plot3logit: Ternary Plots for Interpreting Trinomial Regression Models

Flavio Santi
University of Trento

Maria Michela Dickson
University of Padua

Giuseppe Espa
University of Trento

Diego Giuliani
University of Trento

Abstract

This paper presents the R package **plot3logit** which enables the covariate effects of trinomial regression models to be represented graphically by means of a ternary plot. The aim of the plot is helping the interpretation of regression coefficients in terms of the effects that a change in values of regressors has on the probability distribution of the dependent variable. Such changes may involve either a single regressor, or a group of them (composite changes), and the package permits both cases to be handled in a user-friendly way. Moreover, **plot3logit** can compute and draw confidence regions of the effects of covariate changes and enables multiple changes and profiles to be represented and compared jointly. Upstream and downstream compatibility makes the package able to work with other R packages or applications other than R.

Keywords: plotting software, ternary diagrams, R, **plot3logit**.

1. Introduction

The interpretation of the covariate effect on the probability distribution of the dependent variable of a multinomial regression model is usually neither immediate nor easy. In case of multinomial logit regression, the coefficient of a covariate x referred to the category $\nu^{(m)}$ of the dependent variable determines the effect of a unitary change in the value of x on the logarithm of the ratio between the probability of category $\nu^{(m)}$ and the probability of the reference category $\nu^{(1)}$ of the dependent variable. This entails that the relation between the covariate coefficients and the probability distribution of the dependent variable is non-linear and depends also on covariate coefficients of the other regressors (see Santi, Dickson, and Espa 2019, Equations 5 and 6).

The interpretive difficulty of the parameters of multilogit models is the reason why the coefficient estimates are usually complemented by some estimates or graphical representations of covariate marginal effects. Indeed, both approaches turned out to be fruitful and led to a wide myriad of variants which have been studied from a methodological point of view (see, among the others, Agresti 2013; Fox and Weisberg 2019, 2018; Fox 2003), and have been implemented in R packages such as **effects** (Fox and Hong 2009), **lsmeans** (Lenth 2016), **emmeans** (Lenth 2020), **MNLpred** (Neumann 2020), **DAMisc** (Armstrong 2020).

Yet, both estimates and graphical representations of marginal effects are computed and plotted conditionally to some specific values of the covariates (or a subset of them), thus they cannot exhaustively describe the effect of a covariate over the whole space of regressors. In order to overcome this limitation, Tutz and Schauberger (2013) proposed a diagram, which allows for a representation of the direction (increase vs decrease) and the relative magnitude of the conditional effect of covariates on the probability distribution of the dependent variable. The method, implemented in the R package **EffectStars2** (Schauberger 2019), produces a very appealing and intuitive graph, which can be drawn for multinomial models with any number of categories on the dependent variable, however it relies on a reparametrisation of the multinomial logit model based on the symmetric side constraint, which, in some circumstances, may be unfeasible or undesirable.

In case of multinomial logit models where the dependent variable can take only three values (i.e., the trinomial logit models), Santi *et al.* (2019) show that it is possible to represent the effects of covariates in terms of changes in the probability distribution of the dependent variable by means of a vector field drawn over a ternary plot. Such a representation is possible both conditionally and unconditionally to the values of the covariates, and it can be obtained for changes involving two or more covariates (composite changes).

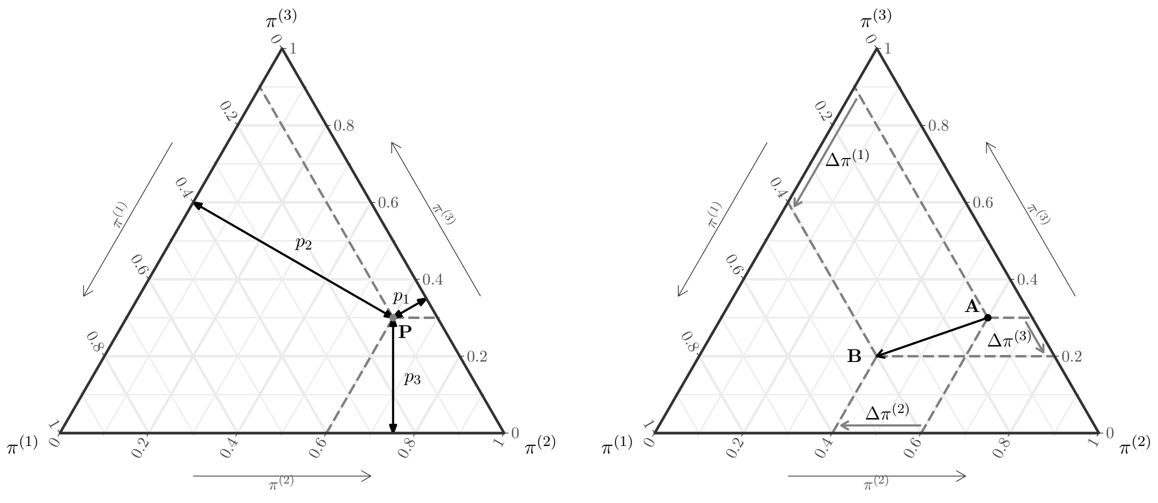
The graphical representation proposed in Santi *et al.* (2019) is implemented in R (R Core Team 2020) through package **plot3logit** (Santi, Dickson, Espa, and Giuliani 2024), available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=plot3logit` since January 2019.

Package **plot3logit** can read the results of both categorical and ordinal trinomial logit regression fitted by various functions (see Section 3) and creates a '`multifield3logit`' object which may be represented by means of functions either based on standard R graphics or based on the grammar of graphics (Wilkinson 2005). Composite changes and multiple changes of covariates can be easily represented through a simple and flexible syntax, whereas the analysis proposed in Santi *et al.* (2019) has been extended by including functions for adding confidence regions of the covariate effects to the plots, in order to enrich and improve the interpretation of the results.

The paper is organised as follows. Section 2 briefly shows how to read ternary plots and how the effects of covariate changes on the probability distribution of the dependent variable in a trinomial logit regression can be represented by means of vector fields and arrows on a ternary plot. Section 3 summarises the features of the package **plot3logit**. Section 4 illustrates how **plot3logit** reads estimates from fitted models, and how the vector fields can be customised, computed and represented graphically. Section 5 illustrates how confidence regions are computed and drawn. Section 6 introduces some wrappers. Finally, Section 7 concludes.

2. Ternary plots and trinomial logit regression

Ternary diagrams were firstly proposed in Bancroft (1897) as a method for representing sets of three numbers from bounded non-negative intervals subject to a constraint on their sum. This is the case of composition data as well as the probabilities of a trinomial random variable. Here we briefly sum up how ternary diagrams work; a more detailed illustration is available in Santi *et al.* (2019), whereas Howarth (1996) offers a valuable and intriguing history of ternary



(a) (b)

Figure 1: Figure (a) shows how the coordinates of a point $P = (p_1, p_2, p_3)$ can be read in a ternary diagram. Figure (b) shows how a change in the probability distribution of a trinomial random variable from $A = (0.1, 0.6, 0.3)$ to $B = (0.4, 0.4, 0.2)$ can be represented, and decomposed in terms of changes of ternary coordinates. Both graphs are taken from Santi *et al.* (2019).

diagrams.

Consider a random element N which takes values in a set of three labels $\{\nu^{(1)}, \nu^{(2)}, \nu^{(3)}\}$ with probability $\pi^{(m)} \equiv \mathbb{P}[N = \nu^{(m)}]$, $m = 1, 2, 3$. The probability distribution of N can be represented through the triplets $(\pi^{(1)}, \pi^{(2)}, \pi^{(3)}) \in [0, 1]^3$, however the parameter space is actually 2-dimensional, as the sum $\pi^{(1)} + \pi^{(2)} + \pi^{(3)}$ is constrained to equal one, thus if $\pi^{(1)}$ and $\pi^{(2)}$ are given, $\pi^{(3)}$ automatically equals $1 - \pi^{(1)} - \pi^{(2)}$.[1] Mathematically, triplets $(\pi^{(1)}, \pi^{(2)}, \pi^{(3)})$ which are valid probability distributions define a 2-dimensional simplex in the 3-dimensional space $[0, 1]^3$, which is denoted by S in the rest of the paper. Formally:

$$S = \{(\pi^{(1)}, \pi^{(2)}, \pi^{(3)}) \in [0, 1]^3 \colon \pi^{(1)} + \pi^{(2)} + \pi^{(3)} = 1\}\,. \tag{1}$$

The simplex S is the equilateral triangle which constitutes the ternary diagram (see Figure 1).

Figure 1a shows how the Cartesian coordinates of a point $P = (p_1, p_2, p_3)$ in the 3-dimensional space $[0, 1]^3$ are transposed over the 2-dimensional simplex (the ternary diagram). Note that the value of a coordinate of the point P (say, p_3) is the distance between P and the side opposite the vertex labelled with that component (that is, $\pi^{(3)}$).

