Python Workshop
 Plotting with `matplotlib` 

Joseph D. Hughes

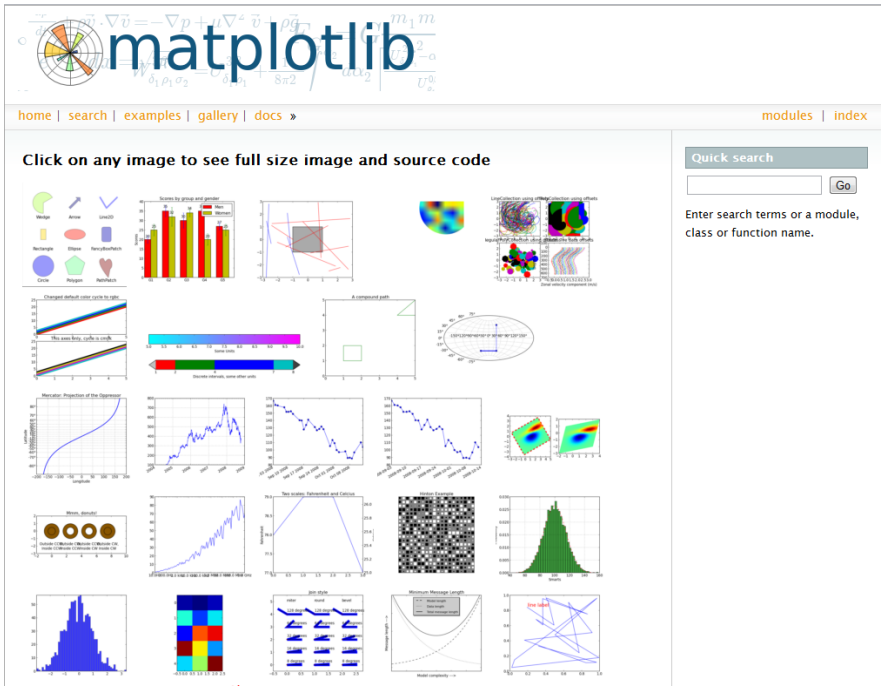
U.S. Geological Survey
Florida Water Science Center, Tampa, Florida USA

USGS National Groundwater Workshop, August 2012

Background information

`matplotlib` resources:

`http://www.matplotlib.sourceforge.net`



USGS science for a changing world

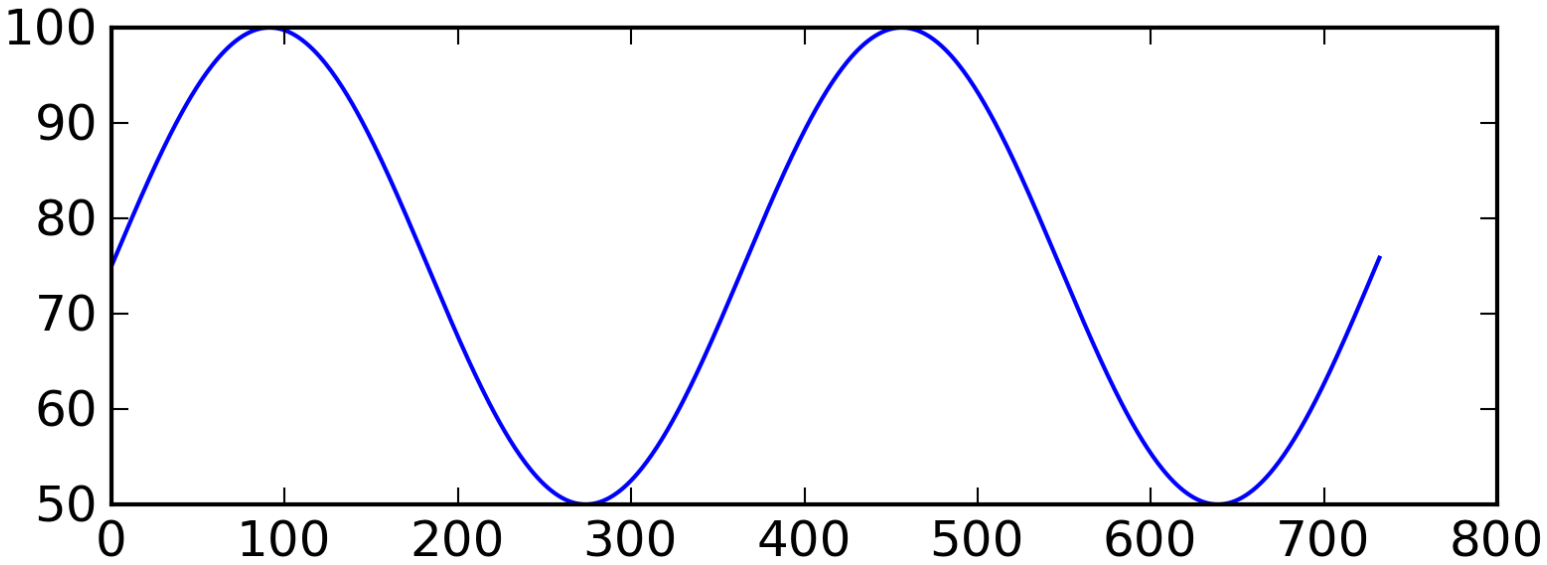
Creating a super simple plot (1)

