Package: learnPopGen (via r-universe)

August 20, 2024

Version 1.0.6

Date 2022-12-29

Title Population Genetic Simulations & Numerical Analysis

Author Liam J. Revell

Maintainer Liam J. Revell <liam.revell@umb.edu>

Depends R (>= 2.6)

Imports grDevices, graphics, gtools, methods, phytools, stats

ZipData no

Description Conducts various numerical analyses and simulations in population genetics and evolutionary theory, primarily for the purpose of teaching (and learning about) key concepts in population & quantitative genetics, and evolutionary theory.

License GPL (>= 2)

URL http://github.com/liamrevell/learnPopGen

Date/Publication 2019-06-10 12:00:00 EST

Repository https://liamrevell.r-universe.dev

RemoteUrl https://github.com/liamrevell/learnpopgen

RemoteRef HEAD

RemoteSha de077491702aaaa07eaedb72a6e0eb248f054219

Contents

clt . . . 2
coalescent.plot . . . 3
drift.selection . . . 4
founder.event . . . 5
freqdep . . . 7
genetic.drift . . . 8
hardy.weinberg . . . 9
hawk.dove . . . 10

msd 11
mutation.selection 12
phenotype.freq 13
rcd 14
selection 15
sexratio 17

Index **19**

`clt` *Illustrates the concept of the Central Limit Theorem*

Description

The *Central Limit Theorem* tells us that when independent random variables are added together, the distribution of their sum tends towards a normal distribution, regardless of the shape of their individual distributions. This function attempts to illustrate this concept by allowing the user to visualize the sum of an arbitrary number of different independent random variables with different underlying distributions.

Usage

```
 clt(nvar=1, nobs=1000, df=c("normal","uniform","exponential","binomial"),
  theta=NULL, breaks="Sturges", show=c("sum","mean"))
 ## S3 method for class 'clt'
print(x, ...)
 ## S3 method for class 'clt'
plot(x, ...)
```

Arguments

`nvar`	number of random variables to sum (1 or more).
`nobs`	total number of observations (per random variable).
`df`	distribution functions of individual random variables to sum. These can be `"normal"`, `"uniform"`, `"exponential"`, or `"binomial"`.
`theta`	parameter of the distribution functions: variance in the case of `df = "normal"`, maximum value in the case of `"uniform"` (the minimum value will be assumed to be 0), rate in the case of `"exponential"`.
`breaks`	breaks (see `hist`).
`show`	whether to show the row-wise *sum* of the independent random variables (`show="sum"`), or their mean (`show="mean"`).
`x`	object of class `"clt"` for `print` and `plot` methods.
`...`	optional arguments for `print` and `plot` methods.

Details

The central limit theorem (CLT) establishes that when independent random variables are added together their (normalized) sum will tend towards a normal distribution, regardless of the distribution of the original random variables. That is to say if we were to generate a set of `nvar` (say) independent, uniform, random variables, normalize each one to have the same variance, and then sum or average the variables by observation, this sum or average will tend towards a normal distribution as the number of random variables (`nvar` in this function) is increased.

Value

Creates a plot showing the observation-wise distribution of the sum or average of the independent random variables.

The distribution of the observation-wise sum or average and the underlying data are also returned invisibly to the user in the form of an object of class `"clt"`. This object can in turn be printed or re-plotted using custom `print` and `plot` methods. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

phenotype.freq

Examples

```
clt(nvar=1,df="exponential")
clt(nvar=10,df="exponential")
object<-clt(nvar=40,df="exponential")
print(object)
plot(object)
```

---

coalescent.plot *Creates a (usually animated) simulation of gene coalescence within a population*

Description

Coalescence or coalescent theory is a model for genetic drift within a population in which we envision gene copies merging or "coalescing" into ancestors in the past. This function generates a(n) (optionally animated) visualization of this process of coalescence within a population.

Usage

```
coalescent.plot(n=10, ngen=20, colors=NULL, ...)
## S3 method for class 'coalescent.plot'
print(x, ...)
## S3 method for class 'coalescent.plot'
plot(x, ...)
```

Arguments

n	number of haploid individuals or gene copies.
ngen	number of generations.
colors	colors to use for plotting individuals and lines. By default, the function tries to use a contrasting color scheme such that adjacent allele copies are dissimilar (to facilitate visualization of the coalescent process.)
x	object of class "coalescent.plot" for print and plot methods.
...	optional arguments. For coalescent.plot optional arguments include: sleep, the time to pause between generations (set to 0.2s by default); lwd, the line width for parent-offspring lines in the coalescent geneology; and col.order, if colors=NULL, whether to use 'sequential' (col.order="sequential", the default) or 'alternating' (col.order="alternating") colors for adjacent alleles.

