Package 'PROSPER'

July 21, 2025

Type Package

Title Simulation of Weed Population Dynamics

Version 0.3.3

Date 2020-06-23

Maintainer Christoph v. Redwitz <christoph.redwitz@posteo.de>

Description An environment to simulate the development of annual plant populations with regard to population dynamics and genetics, especially herbicide resistance. It combines genetics on the individual level (Renton et al. 2011) with a stochastic development on the population level (Daedlow, 2015).
Renton, M, Diggle, A, Manalil, S and Powles, S (2011) <doi:10.1016/j.jtbi.2011.05.010>
Daedlow, Daniel (2015, doctoral dissertation: University of Rostock, Faculty of Agriculture and Environmental Sciences.)

License GPL-3

LazyData yes

Imports methods, data.table

RoxygenNote 7.1.0

NeedsCompilation no

Author Christoph v. Redwitz [aut, cre],
Friederike de Mol [aut]

Repository CRAN

Date/Publication 2020-06-27 14:30:02 UTC

Contents

`gen_check` *Checking the plausibility and conversion of dom and af* `link{struc_preparation}`

Description

Before calling `struc_preparation` it is necessary to bring the genetics into the correct form, which is mainly the correct dimensions.

Usage

```
gen_check(Rmx, af, dom, epis)
```

Arguments

`Rmx`	maximum resistance value, if all gene loci under consideration are homozygous resistant. `numeric`, ≥ 1.
`af`	initial frequency of resistance alleles in the population. `numeric vector` with length of `ng` (number of genes) and elements in [0,1].
`dom`	dominance of resistance alleles. `numeric vector` with length of `n_loci` (number of genes) and elements in [0,1].
`epis`	epistasis value, describing the interaction between resistance alleles. `numeric`. `epis` = 0: no interaction i.e. additive effects of resistance alleles, `epis` < 0: effect of resistance alleles is smaller than additive, `epis` > 0: effect of resistance alleles is higher than additive.

Details

If no genetics are required `af` has to be set to `NA`. The value of `dom` is adjusted to the length of `af`. If there are mismatches, the `dom` is cut or the first given value is repeated to fit the number of given alleles in `af`. The variables af and `dom` are corrected and `n_loci` is created, which is 0 when no genetics is included. If no value is given to `epis` it is set to 0.

gen_diploid *Diploid genetics: recombination of alleles*

Description

gen_diploid recombines the alleles of weeds with diploid genome.

Usage

```
gen_diploid(
  start,
  start_comb = NA,
  result,
  newseeds = get0("newseeds", envir = parent.frame(n = 1)),
  max_vec_length = 1e+07
)
```

Arguments

start	column names of parental cohorts. character vector.
start_comb	all named columns in start are joined and returned as a column with the name of start_comb (character). Only used when start longer than 1.
result	name of the results column. character.
newseeds	number of new seeds calculated by pop_reprod. positive integer.
max_vec_length	used internally, a technical term, defining the maximum length of vectors to be used.

Details

gen_diploid assumes independent allele recombination.

Value

A column with the name of result and, if necessary, a column with the name of start_comb is added to the data.frame dfgenotype.

warning

newseeds are always coerced to a whole number.

See Also

pop_reprod

Examples

```
## Not run:
# generate a 'dfgenotype' data.frame:
struc_preparation2(Rmx=10, af=c(0.01, 0.2), epis=0, dom=1)
#Distribute 10000 individuals of the starting population across the genotypes.
#The two gene loci have initial frequencies of 0.01 and 0.8.
gen_freq(af=c(0.01,0.8), n_seeds=10000)
# The column "initialSB" represents the parent generation, which recombines and
#therefore defines the genetics of the new seeds
newseeds <- 10000
gen_diploid(start="initialSB", result="followingSB")
# If a second cohort is reprodusing in the same time
gen_diploid(start=c("initialSB", "followingSB"),
                  start_comb="two_cohorts_combind", result="followingSB2")
rm(dfgenotype, mf, newseeds, xprobab)

## End(Not run)
```

`gen_freq` *generating the start values for PROSPER models*

Description

`gen_freq` generates the numbers of genotypes at the beginning of the first simulated year.

Usage

```
gen_freq(
  af,
  n_seeds,
  result = "initialSB",
  distribution = NA,
  max_vec_length = 1e+07
)
```

Arguments

`af`	initial frequency of resistance alleles in the population. `numeric vector` with length of `ng` (number of genes) and elements in [0,1].
`n_seeds`	initial number of weed seeds. They will be allocated to the genotypes. `integer`.
`result`	names of the results columns. `character`.
`distribution`	defines the proportion with `n_seeds` is distributed among multiple cloumns, if result defines multiple columns. Has to be of the same length as `result`. `numeric` between 0 and 1 or "equal". See details.
`max_vec_length`	used internally, a technical term, defining the maximum length of vectors to be used.

Details

The start values for a model include the initial frequencies of alleles and the initial seedbank, i.e. the number of seeds in the soil. `gen_freq` allocates the `n_seeds` individuals to the different genotypes as provided by `dfgenotype`, which is created by `struc_preparation2()`. If `af == NULL` the function gives all seed to the only "genotype". The distribution can be splitted to multiple cohorts using `distribution`. If `distribution` is "equal", n_seeds are distributed equaly amoung the cohorts defindes with `result`.

