2014_Osler_Data_Analysis

November 27, 2013

Part I

Analysis of Simpson Island paleomagnetic data

Corresponding Author: Nicholas L. Swanson-Hysell (swanson-hysell@berkeley.edu)

1 Introduction

This IPython notebook contains data analysis on paleomagnetic data developed from ca. 1.1 billion year old lava flows of the Osler Volcanic Group that are part of the North American Midcontinent Rift. This analysis accompanies a manuscript under review at **Geochemistry, Geophysics, Geosystems** entitled "Significant plate motion during the early magmatic stage of North American Midcontinent Rift development" by N. L. Swanson-Hysell, A. A. Vaughan, M. R. Mustain and K. Asp. This notebook is part of the supporting online materials and is available at https://github.com/Swanson-Hysell/2014_Swanson-Hysell-et-al_Osler along with the necessary files to execute all of the code. Within this github repository are the following folders:

1. 2014_Osler_Code This folder contains this notebook file as well as necessary libraries.
2. 2014_Osler_Data This folder contains the data generated in the study at the specimen level as well as the flow means that are imported into this notebook for data analysis.
3. 2014_Osler_Manuscript This folder contains that manuscript text and figures.

If you are viewing this supporting material as a PDF document and want to run the code in the notebook you will need to download the code from the github repository and you will also need a Python distribution that includes IPython (http://ipython.org). Alternatively you can view the notebook online with the the IPython nbviewer at this link: http://bit.ly/19XRdjJ

2 Import libraries

This code blocks imports necessary libraries that define functions that will be used in the data analysis below. The code below uses the pmag.py and pmagplotlib.py files of the the PmagPy software package (version pmagpy-2.206) authored by Lisa Tauxe (https://github.com/ltauxe/PmagPy). The check_updates and get_version functions of the PmagPy libraries cause errors in the interactive environment of IPython so the code below calls a slightly modified version that is included in the Github repository for this work. There are other functions that are necessary for the interactive plotting of paleomagnetic data (some of them edited from PmagPy) that are within the IPmag.py library that is imported in the code block below.

In [1]:
```
#import paleomagnetic specific libraries
import pmag, pmagplotlib, IPmag
```

This notebook runs with pylab inline which imports the numpy, scipy and matplotlib functions and allows for the plots to be viewed inline in the IPython notebook (instead of opening up in another window).

In [2]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

3 Import Simpson Island paleomagnetic data

The data file FlowDataAll.csv has data in the format:

SITE

STRAT_HEIGHT

DEC_GEO

INC_GEO

Dec_TC

INC_TC

A95

n

Plat

ABS_Plat

VGP_lat

VGP_long

These data are displayed in a table below and imported by variable in the cell below the table.

In [3]:
```
import pandas
from IPython.core.display import HTML
pandas.set_option('max_columns', 10)

data = pandas.read_csv('../2014_Osler_Data/SimpsonIsland_OslerData.csv')
display(HTML(data.to_html()))
```

```
<IPython.core.display.HTML at 0x100436f90>
```

In [4]:
```
data_file='../2014_Osler_Data/SimpsonIsland_OslerData.csv'
SimpsonIsland_Strat = np.genfromtxt(data_file,delimiter=",", skip_header=1)

strat_height=SimpsonIsland_Strat[:,1]
Dec_TC=SimpsonIsland_Strat[:,6]
Inc_TC=SimpsonIsland_Strat[:,7]
A95=SimpsonIsland_Strat[:,8]
plat=SimpsonIsland_Strat[:,10]
abs_plat=SimpsonIsland_Strat[:,11]
VGP_lat=SimpsonIsland_Strat[:,12]
VGP_long=SimpsonIsland_Strat[:,13]
```

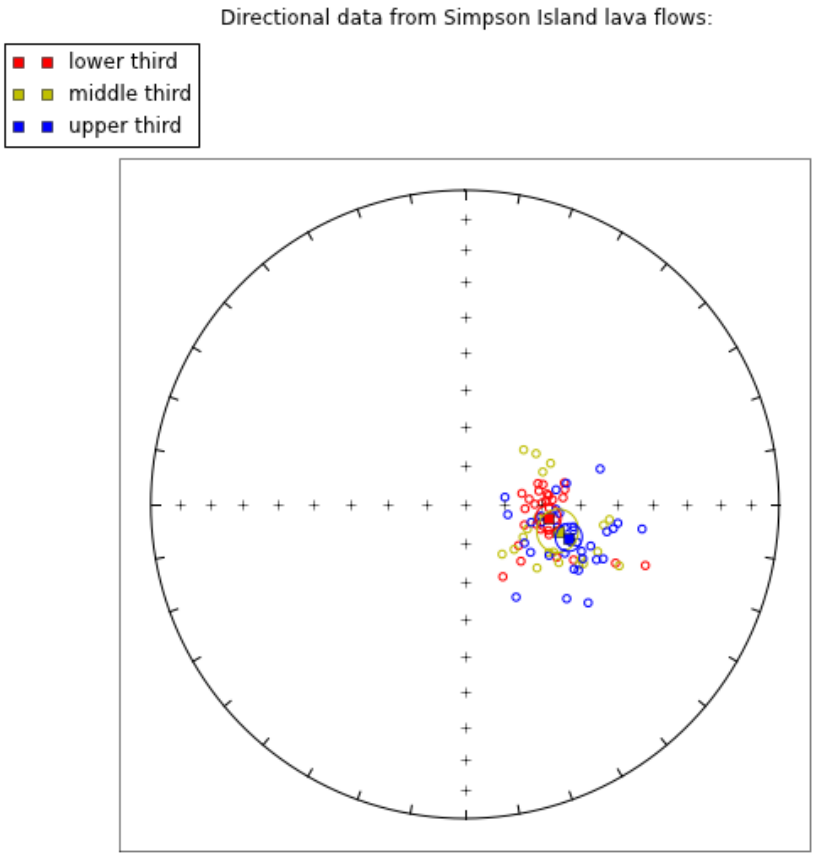
4 Comparison between data from the lower 1/3, middle 1/3 and upper 1/3 of Simpson Island stratigraphy

Let's consider the data by stratigraphically grouping flow data from the lower third (0 to 1041 meters), middle third (1041 to 2083 meters) and the upper third (2083 to 3124 meters) of the stratigraphy. The Fisher means for these groups are calculated with the data being displayed on equal area plots with these means.