SuperSimplePlot.py

```
1  import numpy as np
2  import pylab as pl
3  import matplotlib as mpl
4  #--load flow data
5  q = np.genfromtxt( '..\\data\\USInflow.dat', skip_header=4 )
6  #--create figure of upstream inflow
7  fig = pl.figure( figsize=(6.0, 2.0), facecolor='w' )
8  #--define the subplot
9  ax = fig.add_subplot(1,1,1)
10 #--plot the data
11 ax.plot(q[:,0],q[:,1])
12 #--output figure
13 #--png
14 fig.savefig('..\\figures\\SuperSimplePlot.png',dpi=300)
```

Creating a super simple plot (2)

SuperSimplePlot.py



Creating a not so simple plot (1)

SimplePlot.py

```
1  import sys
2  import string
3  import math
4  import numpy as np
5  import pylab as pl
6  import matplotlib as mpl
7  from matplotlib.font_manager import FontProperties
8  #--general specification data for matplotlib
9  mpl.rcParams['font.sans-serif']          = 'Univers 57 Condensed'
10 mpl.rcParams['font.serif']               = 'Times'
11 mpl.rcParams['font.cursive']             = 'Zapf Chancery'
12 mpl.rcParams['font.fantasy']             = 'Comic Sans MS'
13 mpl.rcParams['font.monospace']           = 'Courier New'
14 mpl.rcParams['mathtext.default']         = 'regular'
15 mpl.rcParams['pdf.compression']          = 0
16 mpl.rcParams['pdf.fonttype']             = 42
17 #--figure text sizes
18 mpl.rcParams['legend.fontsize']  = 7
19 mpl.rcParams['axes.labelsize']   = 8
20 mpl.rcParams['xtick.labelsize']  = 7
21 mpl.rcParams['ytick.labelsize']  = 7
```

Creating a not so simple plot (2)

SimplePlot.py

```
22  #--create upstream inflow data
23  #--temporal dimensions
24  nper    = 365 * 2 + 1
25  ntsp    = np.ones( (nper), np.int )
26  tsp_len = 1.0 #day
27  tmax    = float( nper ) * tsp_len
28  simtime = np.arange(0.0,tmax+2.*tsp_len,tsp_len)
29  #--generate a sinusoidal inflow function
30  q = np.zeros( (len(simtime)), np.float )
31  qbase, qptrb = 75.00, 25.00
32  tp   = 365.
33  ipos = 0
34  for ipos,t in enumerate( simtime ):
35      qp = qptrb * math.sin( 2.0 * math.pi * t / tp )
36      q[ipos] = qbase + qp
```

Creating a not so simple plot (3)

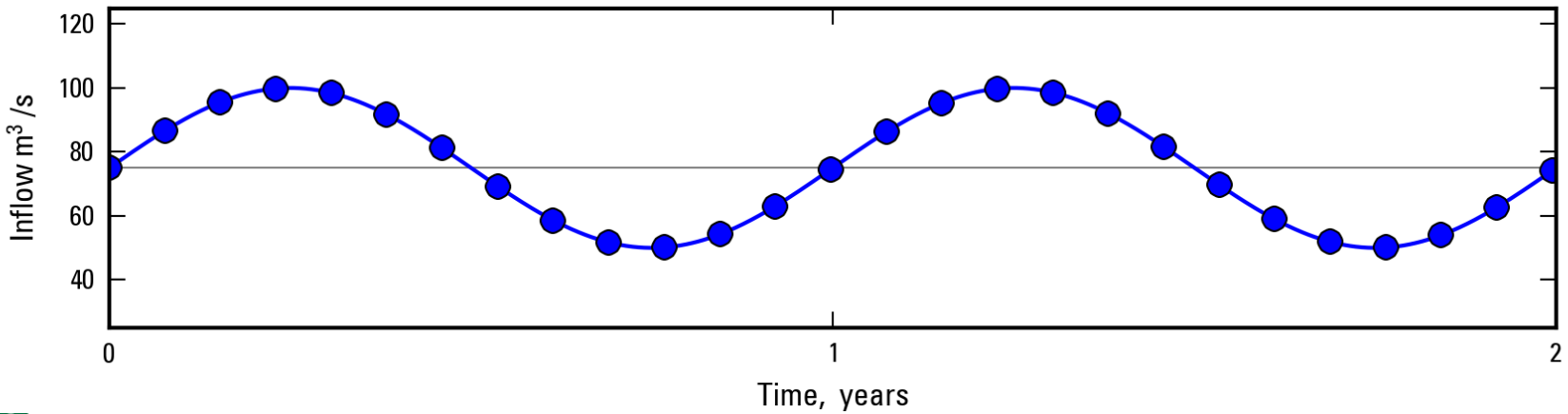
SimplePlot.py

```
37  #--create figure of upstream inflow
38  #--how big to make the figure and where to place it
39  fwid, fhgt = 6.00, 1.50
40  flft, frgt = 0.10, 0.95
41  fbot, ftop = 0.20, 0.95
42  fig = pl.figure( figsize=(fwid, fhgt), facecolor='w' )
43  fig.subplots_adjust(wspace=0.25,hspace=0.25,left=flft,right=frgt,bottom=fbot,top=ftop)
44  #--define the subplot
45  ax = fig.add_subplot(1,1,1)
46  #--plot the data
47  ax.plot([0,10], [qbase,qbase], color='0.5', linewidth=0.5, label='_Zero')
48  ax.plot(simtime/365,q, color='b', linewidth=1, label='Inflow', marker='o', markevery=28)
49  #--titles and axes
50  ax.set_ylabel( r'Inflow $m^3/s$' )
51  ax.set_ylim(25,125)
52  ax.set_xlabel('Time,  years')
53  ax.set_xlim(0,2)
54  ax.set_xticks( np.arange(0,3,1) )
```