Since all (and only) the admittable probability distributions of a trinomial random variable can be drawn as a point of the simplex of the ternary diagram, a change in any probability distribution can be represented through an arrow from a reference starting point A towards

[1]Random element N is typically modelled by means of a random vector which is distributed according to a single-trial multinomial law and it is defined through indicator functions. See Santi *et al.* (2019) for this formalisation of the problem, Johnson, Kemp, and Kotz (2005) (pp. 505–524) on the multinomial probability distribution, and Agresti (2013) on the modellisation of categorical responses.

a final point B. Figure 1b depicts the change of the probability distribution just described, and synthesises the basic idea for representing the effect of one or more covariates on the probability distribution of the dependent variable of a trinomial logit regression.

In order to make notation clear, the trinomial logistic regression is briefly introduced; a more detailed discussion of the model and the notation adopted in this paper is available in Santi *et al.* (2019), whereas a wide and in-depth dissertation on the multinomial logit regression can be found in Agresti (2013).

The multinomial logistic (or logit) regression aims at explaining the probability distribution of a multinomial variable by means of a set of regressors, which may be either quantitative or qualitative. If the number of possible values of the dependent variable equals three, we may refer to it as trinomial, and the model as trinomial logit regression.

The multinomial probability distribution belongs to the exponential family (Lehmann and Casella 1998, pp. 24–25), and, in case of the trinomial distribution, it is identified by means of natural parameters $(\eta_2, \eta_3) \in \mathbb{R}^2$, which are defined as follows:

$$\eta_m = \ln \frac{\pi^{(m)}}{\pi^{(1)}}\,, \qquad m = 2, 3. \tag{2}$$

Thus, the trinomial logit regression models the natural parameters $(\eta_2, \eta_3) \in \mathbb{R}^2$ as a linear transformation of the covariates $x \in \mathbb{R}^p$:

$$\eta(x) = B^\top x = \begin{bmatrix} \beta^{(2)} & \beta^{(3)} \end{bmatrix}^\top x = \begin{bmatrix} x^\top \beta^{(2)} \\ x^\top \beta^{(3)} \end{bmatrix} \tag{3}$$

where $\beta^{(2)} \in \mathbb{R}^p$ and $\beta^{(3)} \in \mathbb{R}^p$ are the regression coefficients.

Equations (2) and (3) justify the interpretation of regression coefficients $\beta_j^{(m)}$ as the effect of a unitary change of the j-th covariate on the logarithm of the ratio between $\pi^{(m)}$ and $\pi^{(1)}$.

Now, consider a trinomial logit regression on p covariates $x = (x_1, x_2, \ldots, x_p)$ (including a constant term) and a profile $x_0 \in \mathcal{X} \subseteq \mathbb{R}^p$, so that $(\pi^{(1)}_{(x_0)}, \pi^{(2)}_{(x_0)}, \pi^{(3)}_{(x_0)})$ is the probability distribution associated to $x = x_0$. It can be shown (see Santi *et al.* 2019, equation 6) that, when $x = x_0 + \Delta$, the probability distribution of the dependent variable changes as follows:

$$\pi^{(m)}_{(x_0+\Delta)} = \left[1 - \sum_{h=2}^{3} \left(1 - \mathrm{e}^{\Delta^\top \beta^{(h)}}\right) \pi^{(h)}_{(x_0)}\right]^{-1} \mathrm{e}^{\Delta^\top \beta^{(m)}} \pi^{(m)}_{(x_0)}\,, \tag{4}$$

with $m = 1, 2, 3$, where $\Delta \in \mathbb{R}^p$ is the change of covariates, and $\beta^{(1)} = 0 \in \mathbb{R}^p$ by construction (see Santi *et al.* 2019).

As Equation (4) shows, the probability distribution after the covariate change Δ only depends on the probability distribution before change $\pi^{(m)}_{(x_0)}$ $(m = 1, 2, 3)$, and on the coefficients of the trinomial regression, whereas there is not dependence on x_0 other than through $\pi^{(m)}_{(x_0)}$. Relation (4) is thus the theoretical basis which justifies the graphical method proposed in Santi *et al.* (2019), as it allows one to represent and analyse the regression coefficients $\beta^{(2)}, \beta^{(3)}$ over the (2-dimensional) simplex S, instead of the (k-dimensional) space of regressors $\mathcal{X}$.

In the following, an example of the method is provided in order to illustrate some of the capabilities of the package **plot3logit**, which are discussed in depth in the next sections of the paper.

A trinomial regression is fitted on self-reported votes for US presidential elections in 2016. Data are provided in Democracy Fund Voter Study Group (2017), where a broad and detailed questionnaire was administered to a sample consisting of 8000 people. In this paper, a dataset where only some information collected by Democracy Fund Voter Study Group (2017) is used. The dataset is made available through the package **plot3logit** under the name `USvote2016`.

In the following we consider a trinomial logit regression which models the self-reported vote (which may take values "Trump", "Clinton", and "Other") over some voters' characteristics (education level, gender, race, and decade when the voter was born). Here there are the R commands for fitting the model through the package **nnet**:

```
library("nnet")
data("USvote2016", package = "plot3logit")
modVote <- multinom(vote ~ educ + gender + race + birthyr,
  data = droplevels(USvote2016), trace = FALSE)
```

Table 1 shows point estimates and standard errors of regression coefficients.

Consider, for example, the coefficients on the regressor `genderFemale`. As the estimates in Table 1 show, both coefficients are negative and statistically different from zero, meaning that, ceteris paribus, female voters had a preference towards Hillary Clinton. Such a preference results in an increase (with respect to male voters with the same characteristics) of the probability to vote for Hillary Clinton to the detriment of Donald Trump and all other candidates. What is hard to assess is the actual effect of gender on the probability distribution of voter's choice, Figure 2a helps in that by representing the effect of covariate `genderFemale` through a vector field over a ternary diagram.

The direction of arrows in Figure 2a is consistent with the conclusion outlined before, although the diagram shows also that the direction is not constant over the simplex. On the other hand, arrow lengths enable to assess the magnitude of the effect, which is not constant and cannot be directly appraised from estimates in Table 1.

Figure 2b includes also the 95% confidence regions in order to assess also the degree of uncertainty of the estimates and how uncertainty on regression parameters determines the uncertainty on the effects (note how shapes and sizes of confidence regions changes over the simplex).

Confidence regions are particularly useful when the effect of a covariate change is analysed for some specific profiles (see Figure 3), or when multiple effects are compared with respect to a single (common) profile, as in Figure 4.

Figure 3 shows the effects of gender on five voter profiles distinguished only by the racial/ethnic group they belong to. The graph shows how the magnitude of the gender effect changes amongst different groups.

Figure 4 shows the effects of covariates on race with respect to a white voter having the same probability of choosing Clinton (33.3%), Trump (33.3%) or other candidates (33.3%). Ternary diagram enables the reader to assess the direction and the magnitude of differences of voters' preferences by voters' race as well as the degree of uncertainty of the estimates by means of 95% confidence regions.

In the rest of the paper it is illustrated and discussed how diagrams like those in Figure 2, 3, and 4 can be drawn by means of package **plot3logit**.

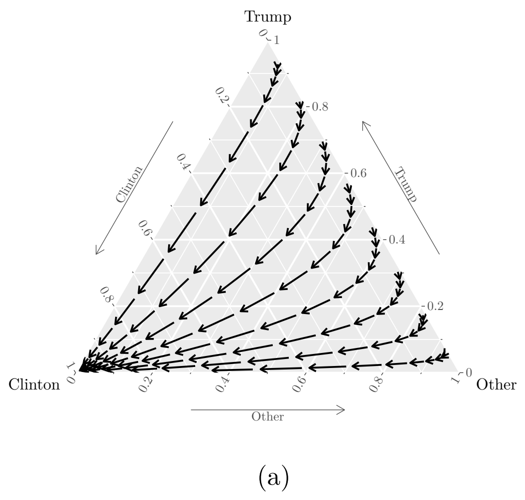


Figure 2: Vector field on the effect of gender (covariate `genderFemale`) on the probability distribution of voter's choice (Figure 2a). Figure 2b shows the same vector field with 95% confidence regions. Coefficient estimates are reported in Table 1.

