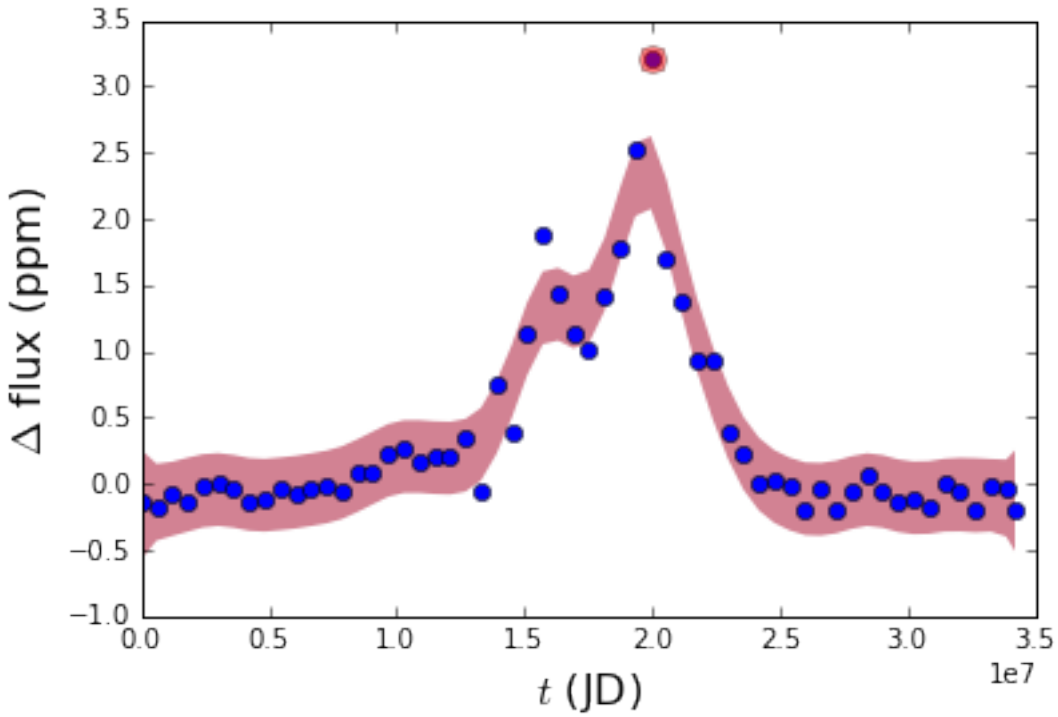
looking_for_outliers

March 17, 2015

In [177]:
```
#Example adapted from
#  http://bit.ly/REH4qj

%matplotlib inline

import astropy.io.fits as fits
import pylab as pl
import numpy as np

from sklearn.gaussian_process import GaussianProcess

hdulist = fits.open('latspec_2FGLJ0221.0+3555_week.fits')
data = hdulist[1].data
y = (data.field('flux') - np.median(data.field('flux')))*1e6
x = data.field('time') - np.min(data.field('time'))

#Plot the original data
pl.plot(x, y, label='orig data', marker='o', ls='')
pl.xlabel("$t$ (JD)", fontsize=16)
pl.ylabel("$\Delta$ flux (ppm)", fontsize=16)

GP_x = np.atleast_2d(x).T
GP_y = np.atleast_2d(y).T

gp = GaussianProcess(corr='cubic', theta0=1e-2, thetaL=1e-4, thetaU=1e-1,
                     nugget=1e-6)

# Fit to data using Maximum Likelihood Estimation of the parameters
gp.fit(GP_x, GP_y)

# Make the prediction on the meshed x-axis (ask for MSE as well)
y_pred, MSE = gp.predict(GP_x, eval_MSE=True)
y_pred = y_pred[:,0]
sigma = np.sqrt(MSE)
confidence_interval = 2.236 * sigma
pl.fill_between(x, y_pred - confidence_interval, y_pred + confidence_interval,
                alpha=.5, facecolor='#A60628', edgecolor='None')

pl.plot(x[(y - y_pred)/confidence_interval > 3.], y[(y - y_pred)/confidence_interval > 3.],
        marker = 'o', ms = 10, color='red', alpha=0.5)
```