Value

Returns a `data.frame` containing the genotypes and their frequencies provided by `dfgenotype`.

See Also

`sel_resist` `struc_preparation2`

Examples

```
# generate a 'dfgenotype' data.frame:
struc_preparation2(Rmx=10, af=c(0.01,0.8), epis=0, dom=1)
#Distribute 10000 individuals of the starting population across the genotypes.
#The two gene loci have initial frequencies of 0.01 and 0.8.
gen_freq(af=c(0.01,0.8), n_seeds=10000)
rm(dfgenotype, mf, xprobab)
```

---

herb_check *Checking the main parameters* `link{sel_herb}`

Description

Is called within `sel_herb()` to check for the existence of the necessary data.

Usage

```
herb_check(put, sdrate, thresh, rate)
```

Arguments

`put`	probability of a weed to be untouched by the herbicide. `numeric`, $0 \leq$ `put` ≤ 1.
`sdrate`	variance of the herbicide rate reaching the weed. `positive numeric`, 1 = 1 unit standard deviation.
`thresh`	threshold herbicide rate to kill weeds without resistance. `numeric`, $0 \leq$ `thresh` ≤ 1.
`rate`	percentage (%) of the registered herbicide dose. `positive numeric`, can exceed 100 %.

`intern_herbicide` *Surviving the Herbicide*

Description

Utility function internally used. It's used only by `sel_herb`, and usually there is no reason to change it. Calculates the surviving number of weeds according to the specific genotype.

Usage

```
intern_herbicide(resist, n_samples, put, rate, sdrate, thresh)
```

Arguments

`resist`	numeric, value of resistance for the genotype (defined by Renton et al. 2011). Shall be ≤ 1.
`n_samples`	integer, the number of weeds with one specific genotype.
`put`	probability of a weed to be untouched by the herbicide. `numeric`, $0 \leq$ `put` ≤ 1.
`rate`	percentage (%) of the registered herbicide dose. `positive numeric`, can exceed 100 %.
`sdrate`	variance of the herbicide rate reaching the weed. `positive numeric`, 1 = 1 unit standard deviation.
`thresh`	threshold herbicide rate to kill weeds without resistance. `numeric`, $0 \leq$ `thresh` ≤ 1.

Details

`intern_herbicide` is used in `sel_herb`. Firstly, it calculates the number of weeds that are untouched by the herbicide by chance (probability=`put`). In the second step, the herbicide rate that reached an individual is compared with the resistance value (calculated after Renton et al. 2011, `sel_resist`). If the resistance value is lower than the dose, the weed dies. All surviving weeds are summed up.

Value

The number of weeds surviving the herbicide.

References

Renton, M.; Diggle, A.; Manalil, S. & Powles, S. (2011): Does cutting herbicide rates threaten the sustainability of weed management in cropping systems? Journal of Theoretical Biology, 283, 14-27.

See Also

`sel_herb` `sel_resist`

Examples

```
#How many of 1000 weeds of a genotype with resistance value 5.5 survive a herbicide application
#with full dose? 'It is assumed that weeds reseaving less than 20 \% of the full dose survive
#independently of their resistant value.
intern_herbicide(resist=5.5, n_samples=1000, put=0.04, rate=100, sdrate=0.4, thresh=20)
```

---

mod_check *Checking the main parameters* struc_preparation2

Description

Before the preparation (struc_preparation2) it is necessary to check, whether the necessary information is provided to principally conduct a simulation run. Furthermore the crop.list is brought to the correct form.

Usage

```
mod_check(
  param.weed = NA,
  area = NA,
  duration = NA,
  repetitions = NA,
  crop_list = NA,
  max_vec_length = NA
)
```

Arguments

param.weed	A data.frame with population dynamic parameters with or without stochasticity. The structure of param.weed is essential (see details). The easiest way to create the data.frame is to adopt an example (param.ECHCG).
area	number of area units. positive numeric.
duration	maximum number of simulation loops in the simulation. positive integer.
repetitions	number of repetitions of the simulation. positive integer
crop_list	crop rotation. character vector, elements must fit to the names in the data.frame weed.
max_vec_length	used internally, a technical term, defining the maximum length of vectors to be used.

Details

If no genetics are required af has to be set to NA. The value of dom is adjusted to the length of af. The dom is cut or the first given value is repeated to fit the number of given alleles in af. The variables af and dom are corrected and n_loci is created, which is 0 when no genetics is included.

Examples

```
mod_check(param.weed=param.GALAP, area=100, duration=2,
     repetitions=1, crop_list=c("corn"), max_vec_length=1000000)
```

---

plot *Simple plot for prosper simulation results*

Description

This function draws three figures. i)Numerics: mean number of individual plants of a specified development stage for each simulation cycle, summing up the genotypes to one number per cycle. ii)Raw counts: same as i) using the results of the repetitions instead of the mean. iii) Mean of the proportion of plants with only alleles for sensitivity, proportion of plants with only alleles for resistance, proportion of plants with mixed alleles for resistance and sensitivity, proportion of R alleles in the population.