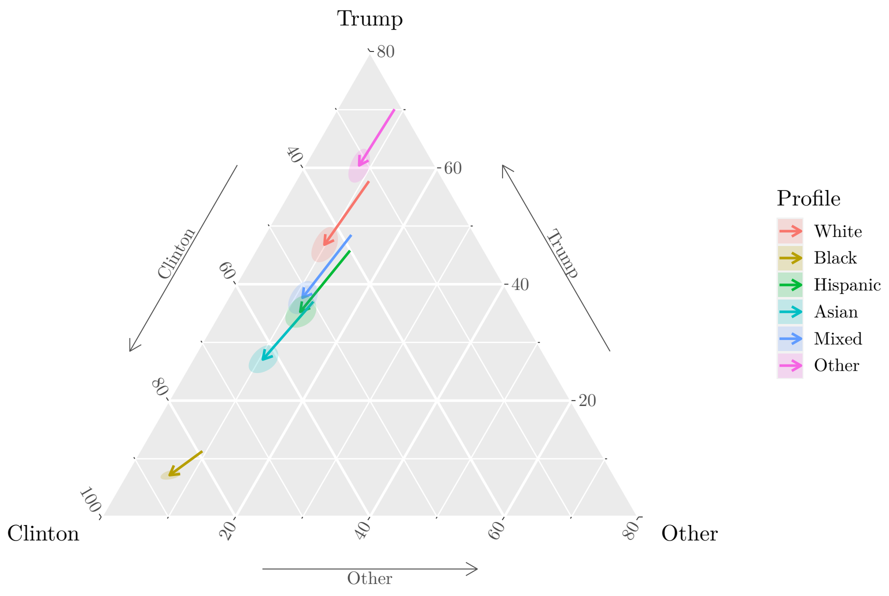


Figure 3: Effect of gender on the probability distribution of voter's choice born in the Seventies and graduated at the high school, distinguished by racial or ethnic group. 95% confidence regions are drawn. Coefficient estimates are reported in Table 1. Note that only a portion of the simplex is represented in this graph.

Table 1: Trinomial logit regression of voter choice (reference level is "Clinton") based on 7590 observations of the Democracy Fund Voter Study Group (2017). All regressors are qualitative: education (ref.: "no high school"), gender (ref.: "male"), race/ethnicity (ref.: "white"), birth year (ref.: $[1920, 1940)$). Table shows point estimates followed by significance symbols (* for p-values between 0.05 and 0.1; ** for p-values between 0.01 and 0.05; *** for p-values smaller than 0.01). Standard errors are reported in parenthesis.

Regressor	*Levels*			
	Trump		Others	
Constant	1.293***	(0.224)	−1.712***	(0.445)
Education				
High school grad.	−0.379*	(0.214)	−0.499	(0.393)
Some college	−0.739***	(0.215)	−0.670*	(0.391)
2-year college	−0.640***	(0.223)	−0.602	(0.405)
4-year college	−0.934***	(0.215)	−0.542	(0.387)
Post-grad	−1.252***	(0.217)	−0.733*	(0.391)
Gender (female)	−0.536***	(0.051)	−0.440***	(0.092)
Race/ethnicity				
Black	−2.560***	(0.150)	−1.087***	(0.191)
Hispanic	−0.479***	(0.114)	−0.004	(0.184)
Asian	−0.912***	(0.246)	−0.296	(0.321)
Mixed	−0.383**	(0.180)	−0.042	(0.281)
Other	0.579***	(0.162)	0.141	(0.303)
Birth year (decade)				
$[1940, 1950)$	−0.058	(0.109)	0.117	(0.291)
$[1950, 1960)$	0.012	(0.104)	0.638**	(0.270)
$[1960, 1970)$	0.081	(0.107)	1.058***	(0.268)
$[1970, 1980)$	−0.302**	(0.123)	1.166***	(0.279)
$[1980, 2000)$	−1.001***	(0.141)	1.365***	(0.280)

3. Features

In summary, the package **plot3logit** can:

- read the trinomial logit models fitted by functions `clm` and `clm2` of package `ordinal` (Christensen 2019), function `multinom` of package **nnet** (Venables and Ripley 2002), function `polr` of package **MASS** (Venables and Ripley 2002), function `mlogit` of package **mlogit** (Croissant 2020),[2] and function `vgam` and `vglm` of package **VGAM** (Yee 2010). Moreover, estimates obtained from other packages or software can be passed explicitly through a properly structured list and processed by **plot3logit**;
- handle several sintaxes for expressing the covariate changes and represent them graphically. The current implementation enables the covariate changes to be passed to func-

[2]The current version of **plot3logit** can only read and represent the results of pure trinomial models returned by `mlogit()`.

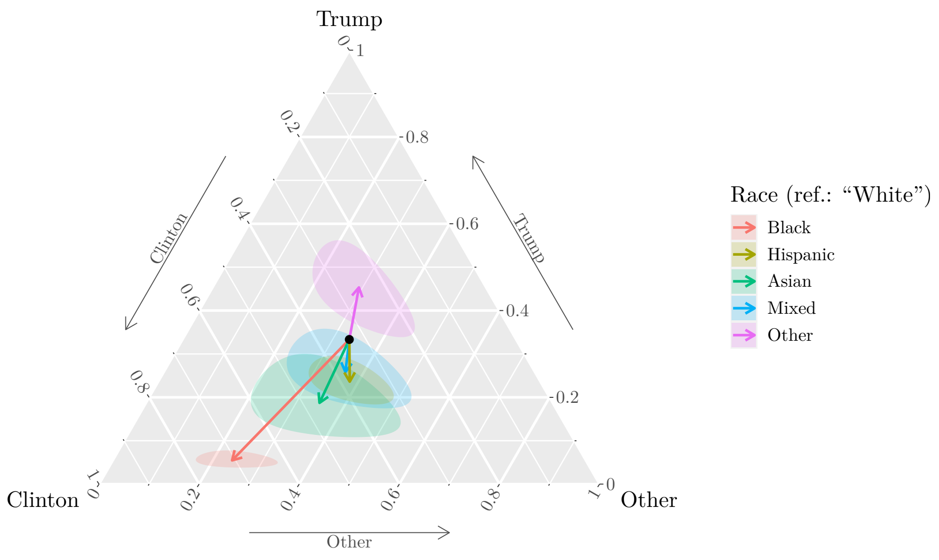


Figure 4: Effect of race on the probability distribution of voter's choice with respect to a white voter having the same probability of choosing Clinton (33.3%), Trump (33.3%) or other candidates (33.3%). 95% confidence regions are drawn. Coefficient estimates are reported in Table 1.

tion `field3logit` either as numeric vectors, named numeric vectors, or mathematical expressions (through R code);

- work both under standard R graphics paradigm through package **Ternary** (Smith 2017), and under the paradigm of the grammar of graphics (Wilkinson 2005) through packages **ggtern** (Hamilton and Ferry 2018) and **ggplot2** (Wickham 2016). Moreover, methods `as.data.frame`, `as_tibble`, `fortify` and `tidy` enable the graphical data to be easily exported in a standardised format which may be used for drawing ternary fields through other packages or software;
- fully customise any feature of ternary fields, including position, number, and alignment of arrows;
- draw and handle several fields over the same plot, so that the effects of different changes of covariates (possibly) with respect to different profiles can be compared;
- compute and draw confidence regions for each effect of covariate change, so that uncertainty about estimates of effects can be shown visually;
- quickly compute and draw ternary fields and confidence regions under standard settings through several wrappers which make the code shorter and easier to write and read.

4. Computation and representation of vector fields

4.1. Computation of vector fields

Function `field3logit` computes the vector field, which represents the effects of covariate changes on the probability distribution of the dependent variable, according to a fitted model. It follows that the two most important arguments of `field3logit` are the parameter estimates of the model (argument `model`) and the change of covariate values (argument `delta`). Further arguments (`p0`, `nstreams`, `narrows`, `edge`) define other characteristics of the vector field. In this section it is illustrated how all these arguments can be set.

Read model estimates

Model estimates are passed to `field3logit` by means of argument `model`; when the trinomial logit model is fitted through any of these functions:

- `clm`, `clm2` of package **ordinal** (Christensen 2019)
- `multinom` of package **nnet** (Venables and Ripley 2002)
- `polr` of package **MASS** (Venables and Ripley 2002)
- `mlogit` of package **mlogit** (Croissant 2020)
- `vgam`, `vglm` of package **VGAM** (Yee 2010)

`field3logit` internally invokes the generic `extract3logit` which automatically extracts all relevant information from the objects returned by those functions.[3]

On the other hand, if estimates are not available as output of the previous functions, they may be passed to argument `model` as a named list consisting of the following components (the order is not relevant):

- `B`: matrix of regression coefficients. It should be a numeric matrix (or any coercible object) with two columns if the model is cardinal, with only one column if the model is ordinal. The number of rows should be equal to the number of covariates and the names of covariates should be added as row names. The intercepts should be included only in case of categorical models, whereas column names, if provided, are ignored.
- `alpha`: intercepts of ordinal models. It should be a numerical vector of length two if the the model is ordinal, otherwise this component should be either set to `NULL` or missing.
- `levels`: vector of possible values of the dependent variable. It should be a character vector of length three, whose first element is interpreted as the reference level, whereas the second and the third elements are associated to the first and second columns of matrix `B` respectively.
- `vcovB`: covariance matrix of regression coefficients. This component is required only if the computation of confidence regions is needed (see Section 5); it should be a numeric matrix (or any coercible object) where the number of rows and columns equals the number of elements of `B`. Rows and columns should be ordered according to the labels of the dependent variable (slower index), and then to the covariates (faster index).

