The Stata Journal (2009)
9, Number 3, pp. 374–387

Graphical representation of multivariate data using Chernoff faces

Rafal Raciborski[1]
Department of Political Science
Emory University
Atlanta, GA
rafal.raciborski@emory.edu

Abstract. Chernoff (1971, Technical Report 71, Department of Statistics, Stanford University; 1973, *Journal of the American Statistical Association* 68: 361–368) proposed the use of cartoon-like faces to represent points in k dimensions. This article describes a Stata implementation of a face-generating algorithm using the method proposed by Flury (1980, Technical Report 3, Institute of Mathematical Statistics and Actuarial Science, Bern University), Schüpbach (1987, Technical Report 25, Institute of Mathematical Statistics and Acturial Science, Bern University), and Friendly (1991, http://www.math.yorku.ca/SCS/sasmac/faces.html). I present examples of applying Chernoff faces to data clustering and outlier detection.

Keywords: gr0038, chernoff, Chernoff faces, graphs

1 Introduction

Chernoff (1971, 1973) proposed a method of representing multivariate data as cartoon faces. The main use of face graphs was to enhance "the user's ability to detect and comprehend important phenomena" and to serve "as a mnemonic device for remembering major conclusions". Other advantages of faces include the ease of monitoring the sensitivity of variables to each other, fast identification of key differentiating dimensions, and the detection of longitudinal trends. Recent applications of Chernoff faces include tracking changes in laboratory data (Lott and Durbridge 1990), consumer perception of brand image (Golden and Sirdesai 1992), classification of forest tree clones (Camussi, Raddi, and Raddi 1992), portrayal of service quality data (Nel, Pitt, and Webb 1994), attitudes toward environmental protection policies (Apaiwongse 1995), and classification of drinking water samples (Astel et al. 2006).

However, the method of constructing faces as proposed by Chernoff suffers from certain disadvantages. First, to make all the faces of equal size, the width and length of each face need to be normalized, which almost obliterates the effect of the variables assigned to those two features (Chernoff 1971, 19). Second, extreme values of certain parameters compress the range of other parameters, which results in "artificial dependencies not originating from the data being represented" (Flury and Riedwyl 1981, 757). The

---

[1] I would like to thank Taha Kass-Hout and an anonymous referee for helpful suggestions.

algorithm described in this article was developed by Flury (1980), Schüpbach (1987), and Friendly (1991), and avoids the above mentioned pitfalls.

The article proceeds as follows. Section 2 describes the `chernoff` command and its options. Section 3 presents examples of data clustering and outlier detection. Section 4 offers general advice on drawing and interpreting Chernoff faces. Section 5 concludes.

2 The chernoff command

2.1 Syntax

`chernoff` [, `isize(`*exp*`)` `iangle(`*exp*`)` `ihor(`*exp*`)` `ivert(`*exp*`)` `psize(`*exp*`)`
`ppos(`*exp*`)` `bcurv(`*exp*`)` `bdens(`*exp*`)` `bhor(`*exp*`)` `bvert(`*exp*`)` `fline(`*exp*`)`
`hupper(`*exp*`)` `hlower(`*exp*`)` `hdark(`*exp*`)` `hslant(`*exp*`)` `nose(`*exp*`)` `msize(`*exp*`)`
`mcurv(`*exp*`)` `gmin(`#`)` `gmax(`#`)` [`lhalf` | `rhalf`] `hspace(`#`)` `ititle(`*varname*`)`
`inote(`*varname*`)` `ilabel(`*varname*`)` `lsize(`*textsizestyle*`)` `xlabel(`#`)` `ylabel(`#`)`
`placement(`*clockdirstyle*`)` `justification(`*jstyle*`)` `iscale(`*varname*`)`
`imargin(`*marginstyle*`)` `iregion(`*marginstyle*`)` `xface(`#`)` `yface(`#`)` `show`
`saveall` `rescale(`#`)` `legend({2|3}` [`nolabel`]`)` `order(`*varlist*`)` `rows(`#`)`
`cols(`#`)` `colfirst` `xcombined(`#`)` `ycombined(`#`)` `nocombine` `title(`*tinfo*`)`
`subtitle(`*tinfo*`)` `note(`*tinfo*`)` `nodraw` `saving(`*filename*`, replace)` `timer`]