Creating a not so simple plot (4)

SimplePlot.py

```
55  #--output figure
56  #--png
57  outfigpng = '..\\figures\\Inflow.png'
58  fig.savefig(outfigpng,dpi=300)
59  print 'created...', outfigpng
60  #--pdf
61  outfigpdf = '..\\figures\\Inflow.pdf'
62  fig.savefig(outfigpdf,dpi=300)
63  print 'created...', outfigpdf
```



USGS science for a changing world

Creating a not so simple plot (5)

http://matplotlib.sourceforge.net/api/pyplot_api.html#matplotlib.pyplot.plot

The *kwargs* can be used to set line properties (any property that has a `set_*` method). You can use this to set a line label (for auto legends), linewidth, anitialising, marker face color, etc. Here is an example:

```
plot([1,2,3], [1,2,3], 'go-', label='line 1', linewidth=2)
plot([1,2,3], [1,4,9], 'rs',  label='line 2')
axis([0, 4, 0, 10])
legend()
```

If you make multiple lines with one plot command, the kwargs apply to all those lines, e.g.:

```
plot(x1, y1, x2, y2, antialised=False)
```

Neither line will be antialiased.

You do not need to use format strings, which are just abbreviations. All of the line properties can be controlled by keyword arguments. For example, you can set the color, marker, linestyle, and markercolor with:

```
plot(x, y, color='green', linestyle='dashed', marker='o',
     markerfacecolor='blue', markersize=12).  See
     :class:`~matplotlib.lines.Line2D` for details.
```

The kwargs are Line2D properties:

Property	Description
agg_filter	unknown
alpha	float (0.0 transparent through 1.0 opaque)
animated	[True | False]
antialiased or aa	[True | False]
axes	an Axes instance
clip_box	a matplotlib.transforms.Bbox instance
clip_on	[True | False]
clip_path	[(Path, Transform) | Patch | None]
color or c	any matplotlib color
contains	a callable function
dash_capstyle	['butt' | 'round' | 'projecting']
dash_joinstyle	['miter' | 'round' | 'bevel']
dashes	sequence of on/off ink in points
data	2D array (rows are x, y) or two 1D arrays
drawstyle	['default' | 'steps' | 'steps-pre' | 'steps-mid' | 'steps-post']
figure	a matplotlib.figure.Figure instance
fillstyle	['full' | 'left' | 'right' | 'bottom' | 'top']

USGS science for a changing world

Creating a bar chart (1)

MeterologicData.csv

```
1    Daily Date,M6888_Rain_inpd,OH515_EPT_mmpd,TA613_AIRT_MIN_C,TA613_AIRT_Max_C
2    1/1/2001,0,3.51,3.45,17.75
3    1/2/2001,0,3.15,7.51,20.15


4016 12/29/2011,0,3.07,13.741,22.76
4017 12/30/2011,0,3.31,13.551,24.887
4018 12/31/2011,0,3.25,16.624,26.572
```

