hrs,
  noct_activity
)
```

Arguments

`n_turbines`	An integer, the number of turbines on the wind farm (T).
`rotor_radius`	A numeric value, the radius of the rotor (R), in metres.
`flight_speed`	A numeric value, the bird flight speed (v), in metres/sec.
`bird_dens`	A numeric vector with daytime in-flight bird densities (D_A), for each month, in birds/km^2.
`daynight_hrs`	A data frame with the total number of daylight hours and night hours at the wind farm site's location, in each month. It must contain, at least, the following columns: • `Month`, name of the month. • `Day`, daylight duration, in decimal hours. • `Night`, night time duration, in decimal hour
`noct_activity`	A numeric value. The nocturnal flight activity level, expressed as a proportion of daytime activity levels (f_night).

Details

The flux factor is used for other model calculations. Methodology and assumptions underpinning `get_flux_factor` are described in "Stage B" of Band (2012)

Value

The number of bird flights potentially transiting through rotors at each time period (assuming no avoidance), if all flights occur within the rotor's circular area.

Examples

```
get_flux_factor(
    n_turbines = 100,
    rotor_radius = 120,
    flight_speed = 13.1,
    bird_dens = c(1.19,0.85,1.05,1.45,1.41,1.45,1.12,1.45,0.93,0.902,1.06,1.23),
    daynight_hrs = Day_Length(52),
    noct_activity = 0.5
    )
```

`get_lac_factor` *Large array correction factor*

Description

A correction factor for large windfarms with a large array of turbines, to take into account the depletion of bird density in later rows of the site.

Usage

```
get_lac_factor(
  n_turbines,
  rotor_radius,
  avoidance_rate,
  avg_prob_coll,
  avg_prop_operational,
  wf_width
)
```

Arguments

`n_turbines`	Integer, the number of turbines on the wind farm (T).
`rotor_radius`	Numeric value, the radius of the rotor (R), in metres.
`avoidance_rate`	Numeric value within the interval $[0, 1]$. The species avoidance rate, expressing the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision. (A).
`avg_prob_coll`	Numeric value, the average probability of collision for a single bird transit through a rotor, assuming no avoidance action ($p_average$).
`avg_prop_operational`	Numeric value, the average proportion of time turbines are operational through the year.
`wf_width`	Numeric value, the approximate longitudinal width of the wind farm, in kilometres (w).

Value

The large array correction factor to be applied

Examples

```
get_lac_factor(
   n_turbines = 100,
   rotor_radius = 120,
   avoidance_rate = 0.989,
   avg_prob_coll = 0.1494609,
   avg_prop_operational = 0.6388292,
```

```
    wf_width = 52
    )
```

`get_mig_flux_factor` *Migration Flux factor*

Description

Returns the migratory flux factor, expressing the total number of bird flights through the rotors of the wind farm per month, if all flights occur within the rotor's circle area of all turbines, and assuming birds take no avoiding action.

Usage

```
get_mig_flux_factor(n_turbines, rotor_radius, wf_width, popn_est)
```

Arguments

`n_turbines`	An integer. The number of turbines on the wind farm (T).
`rotor_radius`	An integer. The radius of the rotor (R), in metres
`wf_width`	An integer. The width (in km) of the
`popn_est`	An integer. The population estimate from the spatial line sampling technique

Details

The flux factor is used for other model calculations. Methodology and assumptions underpinning `get_mig_flux_factor` are described in "Stage B" of Band (2012)

Value

The number of bird flights potentially transiting through rotors at each time period (assuming no avoidance), if all flights occur within the rotor's circular area

Examples

```
get_mig_flux_factor(
     n_turbines = 100,
     rotor_radius = 120,
     wf_width = 51,
     popn_est = 10000
)
```

get_pcoll_grid *Grid of probabilities of single transit collision at points in rotor circle*

Description

Calculates single transit collision risk at grid points (X, Y) inside the left-half of the rotor circle, for a given flight direction (upwind or downwind), and assuming no avoidance action. The origin of the (X, Y) coordinates is the rotor center

Usage

```
get_pcoll_grid(
  rotor_grids,
  direction,
  rotor_radius,
  blade_width,
  rotor_speed,
  blade_pitch,
  flight_type,
  n_blades,
  flight_speed,
  wing_span,
  body_lt
)
```

Arguments

rotor_grids	A list object containing geometric attributes of the rotor at equidistant points within its unit circle. This object should be built via the function `generate_rotor_grids()`.
direction	an integer, indicating the flight direction: 1 for upwind; -1 for downwind.
rotor_radius	Numeric value. The radius of the rotor (R), in metres.
blade_width	Numeric value, giving the maximum blade width, in metres.
rotor_speed	Numeric value. The operational rotation speed, in revolutions/min.
blade_pitch	Numeric value. The average blade pitch angle, the angle between the blade surface and the rotor plane (γ), in radians.
flight_type	A character string, either 'flapping' or 'gliding', indicating the species' characteristic flight type.
n_blades	An integer, the number of blades in rotor (b).
flight_speed	Numeric value. The bird flying speed (v), in metres/sec.
wing_span	Numeric value. The wingspan of the bird (W), in metres.
body_lt	Numeric value. The length of the bird (L), in metres.

Details

Methodology and assumptions underpinning `get_pcoll_grid` are described in section "Extended Approach Taking Account Of Flight Heights" of Band (2012). Note that collision risk calculations are based on equation (3) in "Stage C" section, but using (X, Y) coordinates to reference transit points instead of the equivalent (r, phi) coordinates.

Value

a matrix. The probability of collisions at different flight height bands

See Also

generate_rotor_grids

Examples

```
rotor_grids <- generate_rotor_grids(yinc = 0.05, xinc = 0.05, chord_prof_5MW)

get_pcoll_grid(
  rotor_grids = rotor_grids,
  direction = 1,
  rotor_radius = 120,
  blade_width = 5,
  rotor_speed = 15,
  blade_pitch = 15,
  flight_type = "flapping",
  n_blades = 3,
  flight_speed = 13.1,
  wing_span = 1.01,
  body_lt = 0.85)
```

---

get_phi_grid *Grid with angles between points (x, y) and the rotor's vertical axis*

Description

Given grids of vertical and horizontal distances from rotor axes at points (x, y) inside the rotor circle, `get_phi_grid` calculates the associated radial angles, relative to the rotor vertical axis.

Returned grid represents the left-half of the rotor's circle.

Usage

```
get_phi_grid(x_grid, y_grid)
```

Arguments

x_grid	A 2D array, with horizontal distances x from the rotor's vertical axis, expressed as the proportion of rotor radius, for the left-half of the rotor circle area.
y_grid	A 2D array, with vertical distances (y from the rotor's horizontal axis, expressed as the proportion of rotor radius, for the left-half of the rotor circle area.

Value

A 2D array, giving a grid of angles (in radians) between points (x, y) inside the rotor circle and the rotor center, for the left-half of the rotor circle area.

See Also

get_x_grid(), get_y_grid()

Examples

```
x_grid <- get_x_grid(yinc=0.05, xinc=0.05)
y_grid <- get_y_grid(x_grid,yinc=0.05)
get_phi_grid(x_grid,y_grid)
```

---

get_prop_crh_fhd *Calculate the total proportion of bird flights at collision risk based on a flight height distribution*

Description

Calculate the expected proportion of bird flights at collision risk height (i.e. at rotor height, between bottom and top of the rotor) based on the bird's flight height distribution (Q'_{2R}).

Usage

```
get_prop_crh_fhd(d_y)
```

Arguments

d_y	Numeric vector with the proportion of birds at height bands across the rotor disc

Value

The total proportion of birds at collision risk height derived from a flight height distribution

Examples

```
gen_fhd_dat <- Johnston_Flight_heights_SOSS %>%
     dplyr::filter(variable=="Gannet.est") %>%
     dplyr::select(height,prop)

gen_fhd <- gen_fhd_dat$prop

d_y <-
   get_fhd_rotor(
     hub_height = 150,
     fhd = gen_fhd,
     rotor_radius = 120,
     tidal_offset = 2.5,
     yinc = 0.05)

prop_chr_fhd <- get_prop_crh_fhd(d_y)
```

---

get_risk_y *Single transit collision risk along the chord of the rotor at height band y*

Description

Calculates the single transit collision risk/probability along the horizontal chord of the rotor at height y via numerical integration.

Usage

```
get_risk_y(x_at_y, pcoll_doty)
```

Arguments

x_at_y	a vector, sequence of horizontal distances from rotor's vertical axis to points x along half of the rotor circle, expressed as the proportion of rotor radius, at height band y
pcoll_doty	a vector, the single transit collision risk at horizontal distances x, at height band y

Value

a numeric value, the single transit collision risk along the whole horizontal chord of the rotor circle at height band y

See Also

get_x_grid(), get_pcoll_grid()

Examples