Usage

```
## S4 method for signature 'prosper,missing'
plot(x, y, plot_var = "SB_autumn", ...)
```

Arguments

x	prosper, the result of a prosper simulation.
y	not used.
plot_var	variable, i.e. a column name of the result data.frame, to be plotted. character.
...	other graphical parameters.

See Also

summary

Examples

```
data(param.LOLRI)
mod_lolri    <- prosper.LOLRI(param.weed=param.LOLRI, area=20,  af = c(0.005, 0.01),
                                      duration=3, repetitions=2)
plot(mod_lolri)
```

pop_germc *Germination*

Description

pop_germc describes germination as random event for individual seeds. The function considers different cohorts and dormancy.

Usage

```
pop_germc(init_sb, germ, max_vec_length = 1e+07)
```

Arguments

init_sb	column name of the initial seed bank in the data.frame dfgenotype that is delivered by struc_preparation. character.
germ	germination probabilities for the different cohorts. See details.
max_vec_length	used internally, a technical term, defining the maximum length of vectors to be used.

Details

Each individual has a chance to germinate or to stay dormant. In case of germination, it emerges in one of the cohorts. The distribution of individual seeds to cohorts or dormancy is random. The function uses the columns init_SB of the data.frame dfgenotype as input. The output values are the numbers of seedlings of each genotype and each cohort.

germ must be given as a numeric vector or - in case of multiple columns in init_sb - data.frame or matrix. In that case the row number must fit to the length of init_sb. If the columns in init_sb represent cohorts, the rows of germ give the germination probabilities for these specific cohorts. The sum of one row of germ shall be ≥ 0 and ≤ 1. The difference of 1 and the sum of on row germ is the probability of dormancy.

Value

Columns are added to dfgenotype: "germ_dorm" contains the numbers of each genotype that remain dormant, "germ1" to "germX" contain the numbers for each of X cohorts.

See Also

pop_reprod pop_step quanti

Examples

```
struc_preparation2(Rmx=10, af=c(0.01,0.8), epis=0, dom=1)
ls()
gen_freq( af=c(0.01,0.8), n_seeds=10000)
#Distribute the individuals to three cohorts with the germination
```

```
#probabilities 0.2, 0.4 and 0.4.
pop_germc( init_sb="initialSB", germ=c(0.2,0.4,0.4))
rm(mf, dfgenotype, xprobab)
```

---

pop_reprod *Seed production and crop yield*

Description

Calculates the produced seeds and optionally the proportion of crop yield that is realized relative to weed free yield (Renton at al. 2011).

Usage

```
pop_reprod(start, area, kw, pen_co, kc, dc, crop_inr, SSmax, yield = FALSE)
```

Arguments

start	column names of parental cohorts. character vector.
area	number of area units. positive numeric.
kw	dimensionless weed competition parameter. numeric.
pen_co	penalty values for different weed cohorts. numeric vector, each element in [0,1].
kc	dimensionless crop competition parameter. numeric.
dc	crop sowing density, seeds per unit area. numeric.
crop_inr	position of the crop in the crop rotation. positive integer $\leq$ length of the crop rotation.
SSmax	maximum of weed seed production per unit area. positive integer.
yield	logical, whether the percentage of yield gained should be calculated.

Details

The number of produced seeds is calculated for 1m^2 by the formula:

$$over\_all = 1 + kc * dc[crop\_inr] + sum(kw * dw * pen\_co)$$

$$producedseeds = round(sum((SSmax[crop\_inr] * kw * dw * pen_co)/over\_all), digits = 0)$$

$$propyield = (1 + kc * dc[crop\_inr])/over\_all$$

The weed density dw is calculated for each squaremeter derived from the current simulation run (start). The used parameters values apply to wheat and ryegrass (Pannell et al. 2004, cited in Renton et al. 2011).

References

Renton, M.; Diggle, A.; Manalil, S. & Powles, S. (2011): Does cutting herbicide rates threaten the sustainability of weed management in cropping systems? Journal of Theoretical Biology, 283, 14-27. Pannell, D. J.; Stewart, V.; Bennett, A.; Monjardino, M.; Schmidt, C. & Powles, S. B. (2004): RIM: a bioeconomic model for integrated weed management of Lolium rigidum in Western Australia Agricultural Systems, 79, 305-325

See Also

pop_reprod pop_step

Examples

```
struc_preparation2(Rmx=10, af=c(0.01,0.8), epis=0, dom=1)
#Distribute 10000 individuals of the starting population across the genotypes provided by tmp.
#The two gene loci have initial frequencies of 0.01 and 0.8.
gen_freq(af=c(0.01,0.8), n_seeds=10000, max_vec_length=1e+07)
pop_reprod("initialSB",  area=100, kw=0.5, pen_co=1, kc=0.05, dc=100,
                         crop_inr="wheat", SSmax=3000, yield=TRUE)
rm(producedseeds, dfgenotype, mf, xprobab, propyield)
```

---

pop_step *Surviving a non-selective process*

Description

pop_step picks the individuals that will pass to the next development stage. This is a random process for every individual, which does not exert any selection pressure.