Value

Creates a plot or animation.

Invisibly returns an object of class "coalescent.plot" containing the alleles (coded numerically) and the parent-offspring relationships from the coalescent simulation. This object can be printed or re-plotted using print and plot methods. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

drift.selection, genetic.drift

Examples

```
coalescent.plot()
## Not run:
 coalescent.plot(n=20,ngen=30,col.order="alternating")
 object<-coalescent.plot()
 print(object)
 plot(object)

## End(Not run)
```

---

drift.selection *Simulation of genetic drift & natural selection at a biallelic locus*

Description

Simulates drift and natural selection at a single biallelic locus within one or various populations.

Usage

```
drift.selection(p0=0.5, Ne=100, w=c(1,1,1), ngen=400, nrep=10, colors=NULL, ...)
```

Arguments

`p0`	starting frequency for the *A* allele.
`Ne`	effective population size.
`w`	fitnesses of the three genotypes: *AA*, *Aa*, and *aa*.
`nrep`	number of replicate simulations.
`ngen`	total time, in number of generations, for the simulation.
`colors`	colors to use for plotting.
`...`	optional arguments. Presently the only arguments are `type` (e.g., `"l"`, `"s"`) and `lwd` in the `plot` method.

Value

The function creates a plot and returns an object of class `"drift.selelction"` consisting of list containing the allele frequency through time for each simulation. This object can be printed or plotted using corresponding methods. (See examples.)

Author(s)

Liam Revell `<liam.revell@umb.edu>`

See Also

`genetic.drift`, `selection`

Examples

```
drift.selection()
p<-drift.selection(p0=0.01,Ne=100,w=c(1,0.9,0.8),ngen=200,nrep=5)
print(p)
plot(p)
```

`founder.event` *Simulation of a founder event or population bottleneck*

Description

This function simulates genetic drift with a founding event / population bottleneck at time `etime`.

Usage

```
founder.event(p0=0.5, Ne=1000, Nf=10, ttime=100, etime=50, show="p", ...)
```

Arguments

p0	Starting frequency for the *A* allele.
Ne	Effective population size at the start of the simulation and after the founding event.
Nf	Size of the founding population.
ttime	Total time of the simulation.
etime	Time for the founding event. Can either be a single generation, or a sequence of generations (e.g., etime=40:50) for a prolonged founder event. Note that to simulate a prolonged bottleneck the user *must* supply the sequence of generations during which the population is to be bottlenecked and *not* merely the start and end times of the bottleneck.
show	Two different options for plotting. "p" shows the frequency of the A allele through time; "var" shows the genetic variation in the population, calculated as p*(1-p). The default is show="p".
...	optional arguments. Presently, the only optional argument in founder.event is ltype which specifies the line type and defaults to "s".

Value

The function creates one of two different plots, depending on the value of show.

The function also invisibly returns an object of class "founder.event" which can be printed or plotted using corresponding print and plot methods. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

drift.selection, genetic.drift

Examples

```
founder.event()
p<-founder.event(show="variation")
print(p)
plot(p,show="p",ltype="l")
```

freqdep *Numerical analysis of a frequency dependent seletion model*

Description

This function performs numerical analysis of a frequency dependent selection model based on Rice (2004; *Evolutionary Theory: Mathematical & Conceptual Foundations*). The fitnesses of the three genotypes in the model are as follows, where f(*XX*) denotes the frequency of the *XX* genotype: w(*AA*)=1-3*f(*Aa*)+3*f(*aa*); w(*Aa*)=1-s*f(*Aa*); and w(*aa*)=1-3*f(*Aa*)+3*f(*AA*). As shown in Rice (2004), though entirely deterministic, the model can exhibit chaotic behavior under some values for `s`.

