Paper 10760-2016

Visualizing Eye-Tracking Data with SAS®: Creating Heat Maps on Images

Matthew Duchnowski, Educational Testing Service

ABSTRACT

Few data visualizations are as striking or as useful as heat maps. Fortunately, there are many applications for them. A heat map that displays eye-tracking data is a particularly potent example: the intensity of a viewer's gaze is quantified and superimposed over the image being viewed. The resulting display is often stunning and informative.

Using Base SAS®, the author shows how to prepare, smooth and transform eye-tracking data and ultimately render it on top of the corresponding image. By customizing a graphical template and using specific Graphical Template Language (GTL) options, a heat map can be drawn precisely so that the user maintains pixel-level control. In this paper, eye-tracking data will be used, but the techniques provided are easily adapted to data from other fields such as sports, weather and marketing.

INTRODUCTION

Eye-tracking data is collected by observing a person and logging the coordinates of his/her gaze on an image. Historically, this has been done with a device mounted to the subject's head, but more recently, less invasive technologies have been used such as optical tracking. Figure 1 shows an image of skydivers. The image was presented to subjects and their gaze was recorded. The intensity of their gaze is represented through varying shades of color. In high-intensity areas – for instance, around the face and torso – the map glows a yellowish red. In low-intensity areas the map glows blue. Essentially, the result is a heat map drawn on top of an image.

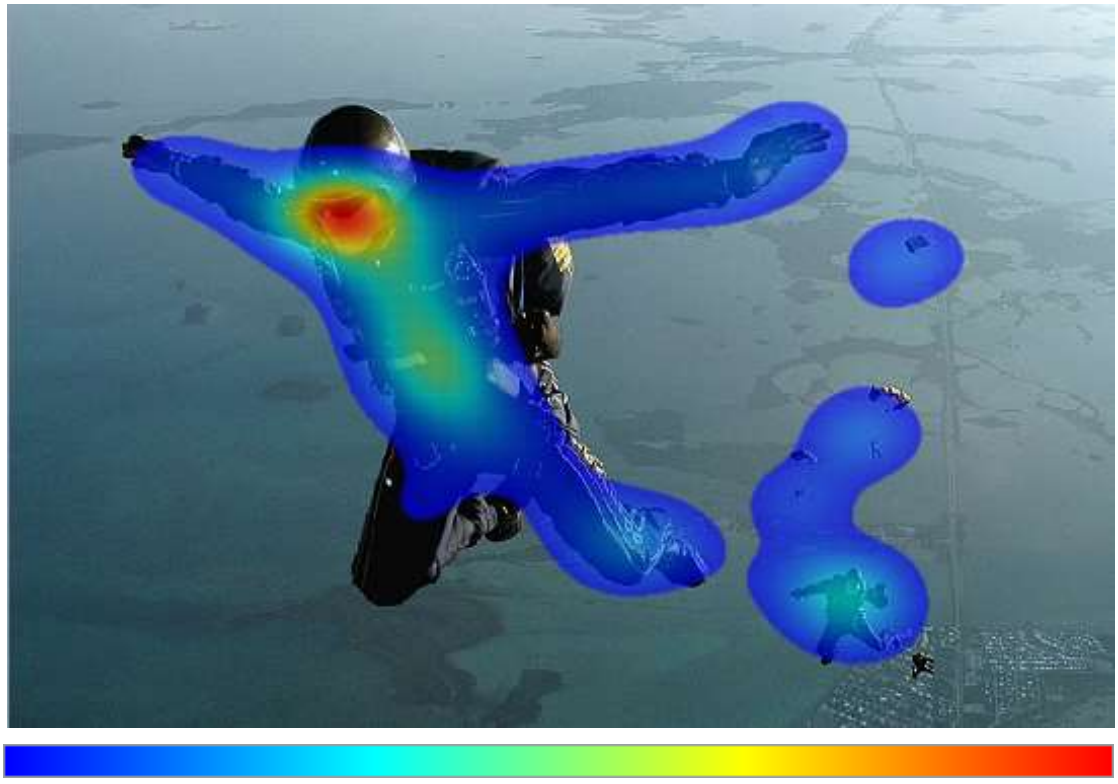


Figure 1. Eye-tracking image created using SAS GTL

WHAT IS A HEAT MAP?