In [5]:
```
SI_Directions=[]
SI_Poles=[]

for n in range(0,84):
    Dec,Inc=Dec_TC[n],Inc_TC[n]
    SI_Directions.append([Dec,Inc,1.])
    Plong,Plat=VGP_long[n],VGP_lat[n]
    SI_Poles.append([Plong,Plat,1.])

SI_LowerThird_Directions=SI_Directions[0:30]
SI_LowerThird_Poles=SI_Poles[0:30]
SI_MiddleThird_Directions=SI_Directions[30:50]
SI_MiddleThird_Poles=SI_Poles[30:50]
SI_UpperThird_Directions=SI_Directions[50:84]
SI_UpperThird_Poles=SI_Poles[50:84]

#calculate and display the Fisher means for each subset of data
pars_1=pmag.fisher_mean(SI_LowerThird_Directions)
pars_2=pmag.fisher_mean(SI_MiddleThird_Directions)
pars_3=pmag.fisher_mean(SI_UpperThird_Directions)

print 'The fisher mean parameters for SI_LowerThird_Directions are: '
print str(pars_1)
print ''
print 'The fisher mean parameters for SI_MiddleThird_Directions are: '
print str(pars_2)
print ''
print 'The fisher mean parameters for SI_UpperThird_Directions are: '
print str(pars_3)

#plot the direction of each flow mean on an equal area plot
fignum = 1
pylab.figure(num=fignum,figsize=(10,10),dpi=160)
pmagplotlib.plotNET(fignum)
IPmag.iplotDI(SI_LowerThird_Directions,'r')
IPmag.iplotDI(SI_MiddleThird_Directions,'y')
IPmag.iplotDI(SI_UpperThird_Directions,'b')
title('Directional data from Simpson Island lava flows:')

#plot the direction and alpha_95 of the stratigraphic groups
SI_LowerThird_mean=[pars_1['dec'],pars_1['inc'],pars_1["alpha95"]]
SI_MiddleThird_mean=[pars_2['dec'],pars_2['inc'],pars_2["alpha95"]]
SI_UpperThird_mean=[pars_3['dec'],pars_3['inc'],pars_3["alpha95"]]

SI_LowerThird_dimap=pmag.dimap(pars_1['dec'],pars_1['inc'])
pylab.plot(SI_LowerThird_dimap[0],SI_LowerThird_dimap[1],
           'rs',label='lower third')
SI_MiddleThird_dimap=pmag.dimap(pars_2['dec'],pars_2['inc'])
pylab.plot(SI_MiddleThird_dimap[0],SI_MiddleThird_dimap[1],
           'ys',label='middle third')
SI_UpperThird_dimap=pmag.dimap(pars_3['dec'],pars_3['inc'])
pylab.plot(SI_UpperThird_dimap[0],SI_UpperThird_dimap[1],
           'bs',label='upper third')
legend(loc=2)

Xcirc,Ycirc=[],[]
Da95,Ia95=pmag.circ(float(pars_1["dec"]),
```

```
                    float(pars_1["inc"]),float(pars_1["alpha95"]))
for k in  range(len(Da95)):
    XY=pmag.dimap(Da95[k],Ia95[k])
    Xcirc.append(XY[0])
    Ycirc.append(XY[1])
pylab.plot(Xcirc,Ycirc,'r')
Xcirc,Ycirc=[],[]
Da95,Ia95=pmag.circ(float(pars_2["dec"]),
                    float(pars_2["inc"]),float(pars_2["alpha95"]))
for k in  range(len(Da95)):
    XY=pmag.dimap(Da95[k],Ia95[k])
    Xcirc.append(XY[0])
    Ycirc.append(XY[1])
pylab.plot(Xcirc,Ycirc,'y')
Xcirc,Ycirc=[],[]
Da95,Ia95=pmag.circ(float(pars_3["dec"]),
                    float(pars_3["inc"]),float(pars_3["alpha95"]))
for k in  range(len(Da95)):
    XY=pmag.dimap(Da95[k],Ia95[k])
    Xcirc.append(XY[0])
    Ycirc.append(XY[1])
pylab.plot(Xcirc,Ycirc,'b')
plt.show()
```

```
The fisher mean parameters for SI_LowerThird_Directions are:
{'csd': 10.27016991125803, 'k': 62.203494067439664, 'n': 30, 'r':
29.533788247191406, 'alpha95': 3.3588968729980548, 'dec':
99.338721094603642, 'inc': -68.042883902179256}

The fisher mean parameters for SI_MiddleThird_Directions are:
{'csd': 13.415586919446101, 'k': 36.454461136308502, 'n': 20, 'r':
19.478801787003345, 'alpha95': 5.4795120053251054, 'dec':
105.906122721936, 'inc': -64.898301440038821}

The fisher mean parameters for SI_UpperThird_Directions are:
{'csd': 11.452725615164553, 'k': 50.020994936019207, 'n': 34, 'r':
33.340277016836438, 'alpha95': 3.5142065682074946, 'dec':
107.62449997258365, 'inc': -61.552568868987819}
```



pmagpy-2.206

4.1 Test for a common mean between paleomagnetic data from the lower, middle and upper thirds of the stratigraphy

The code below will compare the subsets of data using:

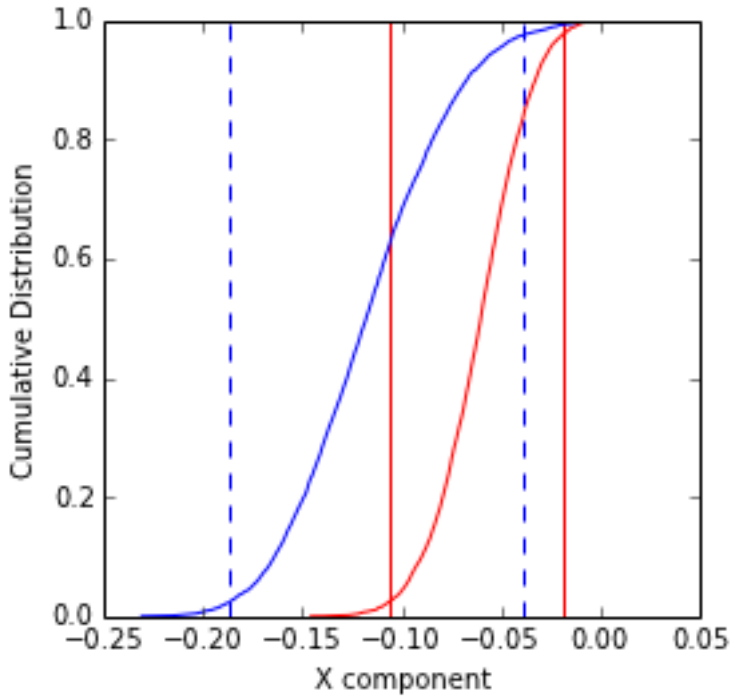
1. The V_w test of Watson (1983) slightly modified from the implementation of PmagPy (Tauxe, 2013). The test is comprised of calculating the V_w statistic for the two data sets and then determining the critical value of V_w through a Monte Carlo simulation. For the simulation, two Fisher-distributed data sets with a common mean are simulated using the k_1 and k_2 precisions and the N_1 and N_2 number of points of the datasets being evaluated. Then the V_w statistic that comes from comparison between the two simulated data sets is calculated. A large number of simulations are done (default is 1000; it can be set as the NumSims parameter of the function) in order to determine the V_w values that would result by chance through sampling distributions with the same direction. The critical value of V_w is at the 95% level of confidence (i.e. if 1000 V_w values are simulated, it is the 950th one).
2. The bootstrap fold test of Tauxe (developed by Tauxe et al., 1991 and implemented per Tauxe (2010)). This approach determines the cumulative distributions of the Cartesian coordinates of bootstrapped means and com-

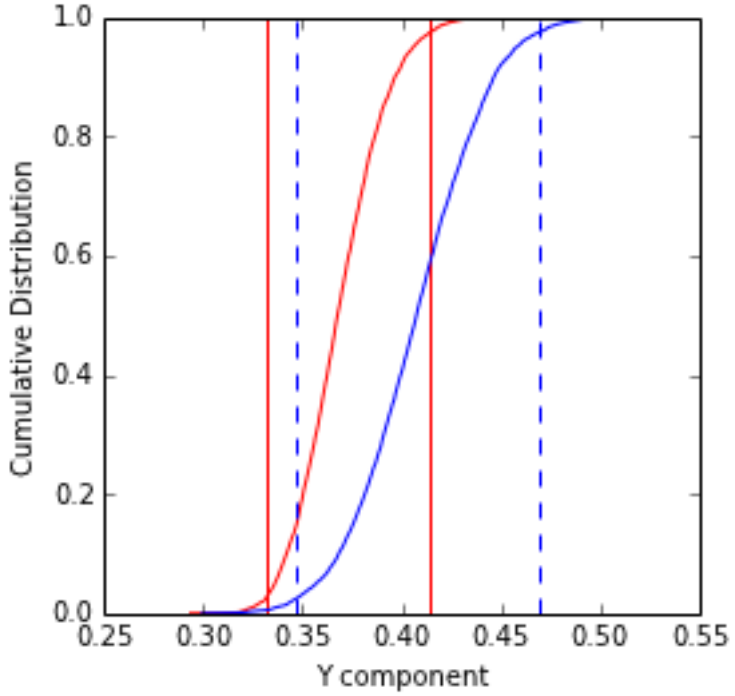
pares the cumulative distributions to see if the confidence intervals overlap. If they do all overlap, then the two means cannot be distinguished at the 95% level of confidence and they pass the bootstrap test for a common mean. Otherwise, if the two sets of directions are distinct in the X, Y or Z component the two means can be distinguished at the 95% confidence level.