```
rotor_grids <- generate_rotor_grids(yinc = 0.05, xinc = 0.05, chord_prof_5MW)

y_lt <- dim(rotor_grids$r_grid)[1]

pcollxy_grid_up <- get_pcoll_grid(
  rotor_grids = rotor_grids,
  direction = 1,
  rotor_radius = 120,
  blade_width = 5,
  rotor_speed = 15,
  blade_pitch = 15,
  flight_type = "flapping",
  n_blades = 3,
  flight_speed = 13.1,
  wing_span = 1.01,
  body_lt = 0.85)

  risk_up <- rep(NA, y_lt)
  for(i in 1:y_lt){
    risk_up[i] <- get_risk_y(rotor_grids$x_grid[i, ], pcollxy_grid_up[i, ])
  }
```

---

get_x_grid *Grid of horizontal distances from points in the rotor circle to its vertical axis*

Description

Taking the center of the rotor circle as the origin, `get_x_grid` generates a grid containing horizontal distances x (by `xinc` increments) from the y-axis up to the outer edge of the rotor circle, at equidistant height bands y (by `yinc` increments) between minimum and maximum rotor height.

Distances are expressed as proportion of rotor radius (i.e. x is dimensionless).

Returned grid represents the left-half of the rotor's circle.

Usage

```
get_x_grid(xinc = 0.05, yinc = 0.05)
```

Arguments

`xinc, yinc`	numeric values, the grid increments along the y-axis and x-axis (i.e. grid cell dimensions)

Value

A 2D array giving a grid of horizontal distances x from rotor's vertical axis, expressed as the proportion of rotor radius (i.e. $[0, 1]$), for the left-half of the rotor circle area.

Examples

```
get_x_grid(xinc=0.05,yinc=0.05)
```

`get_y_grid` *Grid of vertical distances from points in the rotor circle to its horizontal axis*

Description

Taking the center of the rotor circle as the origin, `get_y_grid` generates a grid of vertical distances y (by `yinc` increments) from the x-axis to the outer edge of the rotor circle, across width intervals x (by `xinc` increments) between the center and maximum rotor width.

Returned grid represents the left-half of the rotor's circle.

Distances are expressed as proportion of rotor radius (i.e. y is dimensionless).

Usage

```
get_y_grid(x_grid, yinc = 0.05)
```

Arguments

`x_grid`	A 2D array, with horizontal distances from the rotor's vertical axis, expressed as the proportion of rotor radius, for the left-half of the rotor circle area.
`yinc`	a numeric value, the grid increment along the y-axis

Value

2D array giving a grid of vertical distances y from the rotor's horizontal axis, expressed as the proportion of rotor radius, for the left-half of the rotor circle area. Negative values represent distances from the bottom half of the rotor circle.

Examples

```
x_grid <- get_x_grid(yinc=0.05, xinc=0.05)
get_y_grid(x_grid,yinc=0.05)
```

Johnston_Flight_heights_SOSS *Summarized flight height profiles from Johnston et al (2014)*

Description

A dataframe containing flight height profiles for all species in Johnston et al. (2014). Values are expressed as the proportion of birds in 1 metre height intervals.

Usage

```
Johnston_Flight_heights_SOSS
```

Format

A data frame object with 3 columns containing the maximum likelihood, the median, and Upper/Lower confidence limits of flight height distributions. Flight height bands go from 1 - 300m ASL.

height Height above sea level, in metres. First element represents the 0-1 meters height band, and height interval is 1 metre.

variable The species name and variable from Johnston et al 2014 estimates. E.G., ArcticSkua.est is the maximum likelihood estimate from those models. .est = maximum likelihood, .lcl and .ucl are the lower and upper 95% CLs, and .med is the median estimate.

prop The proportion of birds within the 1m flight height bands.

Source

https://www.bto.org/our-science/wetland-and-marine/soss/projects

mig_stoch_crm *Stochastic migration collision risk model*

Description

Run migration stochastic collision risk model for a single species and one turbine scenario

Usage

```
mig_stoch_crm(
  wing_span_pars,
  flt_speed_pars,
  body_lt_pars,
  prop_crh_pars,
  avoid_bsc_pars,
```

```
  n_turbines,
  n_blades = 3,
  rtn_speed_pars,
  bld_pitch_pars,
  rtr_radius_pars,
  bld_width_pars,
  wf_width,
  wf_latitude,
  flight_type,
  prop_upwind = 0.5,
  popn_estim_pars,
  season_specs,
  chord_profile = chord_prof_5MW,
  trb_wind_avbl,
  trb_downtime_pars,
  n_iter = 10,
  LargeArrayCorrection = TRUE,
  log_file = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`wing_span_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species wingspan, in metres. Assumed to follow a *tnorm-lw0* distribution.
`flt_speed_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species flying speed, in metres/sec. Assumed to follow a Truncated Normal with lower bound at 0 (*tnorm-lw0*).
`body_lt_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species body length, in metres. Assumed to follow a *tnorm-lw0* distribution.
`prop_crh_pars`	A single row data frame with columns `mean` and `sd`. The mean and standard deviation of the proportion of flights at collision risk height.
`avoid_bsc_pars`	Single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species avoidance rate to be used. Avoidance rate expresses the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision, and it is assumed to follow a Beta distribution.
`n_turbines`	An integer. The number of turbines expected to be installed
`n_blades`	An integer. The number of blades per turbine (defaults to 3)
`rtn_speed_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the rotation speed.
`bld_pitch_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the blade pitch angle, i.e. the angle between the blade surface and the rotor plane, in degrees. Assumed to follow a *tnorm-lw0* distribution.

`rtr_radius_pars`
: A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the radius of the rotor, in metres. Assumed to follow a *tnorm-lw0* distribution.

`bld_width_pars`
: A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the maximum blade width, in metres. Assumed to be *tnorm-lw0* distribution.

`wf_width`
: Numeric value, the approximate longitudinal width of the wind farm, in kilometres (w).

`wf_latitude`
: A decimal value. The latitude of the centroid of the windfarm, in degrees.

`flight_type`
: A character. Either "flying" or "gliding" representing the type of flight most commonly used by the species

`prop_upwind`
: Numeric value between 0-1 giving the proportion of flights upwind - defaults to 0.5.

`popn_estim_pars`
: A single row data frame with columns `mean` and `sd`. The population estimate of the species expected to fly through the wind farm area.

`season_specs`
: A data frame defining the seasons for aggregating over collision estimates. It must comprise the following columns:
  - `season_id`, (unique) season identifier,
  - `start_month`, name of the season's first month,
  - `end_month`, name of the season's last month.

`chord_profile`
: A data frame with the chord taper profile of the rotor blade. It must contain the columns:
  - `pp_radius`, equidistant intervals of radius at bird passage point, as a proportion of `rotor_radius`, within the range $[0, 1]$.
  - `chord`, the chord width at `pp_radius`, as a proportion of `blade_width`. Defaults to a generic profile for a typical modern 5MW turbine. See `chord_prof_5MW()` for details.

`trb_wind_avbl`
: A data frame with the monthly estimates of operational wind availability. It must contain the columns:
  - `month`, (unique) month names,
  - `pctg`, the percentage of time wind conditions allow for turbine operation per month.

`trb_downtime_pars`
: A data frame with monthly estimates of maintenance downtime, assumed to follow a *tnorm-lw0* distribution. It must contain the following columns:
  - `month`, (unique) month names,
  - `mean`, numeric, the mean percentage of time in each month when turbines are not operating due to maintenance,
  - `sd`, the standard deviation of monthly maintenance downtime.

`n_iter`
: An integer > 0. The number of iterations for the model simulation.

`LargeArrayCorrection`
: A boolean. Should the large array correction be calculated

log_file	Path to log file to store session info and main model run options. If set to NULL (default value), log file is not created.
seed	Integer, the random seed for random number generation, for analysis reproducibility.
verbose	boolean. TRUE for a verbose output

Details

This function is an adaption of code from Masden(2015) used for estimating the collision risk of seabirds in offshore windfarm sites and is a further adaptation from Band(2012). It is a further adaptation of the stoch_crm function.

The collision risk model evaluates risk for each defined migratory period where flux rate is simply the number of birds travelling through the windfarm.