[3]The vignette "Overview" illustrates some examples where a model is fitted by means of each command listed above, and the result is passed to `field3logit`. Type `vignette("plot3logit-overview")` to browse it.

Here it is an example on how the list should be defined in case of a categorical trinomial logit regression with four covariates (a constant term, X_1, X_2 and X_3) and where the dependent variable takes values "Class A" (reference level), "Class B", "Class C":

```
fittedModel <- list(B = matrix(c(2, 0.3, -0.2, 0.2, 1, 0.1, -0.4, -0.3),
  ncol = 2, dimnames = list(c("(Intercept)", "X1", "X2", "X3"))),
  levels = c("Class A", "Class B", "Class C"))
```

The list `fittedModel` may be passed directly to `field3logit` as argument `model`, anyway, if `fittedModel` is passed to `extract3logit`, an object of class '`model3logit`' is returned:

```
library("plot3logit")
extract3logit(fittedModel)

##
##  Object of class "model3logit"
## -------------------------------------------
## Model has been read from : list
## Type of model            : categorical
## Possible outcomes        : Class A; Class B; Class C
## Reference level          : Class A
## Matrix of coefficients   : 4 x 2
## Covariance matrix        : not available
```

and can then be passed to `field3logit` as argument `model`.

When invoked, `extract3logit` creates a '`model3logit`' object and checks the consistency of the information provided, anyway, there is no advantage in calling `extract3logit` explicitly, as `field3logit` does it in any case on argument `model`.

It is also possible to define new S3 methods for generic `extract3logit`. The code of the new method should collect the information about the fitted model and define a list consisting of the components described above, to which should be added also the following:

- `readfrom`: character with information about the function that returned the estimates in the form `package::function` (for example `nnet::multinom`, `MASS::polr`, ...).

Once the list has been generated, it should be passed to function `extract3logit.default`, which creates a (complete and standardised) '`model3logit`' object and checks on completeness and consistency of the information provided. The output of `extract3logit.default` should then be returned as the output of the new method.

Specification of covariate changes

The change of regressor values may be expressed in three different ways.

Firstly, it may be passed to `field3logit` explicitly as a numeric vector where each component specifies the change of the corresponding regressor. The vector is thus the same denoted by Δ in Equation (4).

Consider, for example, the effect of the dummy variable `genderFemale`, which is the seventh covariate (including the constant term) of the model stored in `modVote`. The vector Δ should be defined as follows:

```
Delta <- rep(0, 17)
Delta[7] <- 1
Delta

##  [1] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
```

then the `field3logit` function enables the vector field in Figure 2a to be computed as follows:

```
field3logit(model = modVote, delta = Delta)

##  Object of class "field3logit"
## -------------------------------
## Label                       : <empty>
## Possible outcomes           : Clinton; Trump; Other
## Reference level             : Clinton
## Type of model               : categorical
## Effect                      : 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
## Model has been read from : nnet::multinom
## Number of stream lines   : 8
## Number of arrows            : 106
## Covariance matrix           : available
## Confidence regions          : not available
```

As an alternative, the change of covariates can be passed to argument `delta` as a named numeric vector where only non-zero changes of covariates are specified:

```
field3logit(model = modVote, delta = c(genderFemale = 1, raceBlack = 1))

##  Object of class "field3logit"
## -------------------------------
## Label                       : <empty>
## Possible outcomes           : Clinton; Trump; Other
## Reference level             : Clinton
## Type of model               : categorical
## Effect                      : genderFemale + raceBlack
## Explicit effect             : 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
## Model has been read from : nnet::multinom
## Number of stream lines   : 8
## Number of arrows            : 23
## Covariance matrix           : available
## Confidence regions          : not available
```

Finally, the change of covariates can be passed to argument `delta` in the form of a character expression in R language. The expression is then evaluated using the covariate names and the implicit vector Δ is computed. For example, the vector field in Figure 2a has been generated through the following command:

```
field3logit(model = modVote, delta = "genderFemale")

##  Object of class "field3logit"
## -------------------------------
## Label                        : <empty>
## Possible outcomes            : Clinton; Trump; Other
## Reference level              : Clinton
## Type of model                : categorical
## Effect                       : genderFemale
## Explicit effect              : 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
## Model has been read from : nnet::multinom
## Number of stream lines       : 8
## Number of arrows             : 106
## Covariance matrix            : available
## Confidence regions           : not available
```

It is worth noting that attribute `Effect` of the ‘`multifield3logit`‘ object obtained from the former command coincides with attribute `Explicit effect` of the latter ‘`field3logit`‘ object.

The use of named numeric vectors and R code (passed as a `character`) for expressing changes of covariates makes the `field3logit` function easy to use, especially when changes are fractional or involve several covariates. Consider, for example, the following two equivalent commands based on the object `fittedModel` previously generated:

```
field3logit(model = fittedModel, delta = c(X1 = 0.5, X2 = -2, X3 = 1))
```

```
field3logit(model = fittedModel, delta = "0.5 * X1 + X3 - 2 * X2")

##  Object of class "field3logit"
## -------------------------------
## Label                        : <empty>
## Possible outcomes            : Class A; Class B; Class C
## Reference level              : Class A
## Type of model                : categorical
## Effect                       : 0.5 * X1 + X3 - 2 * X2
## Explicit effect              : 0 0.5 -2 1
## Model has been read from : list
## Number of stream lines       : 8
## Number of arrows             : 80
## Covariance matrix            : not available
## Confidence regions           : not available
```

The code is easy-to-read, easy-to-write, and does not depend on the order that covariates have in the formula of the fitted model, unlike what happens when the explicit vector of covariate changes is passed to `field3logit`.

In conclusion, if covariate names include some non-alphanumeric character or start with a number, both the syntax based on named vector and the syntax based on R expressions can still be used, provided that the name of the covariate is delimited by single backticks (ASCII decimal code: 96). Here it is an example:

```
field3logit(modVote, delta = "genderFemale + `birthyr[1940,1950)`")

##  Object of class "field3logit"
## -------------------------------
## Label                        : <empty>
## Possible outcomes            : Clinton; Trump; Other
## Reference level              : Clinton
## Type of model                : categorical
## Effect                       : genderFemale + `birthyr[1940,1950)`
## Explicit effect              : 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
## Model has been read from : nnet::multinom
## Number of stream lines   : 8
## Number of arrows             : 101
## Covariance matrix            : available
## Confidence regions           : not available
```

Set up the vector field

In addition to `model` and `delta`, arguments `p0`, `nstreams`, `narrows` and `edge` enable the user to define how many arrows the vector field should consist of, and where they should be placed within the simplex of the ternary plot.

Figure 5 shows four different variations (using package **Ternary** instead of **ggtern**, see Section 4.2) of the field drawn in Figure 2a, and the following is the R code that generated Figure 5:

```
ptsAB <- list(A = c(0.3, 0.4, 0.3), B = c(0.5, 0.1, 0.4))
par(mfrow = c(2, 2), cex = 0.5, mar = rep(0, 4))
# Top-left
plot(field3logit(modVote, "genderFemale", edge = 0.1))
# Top-right
plot(field3logit(modVote, "genderFemale", nstreams = 4))
# Bottom-left
plot(field3logit(modVote, "genderFemale", p0 = ptsAB))
TernaryPoints(ptsAB)
TernaryText(ptsAB, labels = names(ptsAB), pos = 1)
# Bottom-right
plot(field3logit(modVote, "genderFemale", p0 = ptsAB, narrows = 1))
TernaryPoints(ptsAB)
```