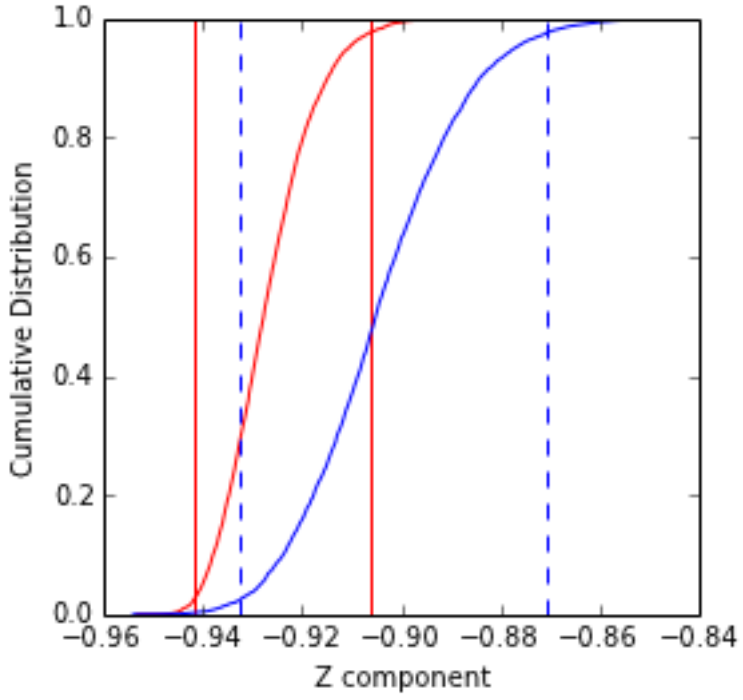
Common mean tests between SI_LowerThird_Directions and SI_MiddleThird_Directions:

In [6]:
```
IPmag.iWatsonV(SI_LowerThird_Directions,SI_MiddleThird_Directions)
IPmag.iBootstrap(SI_LowerThird_Directions,SI_MiddleThird_Directions)
```

```
Results of Watson V test:

Watson's V:           2.6
Critical value of V:  6.3
"Pass": Since V is less than Vcrit, the null hypothesis
that the two populations are drawn from distributions
that share a common mean direction can not be rejected.

M&M1990 classification:

Angle between data set means: 4.1
Critical angle for M&M1990:   6.4
The McFadden and McElhinny (1990) classification for
this test is: 'B'

===============

Here are the results of the bootstrap test for a common mean
```



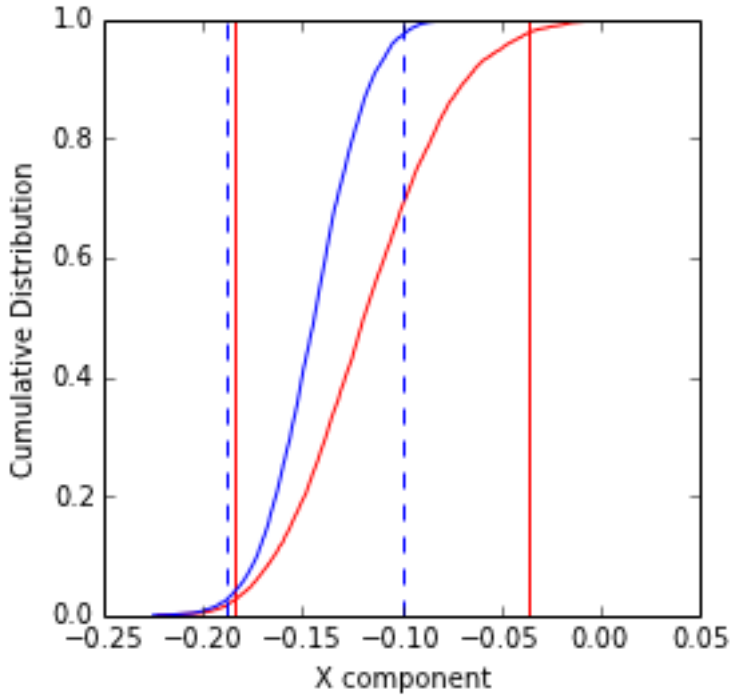


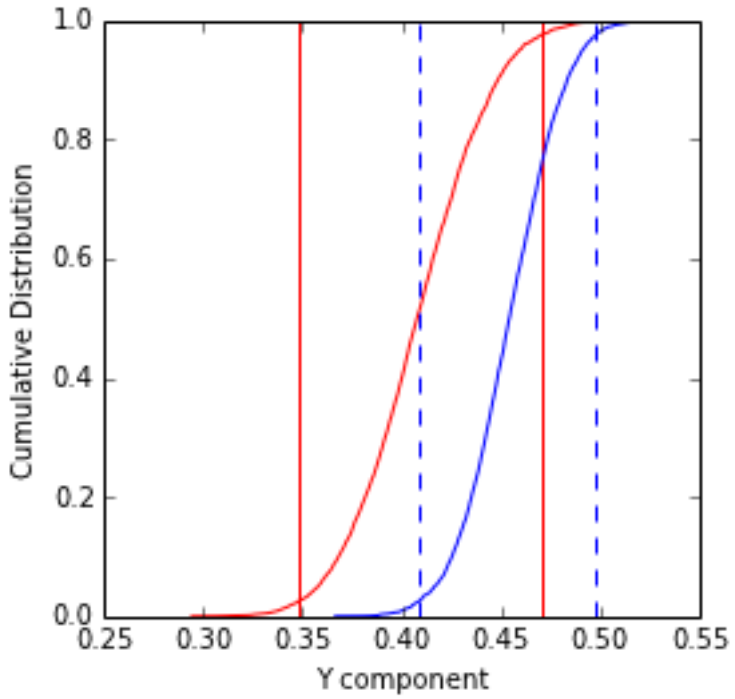


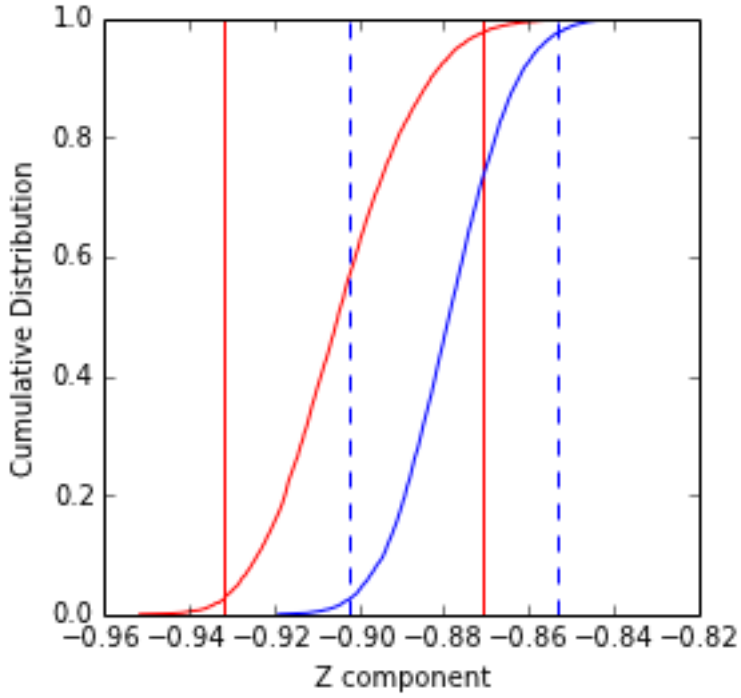
Common mean tests between SI_MiddleThird_Directions and SI_UpperThird_Directions:

In [7]:
```
IPmag.iWatsonV(SI_MiddleThird_Directions,SI_UpperThird_Directions)
IPmag.iBootstrap(SI_MiddleThird_Directions,SI_UpperThird_Directions)
```

```
Results of Watson V test:

Watson's V:           1.8
Critical value of V:  6.1
"Pass": Since V is less than Vcrit, the null hypothesis
that the two populations are drawn from distributions
that share a common mean direction can not be rejected.

M&M1990 classification:

Angle between data set means: 3.4
Critical angle for M&M1990:   6.4
The McFadden and McElhinny (1990) classification for
this test is: 'B'

==============

Here are the results of the bootstrap test for a common mean
```