BarChart.py

```
26   #--read data
27   metnames = ['date','Rain_inpd','ETP_mmpd','AirTMin_C','AirTMax_C']
28   d = np.genfromtxt( '..\\data\\MeterologicData.csv', skip_header=1, delimiter=',', \
29                      missing_values=('MISSING','MISSING','MISSING','MISSING','MISSING'), \
30                      filling_values=(dt.date(1900, 1, 1),0.0,0.0,np.NAN,np.NAN), \
31                      names=metnames, dtype=None, converters={'date':mkdate} )
32   datemin = dt.date(d['date'].min().year  , 1, 1)
33   datemax = dt.date(d['date'].max().year+1, 1, 1)


11   #--function for parsing string into a datetime
12   def mkdate(text):
13       return dt.datetime.strptime(text, '%m/%d/%Y')
```

USGS science for a changing world

Creating a bar chart (2)

BarChart.py

```
49  #--create monthly totals for rainfall and ETP
50  on_date = d['date'][0]
51  monthly_data, c = [], [0.0, 0.0]
52  for ipos,t in enumerate( d['date'] ):
53      if t.month != on_date.month or ipos == len( d['date'] ) - 1:
54          t_month = t_date.month
55          t_day = 1 #int( t_date.day / 2 )
56          t_year = t_date.year
57          monthly_data.append( [ dt.date(t_year, t_month, t_day), c[0], c[1], t_date.day ]
    )
58          c[0] = 0.0
59          c[1] = 0.0
60          on_date = t
61      c[0] += d['Rain_inpd'][ipos]
62      c[1] += d['ETP_mmpd'][ipos]
63      t_date = t
64  monthly_data = np.array( monthly_data )
```

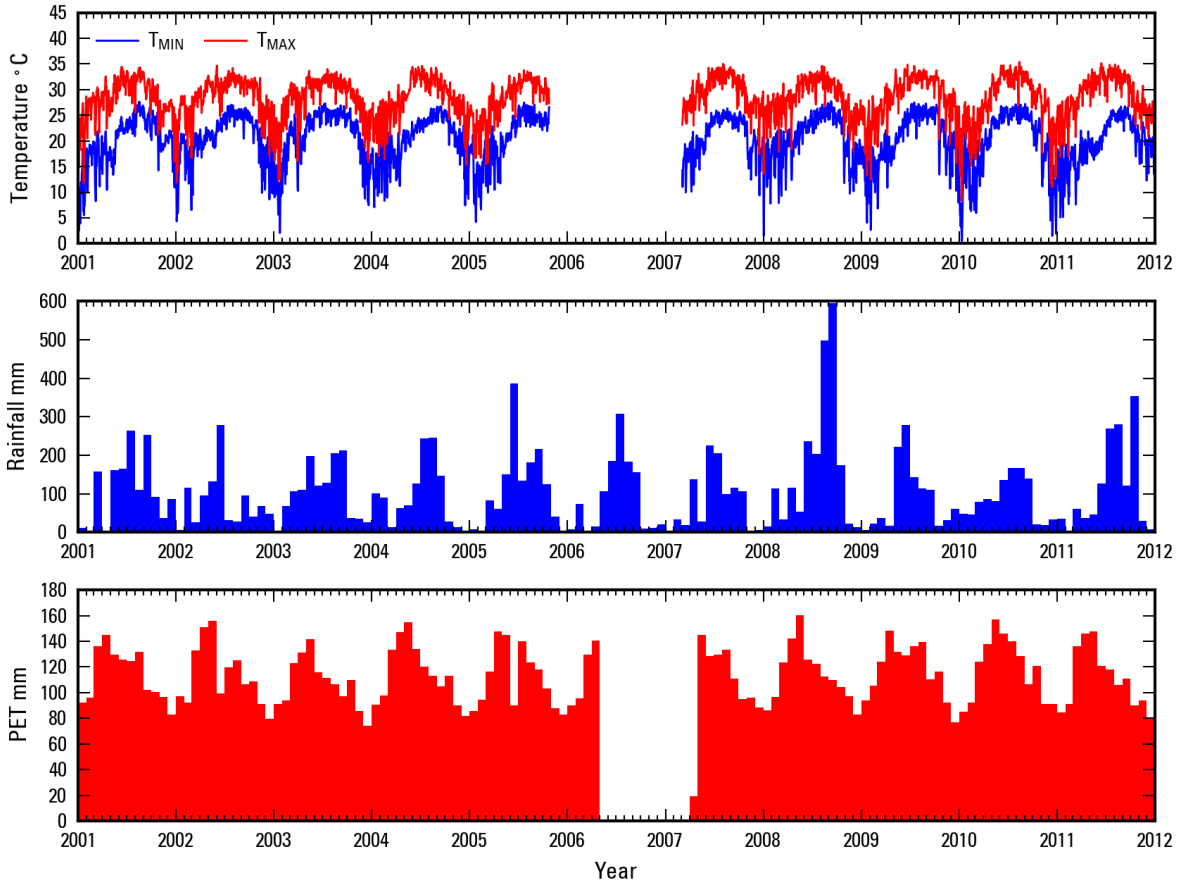
Creating a bar chart (3)