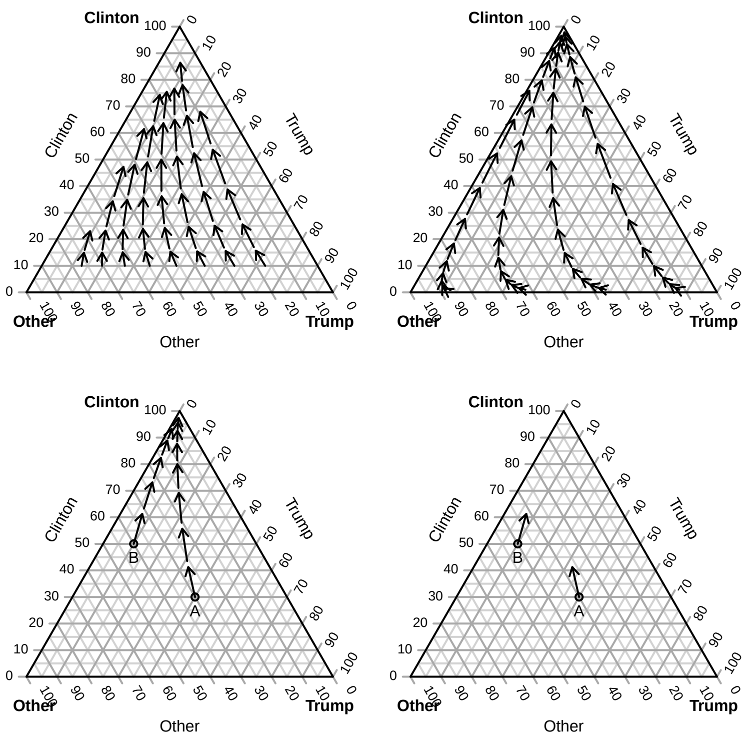


Figure 5: Vector fields on the effect of covariate `genderFemale` (see Figure 2a) generated by `field3logit` with different values of argument `edge` (top-left), `nstreams` (top-right), `p0` (bottom-left), `p0` and `narrows` (bottom-right).

```r
TernaryText(ptsAB, labels = names(ptsAB), pos = 1)
```

The top-left graph in Figure 5 shows the effect of argument `edge`, which sets the minimum distance between the starting and the ending point of each arrow of the field from the sides of the simplex. Vector field represented in Figure 2a has been generated using the default value of `edge` (0.01), whereas the top-left diagram in Figure 5 has been generated with `edge = 0.1`.

As diagram in Figure 2a clearly shows, arrows of ternary fields are arranged along some stream lines. Argument `nstreams` sets the number of stream lines to draw (default value is 8). `field3logit`, when it generates the field, automatically spreads the stream lines over the simplex in order to produce a field which is graphically optimal. Top-right diagram in Figure 5 shows the vector field on the effect of `genderFemale` (see Figure 2a) where `nstreams = 4`.

Argument `p0` enables one to set the starting points of the stream lines, in order to customise the behaviour of `field3logit`. Argument `p0` should be structured as a `list` whose components are `numeric` vectors of ternary coordinates (see object `ptsAB`, defined before). Bottom-left graph in Figure 5 shows an example where points $A = (0.3, 0.4, 0.3)$ and $B = (0.5, 0.1, 0.4)$ are set as starting points of two stream lines.

Finally, argument `narrows` sets the maximum number of arrows which should be computed for each stream line.[4] Bottom-right graph in Figure 5 shows the same field drawn in the bottom-left graph, but `narrows = 1`. Default value of `narrows` is `Inf`, so that arrows are added to a stream line until the edge set through argument `edge` has been reached.

4.2. Representation of vector fields

The vector fields computed by `field3logit` may be represented through functions provided by package **Ternary** (Smith 2017) which is based on standard R graphics, or functions of package **ggtern** (Hamilton and Ferry 2018), which extends package **ggplot2** (Wickham 2016) to ternary diagrams, and it is based on the programming paradigm referred to as "grammar of graphics" (see e.g., Wickham 2016; Wickham and Grolemund 2016) illustrated in Wilkinson (2005).

Plotting by means of package ***Ternary***

Two functions of **plot3logit** enable to draw vector fields of '`field3logit`' objects through package **Ternary**.

Function `TernaryField` takes a '`field3logit`' object as first argument and permits the vector field to be added to an existing ternary diagram created by function `TernaryPlot` of package **Ternary**. Both name and argument structure of `TernaryField` are consistent with other functions defined in package **Ternary** (such as `TernaryPoint`, `TernaryPolygon`, . . .).

The S3 method of generic `plot` takes a '`field3logit`' object as first argument and may either draw the ternary diagram from scratch (if argument `add` is set to `FALSE`), or add the vector field to an existing ternary plot (if `add = TRUE`), and in that case it basically works as a wrapper of `TernaryField`.

Some examples of the graphical rendering of vector fields drawn by means of package **Ternary** are shown in Figure 5.

Clearly, package **plot3logit** does not limit in any way the customisation of the graphs made available by methods of standard R graphics and by package **Ternary** (see manuals of **plot3logit** and **Ternary** for details).

Plotting by means of package ***ggtern***

Vector fields of '`field3logit`' objects can be drawn through package **ggtern** by means of the constructor `gg3logit`, the statistics `stat_field3logit`, `stat_conf3logit`, `stat_3logit`, and the S3 method of generic `autoplot` for class '`field3logit`'.

As opposed to **ggplot2** (and thus **ggtern**) philosophy, which only accepts '`data.frame`'s (or any other object of child classes, such as '`tibble`') as input for argument `data`, package **plot3logit** handles both '`data.frame`'s and '`field3logit`' objects.

This choice has been made in order to make the code simple, as if a '`field3logit`' object is passed to `gg3logit`, the conversion to a `data.frame` and the initialisation of aesthetic parameters (through the function `aes`) passed to argument `mapping` are carried out automatically.

On the contrary, if a `data.frame` (or any coercible object, including objects of child classes) is passed to argument `data` of `gg3logit`, the following aesthetics must be specified:

[4]If the stream line reaches the edge of the simplex, the actual number of arrows may be smaller than `narrows`.

- `x`, `y`, `z` are required by:
  - `stat_field3logit` as ternary coordinates of the starting points of the arrows;
  - `stat_conf3logit` as ternary coordinates of the points on the edge of confidence regions (see Section 5);
- `xend`, `yend`, `zend` are required by `stat_field3logit` as ternary coordinates of the ending points of the arrows;
- `group` is always required as it identifies the groups of the graphical objects (arrows and their confidence regions);
- `type` is always required as it specifies the type of graphical object (arrows or confidence regions) the row of the `data.frame` refers to;

Furthermore, the following variables of a fortified ‘`field3logit`‘ or a ‘`multifield3logit`‘ object (see next section)[5] may be useful for defining other standard aesthetics (such as `fill`, `colour`, ...):

- `label` identifies a field through a label, thus it is useful for distinguishing the fields in a ‘`multifield3logit`‘ object.
- `idarrow` identifies each group of graphical objects (arrows and their confidence regions) *within* every field. Unlike variable `group`, `idarrow` is not a global identifier of graphical objects.

‘`multifield3logit`‘ objects and confidence regions are illustrated in depth in the next sections of the paper.

As a first example on function `gg3logit`, it follows the R code for plotting the ternary diagram in Figure 2a:

```
fieldFemale <- field3logit(modVote, "genderFemale")
gg3logit(fieldFemale) + stat_field3logit()
```

According to the previous code, when a ‘`field3logit`‘ object is passed to `gg3logit`, the syntax is particularly short, as no aestetic has to be set. On the contrary, if a fortified ‘`field3logit`‘ object is passed to `gg3logit`, several aesthetics have to be initialised and the code is longer and less easy to read.

In order to compare the two syntaxes, consider the structure of the fortified object `fieldFemale`:[6]

```
set.seed(3109)
fortfieldFemale <- fortify(field3logit(modVote, "genderFemale"))
set.seed(NULL)
fortfieldFemale
```

[5]An object is referred to as *fortified* whenever it is processed by the method `fortify` (see e.g., Wickham 2016), and thus it is structured as a `data.frame` which contains the information available in the original object. By extension, an object may be referred to as *fortified* whenever it is processed through functions such as `as.data.frame`, `as_tibble`, `tidy`.

[6]The seed of random number generator is set (through `set.seed`) in order to make the results of `fortify` fully reproducible. If the seed is not set, the labels of columns `idarrow` and `group` may be different at each execution of `fortify`.

```
## # A tibble: 106 x 10
##    label idarrow group type  Clinton  Trump  Other Clinton_end
##    <fct> <fct>   <fct> <fct>   <dbl>  <dbl>  <dbl>       <dbl>
##  1 ""    C1A1    H007  arrow  0.0100 0.928  0.0623      0.0169
##  2 ""    C1A10   H087  arrow  0.736  0.222  0.0419      0.824
##  3 ""    C1A11   H103  arrow  0.838  0.134  0.0283      0.897
##  4 ""    C1A12   H020  arrow  0.905  0.0765 0.0182      0.941
##  5 ""    C1A13   H084  arrow  0.946  0.0423 0.0113      0.967
##  6 ""    C1A2    H023  arrow  0.0186 0.913  0.0687      0.0312
##  7 ""    C1A3    H031  arrow  0.0343 0.890  0.0752      0.0568
##  8 ""    C1A4    H018  arrow  0.0623 0.857  0.0812      0.101
##  9 ""    C1A5    H036  arrow  0.111  0.804  0.0854      0.174
## 10 ""    C1A6    H029  arrow  0.188  0.725  0.0864      0.282
## # i 96 more rows
## # i 2 more variables: Trump_end <dbl>, Other_end <dbl>
```

If `fortfieldFemale` is passed to `gg3logit`, the code for drawing the diagram in Figure 2a becomes considerably longer:

```
gg3logit(fortfieldFemale, aes(x = Clinton, y = Trump, z = Other,
  xend = Clinton_end, yend = Trump_end, zend = Other_end, group = group,
  type = type)) + stat_field3logit()
```

The simplicity of the former syntax is apparent, whereas the latter does not provide any practical advantage in terms of greater flexibility, notwithstanding the greater verbosity. This is the reason why the former syntax has been implemented, even though it deviates from orthodox **ggplot2** philosophy, that requires that only ‘`data.frame`‘ objects can be passed to argument `data`.

