Quick Guide to `clustglm` Version 0.2

Lloyd Pledger 10 December, 2015

1 Introduction

To fit models using the `clustglm` function, or to create plots using the various plotting functions, follow the instructions in the `clustglm_run.R` file. This document provides additional information on how to use those functions.

The following information refers to factors ROW and COL, assuming the responses came from an $n \times p$ matrix of data (e.g. Site by Species in an ecological community). The factors ROW and COL may be clustered into specified numbers of groups, e.g. R row groups and/or C column groups.

However, we note that the `clustglm` function may equally well be applied to data from a designed experiment with factors A and B.

The model is of the form $g(\mu_{ij}) = \eta_{ij}$ where $g()$ is the link function, μ_{ij} is $E(Y_{ij})$, the expected value in row i column j, and η_{ij} is a linear predictor with main effects and interactions which may involve ROW, COL, ROWCLUSTER, COLCLUSTER or row or column covariates (see examples later).

2 `clustglm` function

The `clustglm()` command is used to fit a model. The family of distribution, model equation and number of row and column clusters are user-specified. The `clustglm` function goes through a number of procedures to generate possible starting points. For each it runs three EM cycles, and saves that start if it gets to the highest log likelihood found so far. It then uses the best start that was found and runs the EM algorithm until the log likelihood and parameters have stabilised.

A wide range of automatic starting options is used to attempt to generate good starting parameters. Depending on the dataset certain models may not converge, so it is possible to specify starting group-allocation matrices in cases where the model is not converging or where it is suspected that the solution is sub-optimal. If starting clusterings are provided, the random start options are all skipped, and the model goes straight to the EM algorithm.

For the rest of this document, models will have the following classifications:

- Generalized Linear Models (GLMs) are when ($R = 1$ or n) and ($C = 1$ or p)
- Row Clustered Models are when $1 < R < n$ and ($C = 1$ or p)
- Column Clustered Models are when $1 < C < p$ and ($R = 1$ or n)

- Biclustered Models are when both $1 < R < n$ and $1 < C < p$

It should be noted that the GLMs are simply run using the `glm()` function within the `clustlgm()` function, so in the case that a GLM is specified as the desired model, all of the starts and EM algorithm section are skipped.

2.1 How the `clustglm` function works

2.1.1 EM Algorithm:

The EM algorithm alternates betwen the E step, in which estimated probabilities of group membership (rows in row groups, columns in column groups) are updated, and the M step in which maximisation of the log likelihood under (current) complete knowledge gives updates of the parameter estimates. The parameters are those from the component distributions together with π_r = proportion of rows in row-group r and/or κ_c = proportion of columns in column-group c. The main problem with the EM algorithm for finite mixtures is multimodality of the likelihood surface, so that many starts are required to try to find the global maximum.

2.1.2 Start option 1: Kmeans start:

The `kmeans()` function is used to generate the posterior probability matrix for row clustered models (PPR) and the posterior probability matrix for column clustered models (PPC). For biclustered models the `dkm()` function (this is the double Kmeans function which is defined in this package) is used to generate both PPR and PPC matrices.

Using the PPR and PPC matrices, a preliminary M-step is run to generate an array of parameters in the linear predictor plus pi and kappa vectors. Three cycles of the EM algorithm are then run, and the output saved along with the Log Likelihood at the end of the last EM cycle.

The number of starts for the `kmeans()` or `dkm()` function is set to a default of 100, but can be specified. The number of EM cycles run before the output is saved and compared with other starts is set to a default of 3, but can be specified.

2.1.3 Start option 2: Hclust start:

A standard agglomerative hierarchical clustering (with complete linkage and Euclidean metric) is used as a possible start. As with the Kmeans start above, a preliminary M-step then three cycles of the EM algorithm are run. The Log Likelihood at the end of the last EM cycle is then compared to the Log Likelihood saved at the end of the Kmeans start, and if the Hclust has found a better Log Likelihood, then the parameters generated by the Hclust start overwrite the Kmeans start parameters and are stored as the best starting parameters.

2.1.4 Start option 3: Diana method start:

The Diana method from the `cluster` package is a **di**visive cluster **ana**lysis. As above, a preliminary M-step then three cycles of the EM algorithm are run. Again the Log Likelihood is compared against the currently stored best start Log Likelihood, and if a better start has been found the best starting parameters are overwritten.