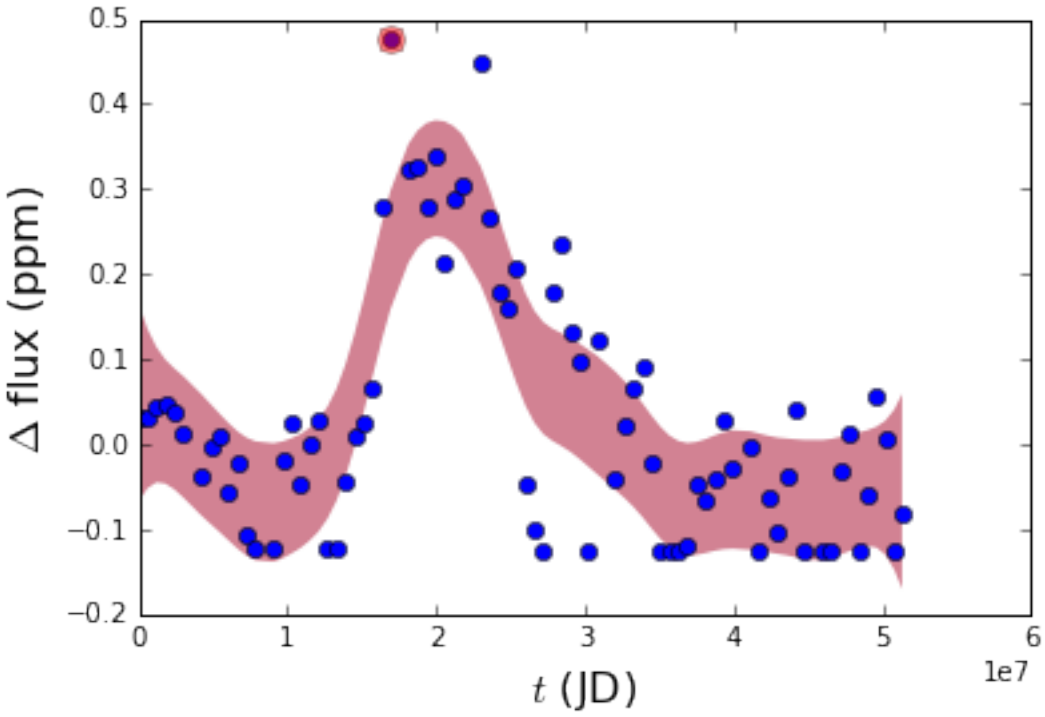
Out[177]:
```
[<matplotlib.lines.Line2D at 0x11862ce90>]
```




```
In [178]: hdulist = fits.open('latspec_2FGLJ0742.6+5442_week.fits')
          data = hdulist[1].data
          y = (data.field('flux') - np.median(data.field('flux')))*1e6
          x = data.field('time') - np.min(data.field('time'))

          #Plot the original data
          pl.plot(x, y, label='orig data', marker='o', ls='')
          pl.xlabel("$t$ (JD)", fontsize=16)
          pl.ylabel("$\Delta$ flux (ppm)", fontsize=16)

          GP_x = np.atleast_2d(x).T
          GP_y = np.atleast_2d(y).T

          gp = GaussianProcess(corr='cubic', theta0=1e-2, thetaL=1e-4, thetaU=1e-1,
                               nugget=1e-6)

          # Fit to data using Maximum Likelihood Estimation of the parameters
          gp.fit(GP_x, GP_y)

          # Make the prediction on the meshed x-axis (ask for MSE as well)
          y_pred, MSE = gp.predict(GP_x, eval_MSE=True)
          y_pred = y_pred[:,0]
          sigma = np.sqrt(MSE)
          confidence_interval = 2.236 * sigma
          pl.fill_between(x, y_pred - confidence_interval, y_pred + confidence_interval,
                          alpha=.5, facecolor='#A60628', edgecolor='None')
```

```
        pl.plot(x[(y - y_pred)/confidence_interval > 3.], y[(y - y_pred)/confidence_interval > 3.],
                marker = 'o', ms = 10, color='red', alpha=0.5,ls='')
```

Out[178]: [<matplotlib.lines.Line2D at 0x1179409d0>]



In [179]:
```
hdulist = fits.open('latspec_2FGLJ2253.9+1609_week.fits')
data = hdulist[1].data
y = (data.field('flux') - np.median(data.field('flux')))*1e6
x = data.field('time') - np.min(data.field('time'))

#Plot the original data
pl.plot(x, y, label='orig data', marker='o', ls='')
pl.xlabel("$t$ (JD)", fontsize=16)
pl.ylabel("$\Delta$ flux (ppm)", fontsize=16)

GP_x = np.atleast_2d(x).T
GP_y = np.atleast_2d(y).T

gp = GaussianProcess(corr='cubic', theta0=1e-2, thetaL=1e-4, thetaU=1e-1,
                     nugget=1e-6)

# Fit to data using Maximum Likelihood Estimation of the parameters
gp.fit(GP_x, GP_y)

# Make the prediction on the meshed x-axis (ask for MSE as well)
y_pred, MSE = gp.predict(GP_x, eval_MSE=True)
y_pred = y_pred[:,0]
```

```
sigma = np.sqrt(MSE)
confidence_interval = 2.236 * sigma
pl.fill_between(x, y_pred - confidence_interval, y_pred + confidence_interval,
                alpha=.5, facecolor='#A60628', edgecolor='None')

pl.plot(x[(y - y_pred)/confidence_interval > 3.], y[(y - y_pred)/confidence_interval > 3.],
        marker = 'o', ms = 10, color='red', alpha=0.5, ls='')
```

Out[179]: [<matplotlib.lines.Line2D at 0x11b908850>]