In this paper, the term "heat map" is applied narrowly as a 2-dimensional plane colored at each coordinate in accordance with a 3rd dimensional value. In a more general sense, heat maps can involve tables, trees, topological maps or other data structures and a corresponding palette of colors, or "color ramp." Each unit in the structure has an associated value that is represented by a color in the color ramp. In this way, heat maps enable an additional dimension of data to be represented through the use of color.

Heat maps rely on the observer's ability to distinguish between different colors or shades of the same color. Consequently, creating them is sometimes more art than science. Often, choices are driven by aesthetics rather than exactitude, which is fine. The goal should be to tell a story with the data and not get bogged down with questions such as, "Is data point A twice as red as data point B?" or "Does orange communicate more intensity to the reader than lime green?"

PROCESSING EYE-TRACKING DATA

Eye-tracking data is stored as a series of *xy*- coordinates that are measured at regular intervals with each coordinate corresponding to a location on an image. These measurements are time-stamped or given an ordering ID. Table 1 shows a sample dataset that logs the location of a subject's gaze at regular intervals.

Time	x	y
1	15	10
2	14	8
3	10	4
…	…	…

Table 1. Basic structure of eye-tracking data

The convention for referencing image pixels is to use a zero-based 2-dimensional array with the origin at the upper left corner of the image.

Before manipulating this data, we will establish two critical global variables: the dimensions of our image.

```
%let width  = 600;
%let height = 400;
```

After reading the information in Table 1 to dataset `infile`, we can use PROC FREQ to calculate the frequency of each pixel, essentially projecting a third dimension, `z`. The variable `z` represents the intensity of the subject's gaze at the corresponding pixel (`x`, `y`).

```
proc freq data=infile noprint;
      tables x*y / out=freq (rename=(COUNT=z) drop=PERCENT);
run;
```

At this point, the output data set `freq` contains a bivariate distribution that is likely to be quite "jagged." While some local neighborhoods might have a high frequency, adjacent pixels might be completely empty. The heat map obtained in such a case would not be very appealing or informative and therefore some data smoothing is usually appropriate.

SMOOTHING DATA WITH PROC KDE

The KDE procedure provides a convenient way to generate kernel density estimates for multivariate data. The following code segment produces an output data set `density` that contains a density estimate for each pixel in our image:

```
ods graphics on;
ods output Levels=Levels;

      proc kde data=freq;
            bivar
                  x (gridl=0 gridu=%sysevalf(&width. -1, int) ngrid=&width.)
                  y (gridl=0 gridu=%sysevalf(&height.-1, int) ngrid=&height.)
            /
            bwm = 1.2 out=density plots=(contour) levels ;
            weight z;
      run;

ods output close;
ods graphics off;
```

By specifying the upper and lower grid limits (`gridu` and `gridl`), we ensure that the output dataset, `density`, represents each pixel in our image along with its new "smoothed" intensity value.

FILTERING OUT NOISE

In addition to `density`, the code has also produced the dataset `levels` that contains the density percentiles for each of the following default levels: 1, 5, 10, 50, 90, 95, 99 and 100. We will use this information to formulate a threshold below which data is considered to be noise.

The following code sets the macro variable `Threshold` and uses that threshold to blank out all density estimates for pixels that compose less than 1% of the bivariate distribution.

```
proc sql noprint;
      select Density into: Threshold from Levels where Percent = 1;
quit;

data image (keep = x y z); set density;
      rename value1=x value2=y density=z;
      if density lt &Threshold. then density = .;
run;
```

We account for these blank data points in our GTL template by using the option `includemissingcolor=FALSE` to prevent these pixels from receiving any additional coloration.

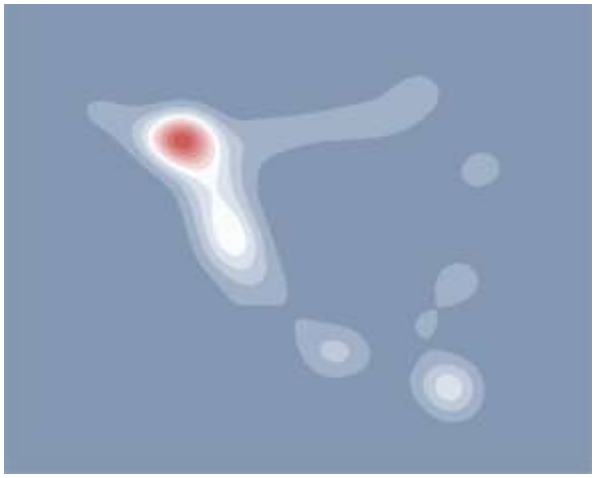
It should be noted that the selection of 1% is arbitrary here. Through experimentation, the reader may find that another threshold produces a more pleasant-looking plot for his or her purpose.

