GLMSCA

This is a short guide for the GLMSCA module in python.

Please see README for description of parameters, attributes and methods.

This guide is made with simulated data:
A 900×50 Y matrix containing response variables was created based on a Poisson distribution. Systematic variation was created from a Standard normal distribution:

$$\boldsymbol{\eta} = \boldsymbol{\beta}_0 + \boldsymbol{X}\boldsymbol{\beta}_1 + \boldsymbol{X}\boldsymbol{\beta}_2.$$

and the Y matrix was given by data drawn from $\boldsymbol{\mu}$:

$$\boldsymbol{\mu} = \exp\{\boldsymbol{\eta}\}$$

The design of the predictor variables are a full factorial design based on the variables in table 1. The 3^2 full factorial design results in 9 combinations,

Table 1: Two factors each with three levels.

George	Michael
Bush	Schumacher
Lucas	Bublé
Clooney	Jordan

which are repeated 100 times. The GLMSCA algorithm is called by:

```
mdl = GLMSCA(X, Y)
```

and the model is fit by:

```
mdl.fit()
```

The raw data can be plotted by the command:

```
mdl.plot_raw()
```

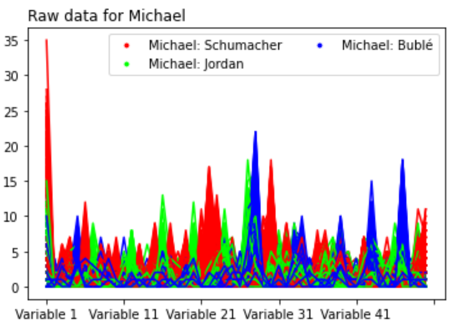
And coloured by specifying the factor, either by:

```
mdl.plot_raw(0)
```

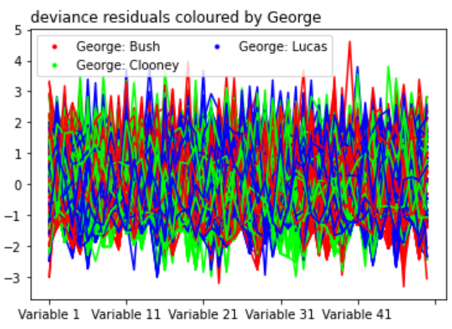
or:

```
mdl.plot_raw('George')
```

for the first factor. The result of the plot is depicted in figure 2. Similarly, the residuals can be plotted and coloured by the specific factor by:



(a) Raw plot of the first factor



(b) residual plot of the second factor

Figure 1: Raw and residual plots

```
mdl.plot_residual('Michael')
```

Which can be seen in figure 1b.
The residual is specified in the `.Options` by:

```
mdl.Options.residual = 'deviance'
```

It is possible to choose from 'anscombe', 'deviance', 'pearson', 'response', 'quantile' or 'working' residuals, where 'working' is the default.
In `.Options` the number of components can be specified, along with whether to have interaction effects:

```
mdl.Options.n_components = 5
```

and

```
mdl.Options.interaction = True
```

Interactions are by default set to `False`.
The distribution type can be chosen from 'Binomial', 'Poisson' or 'NegativeBinomial', where 'Poisson' is the default:

```
mdl.Options.dist = 'Poisson'
```

It is also possible to chose different distributions based on the variable. A list with a length, the same as the number of variables, can be specified in:

```
mdl.Options.dist_list
```

If a a variable do not converge, they are by default omitted from the data set. It is possible to change this by:

```
mdl.Options.keep_conv_only = False
```

Score plots can be plotted by:

```
mdl.plot_scores()
```

which by default returns the scores of the first component. The specific scores can be plotted by

```
mdl.plot_scores(0)
```

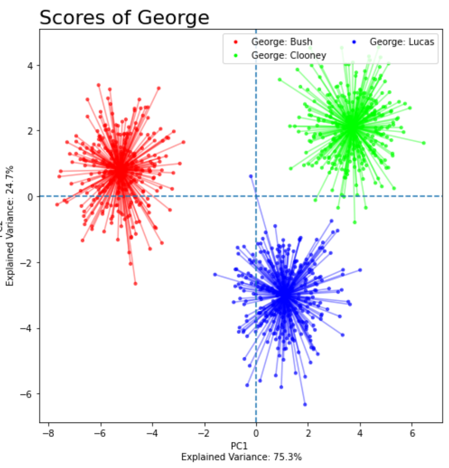
or

```
mdl.plot_scores('George')
```

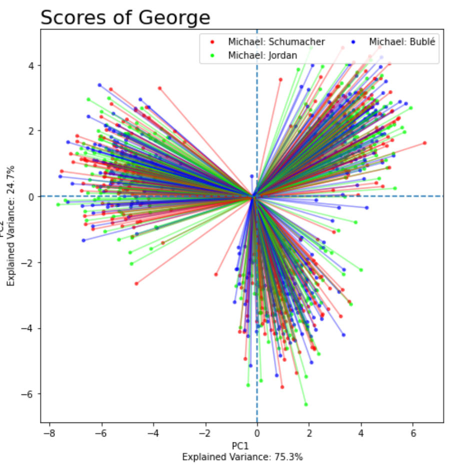
By default the score plots are coloured by the plotted factor, but they can also be coloured by a specific factor:

```
mdl.plot_scores(factor = 'George', group_by = 'Michael')
```

The two resulting score plots are seen in figure 2a and 2b.
The score plots can also be coloured by a different vector, not specified in the model:



(a) Coloured by the first factor 'George'



(b) Coloured by the second factor 'Michael'

Figure 2: Score plots of the first factor 'George'

```
mdl.plot_scores(factor = 'George', group_by = Wham)
```

The component of interest can be specified by the command, component:

```
mdl.plot_scores(factor = 'George', group_by = Wham, component = (0,1))
```

where the default is the first and second component, $(0,1)$. The component is given by a tuple of two integers.
The scores are calculated based on the residuals chosen in `.Options.`
The loadings can be plotted by

```
mdl.plot_loadings()
```

where the default is determined by the first component. Similarly to the other plot types the factor can be specified by

```
mdl.plot_loadings(0)
```

or

```
mdl.plot_loadings('George')
```

The results of the plot can be seen in figure 3.

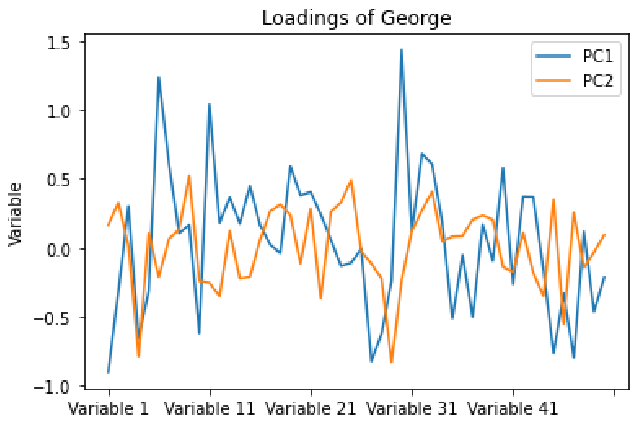


Figure 3: Loading plot of the first factor 'George'