Package 'DeepCNV'

September 24, 2026

Type Package

Title Exploiting Normal Contamination to Infer Copy Number from Deep Sequencing

Version 0.12.3

Date 2026-09-24

Author Mark Zucker, Kevin R Coombes

Maintainer Kevin Coombes <krc@silicovore.com>

Description The DeepCNV package provides a systematic Bayesian framework for inferring copy number from deep sequencing data of one or a few genes.

License Apache License (== 2.0)

Depends R (>= 3.2.0)

Imports methods

Suggests xtable

URL http://silicovore.com/OOMPA/standalone.html

NeedsCompilation no

Contents

`CNVPosterior-class` *Class "CNVPosterior"*

Description

Represents the posterior distribution used to infer copy number from targeted deep sequencing.

Usage

```
makeCNVPosterior(obs, prior)
## S4 method for signature 'CNVPosterior,missing'
plot(x, place="topright", lwd=1, ...)
## S4 method for signature 'CNVPosterior'
hist(x, place="topright", lwd=1, ...)
## S4 method for signature 'CNVPosterior'
summary(object, ...)
## S4 method for signature 'CNVPosteriorSummary'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

obs	Observed read-count data
prior	Prior distribution, represented using the class CNVPrior.
object	object of class CNVPosterior
x	object of class CNVPosterior
place	Character string; where to place the figure legend
lwd	Line width parameter.
row.names	See as.data.frame
optional	See as.data.frame
...	extra arguments for generic or plotting routines

Details

The DeepCNV class is used to fit a Bayesian model to targeted sequencing data from one or a few genes in order to draw inferences about possible copy number changes. Basically, we assume that the observed data consists of a list of triples (K, N, V), one for each variant in a gene. Here K is the number of variant reads, N is the total number of reads, and V is the type of each variant (either a known SNP or a somatic mutation). We model (K, N) using a binomial distribution, where the 'success' parameter depends (in a deterministic way) on the unknown parameters of interest: the fraction ν of normal cells in the sample and the copy number state (Normal, Deleted, or Gained).

The prior distribution consists of a continuous (by default, Beta(α, β)) distribution on ν and a discrete distribution on the copy number state S, which are stored ina n object of the CNVPrior class. The CNVariant computes the success parameter ϕ as a function of the observed data (K,N,V). Then ϕ and thbe observed data are passed on to the cnvLikelihood function, whcih computes the binomial log-likelihood.

We compute the posterior distribution essentially by brute force, in the function makeCNVPosterior. We choopse regularly spaced grid of points on the inerval (0,1) and evaluate the log-likelihood at every grid point foir each conceivable copy number state. These likelihoods are combined with the prior distribution by teh usual application of Bayes Rule.

One complication arises from the variant type V, which can be observed with error. Additioal complications arise because the variant (whether SNP or mutation) can turn out to be either the major or minor allele. These difficulties are mostly hidden from the user, and resolved by rfeplacing an overabundaqnce of states with the maximu a posteriori selection at each variant. This design decision afects the structure of the class, which compute multiple forms of the likelihood:

1. hiddenloglike, for all hidden states
2. snploglike, for each variant separately, collapsiong hidden states

3. `loglike`, for the unified gene

and posterior distribution:

1. `snppost`, for eqach variant separately
2. `posterior`, for the unified gene

Value

The `makeCNVPosterior` constructor returns a valid object of the class.

Slots

`hiddenloglike`: Log-likelihood using the hidden expanded discrete model

`snploglike`: Log-likelihood for each SNP analyzed one at a time

`snppost`: Posterior distribution of parameters for each SNP analyzed one at a time

`loglike`: Log-likelihood merging data from all SNPs

`posterior`: Posterrio distribution merging data from all SNPs

`calls`: List of maximu a posterior variant types used to collapse the hidden model

`observed`: Observed read data

`prior`: Prior distribution

Methods

plot(x, ...) Prints all the information in the object

hist(x, ...) Prints all the information in the object

summary(object, ...) Writes out a summary of the object

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

CNVPosterior

Examples

```
prior <- setCNVPrior(alpha=1.2, beta=4.8, pAbnormal=0.6)
dataset <- simReads(2, 7, 0.17, "Normal")
posterior <- makeCNVPosterior(dataset, prior)
s <- summary(posterior)
s
as.data.frame(s)
plot(posterior)
hist(posterior)
```

CNVPrior-class *Class "CNVPrior"*

Description

Represents the prior distribution used to infer copy number from targeted deep sequencing.

Usage