Usage

```
freqdep(p0=0.01, s=0, time=100, show="p", pause=0, ...)
```

Arguments

`p0`	Starting frequency for the *A* allele.
`s`	Parameter that determines the strength of selection against heterozygotes when they are common.
`time`	Number of generations to run the analysis.
`show`	Various options for plotting. `"p"` shows the frequency of A through time; `"q"` gives the frequency of the *a* allele; `"fitness"` gives the mean population fitness through time; `"surface"` plots the mean fitness as a function of p; `"deltap"` shows the change in p as a function of p; `"cobweb"` creates a cobweb plot showing p(t) by p(t+1). The default is `show="p"`.
`pause`	Pause between generations. `pause=0.01` (for instance) might smooth animation.
`...`	optional arguments. Presently, the only optional argument in `freqdep` is `color`, which can be used to change the color of the lines of the plot. The `plot` method can also accept the optional arguments `type` (e.g., `"l"` or `"s"`) and `lwd`.

Value

The function creates one of several possible plots, depending on the value of `show`.

The use of cobweb plots follows `selection`.

The function also invisibly returns an object of class `"freqdep"` containing the frequency of the allele A through time, if this was calculated by the selected method. This can be printed or plotted using the corresponding methods. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

sexratio

Examples

```
freqdep(time=100)
freqdep(s=1.5,time=100)
p<-freqdep(s=2,show="cobweb",time=100)
plot(p)
```

---

genetic.drift *Genetic drift simulation*

Description

This function simulates genetic drift at a biallelic genetic locus with no selection and no mutation in a sexually reproducing diploid population or set of populations. It is essentially redundant with drift.selection, but in which there is no difference in relative fitness among genotypes; however, it also allows the user to visualize heterozygosity or genetic variation through time - options that are not yet implemented in drift.selection.

Usage

```
genetic.drift(p0=0.5, Ne=20, nrep=10, time=100, show="p", pause=0.1, ...)
```

Arguments

p0	Starting frequency for the A allele.
Ne	Effective population size.
nrep	Number of replicate simulations.
time	Total time, in number of generations, for the simulation.
show	Various options for plotting. "p" shows the frequency of the A allele through time; "genotypes" creates an animated histogram with the frequencies of each of the three genotypes through time; "fixed" shows the fraction of populations that have fixed for each allele, a or A; "heterozygosity" plots the mean heterozygosity and the expected heterozygosity through time. The default is show="p".
pause	Pause between generations. pause=0.01 (for instance) might smooth animation.
...	optional arguments. In genetic.drift the optional arguments are presently: colors (a vector giving the colors to be used to graph the various simulations); and lwd. The plot method of the object class adds the optional argument type (e.g., "l" or "s".)

Value

The function creates one of several possible plots, depending on the value of show.

The function also invisibly returns an object of class "genetic.drift" that can be printed or re-plotted by the user using corresponding print and plot methods. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

drift.selection, founder.event, selection

Examples

```
## Not run:
 genetic.drift()
 object<-genetic.drift(p0=0.7,show="heterozygosity")
 plot(object,show="genotypes")

## End(Not run)
```

---

hardy.weinberg *Computes Hardy-Weinberg frequencies for a multiallelic locus or across multiple loci*

Description

hardy.weinberg computes Hardy-Weinberg frequencies for a multiallelic locus using arbitrary allele frequencies.

multilocus.hw computes multilocus Hardy-Weinberg frequencies for a set of biallelic loci.

Usage

```
hardy.weinberg(p=c(0.5,0.5), alleles=c("A","a"), as.matrix=FALSE)
multilocus.hw(nloci=2, p=NULL)
```

Arguments

p	allele frequencies. In the case of multilocus.hw the frequencies of the *dominant* (in this case, merely uppercase) allele at each locus.
alleles	names of the alleles.
as.matrix	logical argument indicating whether to return the result in the form of a matrix (if TRUE) or a vector.
nloci	for multilocus.hw the number of loci.

Value

Returns a matrix or vector.

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

phenotype.freq

Examples

```
hardy.weinberg()
hardy.weinberg(p=c(0.4,0.3,0.2,0.1),alleles=letters[1:4])
```

---

hawk.dove *Analysis of hawk-dove game theoretic model*

Description

This function performs numerical analysis of a discrete-time hawk-dove model in which "payoff" determines relative fitness in the population.

Usage

```
hawk.dove(p=c(0.01,0.99), M=NULL, time=100)
```

Arguments

p	Starting frequency of hawk & dove phenotypes, respectively. Should correspond with the rows of M. If a single value is given then p will automatically be set to p=c(p,1-p).
M	Payoff matrix. M[i,j] should contain the fitness of i when interacting with j.
time	Number of generations.