2.2 Options

Face feature

`isize(`*exp*`)` through `mcurv(`*exp*`)` represent face features and are described in table 1, below. *exp* is specified as

varname | `.` | `_null_` [, [# | `.`] [# | `.`]]

varname is linearly rescaled to a $(0,1)$ interval before being plotted. If a given face feature is not specified or is specified as missing, it assumes the default value of 0.50. Missing values in *varname* are also assigned the value of 0.50. `_null_` prevents a "logical" set of features from being drawn. For example, specifying any of the eye features as `_null_` will prevent the eyes from being plotted.

The optional part represents a theoretical minimum and maximum for *varname*. For example, if the variable `gpa` in your data ranges from 1.7 to 3.8 but you want to plot it relative to the possible range $(1.0, 4.0)$, you can do so by specifying *facefeature*`(gpa,1 4)`. If you want to specify min only, type *facefeature*`(gpa,1 .)` or *facefeature*`(gpa,1)`. If you want to specify max only, type *facefeature*`(gpa,. 4)`.

Table 1. Explanation of face features

option	face feature	`_null_` group
`isize(`*exp*`)`	eye size	`_null_`
`iangle(`*exp*`)`	eye angle	
`ihor(`*exp*`)`	eye horizontal position	
`ivert(`*exp*`)`	eye vertical position	
`psize(`*exp*`)`	pupil size	`_null_`
`ppos(`*exp*`)`	pupil position	
`bcurv(`*exp*`)`	brow curvature	`_null_`
`bdens(`*exp*`)`	brow density	
`bhor(`*exp*`)`	brow horizontal position	
`bvert(`*exp*`)`	brow vertical position	
`fline(`*exp*`)`	face line	`_null_`
`hupper(`*exp*`)`	hair upper line	`_null_`
`hlower(`*exp*`)`	hair lower line	`_null_`
`hdark(`*exp*`)`	hair darkness	`_null_`
`hslant(`*exp*`)`	hair shading slant	
`nose(`*exp*`)`	nose line	`_null_`
`msize(`*exp*`)`	mouth size	`_null_`
`mcurv(`*exp*`)`	mouth curvature	

Individual face graph

`gmin(`#`)` and `gmax(`#`)` stand for a global minimum and maximum. These options override the "local" minimums and maximums specified within face features. See section 3.2 for more details.

`lhalf` | `rhalf` tells Stata to draw only the left side or the right side of the face, respectively. Only one option may be specified.

`hspace(`#`)` controls the column spacing between half-face graphs. The range of `hspace()` is $(0.50, 1)$, with the default being `hspace(0.75)`. Specifying a lower value will bring the faces closer, and specifying a higher value will spread them apart.

`ititle(`*varname*`)` specifies titles for individual face graphs. The variable may be string or numeric.

`inote(`*varname*`)` specifies notes for individual face graphs. The variable may be string or numeric.

`ilabel(`*varname*`)` tells Stata to label faces with the values of *varname*. The variable may be string or numeric.

`lsize(`*textsizestyle*`)` specifies the size of the label. The default is `lsize(large)`. See [G] ***textsizestyle*** for a list of available styles.

`xlabel(`#`)` and `ylabel(`#`)` denote the (x, y) coordinates of the face label. The default position is $(10, -10)$ for full faces, $(-8, -10)$ for left faces, and $(8, -10)$ for right faces.

`placement(`*clockdirstyle*`)` specifies the position of the face label relative to (`xlabel()`, `ylabel()`) coordinates. Possible values are 0, 1, 2, …, 12. The default value is `placement(9)`, or `placement(3)` if `lhalf` is specified.

`justification(`*jstyle*`)` specifies text justification of the face label. Possible values are `left`, `right`, and `center`. The default value is `justification(left)`, or `justification(right)` if `lhalf` is specified.

`iscale(`*varname*`)` specifies a multiplier that affects the size of `ititle()`, `inote()`, and `ilabel()`. A *varname* is required, which allows for different scales for different faces.