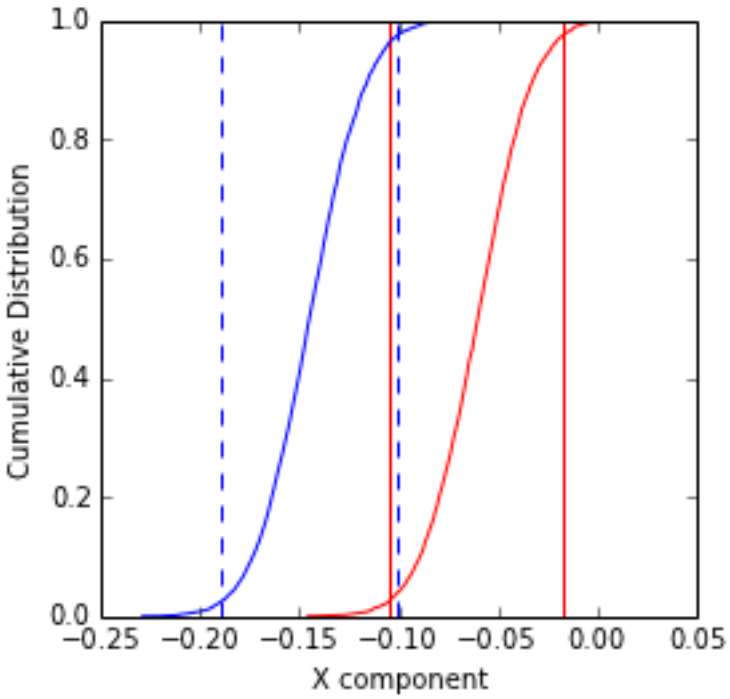


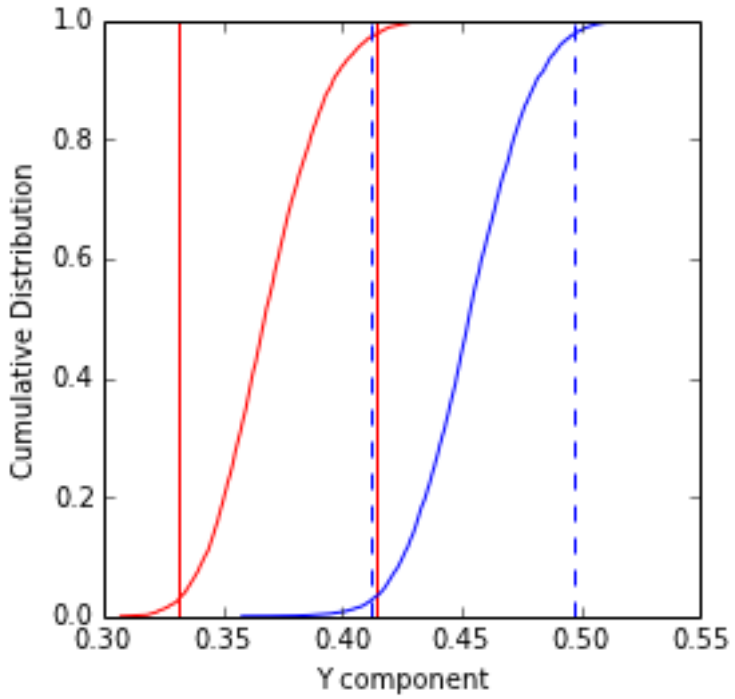


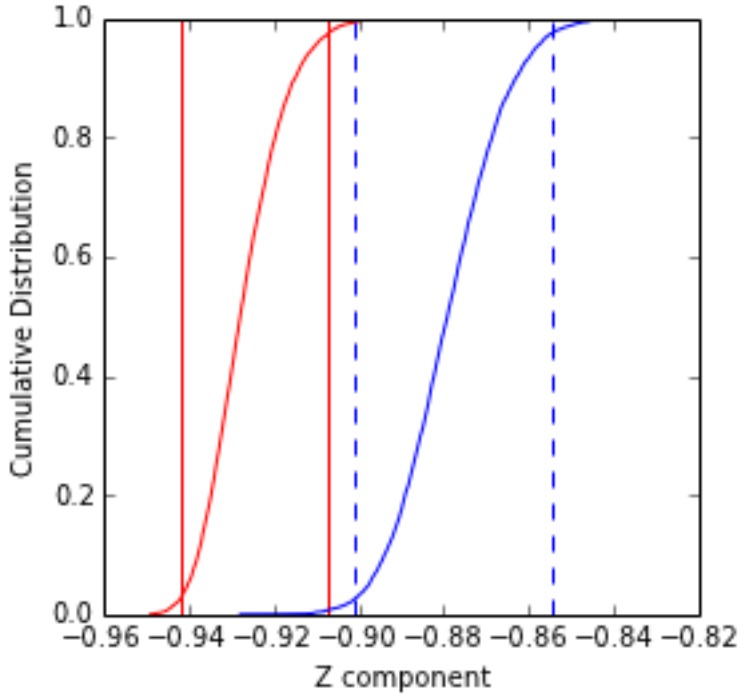
Common mean tests between SI_LowerThird_Directions and SI_UpperThird_Directions:

In [8]:
```
IPmag.iWatsonV(SI_LowerThird_Directions,SI_UpperThird_Directions)
IPmag.iBootstrap(SI_LowerThird_Directions,SI_UpperThird_Directions)
```

```
Results of Watson V test:

Watson's V:           14.5
Critical value of V:  6.4
"Fail": Since V is greater than Vcrit, the two means can
be distinguished at the 95% confidence level.

M&M1990 classification:

Angle between data set means: 7.4
Critical angle for M&M1990:   4.9


===============

Here are the results of the bootstrap test for a common mean
```





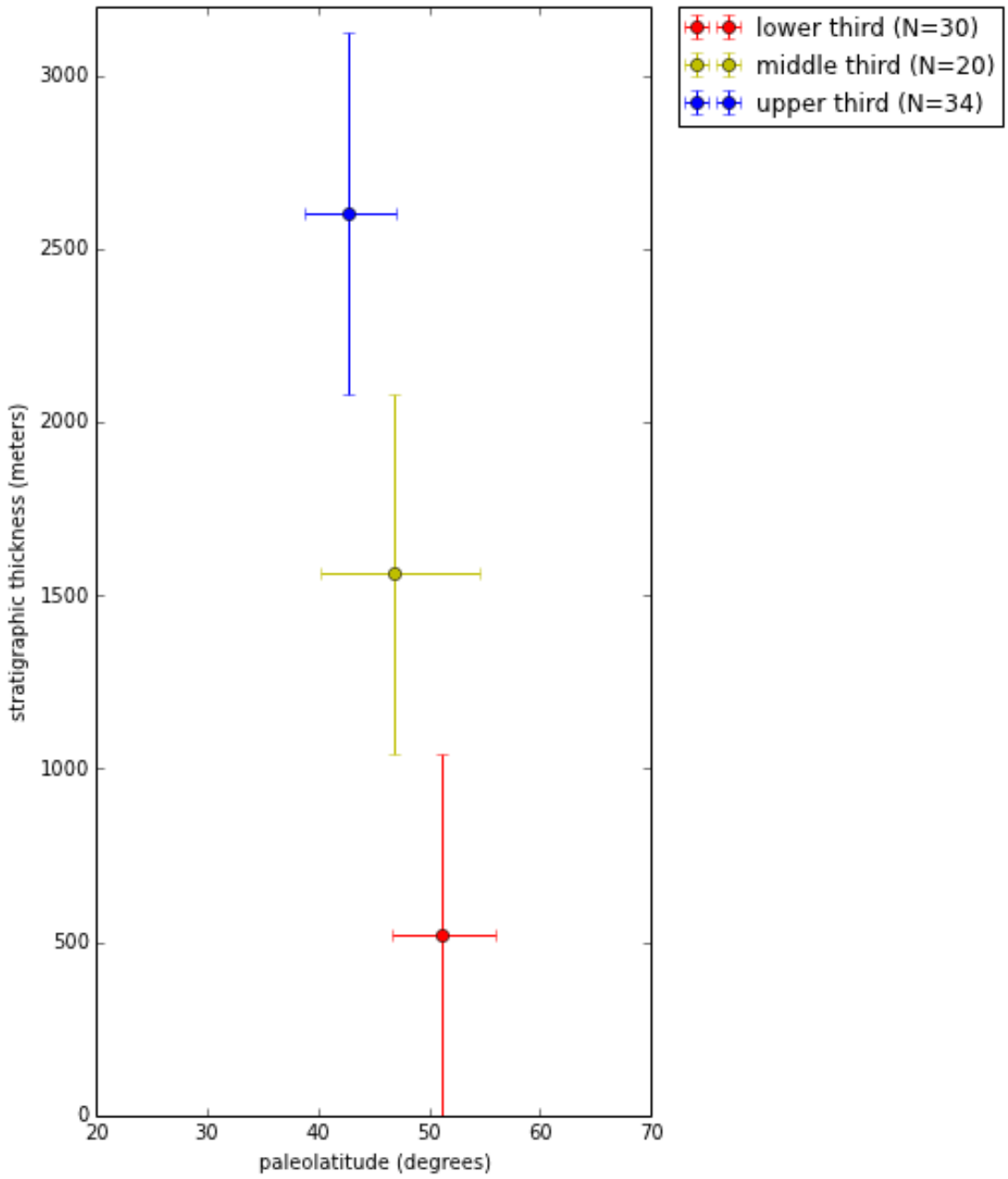


The above results from common mean statistical tests show that the directions from the lower third cannot be distinguished as being obtained from a distribution distinct from that of the middle third nor can the directions from the upper third be distinguished from those of the middle third. In contrast, directions from the lower third of the stratigraphy can be distinguished from those of upper third at the 95% confidence level.