2.1.5 Start option 4: Random start for parameters:

Random starts for parameters is an optional start. The GLM part of the model is fitted, giving starting parameter estimates for those terms. Parameters for the clustered part of the model are started from random samples of residuals from the GLM part, while initial values of π and κ vectors are randomly generated with values approximately equal (e.g. between 0.4–0.6 for two clusters).

A preliminary M-step and three cycles of the EM algorithm are then run. Again the Log Likelihood is compared to the currently stored best start Log Likelihood, and if a better start has been found the best starting parameters are overwritten. This whole process occurs the number of times specified by the randstarts argument (default is 0).

2.1.6 Start option 5: RG split start:

The RG split start is an optional start for row-clustered and biclustered models. It can be used if the random starts from start options 1–4 are not finding starting points that allow the EM algorithm to converge. The way this start option works is it takes the output PPR matrix from a specified model with one fewer row group than the current model, then extends the starting PPR matrix by splitting a single row off into a new group.

A preliminary M-step and three cycles of the EM algorithm are then run. Again the Log Likelihood is compared to the currently stored best start Log Likelihood, and if a better start has been found the best starting parameters are overwritten. This whole process occurs n times, with a different row being moved into the new group each time. To use this optional start, you need to specify the name of the model output to be extended under the `RGsplit` argument.

2.1.7 Start option 6: CG split start:

The CG split start is an optional start that works the same way as the RG split start above, only for column clustered and biclustered models instead. In this case `CGsplit` argument must be used to specify a model output with one fewer column groups than the current model. In the case of a biclustered model, both RGstart and CGstart must be specified.

2.1.8 Given start:

If the model was unable to find satisfactory starting points and failed to converge, the option is also available to supply starting PPR and PPC matrices. These can either be taken from the output of similar models that did converge, or can be manually generated based on knowledge of the dataset being used.

2.2 Specifying a model:

When specifying the model you may use main effects terms to standardize over rows, columns, both or neither. In the discussion and table below, these options are labelled N for no main effects (no standardization, intercept ν only), A for row main effects (row standardization, parameters α_i), B for column main effects (column standardization, β_j) and D for dual standardization (both row and column main effects, α_i and β_j). There are three options for number of row clusters (1, R or n row clusters, $1 < R < n$) and three options for the number of column clusters (1, C or p column clusters, $1 < C < p$). This apparently gives 36 possible types of models. However, there are only 21 distinct types, as some redundancy in the parameterizations means that some models may be simplified to others.

Table 1 lists all 36 model types, but shows where simplifications occur. In the four models with one row group and one column group, the single parameter γ has been set to zero as it is already allowed for in ν. In some cases a parameter is redundant and may be subsumed into another, e.g. in Model AR1 (A = row standardization, R row clusters, one column cluster) the parameter γ_r is redundant as α_i has already allowed for all row differences. Only 17 distinct models are found, the five different GLMs and 12 with non-trivial clustering.

2.3 Interpreting Fitted Parameters:

Although the `clustglm()` function uses a cornerpoint parameterization, the output has been matched to sum-to-zero constraints, which for dual standardization (D) with biclustering gives the following interpretations in the link scale of these parameters in η_{ij}:
$\nu =$ overall average
$\alpha_i =$ deviation from ν for data in row i, with constraint $\Sigma_i \alpha_i = 0$
$\beta_j =$ deviation from ν for data in column j, with constraint $\Sigma_j \beta_j = 0$
$\gamma_{rc} =$ deviation from $\nu + \alpha_i + \beta_j$ for data in row group r and column group c, with constraints $\forall r, \Sigma_c \kappa_c \gamma_{rc} = 0$ and $\forall c, \Sigma_r \pi_r \gamma_{rc} = 0$.

The parameter π_r ($\Sigma \pi_r = 1$) is the proportion of rows in row-group r, and κ_c ($\Sigma \kappa_c = 1$) is the proportion of columns in column-group c. The π and κ parameters are used as weights to ensure the value of γ_{rc} relates to a per-cell deviation for a single cell (i, j). With simpler standardizations N, A or B, either all $\alpha_i = 0$ or all $\beta_j = 0$ or both.