Usage

```
pop_step(
  start,
  start_comb = NA,
  result = NA,
  stepname = NA,
  surv_prob,
  max_vec_length = 1e+07
)
```

Arguments

start	column names of parental cohorts. character vector.
start_comb	all named columns in start are joined and returned as a column with the name of start_comb (character). Only used when start longer than 1.
result	name of the results column. character.
stepname	name of the new column of dfgenotype added by this function. character.

surv_prob	probability to survive this step and reach the next growth stage. numeric.
max_vec_length	used internally, a technical term, defining the maximum length of vectors to be used.

Details

Individuals that reach the next growth stage are picked by using rbinom. In contrast to sel_herb, pop_step does not exert any evolutionary selection pressure. When more than one column is selected with start, they are summed and the result is passed to the picking process. By setting start_comb the sum is added as a column to dfgenotype.

Value

A new column is added to dfgenotype containing the surviving individuals of the different genotypes.

See Also

quanti pop_germc

Examples

```
struc_preparation2(Rmx=10, af=c(0.01,0.8), epis=0, dom=1)
gen_freq(af=c(0.01,0.8), n_seeds=10000)
#How many individuals of each genotype will reach the next growth stage?
pop_step(start="initialSB",  stepname="survivingthewinter",
                            surv_prob=0.4)
```

---

prosper-models *Population dynamic models - Examples*

Description

PROSPER entails full parameterized models, which are described here. Population dynamic models adress different purposes the models differ. The functions prosper.* present different adaptations.

prosper.ECHCG provides the setting for a simulation of the population dynamic of *Echinochloa crus-galli*.

prosper.GALAP provides the setting for a simulation of the population dynamic of *Galium aparine*. No selection process is used.

prosper.LOLRI performs a simulation of PROSPER using the setting presented by Renton at al. (2011). To manipulate the parameters see Details.

Usage

```
prosper.ECHCG(
  param.weed = PROSPER::param.ECHCG,
  area = NA,
  af = NA,
  dom = NA,
  epis = 0,
  put = 0.05,
  sdrate = 0.4,
  thresh = 20,
  Rmx = 10,
  rate = 100,
  duration = NA,
  repetitions = NA,
  crop_list = "corn",
  max_vec_length = 1e+07,
  undersowing = NA
)

prosper.GALAP(
  param.weed = PROSPER::param.GALAP,
  sens_seeds = 400,
  area = 100,
  af = c(0.03, 0.08, 0.02),
  dom = c(0.5, 0.5, 0.5),
  epis = 0,
  put = 0.05,
  thresh = 20,
  Rmx = 10,
  rate = 100,
  sdrate = 0.4,
  duration = 15,
  repetitions = 1,
  crop_list = "wheat",
  max_vec_length = 1e+07
)

prosper.LOLRI(
  param.weed = PROSPER::param.LOLRI,
  area = 100,
  af = c(0.005, 0.01, 0.015, 0.02),
  dom = 0.5,
  epis = 0,
  put = 0.05,
  sdrate = 0.4,
  thresh = 20,
  Rmx = 10,
  dc = 150,
```

```
  kc = 1/11,
  kw = 1/33,
  SSmax = 30000,
  rate = 100,
  pen_co = c(1, 0.5),
  duration = 10,
  repetitions = 1,
  crop_list = c("wheat"),
  max_vec_length = 1e+07
)
```

Arguments

param.weed	A data.frame with population dynamic parameters with or without stochasticity. The structure of param.weed is essential (see details). The easiest way to create the data.frame is to adopt an example (param.ECHCG).
area	number of area units. positive numeric.
af	initial frequency of resistance alleles in the population. numeric vector with length of ng (number of genes) and elements in [0,1].
dom	dominance of resistance alleles. numeric vector with length of n_loci (number of genes) and elements in [0,1].
epis	epistasis value, describing the interaction between resistance alleles. numeric. epis = 0: no interaction i.e. additive effects of resistance alleles, epis < 0: effect of resistance alleles is smaller than additive, epis > 0: effect of resistance alleles is higher than additive.
put	probability of a weed to be untouched by the herbicide. numeric, $0 \leq$ put ≤ 1.
sdrate	variance of the herbicide rate reaching the weed. positive numeric, 1 = 1 unit standard deviation.
thresh	threshold herbicide rate to kill weeds without resistance. numeric, $0 \leq$ thresh ≤ 1.
Rmx	maximum resistance value, if all gene loci under consideration are homozygous resistant. numeric, ≥ 1.
rate	percentage (%) of the registered herbicide dose. positive numeric, can exceed 100 %.
duration	maximum number of simulation loops in the simulation. positive integer.
repetitions	number of repetitions of the simulation. positive integer
crop_list	crop rotation. character vector, elements must fit to the names in the data.frame weed.
max_vec_length	used internally, a technical term, defining the maximum length of vectors to be used.
undersowing	Numerical vector with two values between 0 and 1. See details.
sens_seeds	sensitive seeds added every year.
dc	crop sowing density, seeds per unit area. numeric.
kc	dimensionless crop competition parameter. numeric.

kw	dimensionless weed competition parameter. numeric.
SSmax	maximum of weed seed production per unit area. positive integer.
pen_co	penalty values for different weed cohorts. numeric vector, each element in [0,1].