Value

The function creates a two panel plot. The upper panel shows the relative frequencies of each of the two interacting phenotypes. The lower panel shows mean fitness of the population and of each morph through time.

The function also invisibly returns an object of class "hawk.dove" containing the frequencies of each strategy through time and their fitnesses. This object can be printed or re-plotted using corresponding print and plot methods. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

freqdep

Examples

```
hawk.dove(time=60)
Payoff<-matrix(c(0.5,0.6,1.5,1.0),2,2)
object<-hawk.dove(M=Payoff,time=60)
print(object)
plot(object)
```

---

msd *Migration, drift, and selection*

Description

Simulates migration, natural selection, and genetic drift. Selection can be in opposite directions in the two populations experiencing gene flow.

Usage

```
msd(p0=c(0.5,0.5), Ne=c(100,100), w=list(c(1,1,1),c(1,1,1)),
 m=c(0.01,0.01), ngen=400, colors=c("red","blue"), ...)
```

Arguments

p0	starting frequency for the *A* allele in each of two populations.
Ne	effective population size for each of two populations.
w	fitnesses of the three genotypes (*AA*, *Aa*, and *aa*, in that order) in each of the two populations. w should take the form of a list of two vectors.
m	rates of migration *from* the first population to the second, and from the second population to the first, in that order. This value is best interpreted as the *probability* that an individual born in population 1 will migrate to population 2 before reproduction, and *vice versa*.
ngen	total time, in number of generations, for the simulation.
colors	colors to use for plotting.
...	optional arguments. Presently, the only optional argument for msd is show.legend (which defaults to TRUE). The plot method adds the additional optional arguments of colors (a vector of colors for the two simulated populations), lwd, and type (e.g., "l" or "s").

Value

The function creates a plot and invisibly returns a list containing the allele frequency through time for each of the two simulated populations.

The returned object is of class "msd" and can be printed or re-plotted using corresponding print or plot methods. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

drift.selection

Examples

```
msd()
msd(p0=c(0.25,0.75),w=list(c(1,0.9,0.8),c(0.8,0.9,1)))
object<-msd(p0=c(0.75,0.25),w=list(c(1,0.9,0.8),
 c(0.8,0.9,1)),m=c(0.1,0.1),ngen=100)
print(object)
plot(object,colors=c("black","grey"),lwd=4,type="s")
```

---

mutation.selection *Gene frequencies over time under mutation-selection balance*

Description

This function performs numerical analysis of mutation-selection balance with mutation from *A* to *a* and selection against (either or both of) *Aa* and *aa*.

Usage

```
mutation.selection(p0=1.0, w=c(1,0), u=0.001, time=100, show="q", pause=0,
 ylim=c(0,1))
```

Arguments

p0	Starting frequency for the *A* allele.
w	Fitnesses of the heterozygote (*Aa*) and homozygote deleterious (*aa*) genotypes. The fitness of genotype *AA* is assumed to be 1.0.
u	Rate at which *A* alleles are converted to *a* alleles by mutation.
time	Number of generations to run the analysis.
show	Two options for plotting. "q" shows the frequency of a through time; "fitness" plots the mean fitness over time. The default is show="q".
pause	Pause between generations. pause=0.01 (for instance) might smooth animation.
ylim	Limits on the *y*-axis for plotting.

Value

The function creates one of three possible plots, depending on the value of `show`.

The function also invisibly returns the frequency of the A allele through time and the mean population fitness as an object of class `"mutation.selection"` that can be printed or re-plotted with associated `print` and `plot` methods, respectively. The `plot` method also permits user control over various attributes of the appearance of the plot, such as the color of the plotted lines (`color`), the line widths (`lwd`), the limits of the *y*-axis (`ylim`), and the type of line (e.g., `"l"` vs. `"s"`, via the argument `type`).

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

`genetic.drift`, `selection`

Examples

```
mutation.selection(w=c(1,0),time=100,ylim=c(0,0.1))
```

`phenotype.freq` *Computes phenotypic distribution and its change through time due to natural selection on a polygenic trait*

Description

`phenotype.freq` computes the phenotypic trait distribution for a polygenic trait. Can be used to demonstrate that the phenotypic distribution of a polygenic trait will tend to normality as the number of loci is increased, regardless of the allele frequencies at each locus.