Table 1: Models: linear predictors, model types and details

Model	Linear predictor	Model type	Details
N11	ν	GLM, Null	
NR1	$\nu + \gamma_r$	Rows clust.	
Nn1	$\nu + \gamma_i$	GLM A	Rename γ_i as α_i
N1C	$\nu + \gamma_c$	Cols clust.	
NRC	$\nu + \gamma_{rc}$	Biclust.	
NnC	$\nu + \gamma_{ic}$	Cols clust.	*
N1p	$\nu + \gamma_j$	GLM B	Rename γ_j as β_j
NRp	$\nu + \gamma_{rj}$	Rows clust.	†
Nnp	$\nu + \gamma_{ij}$	GLM, A*B	
A11	$\nu + \alpha_i$	GLM A	
AR1	$\nu + \alpha_i + \gamma_r$	Rows clust.	
An1	$\nu + \alpha_i + \gamma_i$	GLM A	Absorb γ_i in α_i
A1C	$\nu + \alpha_i + \gamma_c$	Cols clust.	
ARC	$\nu + \alpha_i + \gamma_{rc}$	Biclust.	
AnC	$\nu + \alpha_i + \gamma_{ic}$	Cols clust.	$*$
A1p	$\nu + \alpha_i + \gamma_j$	GLM A+B	Rename γ_j as β_j
ARp	$\nu + \alpha_i + \gamma_{rj}$	Rows clust.	††
Anp	$\nu + \alpha_i + \gamma_{ij}$	GLM, A*B	Absorb α_i in γ_{ij}
B11	$\nu + \beta_j$	GLM B	
BR1	$\nu + \beta_j + \gamma_r$	Rows clust.	
Bn1	$\nu + \beta_j + \gamma_i$	GLM A+B	Rename γ_i as α_i
B1C	$\nu + \beta_j + \gamma_c$	Cols clust.	
BRC	$\nu + \beta_j + \gamma_{rc}$	Biclust.	
BnC	$\nu + \beta_j + \gamma_{ic}$	Cols clust.	$**$
B1p	$\nu + \beta_j + \gamma_j$	GLM B	Absorb γ_j in β_j
BRp	$\nu + \beta_j + \gamma_{rj}$	Rows clust.	†
Bnp	$\nu + \beta_j + \gamma_{ij}$	GLM, A*B	Absorb β_j in γ_{ij}
D11	$\nu + \alpha_i + \beta_j$	GLM A+B	
DR1	$\nu + \alpha_i + \beta_j + \gamma_r$	Rows clust.	
Dn1	$\nu + \alpha_i + \beta_j + \gamma_i$	GLM A+B	Rename γ_i as α_i
D1C	$\nu + \alpha_i + \beta_j + \gamma_c$	Cols clust.	
DRC	$\nu + \alpha_i + \beta_j + \gamma_{rc}$	Biclust.	
DnC	$\nu + \alpha_i + \beta_j + \gamma_{ic}$	Cols clust.	$**$
D1p	$\nu + \alpha_i + \beta_j + \gamma_j$	GLM A+B	Rename γ_j as β_j
DRp	$\nu + \alpha_i + \beta_j + \gamma_{rj}$	Rows clust.	††
Dnp	$\nu + \alpha_i + \beta_j + \gamma_{ij}$	GLM, A*B	Absorb α_i, β_j in γ_{ij}

$*$ AnC matches NnC, absorb α_i into γ_{ic}
$**$ DnC matches BnC, absorb α_i into γ_{ic}
† BRp matches NRp, absorb β_j into γ_{rj}
†† DRp matches ARp, absorb β_j into γ_{rj}

2.4 Troubleshooting

2.4.1 Output from model has worse Log Likelihood than expected:

A worse Log Likelihood than expected probably means the model had a poor starting point, which means the maximization process found a local maximum not the global maximum. In this case there is a range of options to find a better starting point. Before running the EM algorithm, the `clustglm` function generates a range of starting points using different methods then runs 3 EM cycles on each to compare them and select the best. You can either increase the number of generated starting points or specify your own. If you specify starting PPR or PPC matrices, this will automatically override the generated starts.

Run extra k-means starts (default is 100):

```
B33.out <- clustglm(glmformula = Y ~ COL, glmfamily = "poisson",
                    data = glm.df, clustfact = c("ROW","COL"),
                    kstarts = 500, numbersclusters=c(3,3),
                    clustfactnames=c("ROWclust","COLclust"),
                    clustformula = Y ~ COL + ROWclust:COLclust)
```

Try running random parameter starts (default is 0):

```
B33.out <- clustglm(glmformula = Y ~ COL, glmfamily = "poisson",
                    data = glm.df, clustfact = c("ROW","COL"),
                    randstarts=100, numbersclusters=c(3,3),
                    clustfactnames=c("ROWclust","COLclust"),
                    clustformula = Y ~ COL + ROWclust:COLclust)
```