Details

prosper.ECHCG() simulates originally the population dynamic of *Echinochloa crus-galli* using the data param.ECHCG. Different cohorts of weed seedlings are the focus of this model. The focus of this model is the effect of weeds that escape the selection pressure of herbicide treatment. These weeds keep the unselected genetic. Can they buffer the selection process? *E. crus-galli* is able to germinate over a long period after maize planting with decreasing reproductive success (Bagavathiannan, 2013). In the model all germinating individuals are represented by two cohorts; an early, major cohort with with high seed production, and a small late emerging with lower reproduction. Only the first cohort is controlled by a herbicide, which is a typical situation in Germany (Rossberg, 2016). The second cohort escapes the herbicide treatment unaffected. However, the second cohort can be suppressed, for example by an undersown crop. Three scenarios with different degrees of suppression, 0%, 30% and 100%, were simulated (Redwitz, 2016). The parameter undersowing describes the probability of surviving a second, not selective pressure on weed seedlings, which germinate after the selective herbicide was applied. prosper.ECHCG provides the setting for a simulation of the population dynamic of *Echinochloa crus-galli*.

prosper.GALAP() simulates originally the population dynamic of *Galium aparine* using the data param.GALAP. Whether sowing of susceptible weed seeds can restore an 'acceptable' resistance level of a population in the early stages of resistance development, is an extraordinary research question. The patchy occurrence of *Galium aparine* and its large seeds result in highly variable population dynamic parameters. Modeling has to take into account this variability. We used a simple population dynamics model structure (Redwitz et al., 2015). A seedbank in spring provides seeds out of which one cohort is germinating. The weeds are selected by herbicides, produce seeds, which are affected by seed predation and return to the seedbank in autumn. Data of a long term field experiment were used for parametrization (Daedlow, 2015).

prosper.LOLRI() performs a simulation of *Lolium rigidum* similar to PERTH (Renton et al. 2011) when it is used with param.LOLRI.

Functions

- prosper.ECHCG: Population dynamic model of *Echinochloa crus-galli*
- prosper.GALAP: Population dynamic model of *Galium aparine*
- prosper.LOLRI: Population dynamic model of *Lolium multiflorum*

Author(s)

Christoph von Redwitz, <christoph.redwitz@uni-rostock.de>

References

Redwitz, C von, Pannwitt H, Gerowitt B (2016): About the interplay of sensitive and resistant biotypes in weed populations - simulation exercises for Echinochloa crus-galli in maize crops. Pro-

ceedings - 28th German Conference on Weed Biology and Weed Control, Julius-Kuehn-Archiv, 93-99, 452.

Redwitz, C von, Daedlow D, Gerowitt B (2015): Simulation exercises on long-term management of widespread herbicide resistance in a field weed population. Proceedings 17th Symposium of the European Weed Research Society, Montpellier, France, 108.

Renton, M., Diggle, A., Manalil, S. & Powles, S. (2011): Does cutting herbicide rates threaten the sustainability of weed management in cropping systems? Journal of Theoretical Biology, Elsevier BV, 283, 14-27.

See Also

weed-parameters

Examples

```
## Not run:
mod_echcg <- prosper.ECHCG(param.weed = param.ECHCG, area=100, af=c(0.001),
                           undersowing=0.2,dom=0.5,duration=7,repetitions=1)


#The model call for Redwitz et al. (2015)
undersowing_prob <- c(1, 0.3, 0) #no undersowing, strong competition, complete dominance
years <- 20
reps  <- 4
####------------------------
simu_collect <- list()
for(simu in 1:3){
simu_collect[[simu]] <- prosper.ECHCG(area          = 100,
                                      param.weed    = param.ECHCG,
                                      thresh        = 20,
                                      duration      = years,
                                      af            = 0.001,
                                      dom           = 1,
                                      undersowing   = undersowing_prob[simu],
                                      repetitions   = reps
                                             )
}

## End(Not run)

mod_galap <- prosper.GALAP(param.weed=param.GALAP, repetitions=2, duration=10)

mod_lolri    <- prosper.LOLRI(param.weed=param.LOLRI, area=100,
                                       duration=15, repetitions=3)
```

PROSPER-pkg *PROSPER-pkg*

Description

PROSPER simulates the development of annual plant populations with regard to population dynamics and genetics, especially herbicide resistance. It combines genetics on the individual level with a stochastic development on the population level. The genetic part is modeled after Renton et al. (2011), the stochastic part follows Daedlow (2015). All parameters delivered with PROSPER are based on an area of 1 squaremeter.

Source

Renton, M.; Diggle, A.; Manalil, S. & Powles, S. (2011): Does cutting herbicide rates threaten the sustainability of weed management in cropping systems? Journal of Theoretical Biology, 283, 14-27.

Daedlow, D. (2015): About the contribution of seed predation on weed demography. (Doctoral dissertation). University of Rostock, Faculty of Agriculture and Environmental Sciences. GBV Gemeinsamer Verbundkatalog (Accession No. 839752644).