`phenotype.selection` computes the change in the phenotypic trait distribution through time under natural selection. Can be used to show that natural selection on a polygenic trait can move the value of the trait well beyond its original distribution in the population.

Usage

```
phenotype.freq(nloci=6, p=NULL, effect=1/nloci)
phenotype.selection(nloci=6, p=NULL, effect=1/nloci, beta=0.1, ngen=20, ...)
```

Arguments

`nloci`	number of loci. For simplicity all loci are assumed to be biallelic.
`p`	allele frequency, *p*, for each locus, in a vector. If not supplied, initially frequencies will be assumed to be 0.5 at all loci.
`effect`	additive effect of an allele substitution. For simplicity, this is assumed to be the same at all loci.

beta	selection gradient.
ngen	number of generations to analyze.
...	optional arguments. Presently the only optional argument in the function phenotype.selection is sleep, which can be used to specify the time delay in seconds between generations.

Value

Creates a plot or animation.

phenotype.freq also invisibly returns an object of class "phenotype.freq" that can be printed or re-plotted using print and plot methods corresponding to the object type. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

clt, selection

Examples

```
## Not run:
 phenotype.freq(n=4)
 object<-phenotype.freq(nloci=6,p=runif(n=6),effect=1/6)
 print(object)
 plot(object)
 object<-phenotype.freq(nloci=10,p=runif(n=10),effect=rexp(n=10))
 print(object)
 plot(object)
 phenotype.selection(ngen=100)

## End(Not run)
```

---

rcd *Simulation of reproductive character displacement in an ecological community*

Description

This function conducts individual-based, genetically explicit numerical simulation of reproductive character displacement in an ecological community. The model is one of multiple species (with fixed relative abundance) competing to utilize the same signal space. There is both stabilizing selection on the signal trait for detectability, as well as (in multi-species simulations) countervailing selection for divergence due to the costs of erroneous mismating attempts.

Usage

```
rcd(nsp=3, nindivs=c(700,400,100), w_t=10, gen=c(500,500), figs="on", pf=100, ...)
```

Arguments

nsp	Number of species in the simulation. If figs="on", nsp must be 1, 2, or 3.
nindivs	A vector of length nsp containing the integer number of individuals in each species of the simulation.
w_t	Shape parameter of the Gaussian selection surface for the male signalling trait.
gen	Vector containing the number of allopatric generations followed by the number of sympatric generations for simulation.
figs	Either "on" if plotting is turned on, or "off" to suppress plotting.
pf	Print frequency for the simulation status to screen.
...	Optional arguments.

Value

The function returns a list containing the mean male signal trait and the mean female preference over time. It also (optionally) plots these.

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

drift.selection, genetic.drift, freqdep, selection

Examples

```
## Not run:
 obj<-rcd(nsp=2,nindivs=c(500,500))

## End(Not run)
```

---

selection *Numerical analysis of biallelic locus frequency independent selection*

Description

This function performs numerical analysis of a simple biallelic selection model.

Usage

```
selection(p0=0.01, w=c(1.0,0.9,0.8), time=100, show="p", pause=0, ...)
```

Arguments

p0	Starting frequency for the *A* allele.
w	Fitnesses for the three genotypes in the following order: *AA*, *Aa*, *aa*.
time	Number of generations to run the analysis.
show	Various options for plotting. "p" shows the frequency of *A* through time; "surface" plots the mean fitness as a function of *p*; "deltap" shows the change in *p* as a function of *p*; "cobweb" creates a cobweb plot showing $p(t)$ by $p(t+1)$. The default is show="p".
pause	Pause between generations. pause=0.01 (for instance) might smooth animation.
...	Optional arguments, including: add, a logical value indicating whether or not to add to the current plot (applies only to show="p"); color, change the color of the plotted line (works nicely with add, for obvious reasons); and equil, a logical value indicating whether or not to show the equilibrium value of *p* using vertical (or horizontal) lines on the graph (defaults to FALSE).

Value

The function creates one of several possible plots, depending on the value of show.

The cobweb plot shows $p(t+1)$ as a function of $p(t)$, with stairsteps giving the changes across generations given the initial value of *p* (p0) and total time (time) that are specified by the user.