4.2 Plotting the data stratigraphically

Using the flow means calculated for each portion of the stratigraphy, the paleolatitude and 95% confidence bounds on paleolatitude can be calculated. The stratigraphic bounds (y-axis error bars) and paleolatitude bounds (x-axis error bars) are displayed for points plotted in the middle of the stratigraphic bin and at the mean paleolatitude.

In [9]:
```
SI_LowerThird_plat=numpy.abs(IPmag.lat_from_i(SI_LowerThird_mean[1]))
SI_LowerThird_plat_max=numpy.abs(IPmag.lat_from_i(SI_LowerThird_mean[1]-
                                                  SI_LowerThird_mean[2]))
SI_LowerThird_plat_min=numpy.abs(IPmag.lat_from_i(SI_LowerThird_mean[1]+
                                                  SI_LowerThird_mean[2]))

SI_MiddleThird_plat=numpy.abs(IPmag.lat_from_i(SI_MiddleThird_mean[1]))
SI_MiddleThird_plat_max=numpy.abs(IPmag.lat_from_i(SI_MiddleThird_mean[1]-
                                                   SI_MiddleThird_mean[2]))
SI_MiddleThird_plat_min=numpy.abs(IPmag.lat_from_i(SI_MiddleThird_mean[1]+
                                                   SI_MiddleThird_mean[2]))

SI_UpperThird_plat=numpy.abs(IPmag.lat_from_i(SI_UpperThird_mean[1]))
SI_UpperThird_plat_max=numpy.abs(IPmag.lat_from_i(SI_UpperThird_mean[1]-
                                                  SI_UpperThird_mean[2]))
SI_UpperThird_plat_min=numpy.abs(IPmag.lat_from_i(SI_UpperThird_mean[1]+
                                                  SI_UpperThird_mean[2]))

SI_LowerThird_meanstrat=521
SI_MiddleThird_meanstrat=521+1041
SI_UpperThird_meanstrat=521+1041+1041

figure()
errorbar(SI_LowerThird_plat, SI_LowerThird_meanstrat, yerr=[[521],[521]],
         xerr=[[SI_LowerThird_plat-SI_LowerThird_plat_min],
               [SI_LowerThird_plat_max-SI_LowerThird_plat]],
         fmt='ro',label="lower third (N=30)")
errorbar(SI_MiddleThird_plat, SI_MiddleThird_meanstrat, yerr=[[521],[521]],
         xerr=[[SI_MiddleThird_plat-SI_MiddleThird_plat_min],
               [SI_MiddleThird_plat_max-SI_MiddleThird_plat]],
         fmt='yo',label="middle third (N=20)")
errorbar(SI_UpperThird_plat, SI_UpperThird_meanstrat, yerr=[[521],[521]],
         xerr=[[SI_UpperThird_plat-SI_UpperThird_plat_min],
               [SI_UpperThird_plat_max-SI_UpperThird_plat]],
         fmt='bo',label="upper third (N=34)")
xlabel('paleolatitude (degrees)')
ylabel('stratigraphic thickness (meters)')
legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(5,10)
pylab.xlim([20,70])
pylab.ylim([0,3200])
plt.show()
```



5 Comparing larger and smaller stratigraphic subsets of the Simpson Island data

5.1 Comparing data from the lowermost 500 meters to data from the uppermost 500 meters

The analysis above demonstrates that the directions from the lower third of the stratigraphy (0 to 1041 meters; 30 flows) and the upper third of the stratigraphy (2083 to 3124 meters; 34 flows) are statistically distinct and thus indicate statistically significant motion to lower latitudes. While it is preferable to have a high total number of flows for such

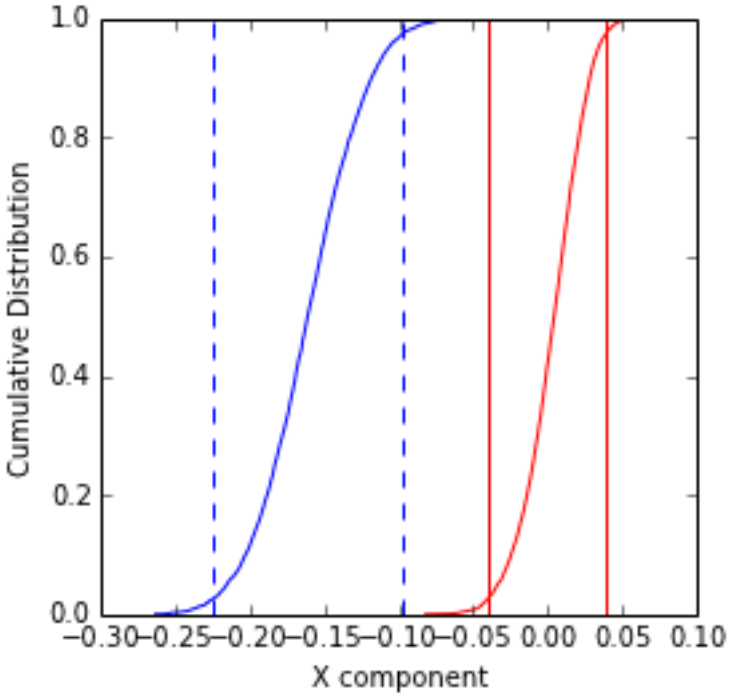
comparisons in order to average out secular variation, let's conduct this same analysis but for an even more restricted range from the bottom 500 meters (17 flows) and top 500 meters (17 flows) of the stratigraphy.

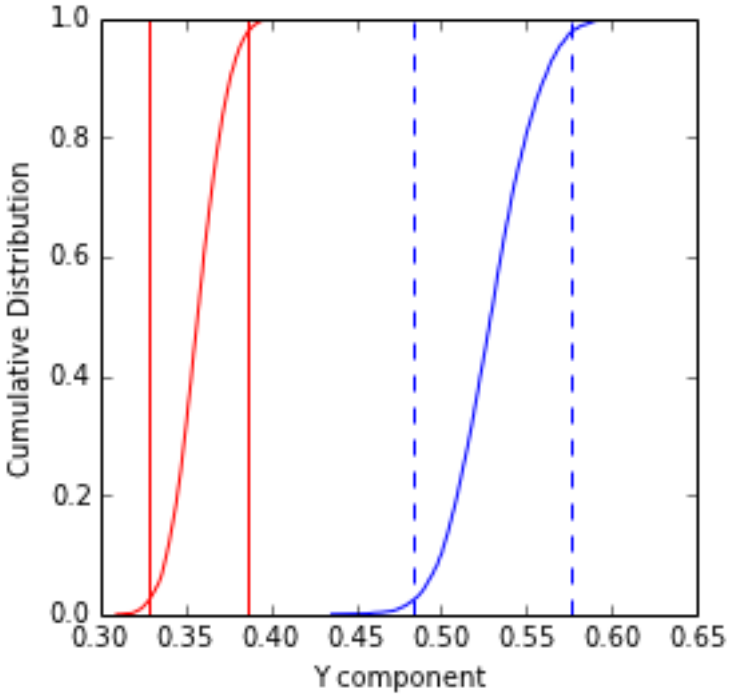
In [10]:
```
SI_Lower500m_Directions=SI_Directions[0:17]
SI_Lower500m_Poles=SI_Poles[0:17]

SI_Upper500m_Directions=SI_Directions[67:84]
SI_Upper500m_Poles=SI_Poles[67:84]

pars_1=pmag.fisher_mean(SI_Lower500m_Directions)
pars_2=pmag.fisher_mean(SI_Upper500m_Directions)

print 'The fisher mean parameters for SI_Lower500m_Directions are: '
print str(pars_1)
print ''
print 'The fisher mean parameters for SI_Upper500m_Directions are: '
print str(pars_2)
```