`quanti` *Step by step: the model develops driven by field data*

Description

This is a working horse of PROSPER. It quantifies the number or the proportion of individuals entering the next development stage using predefined formulas (`formul`). Typically these formulas are the results of experiments.

Usage

```
quanti(
  origin = NA,
  step_name,
  crop,
  proportion = TRUE,
  equal_dis = TRUE,
  res_max = NA,
  res_min = 0,
  formul = NA,
  area,
  addit_variables = NA,
  log_values = TRUE,
  back_log = TRUE
)
```

Arguments

`origin`	numbers or log-numbers of individuals at the start for every genotype. `numeric`. Alternatively the columnaes of dfgenotype can be used.

step_name	step name in the database containing the model parameters. character.
crop	current crop in the crop sequence, must fit to one of the crop names in the data frame with weed-parameters. character.
proportion	logical, TRUE when the output should be a proportion, otherwise it will be an absolute number.
equal_dis	logical, TRUE when the distribution of the weeds is spatially uniform across the area.
res_max	output maximum, ignored when origin=NA. numeric.
res_min	output minimum. numeric.
formul	character. See details.
area	number of area units. positive numeric.
addit_variables	variables used in formul that are not predefined in param.weed. See details. character vector
log_values	logical, TRUE when origin in log-scale.
back_log	logical, TRUE when the output is in log-scale. Only used when proportion==FALSE.

Details

Within PROSPER simulation models are build up with discrete simulation steps. These steps are conducted by functions like pop_step or gen_reprod. These functions affect the complete population and need a count of individuals or a proportion of the population that are affected of the specific simulation step. These numbers are calculated with quanti. The calculation is based on the data provided in weed-parameters. There a model is given for every parameter in the table. The parameter formul allows to use a different model if necessary. If no model is given at all, the simulation step is assumed to consist only of one value. The parameters of param.weed are normal distributed and the SE is used to draw them for the current calculation using rnorm(). These values are used to evaluate the model for simulation step. The resulting count or proportion can be used to perform the simulation step for the population.

Value

Either a (log-)number of individuals or a proportion resp. a rate.

See Also

pop_step pop_germc weed-parameters

Examples

```
#loads the example data for Echinochloa crus-galli
data(param.ECHCG)
param.weed <- param.ECHCG
#how many seeds (natural, not log-scale) produsedby 100 plants in a corn stand on 100 area units?
quanti(origin=100, step_name="seed_prod_first", crop="corn",  proportion=FALSE,
                 area=100, log_values=FALSE, back_log=FALSE)
rm(param.ECHCG)
```

sel_herb *Surviving the herbicide*

Description

sel_herb calculates the surviving number of each genotype. sel_herb selects for resistant individuals.

Usage

```
sel_herb(start, result, thresh, sdrate, rate, put, max_vec_length = 1e+07)
```

Arguments

start	column names of parental cohorts. character vector.
result	name of the results column. character.
thresh	threshold herbicide rate to kill weeds without resistance. numeric, $0 \leq$ thresh ≤ 1.
sdrate	variance of the herbicide rate reaching the weed. positive numeric, 1 = 1 unit standard deviation.
rate	percentage (%) of the registered herbicide dose. positive numeric, can exceed 100 %.
put	probability of a weed to be untouched by the herbicide. numeric, $0 \leq$ put ≤ 1.
max_vec_length	used internally, a technical term, defining the maximum length of vectors to be used.

Details

For every genotype intern_herbicide is called. If no genetics are included, the value from start is returned in result.

See Also

sel_resist

Examples

```
struc_preparation2(Rmx=10, af=c(0.02,0.01), epis=0, dom=1)
gen_freq( af=c(0.01,0.8), n_seeds=10000)
sel_herb(start="initialSB", result="winter",
                      thresh=20, sdrate=0.4, rate=100, put=0.04)
```

`sel_resist` *Calculating resistance values*

Description

Calculates the phenotypic resistance value (Renton et al. 2011) for each genotype in `dfgenotype`.

Usage

```
sel_resist(Rmx, epis, dom)
```

Arguments

`Rmx`	maximum resistance value, if all gene loci under consideration are homozygous resistant. `numeric`, ≥ 1.
`epis`	epistasis value, describing the interaction between resistance alleles. `numeric`. `epis` = 0: no interaction i.e. additive effects of resistance alleles, `epis` < 0: effect of resistance alleles is smaller than additive, `epis` > 0: effect of resistance alleles is higher than additive.
`dom`	dominance of resistance alleles. `numeric vector` with length of `n_loci` (number of genes) and elements in [0,1].

Details

This function is used in `struc_preparation2` to calculate the resistance value using the following term:

$$1 + (Rmx - 1) * (sum(dom)/n\_loci)^{2^{epis}}).$$

Resistance values range from 1 to Rmx. The higher the resistance value is, the less the plant is susceptible to the herbicide. `sel_resist` is only used by `struc_preparation2`.

Value

`numeric vector` with resistance values

References

Renton, M.; Diggle, A.; Manalil, S. & Powles, S. (2011): Does cutting herbicide rates threaten the sustainability of weed management in cropping systems? Journal of Theoretical Biology, 283, 14-27.