BarChart.py

```
57  #--matplotlib date specification
58  years, months = mdates.YearLocator(), mdates.MonthLocator()  #every year, every month
59  yearsFmt = mdates.DateFormatter('%Y')
60  #--define the first subplot
61  ax = fig.add_subplot(3,1,1)
62  #--plot the temperature data
63  ax.plot(pl.date2num(d['date']),d['AirTMin_C'], color='b', linewidth=0.7, label=r'T$_{MIN}$')
64  ax.plot(pl.date2num(d['date']),d['AirTMax_C'], color='r', linewidth=0.7, label=r'T$_{MAX}$')
65  #--legends and axes
66  leg = ax.legend(loc='upper left',ncol=2,labelspacing=0.25,columnspacing=1,\
67                  handletextpad=0.5,handlelength=2.0,numpoints=1)
68  leg._drawFrame=False
69  ax.xaxis.set_major_locator(years), ax.xaxis.set_minor_locator(months)
70  ax.xaxis.set_major_formatter(yearsFmt)
71  ax.set_xlim(datemin, datemax)
72  ax.set_ylabel( r'Temperature $^{\circ}$C' )
73  ax.set_ylim(0,45)
74  #--define the second subplot
75  ax = fig.add_subplot(3,1,2)
76  #--plot the rainfall data
77  ax.bar(pl.date2num(monthly_data[:,0]),monthly_data[:,1]*25.4, \
78         color='b', width=monthly_data[:,3], linewidth=0, label='Rainfall')
```

USGS science for a changing world

Creating a bar chart (4)



Model coordinates

plotHeads.py

```
27  #--problem size
28  nlay, nrow, ncol = 1, 41, 40
29  #--coordinate information
30  dx, dy = 500., 500.
31  xOff, yOff = 0.0, 0.0
32  xcell = np.arange(xOff+dx/2.,xOff+(ncol*dx)+dx/2.0,dx)
33  ycell = np.arange(yOff+dy/2.,yOff+(nrow*dy)+dy/2.0,dy)
34  Xcell,Ycell = np.meshgrid(xcell,ycell)
35  xedge = np.arange(xOff,xOff+float(ncol)*dx+0.001,dx)
36  yedge = np.arange(yOff,yOff+float(nrow)*dy+0.001,dy)
37  Xedge,Yedge = np.meshgrid(xedge,yedge)
38  xmin,xmax = xOff, xOff+float(ncol)*dx
39  ymin,ymax = yOff, yOff+float(nrow)*dy
40  #--read MODFLOW data from external files and invert for plotting
41  ibound  = au.loadArrayFromFile(nrow,ncol,'..\\ref\\ibound.ref')
42  ibound   = np.flipud(ibound)
```

Binary head data

plotHeads.py

```
43  #--get available times
44  headObj = mfb.MODFLOW_Head(nlay,nrow,ncol,head_file)
45  t = headObj.get_gage(1)
46  ntimes = t.shape[0]
47  mf_times = np.zeros( (ntimes), np.float )
48  for i in range(0,ntimes):
49      mf_times[i] = t[i,0]


55  #--create figures for each output time
56  for ipos,on_time in enumerate( mf_times ):
57      #--build output file name
58      output_name = '{0}{1}_{2:05d}.{3}'.format(base_dir,base_name,int(ipos),extension)
59      fnames.append( output_name )
60      #--read head data
61      headObj = mfb.MODFLOW_Head(nlay,nrow,ncol,head_file)
62      totim,kstp,kper,h,success = headObj.get_record(on_time)
63      hd = np.copy( h[0,:,:] )
```

Create the map (1)