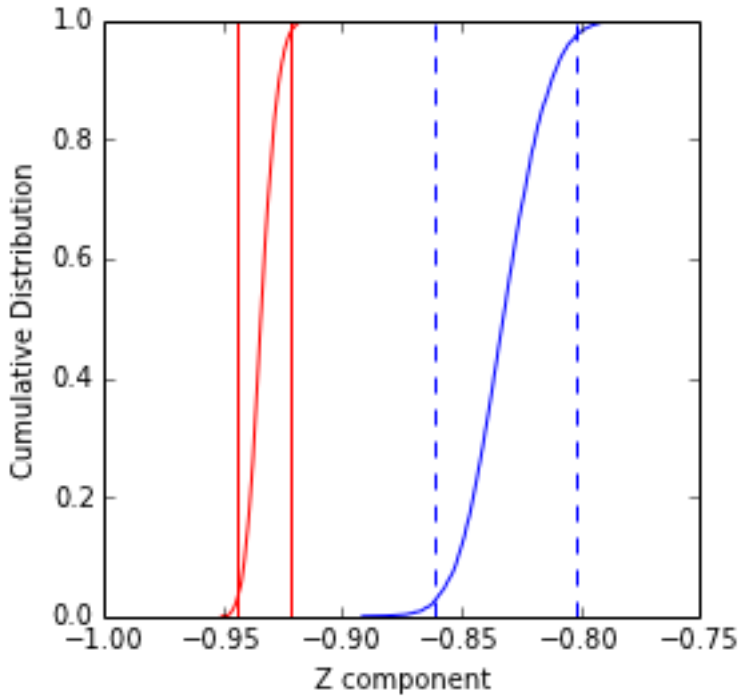
```
The fisher mean parameters for SI_Lower500m_Directions are:
{'csd': 6.1620447880184264, 'k': 172.79068906699587, 'n': 17, 'r':
16.907402417998366, 'alpha95': 2.7213037577811798, 'dec':
89.559211694218391, 'inc': -69.03709846618041}

The fisher mean parameters for SI_Upper500m_Directions are:
{'csd': 10.397718441452353, 'k': 60.686757213380318, 'n': 17, 'r':
16.736351047004497, 'alpha95': 4.6160969512802765, 'dec':
106.9155955500684, 'inc': -56.37171406134344}
```

In [11]:
```
IPmag.iWatsonV(SI_Lower500m_Directions,SI_Upper500m_Directions)
IPmag.iBootstrap(SI_Lower500m_Directions,SI_Upper500m_Directions)
```

```
Results of Watson V test:

Watson's V:           50.4
Critical value of V:  6.7
"Fail": Since V is greater than Vcrit, the two means can
be distinguished at the 95% confidence level.

M&M1990 classification:

Angle between data set means: 14.8
Critical angle for M&M1990:   5.4


===============

Here are the results of the bootstrap test for a common mean
```


1.0
0.8
0.6
0.4
0.2
0.0
Cumulative Distribution
−0.30 −0.25 −0.20 −0.15 −0.10 −0.05 0.00 0.05 0.10
X component


1.0
0.8
0.6
0.4
0.2
0.0
Cumulative Distribution
0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65
Y component



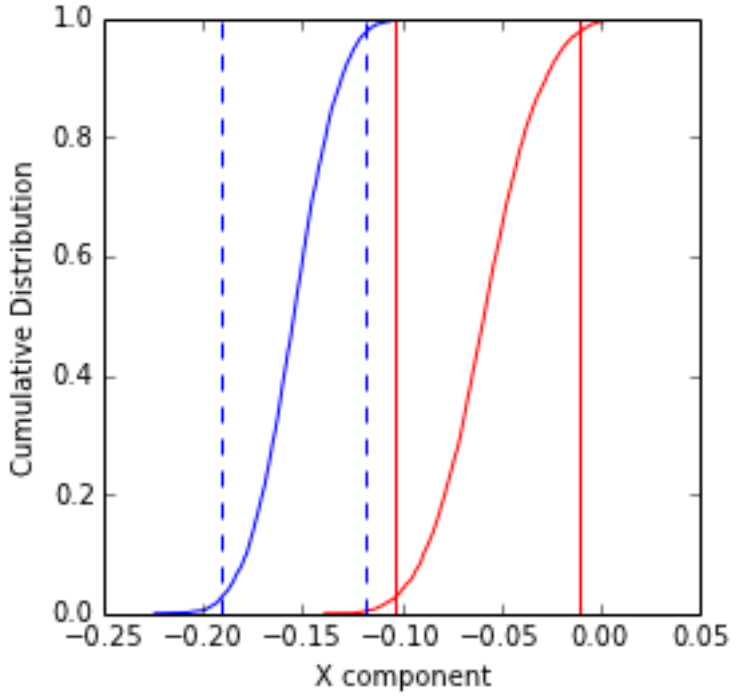
The results of this comparision between the lower 500 meters and upper 500 meters show that they are distinct populations at the 95% confidence level. The difference between the populations is more dramatic in both the Watson V and bootstrap test from the tests on the lower third and upper third subsets that were conducted above. This increase in difference with tighter stratigraphic groups from lower and higher in the sequence is the expected result if the sequence is a record of progressive paleogeographic change.

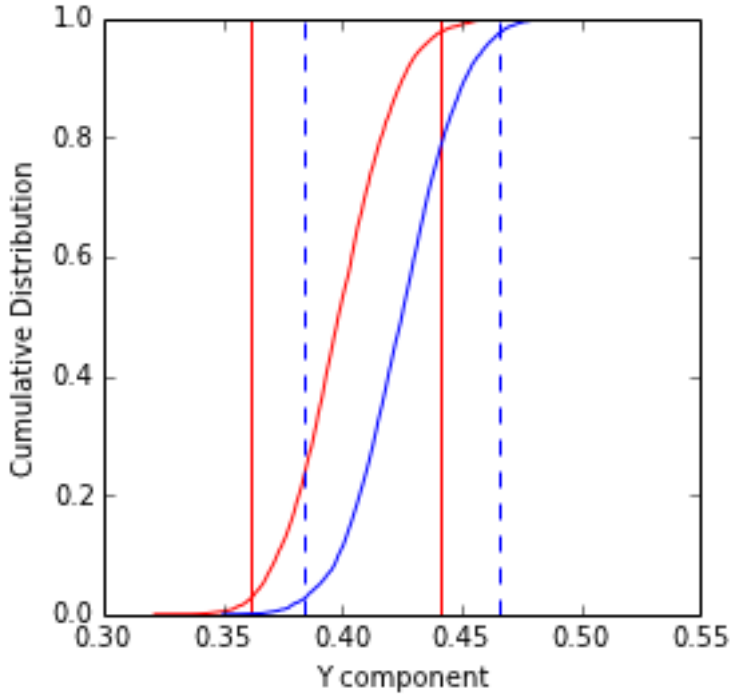
5.2 Comparison between data from the lower 1/2 and upper 1/2 of Simpson Island stratigraphy

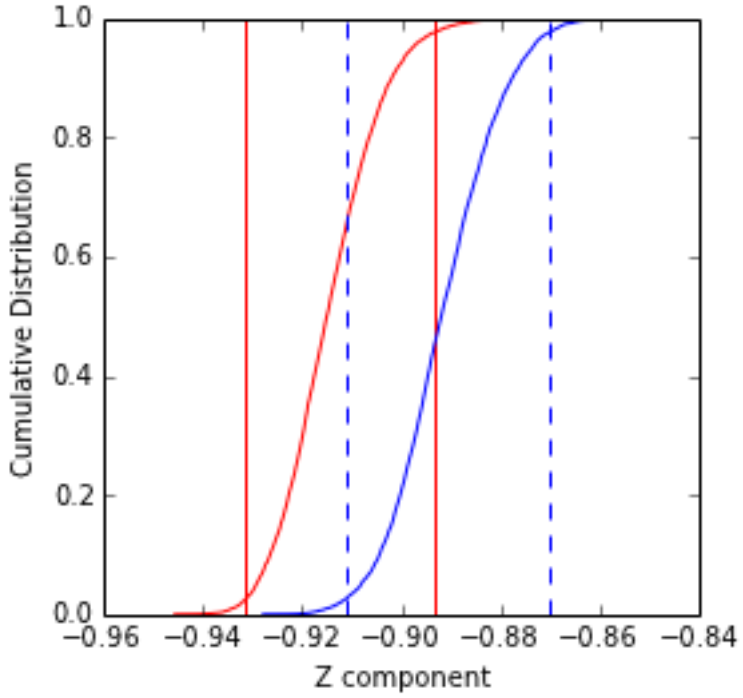
In [12]:
```
SI_Directions=[]
SI_Poles=[]

for n in range(0,84):
    Dec,Inc=Dec_TC[n],Inc_TC[n]
    SI_Directions.append([Dec,Inc,1.])
    Plong,Plat=VGP_long[n],VGP_lat[n]
    SI_Poles.append([Plong,Plat,1.])

SI_LowerHalf_Directions=SI_Directions[0:41]
SI_LowerHalf_Poles=SI_Poles[0:41]
SI_UpperHalf_Directions=SI_Directions[41:84]
SI_UpperHalf_Poles=SI_Poles[41:84]

pars_1=pmag.fisher_mean(SI_LowerHalf_Directions)
pars_2=pmag.fisher_mean(SI_UpperHalf_Directions)

print 'The fisher mean parameters for SI_LowerHalf_Directions are: '
print str(pars_1)
print ''
print 'The fisher mean parameters for SI_UpperHalf_Directions are: '
print str(pars_2)

IPmag.iWatsonV(SI_LowerHalf_Directions,SI_UpperHalf_Directions)
```

```
IPmag.iBootstrap(SI_LowerHalf_Directions,SI_UpperHalf_Directions)
```

```
The fisher mean parameters for SI_LowerHalf_Directions are:
{'csd': 11.890992902895112, 'k': 46.401689960333911, 'n': 41, 'r':
40.137962431234861, 'alpha95': 3.3119762971808, 'dec':
98.368399411680969, 'inc': -66.192087927867661}

The fisher mean parameters for SI_UpperHalf_Directions are:
{'csd': 11.246339102175027, 'k': 51.873755310093365, 'n': 43, 'r':
42.19034201883148, 'alpha95': 3.0524627905260902, 'dec':
109.92657619138177, 'inc': -63.09292348420847}
Results of Watson V test:

Watson's V:           10.4
Critical value of V:  6.5
"Fail": Since V is greater than Vcrit, the two means can
be distinguished at the 95% confidence level.

M&M1990 classification:

Angle between data set means: 5.8
Critical angle for M&M1990:   4.6


===============

Here are the results of the bootstrap test for a common mean
```