Changes in relation to previous top-line function `stoch_crm`

- function will run only option 1 for migratory species

Value

Estimates of number of collisions per migratory season for the n number of iterations specified

Examples

```
# ------------------------------------------------------
# Run with arbitrary parameter values, for illustration
# ------------------------------------------------------
season_specs <- data.frame(
  season_id = c("PrBMigration", "PoBMigration","OMigration"),
  start_month = c("Mar", "May", "Oct"), end_month = c("Apr", "Sep", "Feb")
)

# wind availability
windavb <- data.frame(
  month = month.abb,
  pctg = runif(12, 85, 98)
)
head(windavb)

# maintenance downtime
dwntm <- data.frame(
  month = month.abb,
  mean = runif(12, 6, 10),
  sd = rep(2, 12))
head(dwntm)


mig_stoch_crm(
  # Wing span in m,
  wing_span_pars = data.frame(mean = 1.08, sd = 0.04),
  # Flight speed in m/s
  flt_speed_pars = data.frame(mean = 7.26, sd = 1.5),
```

```
# Body length in m,
body_lt_pars = data.frame(mean = 0.39, sd = 0.005),
# Proportion of birds at CRH
prop_crh_pars = data.frame(mean = 0.06, sd = 0.009),
# avoidance rate
avoid_bsc_pars = data.frame(mean = 0.99, sd = 0.001),
n_turbines = 150,
n_blades = 3,
# rotation speed in m/s of turbine blades
rtn_speed_pars = data.frame(mean = 13.1, sd = 4),
# pitch in degrees of turbine blades
bld_pitch_pars = data.frame(mean = 3, sd = 0.3),
# sd = 0, rotor radius is fixed
rtr_radius_pars = data.frame(mean = 80, sd = 0),
# sd = 0, blade width is fixed
bld_width_pars = data.frame(mean = 8, sd = 0),
wf_width = 100,
wf_latitude = 54.1,
prop_upwind = 0.5,
flight_type = "flapping",
# population flying through windfarm,
popn_estim_pars = data.frame(mean = 21584, sd = 2023),
season_specs = season_specs,
chord_profile = chord_prof_5MW,
trb_wind_avbl = windavb,
trb_downtime_pars = dwntm,
n_iter = 1000,
LargeArrayCorrection = TRUE,
log_file = NULL,
seed = 1234,
verbose = TRUE)
```

`rbeta_dmp` *Customised sampling functions for the Beta distributions*

Description

generate random samples from a beta distribution, parameterized as mean and sd, and returning NAs if conditions are not met

Usage

```
rbeta_dmp(n, p, sd)
```

Arguments

`n`	An integer value. The number of samples to generate
`p`	A decimal value. The value used to calculate parameters for the beta distribution
`sd`	A decimal value. The standard deviation of the beta distribution to simulate

Value

a vector of samples values from the beta distribution

Examples

```
rbeta_dmp(n=100,p=0.9,sd=0.01)
```

`rotor_grids_test` *Sample rotor grids for generated_rotor_grids unit test*

Description

Sample rotor grids for generated_rotor_grids unit test

Usage

```
rotor_grids_test
```

Format

list of data frames

`rtnorm_dmp` *Customised sampling of Truncated Normal distribution*

Description

Wrapper of the msm::rtnorm() function, improving on outputs management and user feedback on edge cases

Usage

```
rtnorm_dmp(n, mean = 0, sd = 1, lower = -Inf, upper = Inf)
```

Arguments

`n`	An integer value. The number of samples to generate
`mean`	A decimal value. The mean for the truncated normal distribution
`sd`	A decimal value. The standard deviation of the distribution to simulate
`lower`	A decimal value. The lower limit for the distribution
`upper`	A decimal value. The upper limit for the distribution

Value

a vector of samples values from the truncated normal distribution

Examples

```
rtnorm_dmp(n=10,mean=0.4,sd=0.2)
```

`sampler_hd` *Customised sampling function wrapper*

Description

Samples a dataset based on inputs for either the rtnorm, rbeta or 'rnorm' distributions

Usage

```
sampler_hd(
  dat,
  mode = "rtnorm",
  n = NULL,
  mean = NULL,
  sd = NULL,
  lower = 0,
  upper = NULL
)
```

Arguments

`dat`	= A decimal value. The SD value to test (from the UI) - if not available in the UI, then do not create a distribution
`mode`	= A string. Either 'rtnorm', 'rbeta' or 'rnorm' to determine which distribution to generate
`n`	An integer value. The number of samples to generate
`mean`	A decimal value. The mean for the truncated normal distribution
`sd`	A decimal value. The standard deviation of the distribution to simulate
`lower`	A decimal value. The lower limit for the distribution
`upper`	A decimal value. The upper limit for the distribution

Value

a vector of samples values from the distribution

Examples

```
sampler_hd(dat=0.1,
     mode='rtnorm',
     n=100,
     mean=9,
     sd=0.1)
```

`sample_parameters` *Parameter sampling whiz*

Description

Generates the random samples of all the stochastic CRM parameters. For internal use.

Usage

```
sample_parameters(
  model_options,
  mod_mths,
  n_iter = 10,
  flt_speed_pars,
  body_lt_pars,
  wing_span_pars,
  avoid_bsc_pars,
  avoid_ext_pars,
  noct_act_pars,
  prop_crh_pars,
  bird_dens_opt = "tnorm",
  bird_dens_dt,
  gen_fhd_boots = NULL,
  site_fhd_boots = NULL,
  rtr_radius_pars,
  air_gap_pars,
  bld_width_pars,
  rtn_pitch_opt = "probDist",
  bld_pitch_pars,
  rtn_speed_pars,
  windspd_pars,
  rtn_pitch_windspd_dt,
  trb_wind_avbl,
  trb_downtime_pars,
  lrg_arr_corr
)
```

Arguments

`model_options`	Character vector, the model options for calculating collision risk (see **Details** section below).
`mod_mths`	character vector, the names of months under modelling
`n_iter`	An integer. The number of iterations for the model simulation.
`flt_speed_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species flying speed, in metres/sec. Assumed to follow a Truncated Normal with lower bound at 0 (*tnorm-lw0*).

`body_lt_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species body length, in metres. Assumed to follow a *tnorm-lw0* distribution.

`wing_span_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species wingspan, in metres. Assumed to follow a *tnorm-lw0* distribution.

`avoid_bsc_pars, avoid_ext_pars`
Single row data frames with columns `mean` and `sd`, the mean and standard deviation of the species avoidance rate to be used in the basic model (Options 1 and 2) and extended model (Options 3 and 4) calculations (see **Details** section). Avoidance rate expresses the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision, and it is assumed to follow a Beta distribution.

`noct_act_pars` A single row data frame with columns `mean` and `sd`, The mean and standard deviation of the species nocturnal flight activity level, expressed as a proportion of daytime activity levels, and assumed to be Beta distributed.

`prop_crh_pars` Required only for model Option 1, a single row data frame with columns `mean` and `sd`. The mean and standard deviation of the proportion of flights at collision risk height derived from site survey, assumed to be Beta distributed.

`bird_dens_opt` Option for specifying the random sampling mechanism for bird densities:

- `"tnorm"`: Sampling of density estimates from a *tnorm-lw0* distribution (default value),
- `"resample"`: Re-sample draws of bird density estimates (e.g. bootstrap samples),
- `"qtiles"`: Sampling from a set of quantile estimates of bird densities.

`bird_dens_dt` A data frame with monthly estimates of bird density within the windfarm footprint, expressed as the number of daytime in-flight birds/km^2 per month. Data frame format requirements:

- If `bird_dens_opt = "tnorm"`, `bird_dens_dt` must contain the following columns:
  - `month`, (unique) month names,
  - `mean`, the mean number of birds in flight at any height per square kilometre in each month,
  - `sd`, idem, for standard deviation.
- If `bird_dens_opt = "resample"`, `bird_dens_dt` columns must be named as months (i.e. `Jan`, `Feb`, ...), each containing random samples of monthly density estimates.
- If `bird_dens_opt = "qtiles"`, `bird_dens_dt` must comply with:
  - First column named as `p`, giving reference probabilities,
  - Remaining columns named as months (i.e. `Jan`, `Feb`, ...), each giving the quantile estimates of bird density in a given month, for the reference probabilities in column `p`.

`gen_fhd_boots` Required only for model Options 2 and 3, a data frame with bootstrap samples of flight height distributions (FHD) of the species derived from general (country/regional level) data. FHD provides relative frequency distribution of bird

flights at 1-+ -metre height bands, starting from sea surface. The first column must be named as `height`, expressing the lower bound of the height band (thus it's first element must be 0). Each remaining column should provide a bootstrap sample of the proportion of bird flights at each height band, with no column naming requirements.

NOTE: generic_fhd_bootstraps is a list object with generic FHD bootstrap estimates for 25 seabird species from Johnson et al (2014) doi:10.1111/1365-2664.12191 (see usage in Example Section below).

`site_fhd_boots` Required only for model Option 4, a data frame similar to `gen_fhd_boots`, but for FHD estimates derived from site-specific data.

`rtr_radius_pars`

A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the radius of the rotor, in metres. Assumed to follow a *tnorm-lw0* distribution.

`air_gap_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the tip clearance gap, in metres, i.e. the distance between the minimum rotor tip height and the highest astronomical tide (HAT). Assumed to follow a *tnorm-lw0* distribution.

`bld_width_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the maximum blade width, in metres. Assumed to be *tnorm-lw0* distribution.

`rtn_pitch_opt` a character string, the option for specifying the sampling mechanism for rotation speed and blade pitch:

- `"probDist"`: sample rotation speed and blade pitch values from a *tnorm-lw0* distribution (default value).
- `"windSpeedReltn"`: generate rotation speed and blade pitch values as a function of wind speed intensity.