Adjusting the Bandwidth Multiplier

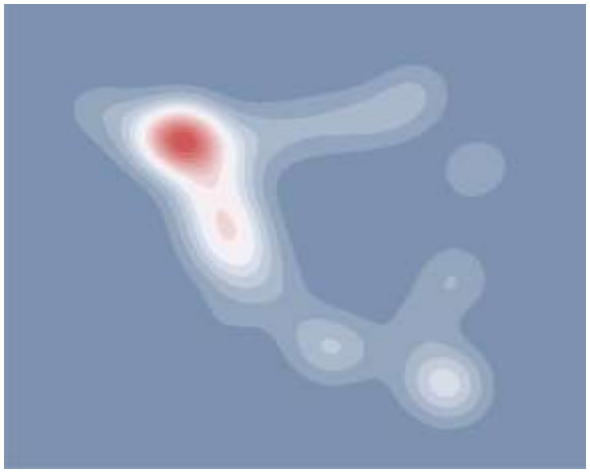
The KDE procedure shown above employs a parameter that warrants further discussion: the bandwidth multiplier, `bwm`. It is likely that the user will wish to alter this value for each individual plot in order to achieve the desired effect. Rather than discuss the technical significance of this multiplier and its effect on the density estimates, it suffices to show example plots produced for various values of `bwm`. The reader might find it useful to think of the bandwidth multiplier as "focus lens" for his or her data; blurring the image to some extent, but not so much that the content is unrecognizable.

bwm = .2

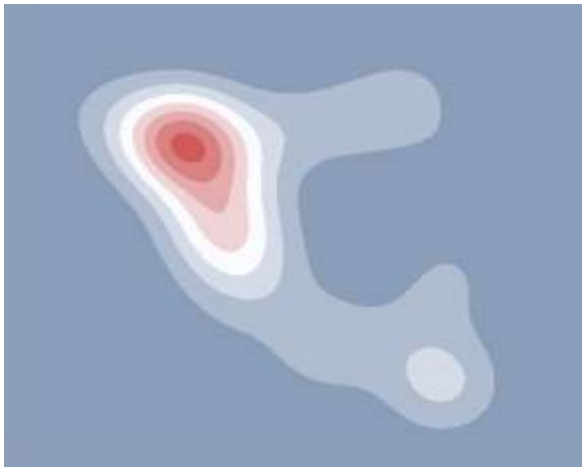
bwm = .6



bwm = .8



bwm = 1.2



Output 1. Contour plots from PROC KDE, shown with various smoothing parameters.

CREATE TEMPLATE USING GRAPHICAL TEMPLATE LANGUAGE (GTL)

Now that the data is prepared, we need only to render a heat map with an image background. Our approach will be to use `PROC TEMPLATE` to create a `statgraph` template called "eyemap." The template will then be used by `PROC SGRENDER`. We will show the template first and then make some notes about key parts of the syntax.

```
proc template; define statgraph eyemap;
    begingraph;
      layout overlay/

      xaxisopts=(
            offsetmin=0 offsetmax=0
            linearopts=(viewmin=0 viewmax=%sysevalf(&width. -1, int))
            label = "X")

      yaxisopts=(
            offsetmin=0 offsetmax=0
            linearopts=(viewmin=0 viewmax=%sysevalf(&height.-1, int))
            label = "Y"

            reverse=true)

      opaque=false
      walldisplay=none
      ;

      drawimage "C:\Pictures\Jumpers.jpg" /

            width =100  widthunit =percent
            height=100 heightunit =percent

            x=50 y=50

            drawspace=wallpercent
            layer=back
            transparency=0
      ;

      heatmapparm x=x y=y colorresponse=z/
            name="map"
            colormodel =(blue cyan yellow red)
            xgap=0 ygap=0
            datatransparency=.8
            xbinaxis=false ybinaxis=false
            includemissingcolor=FALSE;

      continuouslegend "map" /
            title="Intensity of Subject's Gaze"
            location=outside
            valign=bottom ;

      endlayout;
    endgraph;
  end;
run;
```

