Creating contour and ‘heat map’ graphs to display PITCHf/x data

Dave Allen
PITCHf/x Summit
July 11, 2009
Plotting a value over two variables

We often want to display pitchf/x data over two variables. Examples:

- Location of pitches in the strike zone
Plotting a value over two variables

We often want to display pitchf/x data over two variables. Examples:

• Location of pitches in the strike zone
• Home run rate (swing rate, whiff rate, BABIP) by pitch location.
Plotting a value over two variables

We often want to display pitchf/x data over two variables. Examples:

- Location of pitches in the strike zone
- Home run rate (swing rate, whiff rate, BABIP) by pitch location.
- Any of those rates by vertical and horizontal movement of a pitch.
Plotting a value over two variables

We often want to display pitchf/x data over two variables. Examples:

• Location of pitches in the strike zone
• Home run rate (swing rate, whiff rate, BABIP) by pitch location.
• Any of those rates by vertical and horizontal movement of a pitch.
• Speed off the bat by pitch location.
Plotting a value over two variables

We often want to display pitchf/x data over two variables. Examples:

• Location of pitches in the strike zone
• Home run rate (swing rate, whiff rate, BABIP) by pitch location.
• Any of those rates by vertical and horizontal movement of a pitch.
• Speed off the bat by pitch location.
• Vertical angle of the ball off the bat by vertical location and vertical movement of a pitch.
Plotting a value over two variables: Scatter plots

Under certain conditions this is possible with a scatter plot.
Plotting a value over two variables: Scatter plots

Under certain conditions this is possible with a scatter plot.

- When the number of pitches is small.
Plotting a value over two variables: Scatter plots

Under certain conditions this is possible with a scatter plot.

• When the number of pitches is small.
• When they are well clustered.
Plotting a value over two variables: Scatter plots

Under certain conditions this is possible with a scatter plot.

• When the number of pitches is small.
• When they are well clustered.
• When they take categorical values.
Plotting a value over two variables: Scatter plots not appropriate

When these conditions are not met a scatter plot may not be the best way to display the data.
Plotting a value over two variables: Scatter plots not appropriate

When these conditions are not met a scatter plot may not be the best way to display the data.

- Too many pitches
Plotting a value over two variables: Scatter plots not appropriate

When these conditions are not met a scatter plot may not be the best way to display the data.

- Too many pitches
- Non-categorical value
Plotting a value over two variables: Alternative to scatter plots

- I suggest in such cases a contour map or ‘heat map’ is a possible alternative.
Plotting a value over two variables: Alternative to scatter plots

- I suggest in such cases a contour map or ‘heat map’ is a possible alternative.
- These maps do not actually plot the data (pitches), but a surface fit to the data. As if you fit a curve to two dimensional data and then just plot curve, without the points.
Plotting a value over two variables: Alternative to scatter plots

- I suggest in such cases a contour map or ‘heat map’ is a possible alternative.
- These maps do not actually plot the data (pitches), but a surface fit to the data. As if you fit a curve to two dimensional data and then just plot curve, without the points.
- This talk will be broken into two parts:
  1) methods of fitting a surface
  2) creating the contour map once we have the surface.
Fitting a plane

The simplest surface to fit is a plane (analogous to fitting a line).
Fitting a plane

The simplest surface to fit is a plane (analogous to fitting a line).

```
mymodel <- lm( hit_v_ang ~ p_z + pfz_z )
```
The simplest surface to fit is a plane (analogous to fitting a line).

```r
mymodel <- lm( hit_v_ang ~ p_z + pfx_z )
summary(mymodel)
```
Fitting a plane

The simplest surface to fit is a plane (analogous to fitting a line).

mymodel <- lm( hit_v_ang ~ p_z + pfx_z )

summary(mymodel)

| Coefficients: | Estimate | Std. Error | t value | Pr(>|t|) |
|---------------|----------|------------|---------|----------|
| (Intercept)   | -13.55922| 1.11985    | -12.11  | <2e-16   *** |
| p_z           | 6.59482  | 0.40371    | 16.34   | <2e-16   *** |
| pfx_z         | 1.27491  | 0.07063    | 18.05   | <2e-16   *** |