`bld_pitch_pars` Only required if `rtn_pitch_opt = "probDist"`, a single row data frame with columns `mean` and `sd`, the mean and standard deviation of the blade pitch angle, i.e. the angle between the blade surface and the rotor plane, in degrees. Assumed to follow a *tnorm-lw0* distribution.

`rtn_speed_pars` Only required if `rtn_pitch_opt = "probDist"`, a single row data frame with columns `mean` and `sd`, the mean and standard deviation of the operational rotation speed, in revolutions per minute. Assumed to follow a *tnorm-lw0* distribution.

`windspd_pars` Only required if `rtn_pitch_opt = "windSpeedReltn"`, a single row data frame with columns `mean` and `sd`, the mean and the standard deviation of wind speed at the windfarm site, in metres/sec. Assumed to follow a *tnorm-lw0* distribution.

`rtn_pitch_windspd_dt`

Only required if `rtn_pitch_opt = "windSpeedReltn"`, a data frame giving the relationship between wind speed, rotation speed and blade pitch values. It must contain the columns:

- `wind_speed`, wind speed in m/s,
- `rtn_speed`, rotation speed in rpm,
- `bld_pitch`, blade pitch values in degrees.

`trb_wind_avbl` A data frame with the monthly estimates of operational wind availability. It must contain the columns:

- `month`, (unique) month names,
- `pctg`, the percentage of time wind conditions allow for turbine operation per month.

`trb_downtime_pars`
A data frame with monthly estimates of maintenance downtime, assumed to follow a *tnorm-lw0* distribution. It must contain the following columns:

- `month`, (unique) month names,
- `mean`, numeric, the mean percentage of time in each month when turbines are not operating due to maintenance,
- `sd`, the standard deviation of monthly maintenance downtime.

`lrg_arr_corr` Boolean value. If TRUE, the large array correction will be applied. This is a correction factor to account for the decay in bird density at later rows in wind farms with a large array of turbines.

Details

Collision risk can be calculated under 4 options, specified by `model_options`:

- **Option 1** - Basic model with proportion at collision risk height derived from site survey (`prop_crh_surv`).
- **Option 2** - Basic model with proportion at collision risk height derived from a generic flight height distribution (`gen_fhd`).
- **Option 3** - Extended model using a generic flight height distribution (`gen_fhd`).
- **Option 4** - Extended model using a site-specific flight height distribution (`site_fhd`).

Where,

- Basic model - assumes a uniform distribution of bird flights at collision risk height (i.e. above the minimum and below the maximum height of the rotor blade).
- Extended model - takes into account the distribution of bird flight heights at collision risk height.

Value

A list object with each element comprising sampled values of given CRM parameter

Examples

```
bird_dens_dt <- data.frame(
  month = month.abb,
  mean = runif(12, 0.8, 1.5),
  sd = runif(12, 0.2, 0.3)
)

# wind availability
```

```
trb_wind_avbl <- data.frame(
  month = month.abb,
  pctg = runif(12, 85, 98)
)

# maintenance downtime
trb_downtime_pars <- data.frame(
  month = month.abb,
  mean = runif(12, 6, 10),
  sd = rep(2, 12))

# Wind speed relationships
wind_rtn_ptch <- data.frame(
  wind_speed = seq_len(30),
  rtn_speed = 10/(30:1),
  bld_pitch = c(rep(90, 4), rep(0, 8), 5:22)
  )


bird_dens_opt <- "tnorm"
### extract and standardize month format from monthly data sets
b_dens_mth <- switch (bird_dens_opt,
                      tnorm = bird_dens_dt$month,
                      resample = names(bird_dens_dt),
                      qtiles = names(bird_dens_dt)[names(bird_dens_dt) != "p"]
) %>% format_months()
dwntm_mth <- format_months(trb_downtime_pars$month)
windav_mth <- format_months(trb_wind_avbl$month)
### Set months to model: only those in common amongst monthly data sets
mod_mths <- Reduce(intersect, list(b_dens_mth, dwntm_mth, windav_mth))
### Order chronologically
mod_mths <- mod_mths[order(match(mod_mths, month.abb))]

param_draws <- sample_parameters(
  model_options = c(1,2,3),
  n_iter = 10,
  mod_mths = mod_mths,
  flt_speed_pars = data.frame(mean=7.26,sd=1.5),
  body_lt_pars = data.frame(mean=0.39,sd=0.005),
  wing_span_pars = data.frame(mean=1.08,sd=0.04),
  avoid_bsc_pars = data.frame(mean=0.99,sd=0.001),
  avoid_ext_pars = data.frame(mean=0.96,sd=0.002),
  noct_act_pars = data.frame(mean=0.033,sd=0.005),
  prop_crh_pars = data.frame(mean=0.06,sd=0.009),
  bird_dens_opt = "tnorm",
  bird_dens_dt = bird_dens_dt,
  gen_fhd_boots = generic_fhd_bootstraps[[1]],
  site_fhd_boots = NULL,
  rtr_radius_pars = data.frame(mean=80,sd=0),
  air_gap_pars = data.frame(mean=36,sd=0),
  bld_width_pars = data.frame(mean=8,sd=0),
  rtn_pitch_opt = "windSpeedReltn",
  bld_pitch_pars = NULL,
```

```
rtn_speed_pars = NULL,
windspd_pars = data.frame(mean=7.74,sd=3),
rtn_pitch_windspd_dt = wind_rtn_ptch,
trb_wind_avbl = trb_wind_avbl,
trb_downtime_pars = trb_downtime_pars,
lrg_arr_corr = TRUE
)
```

`sample_qtls` *Generate random draws based on empirical c.d.f.*

Description

Generate random draws from a set of quantiles, based on the empirical cumulative density function

Usage

```
sample_qtls(n, probs, qtls)
```

Arguments

`n`	integer, the number of draws to generate
`probs`	numeric vector, the probabilities
`qtls`	numeric vector, the quantiles for the probabilities specified in `probs`

Details

Based on the Inverse Transform Sampling technique, by sampling random probabilities from a uniform distribution and interpolate (cubic) the count samples from the percentiles provided by the user (taken as the empirical cumulative density function)

Value

a numeric vector, with random draws of the approximated distribution underpinning the provided quantiles

Examples

```
sample_qtls(10,c(0.1,0.2,0.3),qtls=c(0.05,0.1,0.95))
```

`sample_turbine_mCRM` *Sampling function for a single turbine in the mCRM*

Description

Samples and aggregates appropriate data for a single wind turbine

Usage

```
sample_turbine_mCRM(
  rtn_speed_pars,
  bld_pitch_pars,
  rtr_radius_pars,
  bld_width_pars,
  season_specs,
  n_iter = 10,
  trb_wind_avbl,
  trb_downtime_pars
)
```

Arguments

`rtn_speed_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the rotation speed.
`bld_pitch_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the blade pitch angle, i.e. the angle between the blade surface and the rotor plane, in degrees. Assumed to follow a *tnorm-lw0* distribution.
`rtr_radius_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the radius of the rotor, in metres. Assumed to follow a *tnorm-lw0* distribution.
`bld_width_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the maximum blade width, in metres. Assumed to be *tnorm-lw0* distribution.
`season_specs`	A data frame defining the seasons for aggregating over collision estimates. It must comprise the following columns: • `season_id`, (unique) season identifier, • `start_month`, name of the season's first month, • `end_month`, name of the season's last month.
`n_iter`	An integer value. The number of samples to generate
`trb_wind_avbl`	A data frame with the monthly estimates of operational wind availability. It must contain the columns: • `month`, (unique) month names,

- pctg, the percentage of time wind conditions allow for turbine operation per month.

trb_downtime_pars

A data frame with monthly estimates of maintenance downtime, assumed to follow a *tnorm-lw0* distribution. It must contain the following columns:

- month, (unique) month names,
- mean, numeric, the mean percentage of time in each month when turbines are not operating due to maintenance,
- sd, the standard deviation of monthly maintenance downtime.

Value

A data frame of all the information sampled for the turbine with nrow = n_iter

Examples

```
season_specs <- data.frame(
  season_id = c("PrBMigration", "PoBMigration","OMigration"),
  start_month = c("Mar", "May", "Oct"), end_month = c("Apr", "Sep", "Feb")
  )

  windavb <- data.frame(
    month = month.abb,
    pctg = runif(12, 85, 98)
  )

  dwntm <- data.frame(
    month = month.abb,
    mean = runif(12, 6, 10),
    sd = rep(2, 12))

  sample_turbine_mCRM(rtn_speed_pars = data.frame(mean = 13.1, sd = 4),
                      bld_pitch_pars = data.frame(mean = 3, sd = 0.3),
                      rtr_radius_pars = data.frame(mean = 80, sd = 0),
                      bld_width_pars = data.frame(mean = 8, sd = 0),
                      season_specs = season_specs,
                      n_iter = 10,
                      trb_wind_avbl = windavb,
                      trb_downtime_pars = dwntm)
```

---

seq_months *Generate sequence of months*

Description

Generate sequence of months

Usage

```
seq_months(start_month, end_month)
```

Arguments

start_month	character string, the three-letter name of the starting month.
end_month	character string, the three-letter name of the finishing month.