```
setCNVPrior(alpha = 1, beta = 1,
            pAbnormal=2/3, pGain = NULL, pLoss = pGain,
            pMutWrong = 0.01, pSnpWrong = 0.01, resn=300)
cnPrior(object)
## S4 method for signature 'CNVPrior,missing'
plot(x, lwd=2, ...)
## S4 method for signature 'CNVPrior'
summary(object, ...)
```

Arguments

alpha	Parameter for the beta prior distribution on the fraction of normal cells.
beta	Parameter for the beta prior distribution on the fraction of normal cells.
pAbnormal	Prior probability of an aberration; split equally between gain and loss.
pGain	Prior probability of a gain in copy number. Defaults to NULL, in which case it is set to half of pAbnormal.
pLoss	Prior probability of a loss of copy number. Defaults to NULL, in which case it is set to half of pAbnormal.
pMutWrong	Prior probability that something called a somatic mutation is actually a known single nucleotide polymorphism (SNP).
pSnpWrong	Prior probability that something called a SNP is actually a somatic mutation.
resn	The number of points at which to compute values of the beta distribution in order to compute the likelihood and posterior.
object	object of class CNVPrior.
x	object of class CNVPrior.
...	extra arguments for generic or plotting routines.
lwd	Line width parameter.

Details

The DeepCNV class is used to fit a Bayesian model to targeted sequencing data from one or a few genes in order to draw inferences about possible copy number changes. Basically, we assume that the observed data consists of a list of triples (K, N, V), one for each variant in a gene. Here K is the number of variant reads, N is the total number of reads, and V is the type of each variant (either a known SNP or a somatic mutation). We model (K, N) using a binomial distribution, where the 'success' parameter depends (in a deterministic way) on the unknown parameters of interest: the fraction ν of normal cells in the sample and the copy number state (Normal, Deleted, or Gained).

The prior distribution consists of a continuous (by default, Beta(α, β)) distribution on ν and a discrete distribution on the copy number state S. The discrete distribution can be defined either by

setting pAbnormal, in which case the probability os divided equally between Deleted and Gained, or by setting both pGain and pLoss.

Note that the settings of pGain and pLoss, if present, override any setting of pAbnormal.

One complication arises from the variant type V, which can be observed with error. In some cases, observations of V are compared with a matched normal sample of DNA from the same subject, in which case the variant type is known with great confidence. In the absence of matched normal samples, however, the variant type may be inferred indirectly from an external database like dbSNP, in which case the confidence decreases. Our confidence in these calls is represented by the prior probabilities assigned in pMutWrong and pSnpWrong. For purposes of our analysis, V is a nuisance parameter that we mostly try to hide from the user. However, the full structure of the discrete prior is revealed when using the summary method to look at the prior.

Value

The setCNVPrior constructor returns a valid object of the CNVPrior class.

The cnPrior function returns a vector of length three containing the prior probabilities of the copy number states.

Slots

discrete: Data frame containing the discrete part of the prior.

pGain: Prior probability of a gain in copy number.

pLoss: Prior probability of a loss of copy number.

pMutWrong Prior probability that something called a somatic mutation is actually a known single nucleotide polymorphism (SNP).

pSnpWrong Prior probability that something called a SNP is actually a somatic mutation.

alpha Parameter for the beta prior distribution on the fraction of normal cells.

beta Parameter for the beta prior distribution on the fraction of normal cells.

grid: Realized grid of points at which the continuous prior is computed.

pdf: Values of the prior distribution at the grid points.

Methods

plot(x, ...) PLots the continuous part of the prior distribution.

summary(object, ...) Writes out a table detailing the discrete part of the prior distribution.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

CNVPosterior

Examples

```
prior <- setCNVPrior(alpha=1.2, beta=4.8, pAbnormal=0.6)
cnPrior(prior)
summary(prior)
plot(prior)
```

MultiCNVPosterior-class

Class "MultiCNVPosterior"

Description

Represents the posterior distribution used to infer copy number from targeted deep sequencing of multiple genes.

Usage