Plotting by means of other packages/software

Besides the integration with packages **Ternary** and **ggtern**, package **plot3logit** guarantees a full downstream compatibility with other R packages or other applications through the S3 methods of generics `as.data.frame`, `as_tibble` (package **tibble**, Müller and Wickham 2020), `fortify` (package **ggplot2**, Wickham 2016), and `tidy` (package **broom**, Robinson and Hayes 2020) for classes ‘`field3logit`‘ and ‘`multifield3logit`‘. All four methods are equivalent, except that `as.data.frame` returns a `data.frame`, whereas the others return a `tibble`.

The mentioned methods enable the graphical information (arrows, confidence regions and labels) of a ‘`field3logit`‘ or a ‘`multifield3logit`‘ object to be exported in a standardised table which can be read by any other R package or can be stored on disk through standard R commands (such as `write.csv`, for example) and then be read by applications other than R.

4.3. Handling multiple fields

When the results of a multinomial regression are analysed, the comparison between the effects of various changes in covariate values may be of interest. Figure 4 shows how this kind of comparisons may be carried out by means of ternary plots.

Each arrow in Figure 4 is associated to a distinct change in the value of one covariate, thus, diagram in Figure 4 may be interpreted as a superimposition of five vector fields consisting of a single arrow each, and having the same profile as a reference point. This is actually the way Figure 4 has been generated.

`multifield3logit` is a S3 class which enables ‘`field3logit`‘ objects to be combined, handled, and represented jointly. Besides the standard constructor `multifield3logit`, objects of class ‘`multifield3logit`‘ can be created and combined through the operator `"+"`.[7]

The following code shows how covariate effects of dummies `raceBlack` and `raceHispanic` are combined in a ‘`multifield3logit`‘ object, when a a single reference profile such that $(\pi^{(1)}, \pi^{(2)}, \pi^{(3)}) = (1/3,\ 1/3,\ 1/3)$ is considered:

```
refprofile <- list(c(1/3, 1/3, 1/3))

fieldBlack <- field3logit(model = modVote, delta = "raceBlack",
  label = "Black", p0 = refprofile, narrows = 1)

fieldHispanic <- field3logit(model = modVote, delta = "raceHispanic",
  label = "Hispanic", p0 = refprofile, narrows = 1)

mfieldrace <- fieldBlack + fieldHispanic
mfieldrace

##  Object of class "multifield3logit"
## ------------------------------------
## Number of fields         : 2
## Labels
##   1. Black     (dX: raceBlack)
##   2. Hispanic  (dX: raceHispanic)
```

The previous example permits also the usage of argument `label` to be clarified, as it is used by graphical functions for distinguishing and labelling the elements of a ‘`multifield3logit`‘ object according to the ‘`field3logit`‘ objects they belong to. This is the reason why, if a single ‘`field3logit`‘ object is defined and used, there is in general no need for initialising the argument `label`, whose default value is an empty character (`""`).

The operator `"+"` permits several (two or more) ‘`field3logit`‘ objects to be combined at once, and ‘`field3logit`‘ objects to be included into an existing ‘`multifield3logit`‘ object:[8]

[7]The package makes available also the S3 methods of generics `"["` and `"[<-"` of class ‘`multifield3logit`‘ for extracting and replacing the ‘`field3logit`‘ objects the ‘`multifield3logit`‘ objects consist of. — See the help of **plot3logit** for details and further information.

[8]Technically, the operator `"+"` has been implemented as a S3 method of class ‘`Hfield3logit`‘ to which both ‘`multifield3logit`‘ and ‘`field3logit`‘ objects belong. This permits a correct method dispatch for generic `"+"`, which is not possible if it is invoked for two objects of different classes (‘`field3logit`‘ and ‘`multifield3logit`‘). This is the only reason why class ‘`Hfield3logit`‘ has been defined.

```
fieldAsian <- field3logit(model = modVote, delta = "raceAsian",
  label = "Asian", p0 = refprofile, narrows = 1)

mfieldrace <- mfieldrace + fieldAsian
mfieldrace

##  Object of class "multifield3logit"
## ------------------------------------
## Number of fields          : 3
## Labels
##   1. Black     (dX: raceBlack)
##   2. Hispanic  (dX: raceHispanic)
##   3. Asian     (dX: raceAsian)
```

When several vector fields have to be generated and combined in a ‘**`multifield3logit`**‘ object, the syntax showed above is unnecessary long and in some cases pleonastic. For this reason, it is possible to rely on function **`field3logit`** by means of the syntax described below. Assume that we are interested in comparing the effects of all dummies on race in the model on United States (US) elections. Let us thus define a list whose elements are lists where only varying arguments to be passed to function **`field3logit`** are specified as named components:

```
race_effects <- list(
  list(delta = "raceBlack", label = "Black"),
  list(delta = "raceHispanic", label = "Hispanic"),
  list(delta = "raceAsian", label = "Asian"),
  list(delta = "raceMixed", label = "Mixed"),
  list(delta = "raceOther", label = "Other")
)
```

If **`race_effects`** is passed to argument **`delta`** of **`field3logit`**, in this way:

```
mfieldrace <- field3logit(model = modVote, delta = race_effects,
  p0 = refprofile, narrows = 1)

mfieldrace

##  Object of class "multifield3logit"
## ------------------------------------
## Number of fields          : 5
## Labels
##   1. Black     (dX: raceBlack)
##   2. Hispanic  (dX: raceHispanic)
##   3. Asian     (dX: raceAsian)
##   4. Mixed     (dX: raceMixed)
##   5. Other     (dX: raceOther)
```

the function `field3logit` is run once for every element of `race_effects`, and the set of ‘`field3logit`‘ objects are combined into a single object of class ‘`multifield3logit`‘. When `field3logit` is applied to each element of `race_effects`, the arguments specified in the parent call of `field3logit` are used as default values, which are then overwritten by those specified in each element of `race_effects`.

The expedient just described enables the ‘`multifield3logit`‘ objects to be generated through a short and efficient syntax even if several ‘`field3logit`‘ objects are involved.

The syntax just described, however, can be simplified further when the fields to be generated involve dummy variables of the same qualitative covariate (encoded as `factor`). In that case, argument `delta` should indicate the name of the original covariate between delimiters `<<` and `>>`, and `field3logit` will create a ‘`multifield3logit`‘ object where each field corresponds to the effect of each dummy variable.

The following code shows how the previous commands can be simplified further:

```
field3logit(model = modVote, delta = "<<race>>", p0 = refprofile,
  narrows = 1)

##  Object of class "multifield3logit"
## ------------------------------------
## Number of fields          : 5
## Labels
##   1. Black     (dX: `raceBlack`)
##   2. Hispanic  (dX: `raceHispanic`)
##   3. Asian     (dX: `raceAsian`)
##   4. Mixed     (dX: `raceMixed`)
##   5. Other     (dX: `raceOther`)
```

If more than one regressor is included between delimiters `<<`, `>>`, all combinations between dummies are generated, and if only some of the fields are actually needed, the ‘`multifield3logit`‘ object can be subsetted through the S3 method `"["`.