Value

character vector. The list of months that falls in between two months

Examples

```
seq_months("Jan", "Apr")
```

---

stochLAB *stochLAB: Stochastic Collision Risk Model*

Description

stochLAB is a tool to run stochastic (and deterministic) Collision Risk Models (CRM) for seabirds on offshore wind farms.

The main functions of stochLAB are:

- `stoch_crm()`: Stochastic Collision Risk Model,
- `band_crm()`: Deterministic Collision Risk Model,
- `mig_stoch_crm()`: Stochastic Migration Collision risk Model

Overview

The {stochLAB} package is an adaptation of the R code developed by Masden (2015) to incorporate variability and uncertainty in the avian collision risk model originally developed by Band (2012).

Code developed under {stochLAB} substantially re-factored and re-structured Masden's (heavily script-based) implementation into a user-friendly, streamlined, well documented and easily distributed tool. Furthermore, this package lays down the code infrastructure for easier incorporation of new functionality, e.g. extra parameter sampling features, model expansions, etc.

Previous code underpinning key calculations for the extended model has been replaced by an alternative approach, resulting in significant gains in computational speed over Masden's code. This optimization is particularly beneficial under a stochastic context, when Band calculations are computed repeatedly. See `generate_rotor_grids()` for further details.

Function `stoch_crm()`

`stoch_crm()` is essentially a replacement function for script *BandModel.r* in Masden's approach. Main changes in terms of user interface include:

- Collision model runs for one single scenario (i.e. one species for one turbine scenario) at each `stoch_crm()` call. Apart from gains in development code structure, this unit-based approach was considered to offer greater end user flexibility for setting up and managing multiple scenarios (including parallel computation).
- Input parameters now entered explicitly into feature-specific arguments, instead of bundled together into wide-column tables. This improves code readability and reduces the quantity of hard-coded parameter names, therefore making referencing errors less likely.
- Outputs no longer saved automatically to external files.

Seasonal Outputs:

`stoch_crm()` now provides an option for user-defined seasonal outputs, allowing for country/region specific seasonal definitions.

Currently implemented CRM calculations produce collision estimates on a monthly basis to reflect changing bird abundance within the windfarm area. Seasonal estimates are obtained by aggregating monthly estimates over each season definition, with uncertainty in collision estimates being suitably propagated to seasonal outputs.

Sampling distributions:

Masden's implementation used Normal and Truncated Normal distributions to generate random parameter values. However, the Normal distribution is unbounded, and so there was the risk of randomly drawing inappropriate values in some cases.

`stoch_crm()` extends the use of bounded probability distributions to all model parameters. Strictly positive features (e.g. bird densities, body length, turbine downtime, etc.) are sampled from Truncated Normal distributions, while features constrained between 0 and 1 (e.g. nocturnal activity, proportion of flights at collision risk height, etc) are assumed to follow Beta distributions.

In addition, new functionality has been incorporated to allow the use of bird density resamples (e.g. bootstrap samples) or quantile estimates from density estimation models in the simulations.

Function `band_crm()`

`band_crm()` performs the core CRM calculations required to estimate the number of collisions, as per Band (2012). While being the computing workhorse of the `stoch_crm()` function, it can also be used alone, providing backward compatibility with the original Band spreadsheet outputs.

See Also

Useful links:

- https://github.com/HiDef-Aerial-Surveying/stochLAB
- Report bugs at https://github.com/HiDef-Aerial-Surveying/stochLAB/issues

Examples

```
# ------------------------------------------------------
# Run with arbitrary parameter values, for illustration
# ------------------------------------------------------

# Setting a dataframe of parameters to draw from
params <- data.frame(
  flight_speed = 13.1,         # Flight speed in m/s
  body_lt = 0.85,              # Body length in m
  wing_span = 1.01,            # Wing span in m
  flight_type = "flapping",    # flapping or gliding flight
  avoid_rt_basic = 0.989,      # avoidance rate for option 1 and 2
  avoid_rt_ext = 0.981,        # extended avoidance rate for option 3 and 4
  noct_activity = 0.5,         # proportion of day birds are inactive
 prop_crh_surv = 0.13,      # proportion of birds at collision risk height (option 1 only)
  prop_upwind = 0.5,           # proportion of flights that are upwind
  rotor_speed = 15,            # rotor speed in m/s
  rotor_radius = 120,          # radius of turbine in m
  blade_width = 5,             # width of turbine blades at thickest point in m
  blade_pitch = 15,            # mean radius pitch in Radians
  n_blades = 3,                # total number of blades per turbine
  hub_height = 150,            # height of hub in m above HAT
  n_turbines = 100,            # number of turbines in the wind farm
  wf_width = 52,               # width across longest section of wind farm
  wf_latitude = 56,            # latitude of centroid of wind farm
  tidal_offset = 2.5,          # mean tidal offset from HAT of the wind farm
  lrg_arr_corr = TRUE          # apply a large array correction?
)

# Monthly bird densities
b_dens <- data.frame(
  month = month.abb,
  dens = runif(12, 0.8, 1.5)
)

# flight height distribution from Johnston et al
gen_fhd_dat <- Johnston_Flight_heights_SOSS %>%
  dplyr::filter(variable=="Gannet.est") %>%
  dplyr::select(height,prop)


# monthly operational time of the wind farm
turb_oper <- data.frame(
  month = month.abb,
  prop_oper = runif(12,0.5,0.8)
)


band_crm(
  model_options = c(1,2,3),
  flight_speed = params$flight_speed,
  body_lt = params$body_lt,
```

```
  wing_span = params$wing_span,
  flight_type = params$flight_type,
  avoid_rt_basic = params$avoid_rt_basic,
  avoid_rt_ext = params$avoid_rt_ext,
  noct_activity = params$noct_activity,
  prop_crh_surv = params$prop_crh_surv,
  dens_month = b_dens,
  prop_upwind = params$prop_upwind,
  gen_fhd = gen_fhd_dat,
  site_fhd = NULL,  # Option 4 only
  rotor_speed = params$rotor_speed,
  rotor_radius = params$rotor_radius,
  blade_width = params$blade_width,
  blade_pitch = params$blade_pitch,
  n_blades = params$n_blades,
  hub_height = params$hub_height,
  chord_prof = chord_prof_5MW,
  n_turbines = params$n_turbines,
  turb_oper_month = turb_oper,
  wf_width = params$wf_width,
  wf_latitude = params$wf_latitude,
  tidal_offset = params$tidal_offset,
  lrg_arr_corr = params$lrg_arr_corr
  )
```

`stoch_crm` *Stochastic collision risk model for a single species and one wind farm scenario*

Description

Runs a Stochastic Collision Risk Model (SCRM) for estimating the number of in-flight collisions with offshore windfarm turbines, for given species and windfarm scenario. Core calculations follow the work developed by Masden (2015). See **Background and Updates** section below for more details.

Usage

```
stoch_crm(
  model_options = c("1", "2", "3", "4"),
  n_iter = 1000,
  flt_speed_pars,
  body_lt_pars,
  wing_span_pars,
  avoid_bsc_pars = NULL,
  avoid_ext_pars = NULL,
  noct_act_pars,
  prop_crh_pars = NULL,
  bird_dens_opt = c("tnorm", "resample", "qtiles"),
```

```
  bird_dens_dt,
  flight_type,
  prop_upwind,
  gen_fhd_boots = NULL,
  site_fhd_boots = NULL,
  n_blades,
  air_gap_pars,
  rtr_radius_pars,
  bld_width_pars,
  bld_chord_prf = chord_prof_5MW,
  rtn_pitch_opt = c("probDist", "windSpeedReltn"),
  bld_pitch_pars = NULL,
  rtn_speed_pars = NULL,
  windspd_pars = NULL,
  rtn_pitch_windspd_dt = NULL,
  trb_wind_avbl,
  trb_downtime_pars,
  wf_n_trbs,
  wf_width,
  wf_latitude,
  tidal_offset,
  lrg_arr_corr = TRUE,
  xinc = 0.05,
  yinc = 0.05,
  out_format = c("draws", "summaries"),
  out_sampled_pars = FALSE,
  out_period = c("months", "seasons", "annum"),
  season_specs = NULL,
  verbose = TRUE,
  log_file = NULL,
  seed = NULL
)
```

Arguments

`model_options`	Character vector, the model options for calculating collision risk (see **Details** section below).
`n_iter`	An integer. The number of iterations for the model simulation.
`flt_speed_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species flying speed, in metres/sec. Assumed to follow a Truncated Normal with lower bound at 0 (*tnorm-lw0*).
`body_lt_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species body length, in metres. Assumed to follow a *tnorm-lw0* distribution.
`wing_span_pars`	A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the species wingspan, in metres. Assumed to follow a *tnorm-lw0* distribution.