KEY FEATURES OF TEMPLATE

DRAWING THE IMAGE

The `drawimage` statement directs the template to the location of our image on the user's system. The option `drawspace=wallpercent` directs the image to be displayed on the display's WALLSPACE and to be specified by the user as a *percentage*. This specification is given as `x=50` and `y=50`, which directs the rendering process to center the image horizontally and vertically. By specifying `layer=back`, the image is ultimately rendered behind the data and made to be completely opaque by setting `transparency=0.`

AXIS, WALLDISLPAY AND OPAQUE OPTIONS

When plotting data, SAS by default supplies some additional padding so that data points are displayed comfortably within the bounds of the axes. Setting `offsetmin=0` and `offsetmax=0` within the `xaxisopts` / `yaxisopts` ensures that there is no such additional padding.

By eliminating the extra padding within the DATASPACE, we are essentially expanding the DATASPACE to the point where it is perfectly aligned with the WALLSPACE, and the WALLSPACE defines the space where our image is rendered. As a result, the axes form a snug frame around both the data and the image.

The plot's data will be visible but the WALLSPACE threatens to block the view of our image and so we need to set `walldisplay=none`. The layout background must be set to translucent, which can be achieved using `opaque=false`.

Lastly, the *y*-axis is reversed in compliance with the convention used in image processing. The image coordinates are zero-based with the origin at the upper left corner of the image. This reversal is accomplished by setting `reverse=true` under the `yaxisopts` specification.

HEATMAPPARM

The heat map parameters are specified within the `heatmapparm` statement. Nonoverlapping cells will fill the entire DATASPACE and will be colored according to the `colorresponse` variable, z. We ensure that these cells are adjacent by setting the gap between them to be of size zero: `xgap=0, ygap=0`.

Since each coordinate previously determined to be "noise" was set to a blank value (`z= .`) , we must now suppress the coloration of these points by setting `includemissingcolor=FALSE` (This feature applies to the second maintenance release of SAS 9.4 and later, and should be omitted entirely if working with an earlier version).

Because we want our image to be seen through the heat map, the data display must be semi-transparent. The author has used `datatransparency=.8`, but encourages individual users to manipulate this value to suit their own needs. This parameter may be set anywhere from 0 (opaque) to 1.0 (invisible).

One unfortunate side effect of setting `datatransparency` is that the legend in the output is also made transparent. The reader may get around this issue by rendering the heat map once with the transparency set to zero and once with the transparency set to the desired value between 0 and 1. The user is then required to combine the two outputs by cutting and pasting the legend from the opaque display.

COLORMODEL

The user must create a color ramp or choose one of four native color ramps: `TwoColorRamp`, `TwoColorAltRamp`, `ThreeColorRamp` or `ThreeColorAltRamp`. A color ramp might be chosen to accentuate a particular finding or it might be chosen to complement/contrast the colors in the background image. As shown in the examples below, a ramp is specified using the `colormodel` option.

```
colormodel = ThreeColorRamp
```

```
colormodel = ThreeColorAltRamp
```

```
colormodel = (blue cyan yellow red)
```

```
colormodel = (white black)
```

```
colormodel = (red orange yellow green blue indigo violet)
```

```
colormodel = (red pink pink pink white)
```

```
colormodel = (red red pink white)
```

```
colormodel = (CX6497EB CXFFFFFF CXDD6060)
```

RENDERING THE IMAGE

The template “eyemap” generated by PROC TEMPLATE is used in a short SGRENDER procedure as shown below. The resulting data graph is very rich with detail and could benefit from antialiasing. The maximum value can be set manually to the product of our image dimensions.

```
ods graphics / antialiasmax=%sysevalf(&width.*&height., int);
      proc sgrender data=image template=eyemap;
      run;
ods graphics off;
```

Despite the user’s effort to employ antialiasing, the rendered image may still suffer minor defects. This depends on image size and limitations of the operating system. In this case, the user may find it beneficial to reproduce the image at varying `height` and `width` values (see `ods graphics` options).