6 Calculating and plotting mean pole positions

6.1 Loading the basemap library in order to display maps and plot poles.

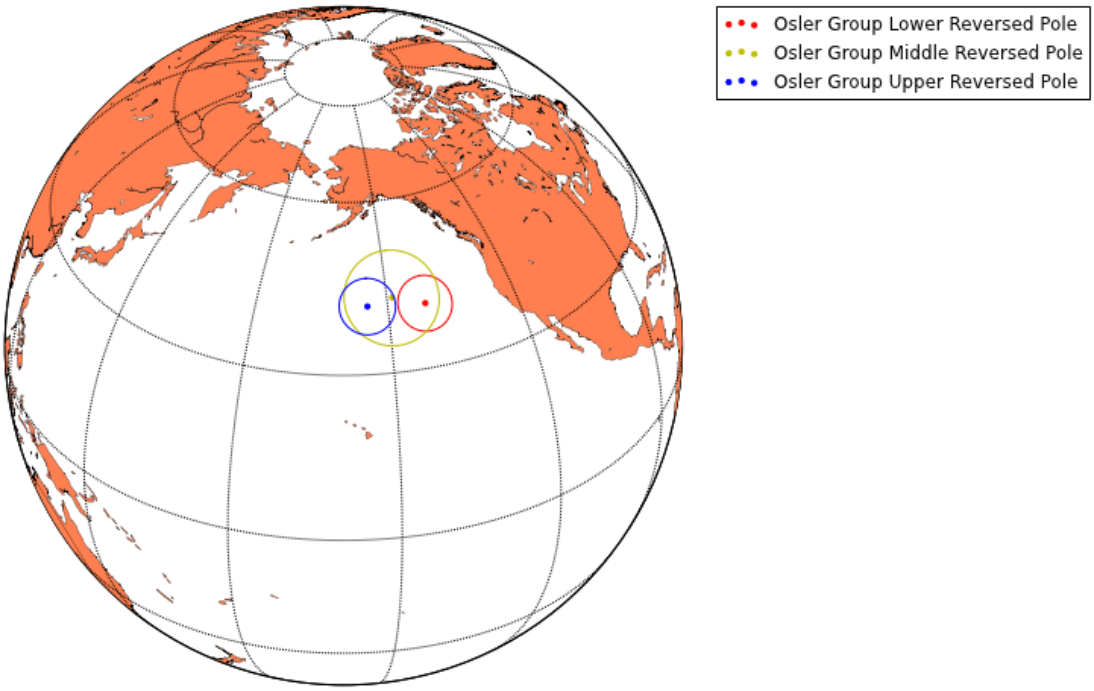
In [13]:
```
from mpl_toolkits.basemap import Basemap
```

The code below calculates the Fisher mean parameters for poles of the stratigraphically grouped virtual geomagnetic poles (VGPs) and then plots them on a “view from space” globe.

In [14]:
```
pars_1=pmag.fisher_mean(SI_LowerThird_Poles)
pars_2=pmag.fisher_mean(SI_MiddleThird_Poles)
pars_3=pmag.fisher_mean(SI_UpperThird_Poles)

print 'The fisher mean parameters for SI_LowerThird_Poles are: '
print str(pars_1)
print ''
print 'The fisher mean parameters for SI_MiddleThird_Poles are: '
print str(pars_2)
print ''
print 'The fisher mean parameters for SI_UpperThird_Poles are: '
print str(pars_3)

figure(figsize=(8, 8))
m = Basemap(projection='ortho',lat_0=35,lon_0=200,resolution='l')
# draw coastlines, country boundaries, fill continents.
m.drawcoastlines(linewidth=0.25)
#map.drawcountries(linewidth=0.25)
m.fillcontinents(color='coral',lake_color='white')
m.drawmapboundary(fill_color='white')
m.drawmeridians(np.arange(0,360,30))
m.drawparallels(np.arange(-90,90,30))

plong,plat=pars_1['dec'],pars_1['inc']
centerlon, centerlat = m(plong,plat)
A95=pars_1['alpha95']
A95_km=A95*111.32
# compute native map projection coordinates of lat/lon grid.
m.scatter(centerlon,centerlat,10,marker='o',color='r',
          label='Osler Group Lower Reversed Pole')
IPmag.equi(m, plong, plat, A95_km,'r')

plong,plat=pars_2['dec'],pars_2['inc']
centerlon, centerlat = m(plong,plat)
A95=pars_2['alpha95']
A95_km=A95*111.32
# compute native map projection coordinates of lat/lon grid.
m.scatter(centerlon,centerlat,10,marker='o',color='y',
          label='Osler Group Middle Reversed Pole')
IPmag.equi(m, plong, plat, A95_km,'y')

plong,plat=pars_3['dec'],pars_3['inc']
centerlon, centerlat = m(plong,plat)
A95=pars_3['alpha95']
A95_km=A95*111.32
# compute native map projection coordinates of lat/lon grid.
m.scatter(centerlon,centerlat,10,marker='o',color='b',
          label='Osler Group Upper Reversed Pole')
IPmag.equi(m, plong, plat, A95_km,'b')

legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()
```

```
The fisher mean parameters for SI_LowerThird_Poles are:
{'csd': 14.439647148676523, 'k': 31.467111290778522, 'n': 30, 'r':
29.078402852679442, 'alpha95': 4.7600623203905919, 'dec':
218.63783260894792, 'inc': 40.93731854099876}

The fisher mean parameters for SI_MiddleThird_Poles are:
```

```
{'csd': 19.701654620387572, 'k': 16.903032828663044, 'n': 20, 'r':
18.875941365517491, 'alpha95': 8.1783359746042343, 'dec':
211.26173612532102, 'inc': 42.737807885172735}

The fisher mean parameters for SI_UpperThird_Poles are:
{'csd': 15.523618638362732, 'k': 27.226016763674149, 'n': 34, 'r':
32.787924054905098, 'alpha95': 4.8039466857390298, 'dec':
205.41022751277507, 'inc': 41.564576182857046}
```



Note that in the print out for the mean pole calculations above 'dec' is really the longitude of the pole (Plong) and 'inc' is actually the latitude of the pole (Plat).