`avoid_bsc_pars, avoid_ext_pars`
: Single row data frames with columns `mean` and `sd`, the mean and standard deviation of the species avoidance rate to be used in the basic model (Options 1 and 2) and extended model (Options 3 and 4) calculations (see **Details** section). Avoidance rate expresses the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision, and it is assumed to follow a Beta distribution.

`noct_act_pars`
: A single row data frame with columns `mean` and `sd`, The mean and standard deviation of the species nocturnal flight activity level, expressed as a proportion of daytime activity levels, and assumed to be Beta distributed.

`prop_crh_pars`
: Required only for model Option 1, a single row data frame with columns `mean` and `sd`. The mean and standard deviation of the proportion of flights at collision risk height derived from site survey, assumed to be Beta distributed.

`bird_dens_opt`
: Option for specifying the random sampling mechanism for bird densities:
  - `"tnorm"`: Sampling of density estimates from a *tnorm-lw0* distribution (default value),
  - `"resample"`: Re-sample draws of bird density estimates (e.g. bootstrap samples),
  - `"qtiles"`: Sampling from a set of quantile estimates of bird densities.

`bird_dens_dt`
: A data frame with monthly estimates of bird density within the windfarm footprint, expressed as the number of daytime in-flight birds/km^2 per month. Data frame format requirements:
  - If `bird_dens_opt = "tnorm"`, `bird_dens_dt` must contain the following columns:
    - `month`, (unique) month names,
    - `mean`, the mean number of birds in flight at any height per square kilometre in each month,
    - `sd`, idem, for standard deviation.
  - If `bird_dens_opt = "resample"`, `bird_dens_dt` columns must be named as months (i.e. `Jan`, `Feb`, ...), each containing random samples of monthly density estimates.
  - If `bird_dens_opt = "qtiles"`, `bird_dens_dt` must comply with:
    - First column named as `p`, giving reference probabilities,
    - Remaining columns named as months (i.e. `Jan`, `Feb`, ...), each giving the quantile estimates of bird density in a given month, for the reference probabilities in column `p`.

`flight_type`
: A character string, either 'flapping' or 'gliding', indicating the species' characteristic flight type.

`prop_upwind`
: Numeric value between 0-1 giving the proportion of flights upwind - defaults to 0.5.

`gen_fhd_boots`
: Required only for model Options 2 and 3, a data frame with bootstrap samples of flight height distributions (FHD) of the species derived from general (country/regional level) data. FHD provides relative frequency distribution of bird flights at 1-+ -metre height bands, starting from sea surface. The first column must be named as `height`, expressing the lower bound of the height band (thus

it's first element must be 0). Each remaining column should provide a bootstrap sample of the proportion of bird flights at each height band, with no column naming requirements.

NOTE: generic_fhd_bootstraps is a list object with generic FHD bootstrap estimates for 25 seabird species from Johnson et al (2014) doi:10.1111/1365-2664.12191 (see usage in Example Section below).

`site_fhd_boots` Required only for model Option 4, a data frame similar to `gen_fhd_boots`, but for FHD estimates derived from site-specific data.

`n_blades` An integer, the number of blades in rotor (b).

`air_gap_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the tip clearance gap, in metres, i.e. the distance between the minimum rotor tip height and the highest astronomical tide (HAT). Assumed to follow a *tnorm-lw0* distribution.

`rtr_radius_pars`

A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the radius of the rotor, in metres. Assumed to follow a *tnorm-lw0* distribution.

`bld_width_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the maximum blade width, in metres. Assumed to be *tnorm-lw0* distribution.

`bld_chord_prf` A data frame with the chord taper profile of the rotor blade. It must contain the columns:

- `pp_radius`, equidistant intervals of radius at bird passage point, as a proportion of `rotor_radius`, within the range $[0, 1]$.
- `chord`, the chord width at `pp_radius`, as a proportion of `blade_width`.

Defaults to a generic profile for a typical modern 5MW turbine. See `chord_prof_5MW()` for details.

`rtn_pitch_opt` a character string, the option for specifying the sampling mechanism for rotation speed and blade pitch:

- `"probDist"`: sample rotation speed and blade pitch values from a *tnorm-lw0* distribution (default value).
- `"windSpeedReltn"`: generate rotation speed and blade pitch values as a function of wind speed intensity.

`bld_pitch_pars` Only required if `rtn_pitch_opt = "probDist"`, a single row data frame with columns `mean` and `sd`, the mean and standard deviation of the blade pitch angle, i.e. the angle between the blade surface and the rotor plane, in degrees. Assumed to follow a *tnorm-lw0* distribution.

`rtn_speed_pars` Only required if `rtn_pitch_opt = "probDist"`, a single row data frame with columns `mean` and `sd`, the mean and standard deviation of the operational rotation speed, in revolutions per minute. Assumed to follow a *tnorm-lw0* distribution.

`windspd_pars` Only required if `rtn_pitch_opt = "windSpeedReltn"`, a single row data frame with columns `mean` and `sd`, the mean and the standard deviation of wind speed at the windfarm site, in metres/sec. Assumed to follow a *tnorm-lw0* distribution.

`rtn_pitch_windspd_dt`
: Only required if `rtn_pitch_opt = "windSpeedReltn"`, a data frame giving the relationship between wind speed, rotation speed and blade pitch values. It must contain the columns:
  - `wind_speed`, wind speed in m/s,
  - `rtn_speed`, rotation speed in rpm,
  - `bld_pitch`, blade pitch values in degrees.

`trb_wind_avbl`
: A data frame with the monthly estimates of operational wind availability. It must contain the columns:
  - `month`, (unique) month names,
  - `pctg`, the percentage of time wind conditions allow for turbine operation per month.

`trb_downtime_pars`
: A data frame with monthly estimates of maintenance downtime, assumed to follow a *tnorm-lw0* distribution. It must contain the following columns:
  - `month`, (unique) month names,
  - `mean`, numeric, the mean percentage of time in each month when turbines are not operating due to maintenance,
  - `sd`, the standard deviation of monthly maintenance downtime.

`wf_n_trbs`
: Integer, the number of turbines on the windfarm.

`wf_width`
: Numeric value, the approximate longitudinal width of the wind farm, in kilometres (w).

`wf_latitude`
: A decimal value. The latitude of the centroid of the windfarm, in degrees.

`tidal_offset`
: A numeric value, the tidal offset, the difference between HAT and mean sea level, in metres.

`lrg_arr_corr`
: Boolean value. If TRUE, the large array correction will be applied. This is a correction factor to account for the decay in bird density at later rows in wind farms with a large array of turbines.

`yinc, xinc`
: numeric values, the increments along the y-axis and x-axis for numerical integration across segments of the rotor circle. Chosen values express proportion of rotor radius. By default these are set to 0.05, i.e. integration will be performed at a resolution of one twentieth of the rotor radius.

`out_format`
: Output format specification. Possible values are:
  - `"draws"`: returns stochastic draws of collisions estimates (default value),
  - `"summaries"`: returns summary statistics of collisions estimates.

`out_sampled_pars`
: Logical, whether to output summary statistics of values sampled for each stochastic model parameter.

`out_period`
: Controls level of temporal aggregation of collision outputs. Possible values are:
  - `"months"`: monthly collisions (default value),
  - `"seasons"`: collisions per user-defined season,
  - `"annum"`: total collisions over 12 months.

season_specs	Only required if out_period = "seasons", a data frame defining the seasons for aggregating over collision estimates. It must comprise the following columns: • season_id, (unique) season identifier, • start_month, name of the season's first month, • end_month, name of the season's last month.
verbose	Logical, print model run progress on the console?
log_file	Path to log file to store session info and main model run options. If set to NULL (default value), log file is not created.
seed	Integer, the random seed for random number generation, for analysis reproducibility.

Details

Collision risk can be calculated under 4 options, specified by model_options:

- **Option 1** - Basic model with proportion at collision risk height derived from site survey (prop_crh_surv).
- **Option 2** - Basic model with proportion at collision risk height derived from a generic flight height distribution (gen_fhd).
- **Option 3** - Extended model using a generic flight height distribution (gen_fhd).
- **Option 4** - Extended model using a site-specific flight height distribution (site_fhd).

Where,

- Basic model - assumes a uniform distribution of bird flights at collision risk height (i.e. above the minimum and below the maximum height of the rotor blade).
- Extended model - takes into account the distribution of bird flight heights at collision risk height.

Value

If out_sampled_pars = FALSE, returns a list with estimates of number of collisions per chosen time periods, with elements containing the outputs for each CRM Option.

If out_sampled_pars = TRUE, returns a list object with two top-level elements:

- collisions, a list comprising collision estimates for each CRM Option,
- sampled_pars, a list with summary statistics of values sampled for stochastic model parameters.