Provide starting PPR or PPC matrices:

```
B33.out <- clustglm(glmformula = Y ~ COL, glmfamily = "poisson",
                    data = glm.df, clustfact = c("ROW","COL"),
                    ppr.start=B3p.out$ppr, ppc.start=Bn3.out$ppc,
                    numbersclusters=c(3,3),
                    clustfactnames=c("ROWclust","COLclust"),
                    clustformula = Y ~ COL + ROWclust:COLclust)
```

Extend the starting PPR or PPC matrices from simpler models (this takes a simpler PPR or PPC matrix with one too few row or column groups, then extends it by moving one row or column at a time into the empty new group):

```
B33.out <- clustglm(glmformula = Y ~ COL, glmfamily = "poisson",
                    data = glm.df, clustfact = c("ROW","COL"),
                    RGsplit=B2p.out, CGsplit=Bn2.out,
                    numbersclusters=c(3,3),
                    clustfactnames=c("ROWclust","COLclust"),
                    clustformula = Y ~ COL + ROWclust:COLclust)
```

2.4.2 Model failing to converge:

If the model runs to its limit of 100 EM cycles, that means it failed to converge. The E-M algorithm failing to converge often means that it has found itself at a flat part of the likelihood surface, either a ridgeline or a wide maximum. This means it will continue to step around in small steps and fail to realise it can stop. If a model has effectively converged, the results will be fine to use, but if you want you can either impose a limit on the maximum number of EM cycles or change the sensitivity parameter for convergence.

Change the maximum number of EM cycles (default is 100 cycles):

```
B33.out <- clustglm(glmformula = Y ~ COL, glmfamily = "poisson",
                    data = glm.df, clustfact = c("ROW","COL"),
                    EMcycles=500, numbersclusters=c(3,3),
                    clustfactnames=c("ROWclust","COLclust"),
                    clustformula = Y ~ COL + ROWclust:COLclust)
```

Change the sensitivity parameter for convergence (default is 1e-04):

```
B33.out <- clustglm(glmformula = Y ~ COL, glmfamily = "poisson",
                    data = glm.df, clustfact = c("ROW","COL"),
                    EMstoppingpar=1e-03, numbersclusters=c(3,3),
                    clustfactnames=c("ROWclust","COLclust"),
                    clustformula = Y ~ COL + ROWclust:COLclust)
```

2.5 Predictor Variables:

These are covariates which may be continuous, discrete or categorical. They may be related to rows (eg. site factors) or columns (eg. species traits). They may be used as alternative to ROW or COL as main effects or as terms in an interaction.

2.6 Arguments:

To run a model, the `clustglm()` command is used. A number of arguments are required and some further ones are optional.

Required Arguments:

- `glmformula`: The GLM formula specifying all unclustered effects in the model. Takes the form `Y ~ ROW + COL` or `Y ~ ROW + Plant.Type` if using predictor variables.
- `data`: Need to specify the data frame to be used.

Optional Arguments:

- `glmfamily`: As with the `glm` function, default is set to "gaussian". Can instead be set to one of bernoulli, binomial or poisson. (Negative binomial and exponential distributions are yet to be included and tested, 10.12.2015.)
- `ntrials`: If glmfamily = "binomial", then the number of trials must be specified.
- `clustfact`: If the model is to have clustering, then you need to specify the names of the existing factors in the data frame which will be clustered (eg `c("ROW","COL")`). Including these existing factors as main effects is optional.
- `numbersclusters`: If the model has clustering, then you need to specify the numbers of row and column clusters (eg `c(10,4)`).
- `clustfactnames`: If the model has clustering, then you need to specify the new names for the clustered versions (eg `c("ROWclust","COLclust")`).
- `clustformula`: A formula including the GLM formula and any clustered terms (eg `Y ~ ROW + COL + ROWclust:COLclust`).
- `kstarts`: The number of starts generated using K-means (by default set to 100)
- `randstarts`: The number of random starts for the parameters (by default set to 0).
- `GF1split`: Used to specify an already generated model output with 1 fewer row clusters than the desired model to create more starting options for the current model (eg `GF1split = B4p.out` if using the output of model B4p to give starts for model B5p).
- `GF2split`: Used to specify an already generated model output with 1 fewer column clusters than the desired model to create more starting options for the current model.
- `ppr.start`: If desired a starting PPR matrix can be specified. This will override all the other starting options.
- `ppc.start`: If desired a starting PPC matrix can be specified. This will override all the other starting options.
- `startEMcycles`: The number of EM cycles that are run to test each starting method (default is 3 EM cycles).