`imargin(`*marginstyle*`)` specifies the margins between the plot area and the outside area of a face graph. This option is equivalent to `graphregion(margin(`*marginstyle*`))`; see [G] ***marginstyle***. The default margin is `imargin(zero)`.

`iregion(`*marginstyle*`)` specifies the axes offset from the contents of the plot. This option is equivalent to `plotregion(margin(`*marginstyle*`))`; see [G] ***region_options***. The default margin is `iregion(medsmall)`.

`xface(`#`)` and `yface(`#`)` denote the size of the individual face graph. The default size is `xface(5.00)` and `yface(6.00)`.

`show` tells Stata to draw individual face graphs. The default is to draw only the final combined graph.

`saveall` tells Stata to save all individual face graphs. This is useful if the user later wants to combine individual face graphs manually. Graphs are saved in the current directory as `FACE1.gph`, `FACE2.gph`, …. Also, a blank graph, `FACE0.gph`, is saved.

`rescale(`#`)` restricts the range of all variables to an interval narrower than $(0, 1)$; use this option only in the rare cases when some features of the face intersect, producing an effect that is not aesthetically pleasing. For example, specifying `rescale(0.95)` compresses all the variables to the range $(0.05, 0.95)$. It is not recommended to use values less than 0.90 because it introduces an artificial reduction in the variation of the data.

Combined graph

`legend({2|3}` [`nolabel`]`)` generates a legend based on variable labels or, if labels are missing, on variable names. The legend is displayed at the bottom of the combined graph. You may specify `legend(2)` or `legend(3)` to display the legend in two or three columns, respectively. If you do not want variable labels to be included in the legend, add the `nolabel` option. The `legend()` option overrides the `note()` option.

`order(`*varlist*`)` tells Stata to draw faces sorted by *varlist*.

`rows(`#`)` and `cols(`#`)` denote the number of rows and columns in a combined face graph. If both options are specified, `cols()` takes precedence.

`colfirst` tells Stata to display face graphs down columns.

`xcombined(`#`)` and `ycombined(`#`)` specify the size of the combined graph.

`nocombine` tells Stata not to construct the final combined graph, which is the most time-consuming part of the `chernoff` command. This option is most useful in conjunction with `saveall` or if the user wants to take a "first peek" as to whether the individual face graphs look "right".

`title(`*tinfo*`)`, `subtitle(`*tinfo*`)`, `note(`*tinfo*`)`, `nodraw`, and `saving(`*filename*`, replace)` are standard Stata graph options; see [G] ***twoway_options***.

Utility

`timer` reports the amount of time the command spent constructing individual face graphs (timer 1) and, once the command concludes, the time spent on constructing the combined graph (timer 2). The first reported time should give you a rough idea of how long it will take the command to complete—in testing, stage 2 took on average about 20 percent longer than the time spent on stage 1.

Figure 1 demonstrates a variety of Chernoff faces using the randomly generated variables `x1`–`x18`. Labels have been assigned randomly to half the variables. The code for this and the next examples can be found in the ancillary files and in the online appendix.[1]

---

[1] The online appendix is located at http://www.roofoos.net.