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25.15 on 11168 degrees of freedom
Multiple R-squared: 0.06024,   Adjusted R-squared: 0.06007
F-statistic: 357.9 on 2 and 11168 DF,  p-value: < 2.2e-16
Fitting a plane
Fitting a ‘step’-surface by binning the data

• The plane is very limited, especially when looking at values of pitch location because the strike zone presents a ‘non-linearity.’
• A 2d step function provides more flexibility.
Fitting a ‘step’-surface by binning the data

• The plane is very limited, especially when looking at values of pitch location because the strike zone presents a ‘non-linearity.’
• A 2d step function provides more flexibility.
Fitting a ‘step’-surface by binning the data
Fitting a ‘step’-surface by binning the data
Fitting a ‘step’-surface by binning the data
Fitting a ‘step’-surface by binning the data
Fitting a ‘step’-surface by binning the data
Fitting a ‘step’-surface by binning the data
Fitting a smoother ‘step’-surface: distance weighted average

- Alternatively put a lattice over the area and find the distance weighted average for each point to the lattice.
- Creates a smoother surface
- With large data sets it can be very computationally intensive
Fitting a smoother ‘step’-surface: distance weighted average
Fitting a smoother ‘step’-surface: distance weighted average
Fitting a smoother ‘step’-surface: distance weighted average
Fitting a smoother ‘step’-surface: distance weighted average
Fitting a smoother ‘step’-surface: distance weighted average
Fitting a smoother ‘step’-surface: distance weighted average

Soriano HRs

Soriano HRs

\[
\frac{1}{\exp(0.229)} = 0.80
\]
Fitting a smoother ‘step’-surface: distance weighted average
Fitting a surface by LOESS regression

- Modern regression method technique that fits a low-degree polynomial to each point in the data set. The polynomial is fit using a weighted least squares, giving more weight to neighboring points nearer the focal point.
Fitting a surface by LOESS regression

- Modern regression method technique that fits a low-degree polynomial to each point in the data set. The polynomial is fit using a weighted least squares, giving more weight to neighboring points nearer the focal point.
- Very flexible, but quite computationally intense and does not produce a closed form mathematical expression for the fit surface like a fit plane (or polynomial surface would).
Fitting a surface by LOESS regression

- Modern regression method technique that fits a low-degree polynomial to each point in the data set. The polynomial is fit using a weighted least squares, giving more weight to neighboring points nearer the focal point.
- Very flexible, but quite computationally intense and does not produce a closed form mathematical expression for the fit surface like a fit plane (or polynomial surface would).

mymodel <- loess( hr ~ p_x + p_z, span=.75 )
Fitting a surface by LOESS regression

- Modern regression method technique that fits a low-degree polynomial to each point in the data set. The polynomial is fit using a weighted least squares, giving more weight to neighboring points nearer the focal point.
- Very flexible, but quite computationally intense and does not produce a closed form mathematical expression for the fit surface like a fit plane (or polynomial surface would).

mymodel <- \texttt{loess}( \texttt{hr \sim p_x + p_z, span=\texttt{.75}} )
Fitting a surface by LOESS regression

- alpha = .75
- alpha = .25
Comparison
Displaying the fit surface in R: image
Displaying the fit surface in R: image

mysurface <- read.csv('~/Documents/pitchfx_pres/mysurface.csv',
                       header=FALSE)
mysurface <- as.matrix(mysurface)
Displaying the fit surface in R: \texttt{image}

mysurface <- \texttt{read.csv} '~/Documents/pitchfx_pres/mysurface.csv', header=FALSE)

mysurface <- \texttt{as.matrix} (mysurface)