Finally, a peculiar behaviour of argument `label` is worth of being reported. When a ‘`multifield3logit`‘ object is generated by `field3logit`, argument `label` works as a prefix of the labels of each vector field. It follows that, if no label is set within argument `delta`, all labels can be set directly through argument `label`:

```
field3logit(model = modVote, delta = c("raceBlack", "raceAsian"),
  label = c("BLACK", "ASIAN"))

##  Object of class "multifield3logit"
## ------------------------------------
## Number of fields          : 2
## Labels
##   1. BLACK  (dX: raceBlack)
##   2. ASIAN  (dX: raceAsian)
```

On the other hand, when argument `delta` uses delimiters `<<`, `>>`, argument `label` can easily help in automatic generation of meaningful labels:

```
mfdecade <- field3logit(modVote, "<<birthyr>>", label = "Born in ")
mfdecade

##  Object of class "multifield3logit"
## -------------------------------------------
## Number of fields         : 5
## Labels
##   1. Born in [1940,1950)  (dX: `birthyr[1940,1950)`)
##   2. Born in [1950,1960)  (dX: `birthyr[1950,1960)`)
##   3. Born in [1960,1970)  (dX: `birthyr[1960,1970)`)
##   4. Born in [1970,1980)  (dX: `birthyr[1970,1980)`)
##   5. Born in [1980,2000)  (dX: `birthyr[1980,2000)`)
```

In any case, if some labels need to be redefined, the S3 method `"labels<-"` will do the job:

```
labels(mfdecade)

## [1] "Born in [1940,1950)" "Born in [1950,1960)" "Born in [1960,1970)"
## [4] "Born in [1970,1980)" "Born in [1980,2000)"

labels(mfdecade) <- c("Fourties", "Fifties", "Sixties", "Seventies",
  "Eighties and Nineties")
mfdecade

##  Object of class "multifield3logit"
## -------------------------------------------
## Number of fields         : 5
## Labels
##   1. Fourties               (dX: `birthyr[1940,1950)`)
##   2. Fifties                (dX: `birthyr[1950,1960)`)
##   3. Sixties                (dX: `birthyr[1960,1970)`)
##   4. Seventies              (dX: `birthyr[1970,1980)`)
##   5. Eighties and Nineties  (dX: `birthyr[1980,2000)`)
```

The ways '`multifield3logit`' objects are graphically represented are similar to those of '`field3logit`' objects, thus S3 method of generics `plot` draws a '`multifield3logit`' object through package **Ternary**, whereas functions `autoplot`, `gg3logit`, `stat_field3logit` make the ternary diagrams through the package **ggtern**. The only remarkable difference in case of function `gg3logit` and its statistics is in the variable `label` which enables various aesthetics to be set accordingly to the vector field of the '`multifield3logit`' object.

For example, the following code generates the diagram of Figure 4 (without confidence regions):

```
gg3logit(mfieldrace, aes(colour = label)) + stat_field3logit() +
  labs(colour = "Race (ref.: White)")
```

5. Confidence regions

Confidence regions of the effects of covariates on the probability distribution of the dependent variable are not considered in Santi *et al.* (2019), however they greatly enrich the information a ternary diagram can provide, and help the interpretation of regression results. For these reasons, they have been implemented in package **plot3logit**. Section 5.1 illustrates how they are mathematically derived and how they can be computed through package **plot3logit**, whereas Section 5.2 shows how they can be represented graphically.

5.1. Computation

Consider a probability distribution π_0 over the simplex S defined in Equation (1). The confidence region $\mathcal{R} \subseteq S$ for a change $\Delta \in \mathbb{R}^p$ in the values of covariates may be defined as it follows:

$$\mathbb{P}((\pi_0 + \hat{\delta}^{(\pi)}) \in \mathcal{R}) = 1 - \alpha \tag{5}$$

where $\hat{\delta}^{(\pi)}$ is the point estimator of the change of the probability distribution π_0.

According to Equation (2), the link function $g\colon S \to \mathbb{R}^2$ of the tinomial logit model and its inverse $g^{\leftarrow}\colon \mathbb{R}^2 \to S$ may be defined as:

$$g(\pi) = g([\pi_1, \pi_2, \pi_3]^\top) = \left[\ln\frac{\pi_2}{\pi_1}\,, \quad \ln\frac{\pi_3}{\pi_1}\right]^\top,$$

$$g^{\leftarrow}(\eta) = g^{\leftarrow}([\eta_2, \eta_3]^\top) = \left[\frac{1}{1+\mathrm{e}^{\eta_2}+\mathrm{e}^{\eta_3}}\,, \quad \frac{\mathrm{e}^{\eta_2}}{1+\mathrm{e}^{\eta_2}+\mathrm{e}^{\eta_3}}\,, \quad \frac{\mathrm{e}^{\eta_3}}{1+\mathrm{e}^{\eta_2}+\mathrm{e}^{\eta_3}}\right]^\top.$$

Bijectivity of g enables confidence region (5) to be restated over the natural parametric space:

$$\mathbb{P}((g(\pi_0) + \hat{\delta}) \in g^{\leftarrow}(\mathcal{R})) = 1 - \alpha\,, \tag{6}$$

where $\hat{\delta}$ is the point estimator of the change of natural parameters, and $g^{\leftarrow}(\mathcal{R}) \overset{\text{def}}{=} \{g^{\leftarrow}(r)\colon r \in \mathcal{R}\}$.

Let $B = [\beta^{(2)}, \beta^{(3)}] \in \mathbb{R}^{k\times 2}$ be the matrix of regression coefficients defined in (3), and let $\hat{B} \in \mathbb{R}^{k\times 2}$ be the point estimate of B. The effect of a change $\Delta \in \mathbb{R}^k$ of covariate vector $x \in \mathbb{R}^k$ on natural parameters $\eta = [\eta_2, \eta_3]$ can then be expressed through the vector $\delta \in \mathbb{R}^2$ as follows:

$$\delta = B^\top \Delta = (I_2 \otimes \Delta)^\top \operatorname{vec}(B)\,,$$

where I_2 is the identity matrix of order 2, $\otimes$ is the Kronecker product, and $\operatorname{vec}(B) \in \mathbb{R}^{2k}$ is the vectorisation of B.

If the point estimate of the variance-covariance matrix of $\operatorname{vec}(B)$ is $\hat{\Xi}$, the variance-covariance matrix of $\hat{\delta} = \hat{B}^\top \Delta$ is:

$$(I_2 \otimes \Delta)^\top\, \hat{\Xi}\, (I_2 \otimes \Delta)\,,$$

it follows that a $(1-\alpha)$-confidence region for δ can be obtained from the following condition on the Wald statistics (Lee, Nyangoma, and Seber 2002; Severini 2000):

$$(\delta - \hat{\delta})^\top [(I_2 \otimes \Delta)^\top \hat{\Xi} (I_2 \otimes \Delta)]^{-1} (\delta - \hat{\delta}) \leq \chi^2_2(1-\alpha)\,, \tag{7}$$

$\chi^2_2(1-\alpha)$ being the quantile function of the probability distribution χ^2_2 (see also Wooldridge 2010).

The confidence region of δ can then be mapped to the simplex S with respect to the reference probability distribution π_0 by means of the inverse link function $g^{\leftarrow}$. Hence, the confidence region $\mathcal{R}$ can be found as it follows:

$$\mathcal{R} = \{g^{\leftarrow}(g(\pi_0) + \delta) \colon \delta \text{ satisfies (7)}\}\,. \tag{8}$$

Clearly, the edge of the confidence region (8) can be found by considering those points associated to the values δ which satisfy condition (7) exactly (i.e., with equality instead of inequality).

The package **plot3logit** enables confidence regions to be computed in two ways, by means of function `field3logit` or through function `add_confregions`.

Function `field3logit` computes the confidence regions for all the arrows in the field according to the value passed to argument `conf`. If `conf` is not set or if it is set to `NA` (default value), confidence regions are not computed. Clearly, the computation is possible only if the variance-covariance matrix of the estimates is available. When computed, confidence regions are part of the '`field3logit`' object returned by `field3logit`.

Function `add_confregions` enables confidence regions to be computed on a '`field3logit`' or a ' `multifield3logit`' object, if not present. Otherwise, it may be used to update confidence regions of a '`field3logit`' or a '`multifield3logit`' object according to a new confidence level. Since `add_confregions` returns an object of class '`field3logit`' (or '`multifield3logit`') equipped with confidence regions, it can be run as follows:

```
mfieldrace <- add_confregions(mfieldrace)
```

By default, argument `conf` is set to 0.95, thus 95% confidence regions are computed, if not differently specified. As in case of `field3logit`, confidence regions can be computed only if variance-covariance matrix of coefficient estimates is available.

Both function `field3logit` and `add_confregions` have an argument named `npoints` which allows the user to set the number of points used for drawing the edges of confidence regions.

5.2. Representation

Confidence regions can be drawn both through package **Ternary** and **ggtern**.

In the former case, the S3 method of generic `plot` works for both '`field3logit`' and '`multifield3logit`' objects, and it creates a new ternary plot if argument `add` is set to `FALSE` (default value), whilst adds a vector field(s) to an existing ternary plot if `add` is set to `TRUE`. As in the case of vector fields, confidence regions of a '`field3logit`' object can be drawn through the function `TernaryField` (see the help for details).

If package **ggtern** is used, confidence regions of ‘`field3logit`‘ and ‘`multifield3logit`‘ objects can be drawn through the statistic `stat_conf3logit`, which works analogously to `stat_field3logit`.