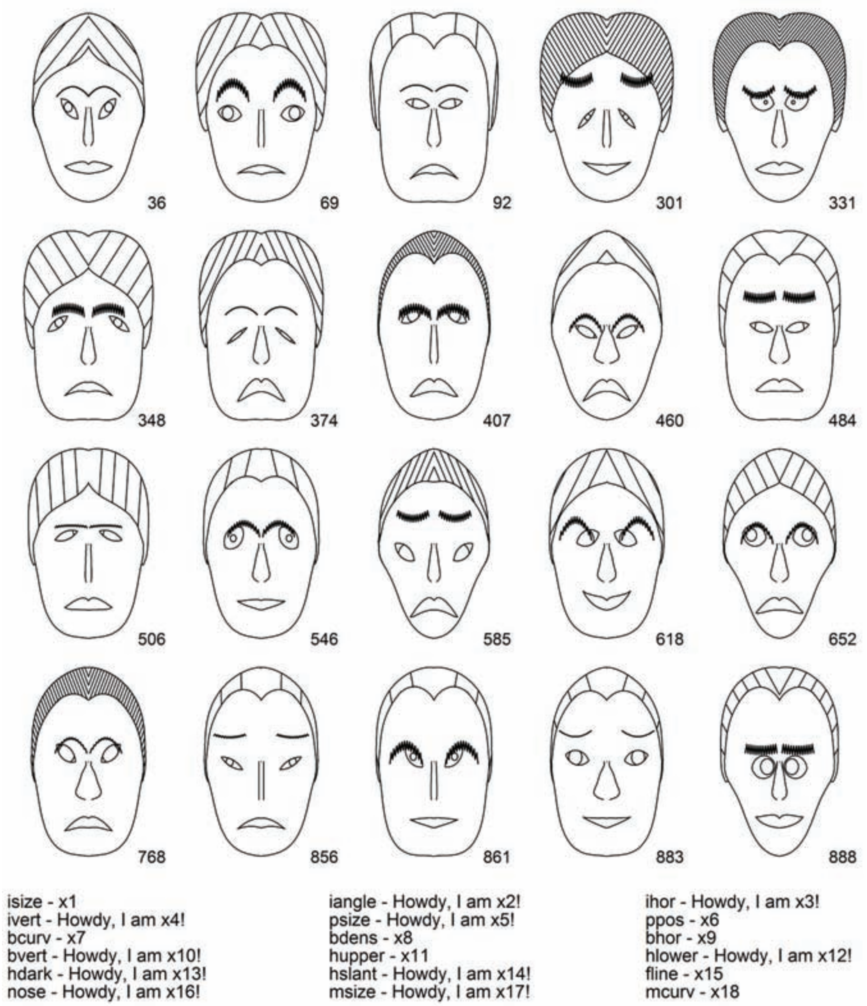


Figure 1. Random faces with a legend

3 Applications

3.1 Classification/clustering

To illustrate the use of Chernoff faces for the purposes of classification, I replicate the example of public utility companies from Johnson and Wichern (2007). The variables are `covr` (fixed-charge coverage ratio, income/debt), `rtrn` (rate of return on capital), `cost` (cost per kW capacity in place), `load` (annual load factor), `grow` (peak kWh demand growth from 1974 to 1975), `sale` (sales, kWh use per year), `nuke` (percent nuclear), and `fuel` (total fuel costs, cents per kWh).

The reader should keep in mind that the purpose of this exercise is to demonstrate how to create sophisticated face graphs in Stata rather than argue about the appropriateness of assignment to a particular group. Table 2 presents the data in a tabular format. It is not clear whether there is any clustering in the data.

Table 2. Public utility data, 1975

`id`	`company`	`state`	`covr`	`rtrn`	`cost`	`load`	`grow`	`sale`	`nuke`	`fuel`
1	Arizona Public	AZ	1.06	9.2	151	54.4	1.6	9077	0	0.63
2	Boston Edison	MA	0.89	10.3	202	57.9	2.2	5088	25.3	1.56
3	C. Louisiana El.	LA	1.43	15.4	113	53	3.4	9212	0	1.06
4	Comm. Edison	IL	1.02	11.2	168	56	0.3	6423	34.3	0.70
5	Con. Edison	NY	1.49	8.8	192	51.2	1	3300	15.6	2.04
6	Florida Power	FL	1.32	13.5	111	60	−2.2	11127	22.5	1.24
7	Hawaiian El.	HI	1.22	12.2	175	67.6	2.2	7642	0	1.65
8	Idaho Power	ID	1.1	9.2	245	57	3.3	13082	0	0.31
9	Kentucky Utils	KY	1.34	13	168	60.4	7.2	8406	0	0.86
10	Madison Gas	WI	1.12	12.4	197	53	2.7	6455	39.2	0.62
11	Nevada Power	NV	0.75	7.5	173	51.5	6.5	17441	0	0.77
12	New England El.	NE	1.13	10.9	178	62	3.7	6154	0	1.90
13	Northern States	MN	1.15	12.7	199	53.7	6.4	7179	50.2	0.53
14	Oklahoma Gas	OK	1.09	12	96	49.8	1.4	9673	0	0.59
15	Pacific Gas	CA	0.96	7.6	164	62.2	−0.1	6468	0.9	1.40
16	Puget Sound	WA	1.16	9.9	252	56	9.2	15991	0	0.62
17	San Diego Gas	CA	0.76	6.4	136	61.9	9	5714	8.3	1.92
18	The Southern Co.	AZ	1.05	12.6	150	56.7	2.7	10140	0	1.11
19	Texas Utils	TX	1.16	11.7	104	54	−2.1	13507	0	0.64
20	Wisconsin El.	WI	1.2	11.8	148	59.9	3.5	7287	41.1	0.70
21	United Illum.	CT	1.04	8.6	204	61	3.5	6650	0	2.12
22	Virginia El.	VA	1.07	9.3	174	54.3	5.9	10093	26.6	1.31