The function invisibly returns an object of class "selection" which can be printed or re-plotted using associated print and plot methods. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

drift.selection, freqdep, msd, mutation.selection

Examples

```
selection(w=c(1.0,0.8,0.8),time=500)
selection(w=c(1.0,1.0,0.0),show="surface")
object<-selection(w=c(0.8,1.1,0.7))
print(object)
plot(object,show="cobweb")
```

`sexratio` *Hypothetical analysis of frequency dependent selection on a sex determining genetic locus*

Description

This function performs numerical analysis of a frequency dependent selection model of a hypothetical diploid sexually reproducing population in which sex is determined by the genotype at a biallelic genetic locus. Genotype *AA* are male, genotype *aa* are female, and genotype *Aa* might be male or female with probabilities that can be specified by the user. (Users may find, for instance, that setting `sex.Aa=c(1,0)` will result in evolution towards an *XY* sex determination system; whereas `sex.Aa=c(0,1)` will evolve towards a *ZW* system.)

Usage

```
sexratio(p0=0.01, time=40, show="p", pause=0, sex.Aa=c(0.5,0.5))
```

Arguments

`p0`	Starting frequency for the *A* allele. Individuals with *AA* genotypes are male, while individuals with *Aa* genotypes are male or female with probability given by `sex.Aa`.
`time`	Number of generations to run the analysis.
`show`	Two different options for plotting. `"p"` shows the frequency of *A* through time; `"fitness"` plots the mean fitness through time; `"sex-ratio"` plots the relative frequency of each sex; and `"genotypes"` plots the frequencies of the three genotypes in the population. The default is `show="p"`.
`pause`	Pause between generations. `pause=0.01` (for instance) might smooth animation.
`sex.Aa`	Probability that individuals with genotype *Aa* are male or female, respectively.

Value

The function creates one of four possible plots, depending on the value of `show`. Numerical analysis of this model shows how frequency dependent selection should favor alleles that tend to produce the rarer sex in the population.

The function invisibly returns an object of class `"sexratio"` that can be printed or re-plotted by the user. (See examples.)

Author(s)

Liam Revell <liam.revell@umb.edu>

See Also

`freqdep`

Examples

```
sexratio()
sexratio(p0=0.001,show="sex-ratio")
sexratio(p0=0.001,show="fitness")
object<-sexratio(p0=0.001,sex.Aa=c(0.9,0.1),
 time=20)
print(object)
par(mfrow=c(2,1))
plot(object,lwd=4,type="s",show="sex-ratio")
plot(object,lwd=4,type="s",show="genotypes")
par(mfrow=c(1,1))
```

Index

∗ **character displacement**
  rcd, 14
∗ **drift**
  drift.selection, 4
  founder.event, 5
  genetic.drift, 8
  msd, 11
  rcd, 14
∗ **evolutionary theory**
  freqdep, 7
  hawk.dove, 10
  mutation.selection, 12
  selection, 15
  sexratio, 17
∗ **game theory**
  hawk.dove, 10
∗ **migration**
  msd, 11
∗ **population genetics**
  coalescent.plot, 3
  drift.selection, 4
  founder.event, 5
  freqdep, 7
  genetic.drift, 8
  hardy.weinberg, 9
  msd, 11
  mutation.selection, 12
  phenotype.freq, 13
  rcd, 14
  selection, 15
  sexratio, 17
∗ **sexual selection**
  rcd, 14
∗ **statistics**
  clt, 2

clt, 2, *14*
coalescent.plot, 3

drift.selection, *4*, 4, *6*, *8*, *9*, *12*, *15*, *16*

founder.event, 5, *9*
freqdep, 7, *11*, *15–17*

genetic.drift, *4–6*, 8, *13*, *15*

hardy.weinberg, 9
hawk.dove, 10
hist, *2*

msd, 11, *16*
multilocus.hw *(*hardy.weinberg*)*, 9
mutation.selection, 12, *16*

phenotype.freq, *3*, *10*, 13
phenotype.selection *(*phenotype.freq*)*, 13
plot.clt *(*clt*)*, 2
plot.coalescent.plot *(*coalescent.plot*)*, 3
print.clt *(*clt*)*, 2
print.coalescent.plot *(*coalescent.plot*)*, 3

rcd, 14

selection, *5*, *7*, *9*, *13–15*, 15
sexratio, *8*, 17