```
multiCNVPosterior(obs, prior)
## S4 method for signature 'MultiCNVPosterior'
summary(object, ...)
## S4 method for signature 'MultiCNVPosterior,missing'
plot(x, place="topright", lwd=1, ...)
```

Arguments

obs	List or data frame of observed read-count data
prior	Prior distribution, represented using the class multiCNVPrior.
object	object of class multiCNVPosterior
x	object of class CNVPosterior
place	Character string; where to place the figure legend
lwd	Line width parameter.
...	extra arguments for generic or plotting routines

Details

The DeepCNV class is used to fit a Bayesian model to targeted sequencing data from one or a few genes in order to draw inferences about possible copy number changes. Details on the algorithm can be found in CNVPrior and CNVPosterior and in the vignettes.

The multiCNVPosterior function is the constructor for the MultiCNVPosterior class, and it implements the Bayesian algorithm on targeted sequencing data from multiple genes. The key point is that estimates of the posterior distribution on the fraction ν of normal cells will be much more reliable by combining the observed variants from several genes. In turn, this improved posterior can yield better copy number calls for the genes.

Limitations: In the current implementation, the same discrete prior distribution on the colpy number state must be used for all genes. However, when multiple genes are sequenced, one should in principle be able to use the read depth of different genes to get a better idea of which ones are likely to be gained or lost, thus customizing the priors (or better yet, formally adding this information to the model). A second shortcoming is that there is currently no easy way to convert the joint posterior distribution on the normal fraction into an object of the class CNVPrior to reuse it.

Value

The makemultiCNVPosterior constructor returns a valid object of the class.

Slots

gposts: List of CNVPosterior objects, one for each gene

cps An object of the CNVPosteriorSummary class.

Methods

summary(object, ...) Writes out a summary of the object

plot(x, ...) Plots the posteriro distribution on the normal fraction

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

CNVPrior, CNVPosterior, cnvLikelihood

Examples

```
prior <- setCNVPrior(alpha=1.2, beta=4.8, pAbnormal=0.6)
obs1 <- simReads(1, 9, 0.23, "Normal")
obs2 <- simReads(1, 7, 0.23, "Deleted")
observed <- list(G1 = obs1, G2=obs2)
rm(obs1, obs2)
prior <- setCNVPrior(alpha=1.2, beta=4.8, pAbnormal=0.6)
mpost <- multiCNVPosterior(observed, prior)
summary(mpost)
plot(mpost, main="Posterior Distribution on Nomral Fraction")
```

---

simReads *Simulating Read-Counts and Working With Priors for DeepCNV*

Description

The DeepCNV class is used to fit a Bayesian model to targeted sequencing data from one or a few genes in order to draw inferences about possible copy number changes. It includes routines to simulate read-counts with known copy number state and known fraction of normal 'contaminating' cells.

Usage

```
simReads(nMut, nVar, nu, cnstate=cnSet, depth=100, sdepth=25, ...)
CNVariant(nu, S = cnSet, V = vtSet, M = 2)
cnvLikelihood(nu, K, N, V, S)
```

Arguments

nMut	integer; the number of somatic mutations in a gene
nVar	integer; the number of variant SNPs in a gene
nu	numeric between 0 and 1; the fr4action of normal cells in the sample
cnstate	the copy number state of the gene; must be one of "Deleted", "Normal", or "Gained", as enumerated in cnSetg
depth	integer; the average read depth at the gene
sdepth	integer; the stnadard deviation of the read depth across varaints within a gene
...	extra parameters to pass from simReads to CNVariant
S	The copy number state, as enumerated in cnSet
V	The variant type. Must be one of "Mutation" or "SNP" as enumerated in vtSet
M	the total number of replicate copies of a (allelic) gene. The default value of 2 corresponds to a gain of one copy
K	Number of variant reads
N	Number of total reads (both variant and refernce)

Details

The DeepCNV class is used to fit a Bayesian model to targeted sequencing data from one or a few genes in order to draw inferences about possible copy number changes. Basically, we assume that the observed data consists of a list of triples (K, N, V), one for each variant in a gene. Here K is the number of variant reads, N is the total number of reads, and V is the type of each variant (either a known SNP or a somatic mutation). We model (K, N) using a binomial distribution, where the 'success' parameter ϕ depends (in a deterministic way) on the unknown parameters of interest: the fraction ν of normal cells in the sample and the copy number state (Normal, Deleted, or Gained).

The functions cnvLikelihood and CNVariant are used to compute the log-likelihood of the unknown parameters given the observed data. CNVariant computes the success parameter ϕ as a function of the observed data (K, N, V), and this parameter is then used to compute the binomial log-likelihood.

The simReads function generates simulated read-count data based on the underlying theoretical binomial model. More details can be found in the vignettes d01-cnvTheory and d02-oneGeneSims.

Value

The simReads function returns a data frame suitable for use by the function makeCNVPosterior.

The CNVariant function returns a real number between zero and one, corresponding to the fraction of reads that are expected to be variants given the variant type (V), the copy number state (S), and the fraction of normal cells (nu).

The cnvLikelihood function returns a real number representing the log-likelihood (yes, I know; it probably should be renamed) of the parameters (S, ν) given the observed data (K, N).

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

CNVPrior, CNVPosterior.

Examples

```
simReads(nMut=2, nVar=7, nu=0.17, "Norm", depth=130 )
# check log-likelihhoods of different copy number states
# for the same observed data
obs <- data.frame(K=c(69, 48), N=c(153, 167))
cnvLikelihood(0.22, obs)
cnvLikelihood(0.22, obs)
cnvLikelihood(0.22, obs)
```

Index