\texttt{image} (x=\texttt{seq} (from=-1.5, to=1.5, length=20),
             y=\texttt{seq} (from=1, to=3.75, length=25),
             z=mysurface,
             col=\texttt{hsv} (h=\texttt{seq} (from=2/3, to=0, length=20), s=1, v=1)
Displaying the fit surface in R: `image`

```r
dsurface <- read.csv '~/Documents/pitchfx_pres/mysurface.csv', header=FALSE)
surface <- as.matrix(surface)
image(x=seq(from=-1.5, to=1.5, length=20),
      y=seq(from=1, to=3.75, length=25),
      z=surface,
      col=hsv(h=seq(from=2/3, to=0, length=20)))
```
mysurface <- read.csv('~/Documents/pitchfx_pres/mysurface.csv',
                   header=FALSE)
mysurface <- as.matrix(mysurface)
image(x=seq(from=-1.5,to=1.5,length=20),
      y=seq(from=1,to=3.75,length=25),
      z=mysurface,
      col= hsv(h=seq(from=2/3,to=0,length=20),
      top <- 3.42
bot <- 1.56
lines(c(-1,-1),c(bot,top))
lines(c(1,1),c(bot,top))
lines(c(-1,1),c(top,top))
lines(c(-1,1),c(bot,bot))
#points(...)
Displaying the fit surface in R: `image`

```r
mysurface <- read.csv('~/Documents/pitchfx_pres/mysurface.csv',
   header=FALSE)
mysurface <- as.matrix(mysurface)
image(x=seq(from=-1.5,to=1.5,length=20),
   y=seq(from=1,to=3.75,length=25),
   z=mysurface,
   col=hsv(h=seq(from=2/3,to=0,length=20),s=1,v=1)
)

top <- 3.42
bot <- 1.56
lines(c(-1,-1),c(bot,top))
lines(c(1,1),c(bot,top))
lines(c(-1,1),c(bot,top))
lines(c(-1,1),c(top,top))
lines(c(-1,1),c(bot,bot))
#points(...)```
Displaying the fit surface in R: `contour`

```r
mysurface <- read.csv '~/Documents/pitchfx_pres/mysurface.csv',
    header=FALSE)
mysurface <- as.matrix(mysurface)
```
Displaying the fit surface in R: `contour`

```r
mysurface <- read.csv('~/Documents/pitchfx_pres/mysurface.csv',
                      header=FALSE)
mysurface <- as.matrix(mysurface)

contour(x=seq(from=-1.5,to=1.5,length=20),
        y=seq(from=1,to=3.75,length=25),
        z=mysurface,
        levels=c(75,77.5,80,81,82,82.5,83,83.5,84,84.5,85)
        #nlevels=15
)
```
Displaying the fit surface in R: `contour`

```r
mysurface <- read.csv('~/Documents/pitchfx_pres/mysurface.csv', header=FALSE)
mysurface <- as.matrix(mysurface)

contour(x=seq(from=-1.5, to=1.5, length=20),
        y=seq(from=1, to=3.75, length=25),
        z=mysurface,
        levels=c(75, 77.5, 80, 81, 82, 82.5, 83, 83.5, 84, 84.5, 85)
        #nlevels=15
)

top <- 3.42
bot <- 1.56
lines(c(-1,-1), c(bot,top),col='red')
lines(c(1,1), c(bot,top),col='red')
lines(c(-1,1), c(top,top),col='red')
lines(c(-1,1), c(bot,bot),col='red')
```
Displaying the fit surface in R: `contour`

given: 

```r
mysurface <- read.csv(‘~/Documents/pitchfx_pres/mysurface.csv’,
header=FALSE)
mysurface <- as.matrix(mysurface)

contour(x=seq(from=-1.5,to=1.5,length=20),
        y=seq(from=1,to=3.75,length=25),
        z=mysurface,
        levels=c(75,77.5,80,81,82,82.5,83,83.5,84,84.5,85)
        #nlevels=15
)

