

# Package: gasplit (via r-universe)

October 16, 2024

**Title** Flexible estimation of 2-category proportions

**Version** 1.0.6

**Author** Mark Bravington <markb2@summerinsouth.net>

**Description** Like a GLM/GAM where the true category of each sample is unknown, but the response data is a continuous variable with known distributions for category-1 and category-2 samples.

**Maintainer** Mark Bravington <markb2@summerinsouth.net>

**Imports** mvbutils, atease, mgcv

**Suggests** offarray

**License** CC-BY

**Repository** <https://markbravington.r-universe.dev>

**RemoteUrl** <https://github.com/markbravington/gasplit>

**RemoteRef** HEAD

**RemoteSha** a28c45e4e49701634b4fa6a8db7a4dc7b0be835a

## Contents

gasplit . . . . .	1
make_fake_ppns . . . . .	3
<b>Index</b>	<b>5</b>

---

gasplit

*Flexible models for 2-category proportions from continuous data*

---

## Description

Suppose your data each come from one or the other of two categories (say, Eastern or Western Atlantic). You don't know for sure which one, but you do have a "measurement" for each one (say, a log-likelihood ratio) which tends to be high for one category and low for the other, with an a-priori-known distribution for each category (say, based on other samples of known category).

Now you want to estimate the proportion of category-1 data in your data, but *stratified* according to covariates of the data (say, year and location)— and/or simply dependent on individual covariates, (say, size). Ideally, you'd like to be able to specify a flexible model formula just as you might in a GLM or GAM. Well, now you can! For example:

```
addmod <- gasplit( LGLR ~ year + region - 1, data= ew_data,
  d1= known_distro_of_measurement_for_category_1,
  d2= ditto_for_2
)
```

to fit a two-way model with no interactions, to a dataset with covariates "year" and "region" based on a measurement "LGLR".

Having fitted your model, you can make a subsequent call to predict the proportion for (presumably) new covariate values, via the `predict_from_previous` argument..

See EXAMPLES of [test\\_gasplit](#) (qv) for practical stuff.

**To do:** Move this to RTMB, and add random effects, so that you could write eg

```
fancymod <- gasplit( LGLR ~ year + region + s( year, region, bs='re'), ...)
```

using mgcv syntax to put a random-effect on each interaction term. It's easy enough in principle, but everything takes time...

## Usage

```
gasplit(formula, data, d1, d2, start=0.001,
  predict_from_previous=NULL)
```

## Arguments

<code>formula</code>	a classic R model formula (eventually allowing mgcv extensions)
<code>data</code>	a dataframe...
<code>d1, d2</code>	these are <i>functions</i> giving the PDF of the response variable for category-1 and category-2 samples. You gotta pre-fit those to other data. See EXAMPLES.
<code>start</code>	Either a scalar (probably close to 0) which will be used for all coefficients, or a vector with length <code>ncol(&lt;model matrix&gt;)</code>
<code>predict_from_previous</code>	Optional result from a previous call to <code>gasplit</code> . If set, then no fitting will happen, but <code>gasplit</code> will make predictions at the (presumably new) values of data, based on the parameters of the previous fit.

## Details

Given parameters  $\beta$  and a model-matrix  $X$  from the formula, the log-likelihood of observation  $i$  with response  $y_i$  is:

$$\text{ppn}_i * d1(y_i) + (1 - \text{ppn}_i) * d2(y_i)$$

where  $\text{ppn}_i = \text{inv.logit}(\sum_j \{X_{\{i,j\}} * \beta_j\})$ .

## Value

A list with elements `par` (raw parameter estimates from `optim`) and `ppn`, which is a vector of estimated proportion-1 for each *observation*. If the "strata" are discrete, there will be lots of identical `ppn` values, but not necessarily the case.

---

make_fake_ppns	<i>Simulate dataset for 'gasplit'</i>
----------------	---------------------------------------

---

## Description

`make_fake_ppns` simulates a dataset for fitting with [gasplit](#), using two factors  $Y$  and  $Z$  and giving a response variable LGLR (so the strata are all combinations of  $Y$  and  $Z$ ). You can control the number of factor-levels, distributions of LGLR for category-1 and category-2 observations (and thus the "power" of LGLR for assignment), the number of samples per stratum, the average strength of each  $Y$ -effect and  $Z$ -effect, the average strength of  $Y*Z$  interactions, etc. It requires the **offarray** package (purely for my programming convenience; it's not apparent in the output).

`test_gasplit` fits a [gasplit](#) model (with no interactions) to the output of `make_fake_ppns`, running the latter itself if required. The output is the (true and) fitted proportions per strata, which can be compared: see **Examples**. Note that it's a bit harsh to simulate with `interpow > 0` and then test with [gasplit](#), coz the latter assumes there's no interactions. (Of course, you can fit a model with interactions if you want.)

## Usage

```
make_fake_ppns(nyears = 4, nzones = 3, interpow = 0.1,
               df_t = 5, mean_nsamp = 100, meanE = 1, prange = 1, seed = 2)
test_gasplit(sim=NULL, ...)
```

## Arguments

<code>nyears, nzones</code>	number of factor levels
<code>interpow</code>	average strength of an interaction relative to a main effect
<code>mean_nsamp</code>	actually there are <i>exactly</i> this many total samples from each stratum
<code>df_t, meanE</code>	<code>df_t</code> degrees-of-freedom for the t-distributions of response by category, which will have variance 1 and means $+\text{meanE}$ and $-\text{meanE}$ . A <code>meanE</code> of 1 corresponds to pretty poor separation; 3 is pretty good.

prange	how strong the effects should be. If they are too strong, then
seed	random number seed. The system seed is restored on completion.
sim, ...	(test_gasplit) sim should come from a previous call to make_fake_ppns. If not supplied, then make_fake_ppns will be run first, using the ... arguments.

### Value

make\_fake\_ppns returns a dataframe that has all the stuff you need, plus an attribute truth which is a list containing the real dope, as stratum-specific offarrays. Note that truth\$samp\_ppnE is the actual ppn of category-1 ("E") samples in each stratum, which will differ from the nominal value given by the factor strengths and interactions because of binomial sampling variability.

### See Also

[gasplit](#)

### Examples

```
if( require( offarray)){
sim2 <- make_fake_ppns(
  meanE= -2, # not-brilliant separation
  mean_nsamp=200, # reasonable samp sizes
  interpow=0) # no interactoins
hist( sim2$LGLR, nc=40) # to see how good the separation is
test2 <- test_gasplit( sim2)
with( test2, plot( tru_ppnE, fit_ppnE))
abline( 0,1)
# Make up some new data...
splodge <- with( test2, simpure[ 1:7,])
# ... and predict. The result of the original call to gasplit() is
# stored in test2$gg1
splodge_pred <- gasplit( data=splodge, predict_from_previous=test2$gg1)
# ... for normal non-test situations, do something more like this:
if( FALSE){
  pfit <- gasplit( myform, mydata)
  ppred <- gasplit( data=newmydata, predict_from_previous=pfit)
}
}
```

# Index

\* **misc**

gasplit, [1](#)

make\_fake\_ppns, [3](#)

gasplit, [1](#), [3](#), [4](#)

make\_fake\_ppns, [3](#)

test\_gasplit, [2](#)

test\_gasplit(make\_fake\_ppns), [3](#)