Figure 2 presents the companies divided into clusters as suggested by Johnson and Wichern (2007). I performed the following assignments:

```
... bdens(cost) fline(nuke) hdark(covr) iangle(load) isize(grow) ///
         mcurv(sale) hslant(rtrn) nose(fuel)
```

As can be seen, companies group largely according to geographical location. On a practical side, I first called `chernoff` with the `nocombine` and `saveall` options and made cluster name graphs by hand. I then put the graphs in their appropriate places by using `graph combine`. The empty spaces in the graph were obtained with the multiple use of `FACE0.gph`.

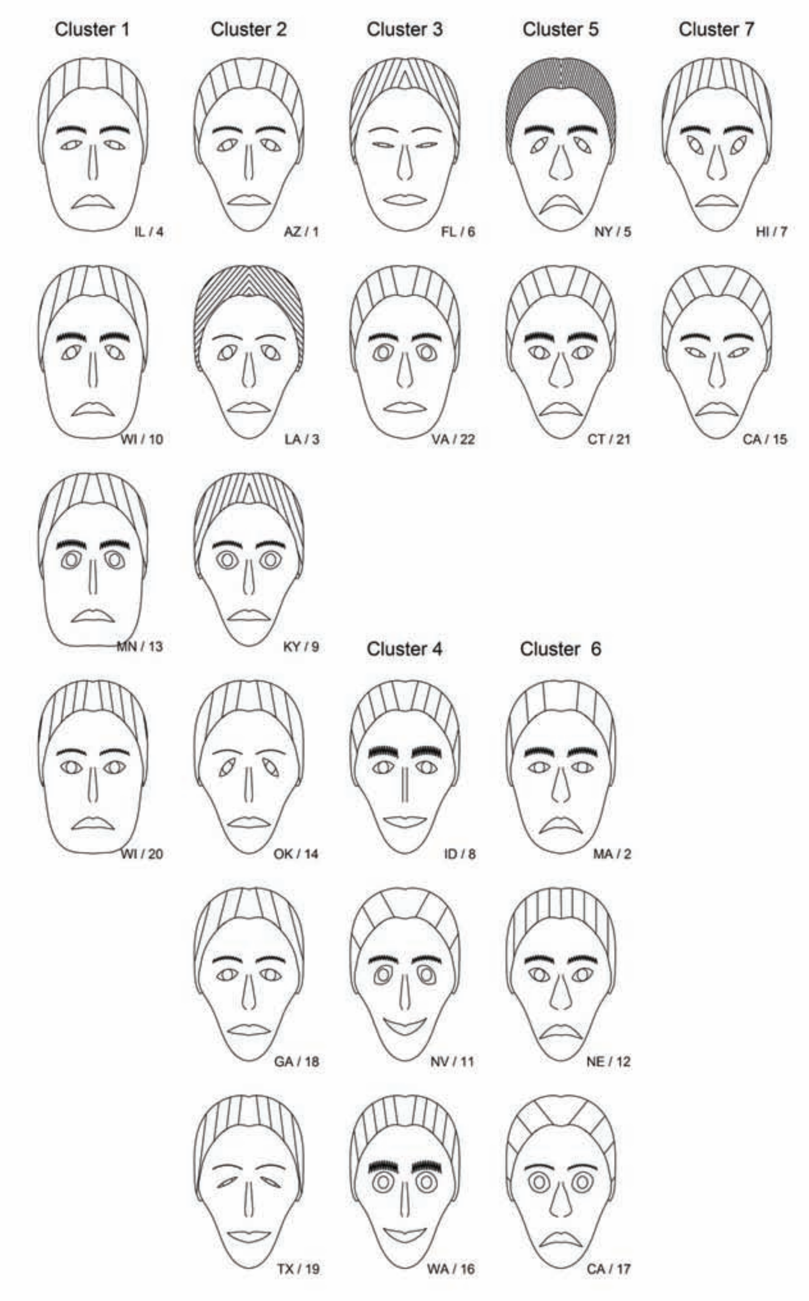


Figure 2. Chernoff faces for 22 public utilities

3.2 Outlier detection

An outlier can have a profound effect on the appearance of face features. Consider a variable with values $0, 1, 2, \ldots, 10$. If I assign this variable to all the 18 face features, the face with the values 0 will look "saddest" and the face with the values 10 will look "happiest".

Now consider adding one more observation with the value of 100 and redrawing the faces. Because internally the values are mapped to a $(0, 1)$ range using the formula

$$x_i^* = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