plotHeads.py

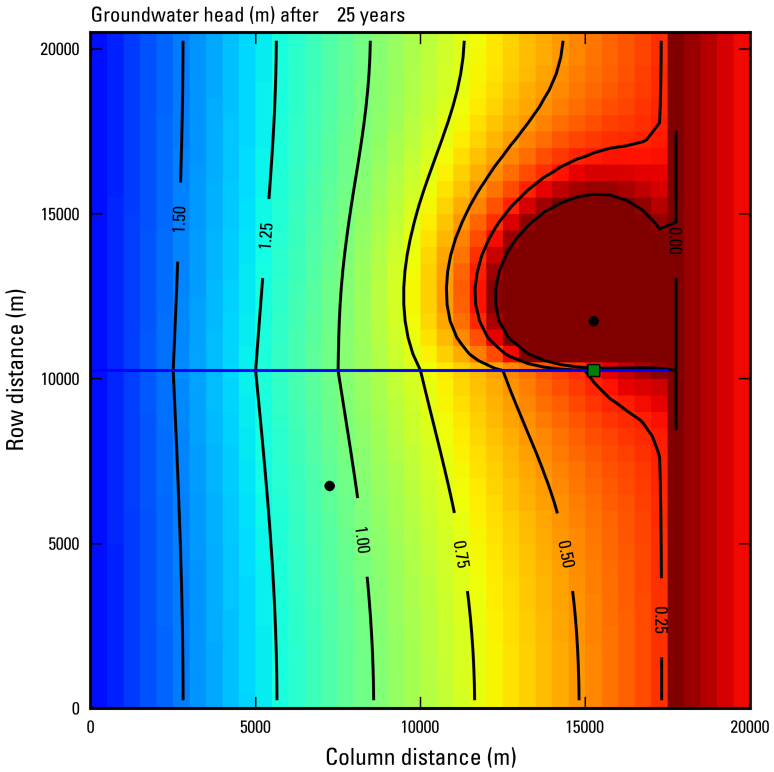
```
64      #--invert rows for plotting and mask data in inactive areas
65      hd       = np.flipud(hd)
66      hd       = np.ma.masked_where(ibound<1,hd)


72      #--figure
73      ztf = figure(figsize=(4.0,4.0), facecolor='w')
74      ztf.subplots_adjust(wspace=0.2,hspace=0.2,left=0.1,right=0.9,bottom=0.1,top=0.9)
75      ax = ztf.add_subplot(1,1,1,aspect='equal')
76      iyears = int( on_time / 365. )
77      ctime = 'years'
78      if iyears == 1:
79          ctime = 'year'
80      ctitle = 'Groundwater head (m) after {0:5d} {1}'.format( iyears, ctime )
81      text(0.0,1.01,ctitle,\
82           horizontalalignment='left',verticalalignment='bottom',size=7,\
83           transform=ax.transAxes)
```

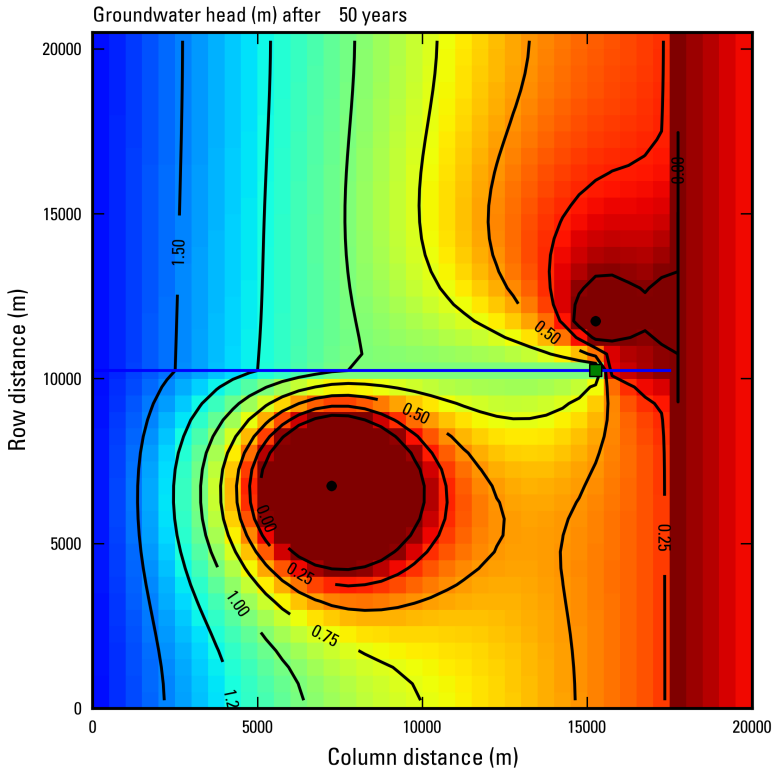
Create the map (2)

plotHeads.py

```
84      hp = ax.pcolor(Xedge,Yedge,hd,\
85                     vmin=0,vmax=2,cmap='jet_r',alpha=1.0,edgecolors='None')
86      ch = ax.contour(xcell,ycell,hd,\
87                      levels=hdcontour,colors='k',linewidths=1)
88      ax.clabel(ch,inline=1,fmt='%5.2f',fontsize=6)
89      ax.plot([xedge[0],xedge[35]],[ycell[20],ycell[20]],linewidth=1,color='b',label='River')
90      ax.plot(xcell[struct_loc[1]],ycell[struct_loc[0]],'gs',markersize=4,label='Structure')
91      ax.plot(xcell[well_loc[0,1]],ycell[well_loc[0,0]],'ko',markersize=3,label='PW-1')
92      ax.plot(xcell[well_loc[1,1]],ycell[well_loc[1,0]],'ko',markersize=3,label='PW-2')
93      #--plot limits
94      ax.set_xlim(xmin,xmax)
95      ax.set_ylim(ymin,ymax)
96      xlabel('Column distance (m)')
97      ylabel('Row distance (m)')
98      #--save figure
99      ztf.savefig(output_name,dpi=600)
100     close(ztf)
```