See Also

`sel_herb` `intern_herbicide`

Examples

```
#dfgenotype is usually generated by the function 'struc_preparation2'.
#Here, a simple example is done by hand.
var1 <- c("00","01","02","10","11","12","20","21","21")
var2 <- c(0,0,0,1,1,1,2,2,2)
var3 <- c(0,1,2,0,1,2,0,1,2)
dfgenotype <- data.frame(genotype=var1,l1=var2,l2=var3, stringsAsFactors = TRUE)
sel_resist(Rmx=10, epis=0, dom=1)
```

`struc_endSim` *Check for some stop conditions.*

Description

Conditions that are necessary to continue the simulation are checked.

Usage

```
struc_endSim(simcycle = year, break_col_names = "SB_autumn")
```

Arguments

`simcycle`	The count of simulation runs. The value is `duration` ≥ 0.
`break_col_names`	The names of the columns in `dfgenotype`, which are the first in the next simulation cycle.

Details

This function combines two conditions to terminate the simulation. The first is the `duration` that is defined for the simulation. The second is the the number of individuals for the next simulation cycle: when the population is extinct the simulation ends.

Value

`logical` TRUE if the simulation has to stop.

See Also

`struc_saveSimData`

`struc_preparation2` *Generating data input and output structures*

Description

`struc_preparation` creates the data input and output structures (data.frames and table) for the simulation run, 'dfgenotype', 'xprobab' and 'mf'.

Usage

```
struc_preparation2(Rmx = NA, af = NA, epis = NA, dom = NA)
```

Arguments

`Rmx`	maximum resistance value, if all gene loci under consideration are homozygous resistant. `numeric`, ≥ 1.
`af`	initial frequency of resistance alleles in the population. `numeric vector` with length of ng (number of genes) and elements in [0,1].
`epis`	epistasis value, describing the interaction between resistance alleles. `numeric`. `epis` = 0: no interaction i.e. additive effects of resistance alleles, `epis` < 0: effect of resistance alleles is smaller than additive, `epis` > 0: effect of resistance alleles is higher than additive.
`dom`	dominance of resistance alleles. `numeric vector` with length of `n_loci` (number of genes) and elements in [0,1].

Details

Prior to the simulation, a data.frame is generated to save results (`dfgenotype`). Additionally, a table with recombination probabilities (`xprobab`) is calculated. During the simulation run, probability values are not computed again but looked up in the table. PROSPER assumes diploid plants and maximum four resistance genes. To calculate the phenotypic resistance value for each genotype `sel_resist` is called. If `n_loci` < `0` the structure is set up for no genetics at all in the simulation.

Value

Returns a `list` of two `data.frame` and a `table`:

1. mf: all possible combinations of parental genotypes (see 'dfgenotype$genotype') are saved in one column 'mf' (male, female). The column 'mf' is a character vector. Each string of the vector has twice the length of the number of resistance loci under consideration.
2. dfgenotype: the structure to save the results of one simulation cycle (year). After each cycle the data is reset to the new start values. The first column 'genotype' is a character vector. Each string of the vector has the length of the number of resistance loci under consideration. Each locus can have 0, 1 or 2 resistance alleles. The second column 'resist' saves resistance values that are calculated according to the equation in the section 'details'.
3. xprobab: the probabilities of occurrence for all possible genotypes in the offspring (F-generation) with all possible parent genotypes (P-generation). Free recombination is assumed. Column names are the combinations of parental genes, row names are genotypes of the offspring.

Warning

The run of `struc_preparation` is time consuming. Duration strongly increases with the number of genes under consideration, `n_loci`.

See Also

`sel_resist` `gen_freq`

Examples

```
#generate the genotype and probability tables for a simulation with two resistance
#loci with one dominant and one partial dominant resistant allele, no epistasis, and a
#maximumx resistance value of 10.
ls()
struc_preparation2(Rmx=10, af=c(0.01,0.5), epis=0, dom=c(1,0.3))
ls()
rm(dfgenotype, mf, xprobab)
```

`struc_saveSimData` *Saving simulation data*

Description

`struc_saveSimData` collects the data of one simulation cycle and returns an object to start a new simulation cycle.

Usage

```
struc_saveSimData(rep_counter, simcycle, start_names, end_names, simstruc)
```

Arguments

`rep_counter`	number of the current repetition. `integer`.
`simcycle`	number of the current simulation cycle. `integer`.
`start_names`	names of the columns with the first population stage in the simulation. `character`.
`end_names`	names of the columns with the last population stage in the simulation. These are taken as the first stage in the next simulation cycle. `character`.
`simstruc`	two numbers, the first defines the number of repetitions for the simulation, the second defines the number of simulation cycles in each repetition of the simulation. `numeric vector` with length 2.

Details

In the first simulation cycle the object `sim_result` of the class `prosper` is created and the first cycle results are saved. This object is the same for all cycles in all repetitions and gets all the results. The function can only be called once per simulation run.

Value

A prosper object. SimData as a data.table with the repetition in the first and the simulation cycle in the second column.