the old mapping $10 \rightarrow 1$ now becomes $10 \rightarrow 0.10$, and all the original faces will get "compressed" and look "sad" compared with the new arrival. I illustrate this phenomenon on biosurveillance data where I have data points arriving at fixed time intervals. Every time period, the user is presented with faces corresponding to the last seven time periods. If the latest data point appears extreme relative to the past data points, it may signify the beginning of a disease outbreak. Table 3 shows weekly flu data representing the frequency of Google Internet search queries containing the word *flu*.[2] In theory, the frequency of search queries ranges from 0 to 100, and because a public health perspective finds lower values to be more desirable, I define and plot `myflu = 100 - flu`.

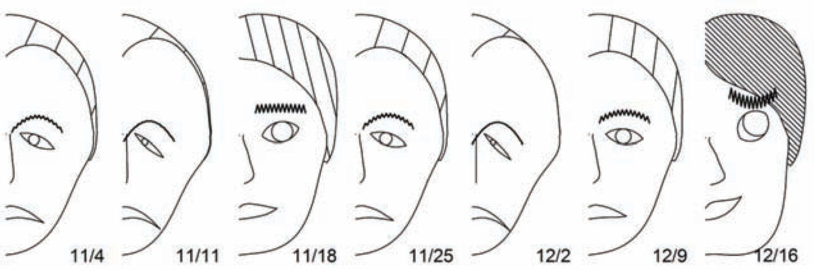
Table 3. Flu data, 2004

`t`	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
`date`	11/4	11/11	11/18	11/25	12/2	12/9	12/16	12/23
`flu`	19.38	20.99	15.71	18.80	21.42	17.73	12.53	97.17
`myflu`	80.62	79.01	84.29	81.20	78.58	82.27	87.47	2.83

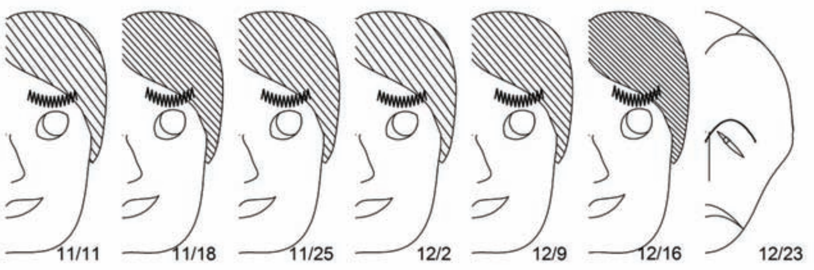
Figure 3a shows faces for $t_1 - t_7$ drawn without considering the theoretical extrema. All the values of `myflu` are relatively high compared with the possible minimum of 0, yet I get an impression that most of the time I should be worried about a flu outbreak. Now consider what happens when `myflu` at t_8 is added. Because the new arrival represents a true departure from the previous pattern, it completely changes the look of the faces (figure 3b). The researcher will certainly be baffled by the resulting picture.