Final maps


Groundwater head (m) after 25 years
Row distance (m)
Column distance (m)
0
5000
10000
15000
20000
1.50
1.25
1.00
0.75
0.50
0.25
0.00

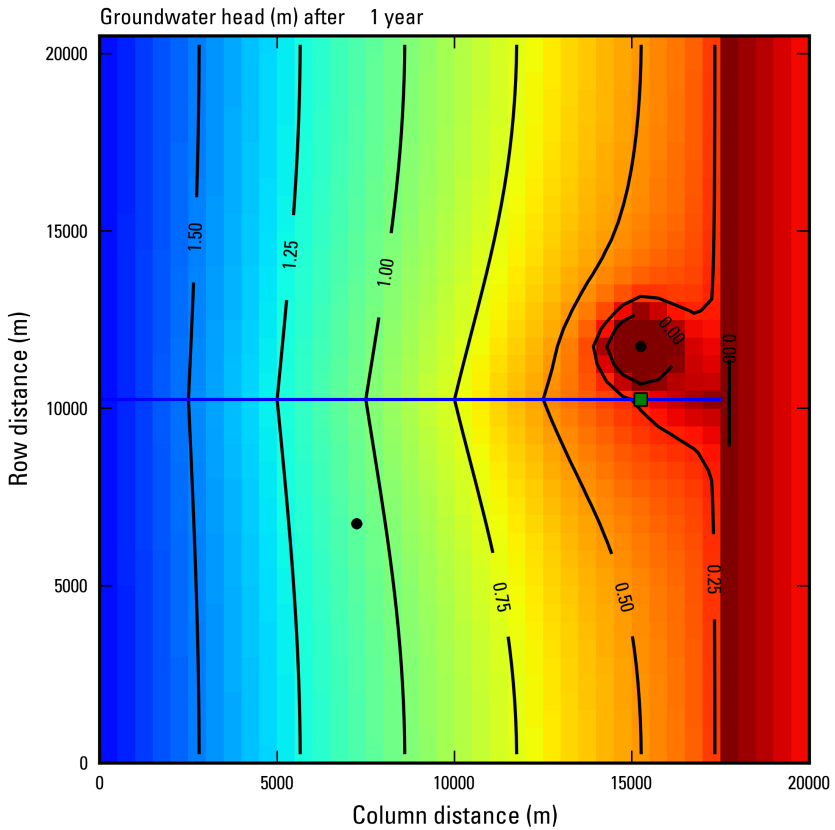

Groundwater head (m) after 50 years
Row distance (m)
Column distance (m)
0
5000
10000
15000
20000
1.50
1.00
0.75
0.50
0.25
0.00

Using `ffmpeg.exe`

plotHeads.py

```
1   import sys
2   import os
3   import subprocess


101 #--animate head data
102 coutf = '{0}{1}.swf'.format(base_dir,base_name)
103 cline = 'ffmpeg.exe -i {0}{1}_%05d.png {2} -y'.format( base_dir,base_name,coutf )
104 try:
105     os.remove(coutf)
106 except:
107     print 'could not remove...{0}'.format( coutf )
108 subprocess.call(cline, stdin=None, stdout=None, stderr=None, shell=False)
109 #--delete temporary png files
110 for f in fnames:
111     os.remove(f)
```