Examples

```
# ------------------------------------------------------
# Run with arbitrary parameter values, for illustration
# ------------------------------------------------------

# ------------------------------------------------------
# Setting some of the required inputs upfront
b_dens <- data.frame(
```

```
  month = month.abb,
  mean = runif(12, 0.8, 1.5),
  sd = runif(12, 0.2, 0.3)
)
head(b_dens)

# Generic FHD bootstraps from Johnson et al (2014)
fhd_boots <- generic_fhd_bootstraps[[1]]
head(fhd_boots)

# wind speed vs rotation speed vs blade pitch
wind_rtn_ptch <- data.frame(
  wind_speed = seq_len(30),
  rtn_speed = 10/(30:1),
  bld_pitch = c(rep(90, 4), rep(0, 8), 5:22)
)
head(wind_rtn_ptch)

# wind availability
windavb <- data.frame(
  month = month.abb,
  pctg = runif(12, 85, 98)
)
head(windavb)

# maintenance downtime
dwntm <- data.frame(
  month = month.abb,
  mean = runif(12, 6, 10),
  sd = rep(2, 12))
head(dwntm)

# seasons specification
seas_dt <- data.frame(
  season_id = c("a", "b", "c"),
  start_month = c("Jan", "May", "Oct"), end_month = c("Apr", "Sep", "Dec")
  )
head(seas_dt)

# ------------------------------------------------------------
# Run stochastic CRM, treating rotor radius, air gap and
# blade width as fixed parameters (i.e. not stochastic)

stoch_crm(
  model_options = c(1, 2, 3),
  n_iter = 1000,
  flt_speed_pars = data.frame(mean = 7.26, sd = 1.5),
  body_lt_pars = data.frame(mean = 0.39, sd = 0.005),
  wing_span_pars = data.frame(mean = 1.08, sd = 0.04),
  avoid_bsc_pars = data.frame(mean = 0.99, sd = 0.001),
  avoid_ext_pars = data.frame(mean = 0.96, sd = 0.002),
  noct_act_pars = data.frame(mean = 0.033, sd = 0.005),
  prop_crh_pars = data.frame(mean = 0.06, sd = 0.009),
```

```
  bird_dens_opt = "tnorm",
  bird_dens_dt = b_dens,
  flight_type = "flapping",
  prop_upwind = 0.5,
  gen_fhd_boots = fhd_boots,
  n_blades = 3,
  rtr_radius_pars = data.frame(mean = 80, sd = 0), # sd = 0, rotor radius is fixed
  air_gap_pars = data.frame(mean = 36, sd = 0),    # sd = 0, air gap is fixed
  bld_width_pars = data.frame(mean = 8, sd = 0),   # sd = 0, blade width is fixed
  rtn_pitch_opt = "windSpeedReltn",
  windspd_pars = data.frame(mean = 7.74, sd = 3),
  rtn_pitch_windspd_dt = wind_rtn_ptch,
  trb_wind_avbl = windavb,
  trb_downtime_pars = dwntm,
  wf_n_trbs = 200,
  wf_width = 15,
  wf_latitude = 56.9,
  tidal_offset = 2.5,
  lrg_arr_corr = TRUE,
  verbose = TRUE,
  seed = 1234,
  out_format = "summaries",
  out_sampled_pars = TRUE,
  out_period = "seasons",
  season_specs = seas_dt,
  log_file = NULL
)
```

`turb_pars_wide_example` *Example of turbine and windfarm parameters stored in wide format*

Description

A data frame of (fake) data on turbine and wind farm features for 3 scenarios.

Usage

```
turb_pars_wide_example
```

Format

A 3 x 51 data frame, with the turbine and windfarm parameters (columns) for each of the 3 development scenarios (rows). Columns include:

TurbineModel The turbine/windfarm scenario ID

Blades Nr of blades

RotorRadius Mean of rotor radius

RotorRadiusSD SD of rotor radius

HubHeightAdd Mean of air gap, the distance between sea surface and lowest tip height.

HubHeightAddSD SD of air gap, as explained above. ...

Details

Intended to illustrate the application of `stoch_scrm()` to a multiple scenario setting, where parameter data is available from tables in wide format.

`validate_inputs` *Input validator*

Description

Input validator

Usage

```
validate_inputs(
  model_options,
  n_iter = NULL,
  flt_speed_pars = NULL,
  flight_speed = NULL,
  body_lt_pars = NULL,
  body_lt = NULL,
  wing_span_pars = NULL,
  wing_span = NULL,
  avoid_bsc_pars = NULL,
  avoid_rt_basic = NULL,
  avoid_ext_pars = NULL,
  avoid_rt_ext = NULL,
  noct_act_pars = NULL,
  noct_activity = NULL,
  prop_crh_pars = NULL,
  bird_dens_opt = NULL,
  bird_dens_dt = NULL,
  chord_prof = NULL,
  dens_month = NULL,
  turb_oper_month = NULL,
  flight_type = NULL,
  prop_upwind = NULL,
  gen_fhd_boots = NULL,
  site_fhd_boots = NULL,
  n_blades = NULL,
  air_gap_pars = NULL,
  rtr_radius_pars = NULL,
  rotor_radius = NULL,
  blade_width = NULL,
```

```
  blade_pitch = NULL,
  hub_height = NULL,
  bld_width_pars = NULL,
  rtn_pitch_opt = NULL,
  bld_pitch_pars = NULL,
  rtn_speed_pars = NULL,
  rotor_speed = NULL,
  n_turbines = NULL,
  windspd_pars = NULL,
  rtn_pitch_windspd_dt = NULL,
  trb_wind_avbl = NULL,
  trb_downtime_pars = NULL,
  wf_n_trbs = NULL,
  wf_width = NULL,
  wf_latitude = NULL,
  tidal_offset = NULL,
  gen_fhd = NULL,
  site_fhd = NULL,
  lrg_arr_corr = NULL,
  xinc = NULL,
  yinc = NULL,
  seed = NULL,
  verbose = NULL,
  out_format = NULL,
  out_sampled_pars = NULL,
  out_period = NULL,
  season_specs = NULL,
  popn_estim_pars = NULL,
  fn = "scrm"
)
```

Arguments

model_options	Character vector, the model options for calculating collision risk (see **Details** section below).
n_iter	An integer. The number of iterations for the model simulation.
flt_speed_pars	A single row data frame with columns mean and sd, the mean and standard deviation of the species flying speed, in metres/sec. Assumed to follow a Truncated Normal with lower bound at 0 (*tnorm-lw0*).
flight_speed	Numeric value. The bird flying speed (v), in metres/sec.
body_lt_pars	A single row data frame with columns mean and sd, the mean and standard deviation of the species body length, in metres. Assumed to follow a *tnorm-lw0* distribution.
body_lt	Numeric value. The length of the bird (L), in metres.
wing_span_pars	A single row data frame with columns mean and sd, the mean and standard deviation of the species wingspan, in metres. Assumed to follow a *tnorm-lw0* distribution.

`wing_span` Numeric value. The wingspan of the bird (W), in metres.

`avoid_bsc_pars, avoid_ext_pars`
Single row data frames with columns `mean` and `sd`, the mean and standard deviation of the species avoidance rate to be used in the basic model (Options 1 and 2) and extended model (Options 3 and 4) calculations (see **Details** section). Avoidance rate expresses the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision, and it is assumed to follow a Beta distribution.

`avoid_rt_basic, avoid_rt_ext`
Numeric values. The avoidance rate for, respectively, the basic model (i.e. required for model Options 1 and 2) and the extended model (i.e. required for Options 3 and 4). Avoidance rate expresses the probability that a bird flying on a collision course with a turbine will take evading action to avoid collision.

`noct_act_pars` A single row data frame with columns `mean` and `sd`, The mean and standard deviation of the species nocturnal flight activity level, expressed as a proportion of daytime activity levels, and assumed to be Beta distributed.

`noct_activity` A numeric value. The nocturnal flight activity level, expressed as a proportion of daytime activity levels (f_night).

`prop_crh_pars` Required only for model Option 1, a single row data frame with columns `mean` and `sd`. The mean and standard deviation of the proportion of flights at collision risk height derived from site survey, assumed to be Beta distributed.

`bird_dens_opt` Option for specifying the random sampling mechanism for bird densities:

- `"tnorm"`: Sampling of density estimates from a *tnorm-lw0* distribution (default value),
- `"resample"`: Re-sample draws of bird density estimates (e.g. bootstrap samples),
- `"qtiles"`: Sampling from a set of quantile estimates of bird densities.

`bird_dens_dt` A data frame with monthly estimates of bird density within the windfarm footprint, expressed as the number of daytime in-flight birds/km^2 per month. Data frame format requirements:

- If `bird_dens_opt = "tnorm"`, `bird_dens_dt` must contain the following columns:
  - `month`, (unique) month names,
  - `mean`, the mean number of birds in flight at any height per square kilometre in each month,
  - `sd`, idem, for standard deviation.
- If `bird_dens_opt = "resample"`, `bird_dens_dt` columns must be named as months (i.e. `Jan`, `Feb`, ...), each containing random samples of monthly density estimates.
- If `bird_dens_opt = "qtiles"`, `bird_dens_dt` must comply with:
  - First column named as `p`, giving reference probabilities,
  - Remaining columns named as months (i.e. `Jan`, `Feb`, ...), each giving the quantile estimates of bird density in a given month, for the reference probabilities in column `p`.