See Also

struc_endSim

Examples

```
struc_preparation2(Rmx=10, af=c(2,1), epis=0, dom=c(1,0.3))
dfgenotype$"SB_autumn" <- c(4,23,0,123,53,98,45,3245,234)
dfgenotype$"SB_autumn_end" <- c(0,2,0,123,434,5234,5678,123,2)
#creation of an example object with data of the first year
struc_saveSimData(rep_counter=1, simcycle=1, simstruc=c(5, 10),
                                  start_names="SB_autumn", end_names="SB_autumn_end")
#creating some fictional population stages
dfgenotype$"SB_autumn" <- c(1,1,1,1,0,0,0,0,4)
dfgenotype$"SB_autumn_end" <- c(67,67,67,67,67,67,67,67,67)
#appending rows with the new results to the first results. necessary columns are inserted.
struc_saveSimData( rep_counter=1, simcycle=2,
                     start_names="SB_autumn", end_names="SB_autumn_end",
                      simstruc=c(repetitions, duration))
sim_result
rm(sim_result, dfgenotype, mf, xprobab)
```

---

summary *Summary for prosper objects*

Description

This function gives a first overview of a prosper simulation result.

Usage

```
## S4 method for signature 'prosper'
summary(object, geneticSumCol = "SB_autumn_end", ...)
```

Arguments

object	prosper, the result of a simulation with PROSPER.
geneticSumCol	column name of the prosper object the genetic summary is built of. character.
...	not in use.

Details

For the population dynamic part, means and standard deviations per simulation cycle for every numeric variable is calculated. Variability is introduced by repetitions. Additionally, a simple overview on factorial variables is given. For further calculations the numbers of individuals for each simulation cycle and repetition are provided.

For the genetic part, means and standard deviations of proportions of individuals with only sensitive alleles (allSalleles), only resistance alleles (allRalleles), mixed alleles (RSalleles), and the proportions of resistance alleles in the population (Ralleles) are calculated for each simulation cycle. The repetitions are responsible for the variability.

Value

A named list of data.tables.

See Also

plot

Examples

```
data(param.LOLRI)
mod_lolri    <- prosper.LOLRI(param.weed=param.LOLRI, area=30, af = c(0.005, 0.01),
                                      duration=3, repetitions=2)
summary(mod_lolri)
```

---

weed-parameters *Parameterisations for population dynamic models - Examples*

Description

PROSPER entails full parameterized models. The parameterisation for these models are described here.

Format

The provided data are given in tables with 10 columns.

weed the weed species.

crop the crop for which the parameters apply, `character`.

variable the name of the variable, simulation step, `character`.

name a naming of parameter, like a variable definition, `character`.

mean the mean of the parameters defined with "name", `numeric`.

se the standard error of the parameters defined with "name", `numeric`.

model a (statistical) model to describe a specific simulation step. The model must be constructed of defined parameters with "name", `character`.

range_low a model might have a minimum level of use, then this value can be defined here, `numeric`.

range_up a model might have a maximum level of use, then this value can be defined here, `numeric`.

source if the value is taken from sources not owned by the author they can be named here.

Details

1. `param.LOLRI` entails the data used for the introduction of PERTH by Renton et al. (2011) using the parameters of Pannell et al. (2004). With `prosper.LOLRI` one of the PERTH models can be reproduced.
2. `param.ECHCG` contains parameters for population dynamic of *Echinochloa crus-galli* (Redwitz et al., 2016).
3. `param.GALAP` contains parameters for population dynamic off *Galium aparine* taken from field experiments (Daedlow, 2015).

References

1. Bagavathiannan, MV, Norsworthy, JK (2013) Postdispersal loss of important arable weed seeds in the midsouthern United States. Weed Science, 61(4), 570-579.
2. Bosnic, AC, Swanton, C (1997) Influence of Barnyardgrass (Echinochloa crus-galli) Time of Emergence and Density on Corn (Zea mays). Weed Science, 45(2), 276-282.
3. Clay, SA, Kleinjan, J, Clay, DE, Forcella, Frank, Batchelor, W (2005) Growth and fecundity of several weed species in corn and soybean. Agronomy journal, 97(1), 294-302.
4. Daedlow, D. (2015): About the contribution of seed predation on weed demography. (Doctoral dissertation). University of Rostock, Faculty of Agriculture and Environmental Sciences. GBV Gemeinsamer Verbundkatalog (Accession No. 839752644).
5. Ogg, AG, Dawson, JH (1984) Time of emergence of eight weed species. Weed Science, 32(3), 327-335.
6. Pannell, DJ, Stewart, V, Bennett, A, Monjardino, M, Schmidt, C, Powles, SB (2004) RIM:a bioeconomic model for integrated weedmanagement of Lolium rigidum in Western Australia. Agricultural Systems, 79, 305-325.
7. Redwitz, C von, Pannwitt, H, Gerowitt, B (2016): About the interplay of sensitive and resistant biotypes in weed populations - simulation exercises for Echinochloa crus-galli in maize crops. Julius-Kuehn-Archiv 452, 93-99.
8. Renton, M.; Diggle, A.; Manalil, S. & Powles, S. (2011): Does cutting herbicide rates threaten the sustainability of weed management in cropping systems? Journal of Theoretical Biology, 283, 14-27.

See Also

`prosper-models` `quanti`

Index