---

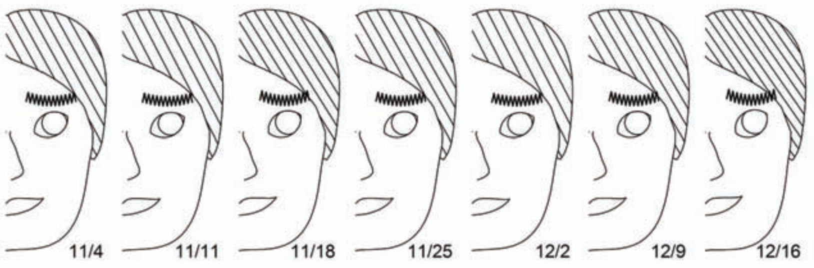
[2] Data courtesy of Taha Kass-Hout and Google Insights for Search.



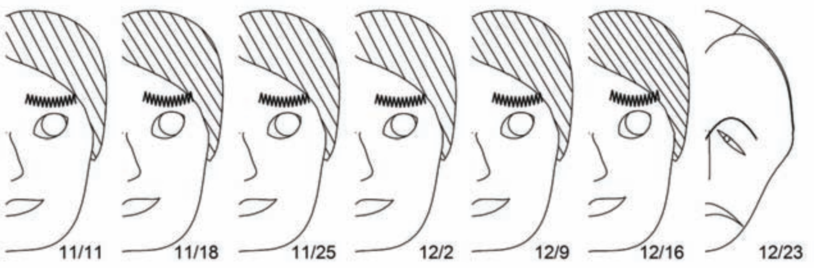
(a) Time period t_1–t_7 without considering global extrema



(b) Time period t_2–t_8 without considering global extrema



(c) Time period t_1–t_7 with adjustment for global extrema



(d) Time period t_2–t_8 with adjustment for global extrema

Figure 3. Flu data

Now let's revisit the data but this time include a theoretical min and max. This can be done two ways. First, you can specify

```
. chernoff, isize(myflu, 0 100) iangle(myflu, 0 100) ... mcurv(myflu, 0 100)
```

which gets the job done but is tedious. This is where the `gmin()` and `gmax()` options come in handy. If all your variables share the same theoretical min or max, you can specify a global min or max for all of them by typing

```
. chernoff, isize(myflu) iangle(myflu) ... mcurve(myflu) gmin(0) gmax(100)
```

Figure 3c presents the same data as figure 3a, but I account for extremes. Now all the faces are correctly depicted as "happy". Also, the arrival of an outlier does not alter face representation for the previous time periods (figure 3d).

The flu example illustrates the importance of including a theoretical minimum and maximum for variables, especially for continually updated or "live" data. This will not always be possible because some phenomena do not have a strictly defined min or max, for example, growth rates. In those cases, the researcher should be aware that an arrival of an outlier can significantly alter the face landscape.

4 Further advice

Over the years, Chernoff faces received their fair share of criticism. However, most pitfalls can be avoided by carefully thinking about the problem at hand and by knowing your data. In this section, I offer several suggestions with respect to the use of Chernoff graphs and the `chernoff` command.

4.1 Inverted scales

Be aware of what your variables represent. If you have a depression score where higher values indicate higher depression, do not assign it to the mouth feature because higher values will result in a smile. The usual solution is to reverse the values before the assignment, similar to what I did in the flu example. Good candidates for inverted scales include employee turnover, occurrence of earthquakes, dropout rate, number of bankruptcies, and measures of corruption. Be aware that some popular indices are coded counterintuitively—for example, the Freedom House index of democracy uses 1 to denote free countries and 7 to denote not free countries.[3]

4.2 No if or in

The sensitivity of face features with respect to outliers is the reason that I have not implemented the `if` or `in` option. You could be tempted to code

```
. chernoff if foreign==1 ...
```

---

[3] The Freedom House web site is http://www.freedomhouse.org.

but does it mean that you want face features to be calculated on the foreign car subsample or do you want calculations to be performed on the whole sample but graph only faces representing foreign cars? To avoid the confusion, `chernoff` does not honor qualifiers. If you wish to plot faces for a particular subsample, you can always drop the unneeded observations, although you should probably specify a min and max from the entire sample.