`chord_prof` A data frame with the chord taper profile of the rotor blade. Function expects two named columns:

- `pp_radius`, equidistant intervals of radius at bird passage point, as a proportion of `rotor_radius`, within the range $[0, 1]$.
- `chord`, the chord width at `pp_radius`, as a proportion of `blade_width`.

Defaults to a generic profile for a typical modern 5MW turbine. See `chord_prof_5MW()` for details.

`dens_month` Data frame, containing estimates of daytime in-flight bird densities per month within the windfarm footprint, in birds/km^2. It must contain the following named columns:

- `month`, the month names.
- `dens`, the number of birds in flight at any height per square kilometre in each month.

`turb_oper_month` Data frame, holding the proportion of time during which turbines are operational per month. The following named column are expected:

- `month`, the month names.
- `prop_oper`, the proportion of time operating, per month.

`flight_type` A character string, either 'flapping' or 'gliding', indicating the species' characteristic flight type.

`prop_upwind` Numeric value between 0-1 giving the proportion of flights upwind - defaults to 0.5.

`gen_fhd_boots` Required only for model Options 2 and 3, a data frame with bootstrap samples of flight height distributions (FHD) of the species derived from general (country/regional level) data. FHD provides relative frequency distribution of bird flights at 1-+ -metre height bands, starting from sea surface. The first column must be named as `height`, expressing the lower bound of the height band (thus it's first element must be 0). Each remaining column should provide a bootstrap sample of the proportion of bird flights at each height band, with no column naming requirements.

NOTE: generic_fhd_bootstraps is a list object with generic FHD bootstrap estimates for 25 seabird species from Johnson et al (2014) doi:10.1111/1365-2664.12191 (see usage in Example Section below).

`site_fhd_boots` Required only for model Option 4, a data frame similar to `gen_fhd_boots`, but for FHD estimates derived from site-specific data.

`n_blades` An integer, the number of blades in rotor (b).

`air_gap_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the tip clearance gap, in metres, i.e. the distance between the minimum rotor tip height and the highest astronomical tide (HAT). Assumed to follow a *tnorm-lw0* distribution.

`rtr_radius_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the radius of the rotor, in metres. Assumed to follow a *tnorm-lw0* distribution.

`rotor_radius` Numeric value. The radius of the rotor (R), in metres.

`blade_width` Numeric value, giving the maximum blade width, in metres.

`blade_pitch` Numeric value. The average blade pitch angle, the angle between the blade surface and the rotor plane (γ), in radians.

`hub_height` A numeric value, the height of the rotor hub (H), given by the sum of rotor radius and minimum blade clearance above the highest astronomical tide (HAT), in metres.

`bld_width_pars` A single row data frame with columns `mean` and `sd`, the mean and standard deviation of the maximum blade width, in metres. Assumed to be *tnorm-lw0* distribution.

`rtn_pitch_opt` a character string, the option for specifying the sampling mechanism for rotation speed and blade pitch:

- `"probDist"`: sample rotation speed and blade pitch values from a *tnorm-lw0* distribution (default value).
- `"windSpeedReltn"`: generate rotation speed and blade pitch values as a function of wind speed intensity.

`bld_pitch_pars` Only required if `rtn_pitch_opt = "probDist"`, a single row data frame with columns `mean` and `sd`, the mean and standard deviation of the blade pitch angle, i.e. the angle between the blade surface and the rotor plane, in degrees. Assumed to follow a *tnorm-lw0* distribution.

`rtn_speed_pars` Only required if `rtn_pitch_opt = "probDist"`, a single row data frame with columns `mean` and `sd`, the mean and standard deviation of the operational rotation speed, in revolutions per minute. Assumed to follow a *tnorm-lw0* distribution.

`rotor_speed` Numeric value. The operational rotation speed, in revolutions/min.

`n_turbines` Integer, the number of turbines on the wind farm (T).

`windspd_pars` Only required if `rtn_pitch_opt = "windSpeedReltn"`, a single row data frame with columns `mean` and `sd`, the mean and the standard deviation of wind speed at the windfarm site, in metres/sec. Assumed to follow a *tnorm-lw0* distribution.

`rtn_pitch_windspd_dt`
Only required if `rtn_pitch_opt = "windSpeedReltn"`, a data frame giving the relationship between wind speed, rotation speed and blade pitch values. It must contain the columns:

- `wind_speed`, wind speed in m/s,
- `rtn_speed`, rotation speed in rpm,
- `bld_pitch`, blade pitch values in degrees.

`trb_wind_avbl` A data frame with the monthly estimates of operational wind availability. It must contain the columns:

- `month`, (unique) month names,
- `pctg`, the percentage of time wind conditions allow for turbine operation per month.

`trb_downtime_pars`
A data frame with monthly estimates of maintenance downtime, assumed to follow a *tnorm-lw0* distribution. It must contain the following columns:

- `month`, (unique) month names,
- `mean`, numeric, the mean percentage of time in each month when turbines are not operating due to maintenance,
- `sd`, the standard deviation of monthly maintenance downtime.

`wf_n_trbs` Integer, the number of turbines on the windfarm.

`wf_width` Numeric value, the approximate longitudinal width of the wind farm, in kilometres (w).

`wf_latitude` A decimal value. The latitude of the centroid of the windfarm, in degrees.

`tidal_offset` A numeric value, the tidal offset, the difference between HAT and mean sea level, in metres.

`gen_fhd, site_fhd`

Data frame objects, with flight height distributions (fhd) of the species - the relative frequency distribution of bird flights at 1-metre height intervals from sea surface. Specifically:

- `gen_fhd`, Data frame with the species' generic fhd derived from combining wider survey data. Only required for model Options 2 and 3
- `site_fhd`, Data frame with the species' site-specific fhd derived from local survey data. Only required for model Option 4

Data frames must contain the following named columns:

- `height`, integers representing height bands from sea surface, in metres. Function expects 0 as the first value, representing the 0-1m band.
- `prop`, the proportion of flights at each height band.

`lrg_arr_corr` Boolean value. If TRUE, the large array correction will be applied. This is a correction factor to account for the decay in bird density at later rows in wind farms with a large array of turbines.

`yinc, xinc` numeric values, the increments along the y-axis and x-axis for numerical integration across segments of the rotor circle. Chosen values express proportion of rotor radius. By default these are set to 0.05, i.e. integration will be performed at a resolution of one twentieth of the rotor radius.

`seed` Integer, the random seed for random number generation, for analysis reproducibility.

`verbose` Logical, print model run progress on the console?

`out_format` Output format specification. Possible values are:

- `"draws"`: returns stochastic draws of collisions estimates (default value),
- `"summaries"`: returns summary statistics of collisions estimates.

`out_sampled_pars`

Logical, whether to output summary statistics of values sampled for each stochastic model parameter.

`out_period` Controls level of temporal aggregation of collision outputs. Possible values are:

- `"months"`: monthly collisions (default value),
- `"seasons"`: collisions per user-defined season,
- `"annum"`: total collisions over 12 months.

season_specs Only required if out_period = "seasons", a data frame defining the seasons for aggregating over collision estimates. It must comprise the following columns:

- season_id, (unique) season identifier,
- start_month, name of the season's first month,
- end_month, name of the season's last month.

popn_estim_pars
A single row data frame with columns mean and sd. The population estimate of the species expected to fly through the wind farm area.

fn a character string specifying the parent function whose inputs are being checked:

- "scrm": checks stoch_crm() inputs
- "crm": checks band_crm() inputs
- "mcrm": checks mig_stoch_crm() inputs

Value

Nothing returned from this function

Examples

```
validate_inputs(model_options=c(1),
             avoid_bsc_pars=data.frame(mean=0.99,sd=0.001),
             prop_crh_pars=data.frame(mean=0.01,sd=0.01),
             air_gap_pars = data.frame(mean=21,sd=0),
             rtr_radius_pars = data.frame(mean=100,sd=0),
             bld_pitch_pars = data.frame(mean=15,sd=0),
             rtn_pitch_opt = "probDist",
             rtn_speed_pars = data.frame(mean=14,sd=5),
             out_period = "months",
             lrg_arr_corr = TRUE,
             fn="scrm")
```

---

wndspd_rtn_ptch_example

Example of data with relationship between wind speed, rotation speed and blade pitch

---

Description

Intended to illustrate the application of stoch_scrm()

Usage

```
wndspd_rtn_ptch_example
```

Format

A 30 x 3 data frame with columns:

wind_speed Wind speed in metres per second.

rtn_speed Blade rotation speed, in revolutions per minute

bld_pitch Blade pitch, in degrees

Index