The following code generates the diagram of Figure 4:

```
gg3logit(mfieldrace) + stat_field3logit(aes(colour = label)) +
  stat_conf3logit(aes(fill = label)) +
  labs(colour = "Race (ref.: White)", fill = "Race (ref.: White)")
```

whereas the following code generates the diagram of Figure 3 from scratch:

```
library("tidyverse")

tibble(race = levels(USvote2016$race), educ = "High school grad.",
    gender = "Male", birthyr = "[1970,1980)"
  ) %>%
  mutate(delta = "genderFemale", label = race) %>%
  group_by(delta, label) %>%
  nest() %>%
  mutate(p0 = map(data, ~list(predict(modVote, .x, type = "probs")))) %>%
  select(-data) %>%
  transpose -> gender_by_race

mfieldGbyR <- field3logit(modVote, gender_by_race, narrows = 1,
  conf = 0.95)

gg3logit(mfieldGbyR) + stat_field3logit(aes(colour = label)) +
  stat_conf3logit(aes(fill = label)) + tern_limits(T = 0.8, R = 0.8) +
  labs(colour = "Profile", fill = "Profile")
```

6. Wrappers

Package **plot3logit** includes two wrappers which aims at simplifying the syntax when a [9] is drawn through package **ggtern**.

The first wrapper is `stat_3logit` which is a wrapper for:

```
stat_field3logit() + stat_conf3logit()
```

`stat_3logit` has arguments `mapping_field` and `mapping_conf` which enables one to specify the aesthetic mappings for `stat_field3logit` and `stat_conf3logit` respectively, whereas arguments `params_field` and `params_conf` allow one to set the graphical parameters of the two layers.

[9]The seed of random number generator is set (through `set.seed`) in order to make the results of `fortify` fully reproducible. If the seed is not set, the labels of columns `idarrow` and `group` will be different at each execution of `fortify`.

The second wrapper is `autoplot` which is a wrapper for:

```
gg3logit() + stat_3logit()
```

and thus for

```
gg3logit() + stat_field3logit() + stat_conf3logit()
```

Just like in case of `stat_3logit`, `autoplot` has arguments `mapping_field`, `mapping_conf`, `params_field`, `params_conf` with the same role described before.

In order to provide an example, the code for drawing the graph in Figure 4 is reported both with and without wrappers.

The following command:

```
gg3logit(mfieldrace) + stat_field3logit(aes(colour = label)) +
  stat_conf3logit(aes(fill = label))
```

is then equivalent to the following:

```
gg3logit(mfieldrace) + stat_3logit(aes(colour = label), aes(fill = label))
```

which, in turn, is equivalent to this:

```
autoplot(mfieldrace, mapping_field = aes(colour = label),
  mapping_conf = aes(fill = label))
```

7. Conclusions

Package **plot3logit** implements the ternary diagrams proposed in Santi *et al.* (2019) for interpreting the coefficient estimates of a trinomial logit regression. The package has been implemented so as to make it easy to use without losing flexibility. Upstream and downstream compatibility of the package enables the user to read model estimates whatever is the package/software that computed them, whereas the implementation of graphical functions based both on standard R graphics and **ggplot2**-based graphics, as well as the export methods (`as.data.frame`, `as_tibble`, `fortify`, `tidy`), provides several graphical tools for drawing the random fields, but does not prevent the user to adopt other graphical packages, or applications other than R.

Acknowledgments

We thank the editor, Yves Croissant, and Jonas Schöley for their valuable comments and suggestions, that significantly improved both the package **plot3logit** and the article published on the *Journal of Statistical Software* (Santi, Dickson, Espa, and Giuliani 2022).

References

Agresti A (2013). *Categorical Data Analysis*. 3 edition. John Wiley & Sons.

Armstrong D (2020). ***DAMisc**: Dave Armstrong's Miscellaneous Functions.* R package version 1.6.1, URL https://CRAN.R-project.org/package=DAMisc.

Bancroft WD (1897). "A triangular diagram." *Journal of Physical Chemistry*, **1**, 403–10.

Christensen RHB (2019). *ordinal: Regression Models for Ordinal Data.* R package version 2019.12-10, URL https://CRAN.R-project.org/package=ordinal.

Croissant Y (2020). *mlogit: Multinomial Logit Models.* R package version 1.1-0, URL https://CRAN.R-project.org/package=mlogit.

Democracy Fund Voter Study Group (2017). "Views of the electorate research survey, December 2016." URL https://www.voterstudygroup.org.

Fox J (2003). "Effect Displays in R for Generalised Linear Models." *Journal of Statistical Software*, **8**(15), 1–27. URL https://www.jstatsoft.org/article/view/v008i15.

Fox J, Hong J (2009). "Effect Displays in R for Multinomial and Proportional-Odds Logit Models: Extensions to the **effects** Package." *Journal of Statistical Software*, **32**(1), 1–24. URL https://www.jstatsoft.org/article/view/v032i01.

Fox J, Weisberg S (2018). "Visualizing Fit and Lack of Fit in Complex Regression Models with Predictor Effect Plots and Partial Residuals." *Journal of Statistical Software*, **87**(9), 1–27. doi:10.18637/jss.v087.i09. URL https://www.jstatsoft.org/article/view/v087i09.

Fox J, Weisberg S (2019). *An R Companion to Applied Regression.* 3rd edition. Sage, Thousand Oaks CA.

Hamilton NE, Ferry M (2018). "ggtern: Ternary Diagrams Using ggplot2." *Journal of Statistical Software, Code Snippets*, **87**(3), 1–17. doi:10.18637/jss.v087.c03.

Howarth RJ (1996). "Sources for a History of the Ternary Diagram." *The British Journal for the History of Science*, **29**(3), 337–356. doi:10.1017/S000708740003449X.

Johnson NL, Kemp AW, Kotz S (2005). *Univariate Discrete Distributions.* 3 edition. John Wiley & Sons.

Lee AJ, Nyangoma SO, Seber GAF (2002). "Confidence regions for multinomial parameters." *Computational Statistics & Data Analysis*, **39**, 329–342.

Lehmann EL, Casella G (1998). *Theory of Point Estimation.* 2 edition. Springer.

Lenth RV (2016). "Least-Squares Means: The R Package **lsmeans**." *Journal of Statistical Software*, **69**(1), 1–33. doi:10.18637/jss.v069.i01.

Lenth RV (2020). ***emmeans**: Estimated Marginal Means, aka Least-Squares Means.* R package version 1.5.0, URL https://CRAN.R-project.org/package=emmeans.

Müller K, Wickham H (2020). *tibble: Simple Data Frames.* R package version 3.0.1, URL https://CRAN.R-project.org/package=tibble.

Neumann M (2020). ***MNLpred** – Simulated Predicted Probabilities for Multinomial Logit Models*. R version 0.0.4, URL https://CRAN.R-project.org/package=MNLpred.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robinson D, Hayes A (2020). *broom: Convert Statistical Analysis Objects into Tidy Tibbles*. R package version 0.5.6, URL https://CRAN.R-project.org/package=broom.

Santi F, Dickson MM, Espa G (2019). "A graphical tool for interpreting regression coefficients of trinomial logit models." *The American Statistician*, **73**(2), 200–207. doi:10.1080/00031305.2018.1442368.

Santi F, Dickson MM, Espa G, Giuliani D (2022). "plot3logit: Ternary Plots for Interpreting Trinomial Regression Models." *Journal of Statistical Software, Code Snippets*, **103**(1), 1–27. doi:10.18637/jss.v103.c01.

Santi F, Dickson MM, Espa G, Giuliani D (2024). *plot3logit: Ternary Plots for Trinomial Regression Models*. R package version 3.2.0.

Schauberger G (2019). ***EffectStars2**: Effect Stars*. R package version 0.1-3, URL https://CRAN.R-project.org/package=EffectStars2.

Severini TA (2000). *Likelihood Methods in Statistics*. Oxford University Press. ISBN 978-0-19-850650-8.

Smith MR (2017). "Ternary: An R Package for Creating Ternary Plots." *Zenodo*.

Tutz G, Schauberger G (2013). "Visualization of Categorical Response Models: From Data Glyphs to Parameter Glyphs." *Journal of Computational and Graphical Statistics*, **22**(1), 156–177. doi:10.1080/10618600.2012.701379.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Fourth edition. Springer-Verlag, New York. ISBN 0-387-95457-0, URL https://www.stats.ox.ac.uk/pub/MASS4/.

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag. ISBN 978-3-319-24277-4.

Wickham H, Grolemund G (2016). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly. ISBN 978-1-491-91039-9.

Wilkinson L (2005). *The Grammar of Graphics*. Statistics and Computing, 2 edition. Springer-Verlag.

Wooldridge JM (2010). *Econometric Analysis of Cross Section and Panel Data*. 2 edition. The MIT Press. ISBN 978-0-262-23258-6.

Yee TW (2010). "The VGAM Package for Categorical Data Analysis." *Journal of Statistical Software, Articles*, **32**(10), 1–34. doi:10.18637/jss.v032.i10.

Affiliation:

Flavio Santi
Department of Economics and Management
University of Trento
Via Inama 5
38122 Trento (TN), Italy
E-mail: `flavio.santi@unitn.it`