4.3 Limits

The main challenge in programming Chernoff faces is that each individual face requires many data points to be plotted. For example, a SAS implementation by Friendly (1991) uses "approximately 800 annotate observations for each face". In my implementation, each face requires 51 temporary variables, but those are recycled for each face; thus the user is not limited by the number of variables Stata can hold. The only restriction is that the number of sersets is capped at 1,999—because each face uses two sersets, this limits the size of the combined graph to 999 faces; see [P] **serset** for more details. Hair shading is stored as a separate serset, thus specifying `hdark(_null_)` or `hslant(_null_)` will allow for 1,999 faces to be plotted simultaneously.

4.4 Number of observations

Several face features are implemented as fifth-degree polynomials. Upper and lower hair line, stored in temporary variables, each needs 121 data points to be plotted. When the `chernoff` command starts, it checks whether there is a sufficient number of observations, and if not, it issues `set obs 121`. The number of observations is restored to the original value when the command finishes. However, if you press the *Break* key in the middle of the execution, the command will not be able to restore the original number of observations. For this reason, at the start the command reports the number of observations being processed. If you interrupted the command and had less than 121 observations to begin with, make sure to check for redundant observations.

4.5 Size of a combined graph

The `chernoff` command is smart enough to calculate the correct dimensions of the combined graph whether you choose to draw full or half faces and whether you specify the number of rows or columns or not. Because Stata does not allow the x size or y size of the graph to exceed 20, the command will rescale both dimensions proportionately if they do. You may need to adjust the size of the combined graph when you add a title and subtitle, change the position of the label, etc.

5 Conclusion

In 1994, Nel, Pitt, and Webb noted that the difficulty with the implementation of Chernoff faces is that "some companies may not have access to the facilities and skills with which to generate faces" (p. 253). They also expressed hope that "recent developments in the field of personal computers and user-friendly software may eventually alleviate this to some extent". The solution is long overdue.

The `chernoff` command, introduced in this article, is straightforward to use and does not require any special statistical skills. To my knowledge, this is a second implementation of Chernoff faces in a mainstream statistical software, with the first one being due to Friendly (1991). I hope that my command will prove useful in many diverse fields that rely on visualization of highly dimensional data to detect patterns, clusters, outliers, and temporal trends.

6 References

Apaiwongse, T. S. 1995. Facial display of environmental policy uncertainty. *Journal of Business and Psychology* 10: 65–74.

Astel, A., K. Astel, M. Biziuk, and J. Namieśnik. 2006. Classification of drinking water samples using the Chernoff's faces visualization approach. *Polish Journal of Environmental Studies* 15: 691–697.

Camussi, A., S. Raddi, and P. Raddi. 1992. Visual identification of forest-tree clones by using Chernoff's faces. *Taxon* 41: 451–458.

Chernoff, H. 1971. The use of faces to represent points in n-dimensional space graphically. Technical Report 71, Department of Statistics, Stanford University.

———. 1973. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association* 68: 361–368.

Flury, B. 1980. Construction of the asymmetrical face to represent multivariate data graphically. Technical Report 3, Institute of Mathematical Statistics and Actuarial Science, Bern University.

Flury, B., and H. Riedwyl. 1981. Graphical representation of multivariate data by means of asymmetrical faces. *Journal of the American Statistical Association* 76: 757–765.

Friendly, M. 1991. Faces: Faces display of multivariate data. Department of Psychology, York University. http://www.math.yorku.ca/SCS/sasmac/faces.html.

Golden, L. L., and M. Sirdesai. 1992. Chernoff faces: A useful technique for comparative image analysis and representation. *Advances in Consumer Research* 19: 123–128.

Johnson, R. A., and D. W. Wichern. 2007. *Applied Multivariate Statistical Analysis.* 6th ed. Upper Saddle River, NJ: Prentice Hall.

Lott, J. A., and T. C. Durbridge. 1990. Use of Chernoff faces to follow trends in laboratory data. *Journal of Clinical Laboratory Analysis* 4: 59–63.

Nel, D., L. Pitt, and T. Webb. 1994. Using Chernoff faces to portray service quality data. *Journal of Marketing Management* 10: 247–255.

Schüpbach, M. 1987. ASYMFACE asymmetrical faces in TurboPascal. Technical Report 25, Institute of Mathematical Statistics and Actuarial Science, Bern University.

About the author

Rafal Raciborski is a graduate student in the Department of Political Science at Emory University. His interests include political economy, applied econometrics, and statistical computing.