- `EMcycles`: The maximum number of cycles the EM algorithm can run for (default is 100).
- `EMstoppingpar`: The sensitivity parameter for convergence of the EM algorithm (default is 1e-04).

3 ICtable function

The `ICtable()` command takes the output from a number of models run using the `clustglm()` function and assembles it into a table allowing for the comparison of the models using different information criteria. The models to be included are supplied in a list.

4 ICplot function

The `ICplot()` command allows for the visual comparison of a range of models using a selected information criterion.

4.1 Arguments:

A number of arguments are required and some further ones are optional.

Required Arguments:

- `plot.list`: Need to specify the models to be plotted.
- `ICtype`: Needs to be set to one of AIC, AICc, BIC. This determines which contour lines are printed.

Optional Arguments:

- `lowlabelposition`: Command used to determine the placement of the lower contour label on the plot. The default position is 1, but if set to 2 it moves the lower contour label more towards the centre of the plot. If set to 0 then no contour label will be shown.
- `highlabelposition`: Command used to determine the placement of the upper contour label on the plot. The default position is 1, but if set to 2 it moves the upper contour label more towards the centre of the plot. If set to 0 then no contour label will be shown.
- `colourvector`: If desired a specific colour vector can be used, otherwise the models will all be plotted in black.
- `plottitle`: If desired a title can be specified, otherwise a default title will be generated.

5 Ordplot function

The `ordplot()` command creates an ordination plot from the selected model. Note that only certain models will allow the creation of a 2-dimensional ordination plot. For row ordination the model needs all rows separate and two column clusters. For column ordination the model needs all columns separate and two row clusters.

5.1 Arguments:

A number of arguments are required and some further ones are optional.

Required Arguments:

- `model.in`: Need to specify the model to be plotted
- `ordtype`: Needs to be set to either rows or columns

Optional Arguments:

- `rlabels`: When ordinating by rows, if rlabels is not specified the rownames of the gamma matrix will be used
- `clabels`: When ordinating by columns, if clabels is not specified the colnames of the gamma matrix will be used
- `xlabel`: Used to specify the label on the x-axis of the plot. If not specified the default label is "Dimension 1"
- `ylabel`: Used to specify the label on the y-axis of the plot. If not specified the default label is "Dimension 2"
- `plottitle`: If desired a title can be specified, otherwise a default title will be generated

6 Profileplot function

The `profileplot()` command creates a profile plot of either the row groups or column groups for the selected model. Different sortings can be applied to the row or column order for easier interpretation.

6.1 Arguments:

A number of arguments are required and some further ones are optional.

Required Arguments:

- `model.in`: Need to specify the model to be plotted.
- `profiletype`: Needs to be set to either rowgroups or colgroups.

Optional Arguments:

- `sorttype`: Can be NULL (no sorting), mean (sorted into ascending mean order), group (sorted into ascending order for the group specified), travellingsalesman (sorted to achieve the shortest paths).
- `sortgroup`: If sorttype = group then sortgroup needs to be specified to determine which group needs to be sorted into ascending order.
- `colourvector`: If desired a specific colour vector can be used, otherwise the default colour vector will be used.
- `xlabel`: Used to specify the label on the x-axis of the plot. If not specified the default label is either "Row" or "Column" depending on whether the profiletype is "colgroups"or "rowgroups".
- `plottitle`: If desired a title can be specified, otherwise a default title will be generated.
- `linetype`: Can be specified in the form `c(1,2,3)`. By default all solid lines will be used (line type set to 1).
- `linewidth`: Can be specified in the form `c(1,0.7,1.2)`. By default all line widths are set to 1.

7 Biplot function

The `biplot()` command creates a biplot from a biclustered model with two row groups and two column groups.

7.1 Arguments:

A number of arguments are required and some further ones are optional.

Required Arguments:

- `model.in`: Need to specify the model to be plotted
- `data`: Need to specify the data matrix to be used

Optional Arguments:

- `colourvector`: If desired a specific colour vector can be used, otherwise the default colour vector will be used
- `plottitle`: If desired a title can be specified, otherwise a default title will be generated
- `myratio`: By default set to 1. May need to be specified to stretch the width of the points plotted to make the graph more easily interpreted (eg. `myratio = 0.1`).
- `centrepoints`: By default set to T. Can be set to F to not plot the centrepoints.