CONCLUSION

Heat maps immediately draw the readers' eye and leave a remarkable impact. Although eye-tracking data was used in this discussion, there are many useful applications that involve identical methods. To inspire the reader, we conclude the discussion with several examples:

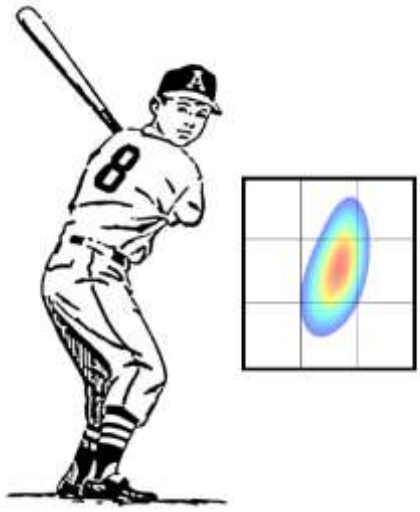


Figure 2a. baseball pitcher performance

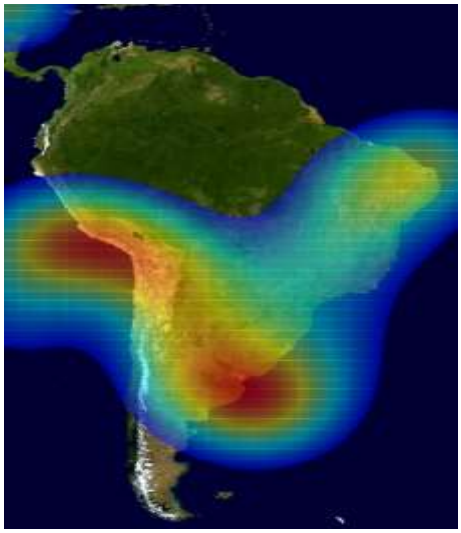


Figure 2b. South American weather patterns

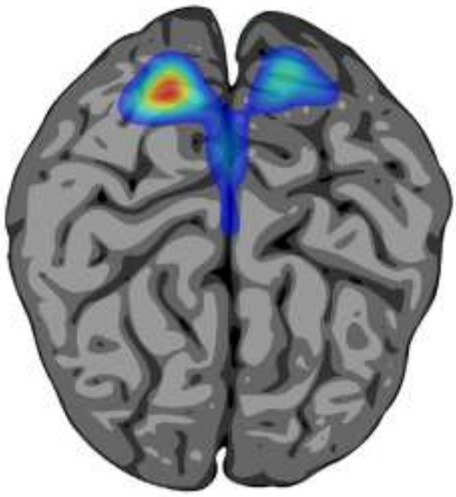


Figure 2c. human brain activity

REFERENCES

Heath, D. (2011). Now you can annotate your statistical graphics procedure graphs. In *Proceedings of the SAS® Global Forum 2011 Conference*. Retrieved from https://support.sas.com/resources/papers/proceedings11/277-2011.pdf

Matange, S. (2014). Annotate your SGPLOT graphs. In *PharmaSUG 2014 conference proceedings*. Retrieved from http://www.pharmasug.org/proceedings/china2014/CC/PharmaSUG-China-2014-CC01.pdf

SAS Institute Inc. (2011). *SAS/STAT® 9.3 user's guide.* Retrieved from http://support.sas.com/documentation/cdl/en/statug/63962/PDF/default/statug.pdf

SAS Institute Inc. (2015). *SAS® 9.4 graph template language: User's guide* (4th ed.). Retrieved from http://support.sas.com/documentation/cdl/en/grstatgraph/67882/PDF/default/grstatgraph.pdf

ACKNOWLEDGMENTS

I would like to thank all friends and colleagues at ETS who are endlessly willing to discuss topics such as those discussed here, particularly Gary Feng who shared some insights early in the process. Special thanks to Amanda Cirillo and Yan Zhang for their feedback on this paper. Lastly, I would like to thank my wife Robin, who provides me with unending support in all that I do.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Matthew Duchnowski
Educational Testing Service
Rosedale Rd, MS 20T Princeton, NJ 08541
(609) 683-2939
mduchnowski@ets.org