# coordinates:

top  <- 3.42
bot  <- 1.56
lines(c(-1,-1),c(bot,top),col=‘red’)
lines(c(1,1),c(bot,top),col=‘red’)
lines(c(-1,1),c(top,top),col=‘red’)
lines(c(-1,1),c(bot,bot),col=‘red’)
```
Displaying the fit surface in R:

```r
mysurface <- read.csv(`~/Documents/pitchfx_pres/mysurface.csv`,
                        header=FALSE)

mysurface <- as.matrix(mysurface)
top <- 3.42
bot <- 1.56
```
Displaying the fit surface in R:

```r
mysurface <- read.csv('~/Documents/pitchfx_pres/mysurface.csv', header=FALSE)

mysurface <- as.matrix(mysurface)
top <- 3.42
bot <- 1.56

filled.contour(x=seq(from=-1.5,to=1.5,length=20),
               y=seq(from=1,to=3.75,length=25),
               z=mysurface,
               nlevels=27,
               col=hsv(h=seq(from=2/3,to=0,length=27),s=1,v=1),
               plot.axes={lines(c(-1,-1),c(bot,top)),
                          lines(c(1,1),c(bot,top)),
                          lines(c(-1,1),c(top,top))
                         lines(c(-1,1),c(bot,bot))
               })
```
Displaying the fit surface in R: `filled.contour`

```r
mysurface <- read.csv('~/Documents/pitchfx_pres/mysurface.csv', header=FALSE)
mysurface <- as.matrix(mysurface)
top <- 3.42
bot <- 1.56
filled.contour(x=seq(from=-1.5,to=1.5,length=20),
y=seq(from=1,to=3.75,length=25),
z=mysurface,
levels=27,
col=hsv(h=seq(from=2/3,to=0,length=27),s=1,v=1),
plot.axes=
lines(c(-1,-1),c(bot,top),
      lines(c(1,1),c(bot,top))
      lines(c(-1,1),c(top,top))
      lines(c(-1,1),c(bot,bot))
)
```

Comparison
Displaying the fit models in R

mymodel <- loess( hr ~ p_x + p_z )
#mymodel <- lm(v_ang ~ p_z + pfx_z )
Displaying the fit models in R

```r
mymodel <- loess( hr ~ p_x + p_z )
#mymodel <- lm(v_ang ~ p_z + pfx_z )

myx <- matrix(data=seq(from=-2,to=2,length=20),nrow=20,ncol=25)
myz <- t(matrix(data=seq(from=-2,to=2,length=20),nrow=20,ncol=25))
```
Displaying the fit models in R

```r
mymodel <- loess( hr ~ p_x + p_z )
#mymodel <- lm(v_ang ~ p_z + pfx_z )

myx <- matrix(data=seq(from=-2,to=2,length=20),nrow=20,ncol=25)
myz <- t(matrix(data=seq(from=-2,to=2,length=20),nrow=20,ncol=25))

mypredict <- predict(object=mymodel, 
                      newdata=data.frame(p_x=as.vector(myx),
                                          p_z=as.vector(myz))
)
```
Displaying the fit models in R

```r
mymodel <- loess( hr ~ p_x + p_z )
#mymodel <- lm(v_ang ~ p_z + pfx_z )

myx <- matrix(data=seq(from=-2,to=2,length=20),nrow=20,ncol=25)
myz <- t(matrix(data=seq(from=-2,to=2,length=20),nrow=20,ncol=25))

mypredict <- predict(object=mymodel, 
                      newdata=data.frame(p_x=as.vector(myx), 
                                        p_z=as.vector(myz) )
)

mypredict<-matrix(data=mypredict,nrow=20,nrow=25)
image(x=seq(from=-2,to=2,length=20),
     y=seq(from=0,to=5,length=25),
     z=mypredict,
     col=...
)
```
Displaying the fit surface in Excel
Displaying the fit surface in Excel
Displaying the fit surface in Excel