Using `ffmpeg.exe`

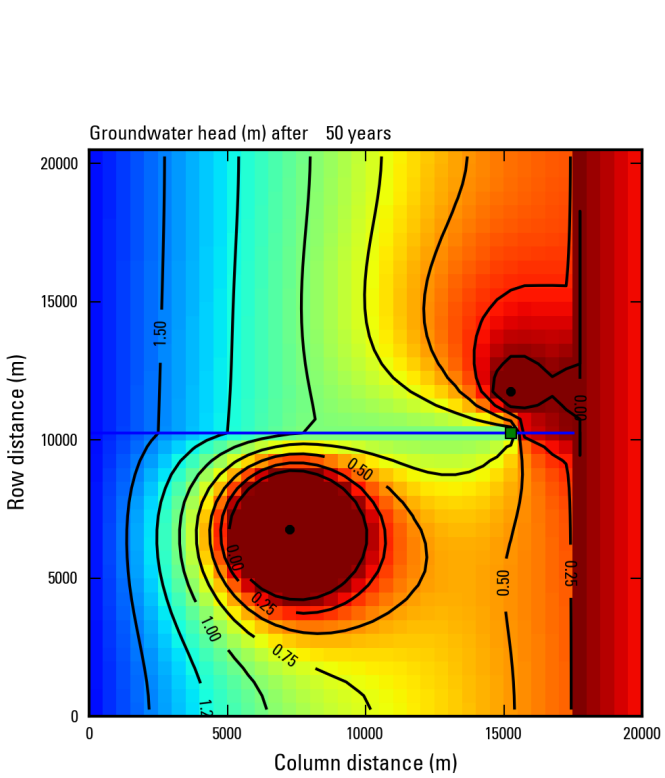


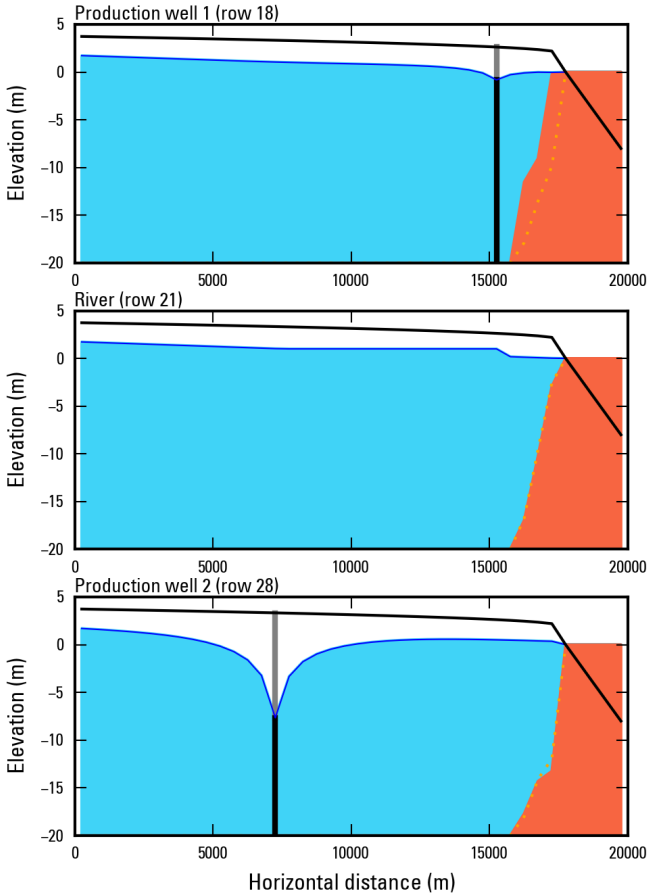
Adding cross-sections (1)

Cross-sectionSample.py

```
1     ix = 23
2     ax = ztf.add_subplot(3,2,2)
3     text(0.0,1.01,'Production well 1 (row 18)',\
4          horizontalalignment='left',verticalalignment='bottom',size=7,transform=ax.transAxes)
5     t = np.copy( top[ix,:] )
6     h = np.copy(  hd[ix,:] )
7     h[35:] = 0.0
8     z = np.copy(  zs[ix,:] )
9     z[35:] = 0.0
10    zt = np.copy(  z_steady[ix,:] )
11    f  = ax.fill_between(xcell,y1=h,y2=z,color='#40d3f7')
12    s  = ax.fill_between(xcell,y1=z,y2=-25.,color='#F76541')
13    ax.plot(xcell,zt,linestyle=':',color='#FFA500')
14    ax.plot(xcell,t,'k-',zorder=100)
15    ax.plot([xcell[30],xcell[30]],[t[30],h[30]],linestyle='solid',color='0.5',linewidth=2)
16    ax.plot([xcell[30],xcell[30]],[h[30],-25.],'k-',linewidth=2)
17    ax.plot(xcell[0:36],h[0:36],'b-',linewidth=0.5)
18    #--plot limits
19    ax.set_xlim(xmin,xmax)
20    ax.set_ylim(-20,5)
21    ylabel('Elevation (m)')
```

Adding cross-sections (2)





Automating model runs and figure preparation

AutomationSample.bat

```
1  rem ***Sample 02
2  cd ..\SWRSample02\
3  mf2005-swr_x64.exe SWRSample02.nam
4  mf2005-swr_x64.exe SWRSample02.02.nam
5  cd ..\Python\
6  python SWRSample02.py
7  python SWRSample02.02.py
8  python SWRSample02v.py
9  python SWRSample02v.02.py
10 rem ***Sample 16
11 cd ..\SWRTestSimulation16\
12 mf2005-swr_x64.exe SWRTestSimulation16.level.nam
13 mf2005-swr_x64.exe SWRTestSimulation16.tilted.nam
14 mf2005-swr_x64.exe SWRTestSimulation16.tilted.IC.nam
15 cd ..\python\
16 python SWRTestSimulation16.level.py
17 python SWRTestSimulation16.tilted.py
18 python SWRTestSimulation16.tilted.IC.py
19 rem END OF BATCH FILE
20 cd ..\
21 pause
```