Pole ID

Plong

Plat

A95

N

SI_Lower

218.6

40.9

4.8

30

SI_Middle

211.3

42.7

8.2

20

SI_Upper

205.4

41.6

4.8

34

6.2 Comparision between data from Halls (1974) study and Simpson Island VGPs

Halls (1974) developed paleomagnetic data from the Osler Volcanic Group in the Nipogon Straits region (bibtex citation below; also see http://earthref.org/MAGIC/9518). These data are predominantly from flows with reversed magnetization below an angular unconformity on Puff Island that seperates the reversedly magnetized flows from younger flows of normal polarity (only a few of which are preserved before the sequence is submerged beneath Lake Superior). The analysis below compares paleomagnetic data from this study with the reversed flows (N=25) from the Halls (1974) data. The result from this analysis is that the Halls (1974) data is statistically distinct from the lower third of the Simpson Island stratigraphy, but is statitically indistinguishable from the upper third of the Simpson Island stratigraphy. This result fits with stratigraphic considerations that place the sites studied by Halls towards the top of the stratigraphy studied herein at Simpson Island. @articleHalls1974a, Author = Halls, H.C., Journal = Canadian Journal of Earth Science, Pages = 1200-1207, Title = A paleomagnetic reversal in the Osler Volcanic Group, northern Lake Superior, Volume = 11, Year = 1974 Import the flow site VGPs from the reversed flows (below the angular unconformity) of the Halls (1974) study

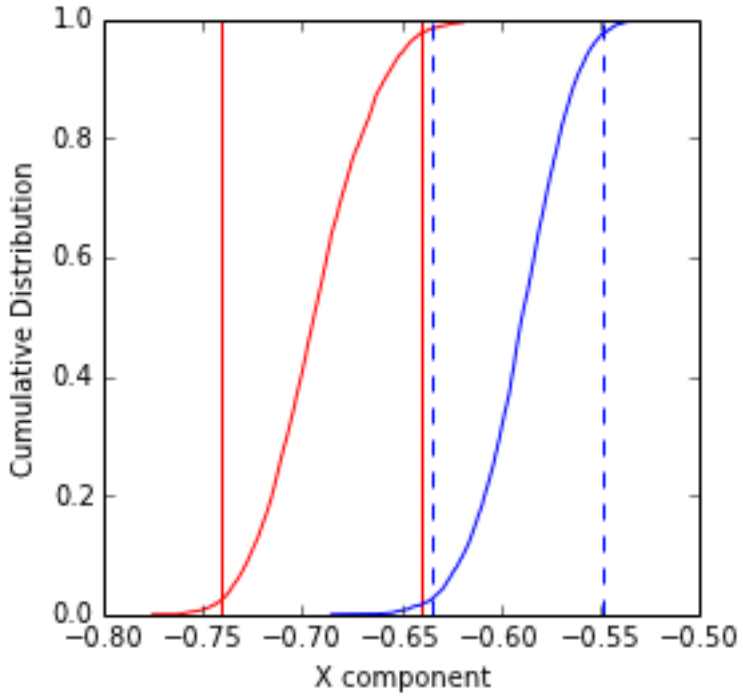
In [15]:
```
OslerHalls_VGP=[]

data_file='../2014_Osler_Data/Halls1974_Osler_reversed.csv'
Halls1974_Osler_reversed = np.genfromtxt(data_file,delimiter=",", skip_header=1)

Halls1974_VGP_long=Halls1974_Osler_reversed[:,0]
Halls1974_VGP_lat=Halls1974_Osler_reversed[:,1]

for VGP in range(0,len(Halls1974_VGP_lat)):
    Plong,Plat=Halls1974_VGP_long[VGP],Halls1974_VGP_lat[VGP]
    OslerHalls_VGP.append([Plong,Plat,1.])
```

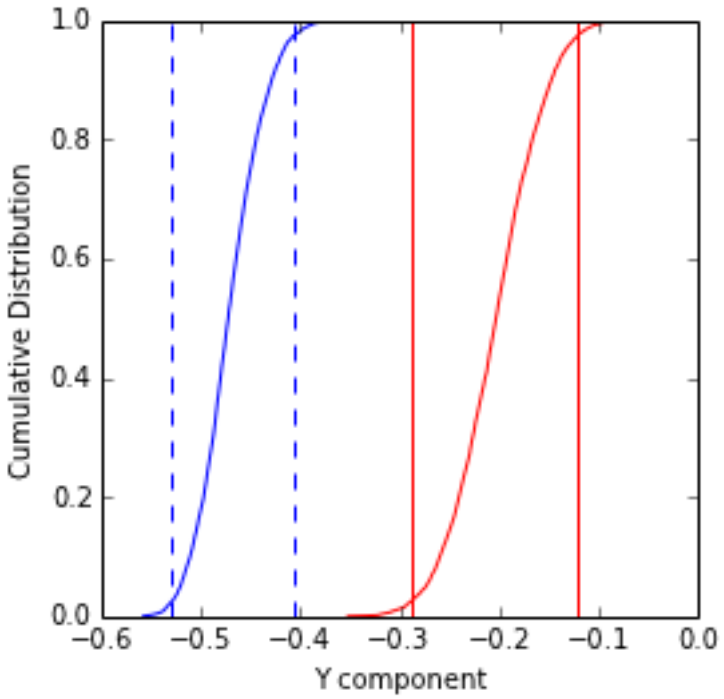
Common mean tests between Halls (1974) data and the data from the lower third of the Simpson Island stratigraphy

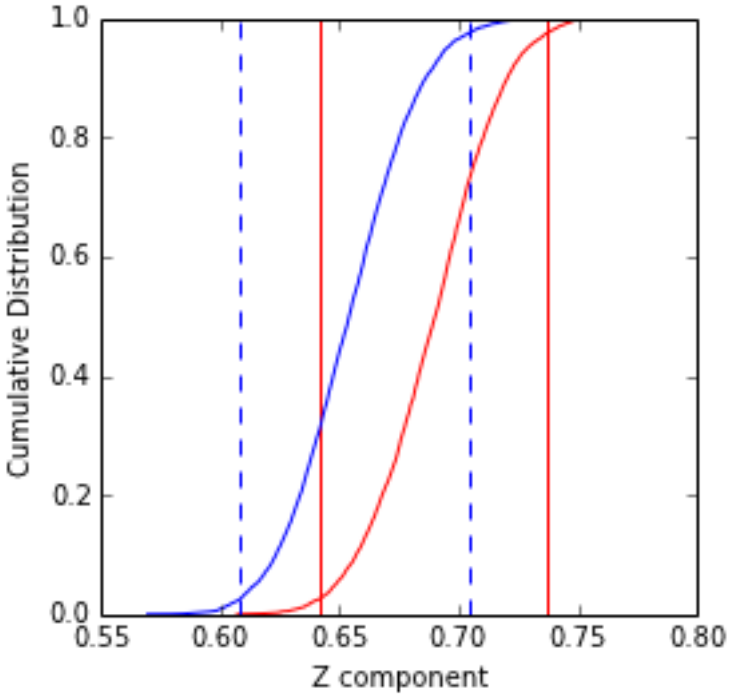
In [16]:
```
IPmag.iWatsonV(OslerHalls_VGP,SI_LowerThird_Poles)
IPmag.iBootstrap(OslerHalls_VGP,SI_LowerThird_Poles)
```

```
Results of Watson V test:

Watson's V:           31.0
Critical value of V:  6.6
"Fail": Since V is greater than Vcrit, the two means can
be distinguished at the 95% confidence level.

M&M1990 classification:
```

```
Angle between data set means: 16.7
Critical angle for M&M1990:   7.7

===============

Here are the results of the bootstrap test for a common mean
```





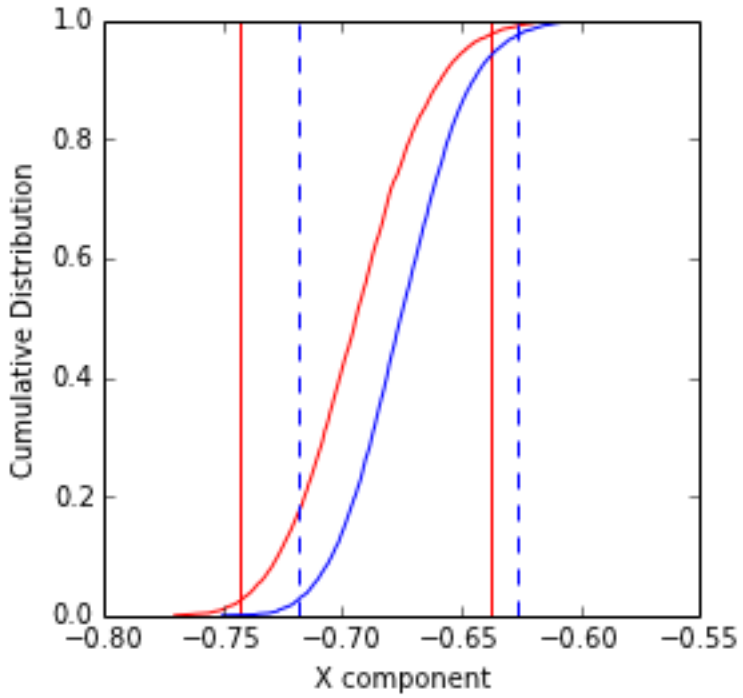


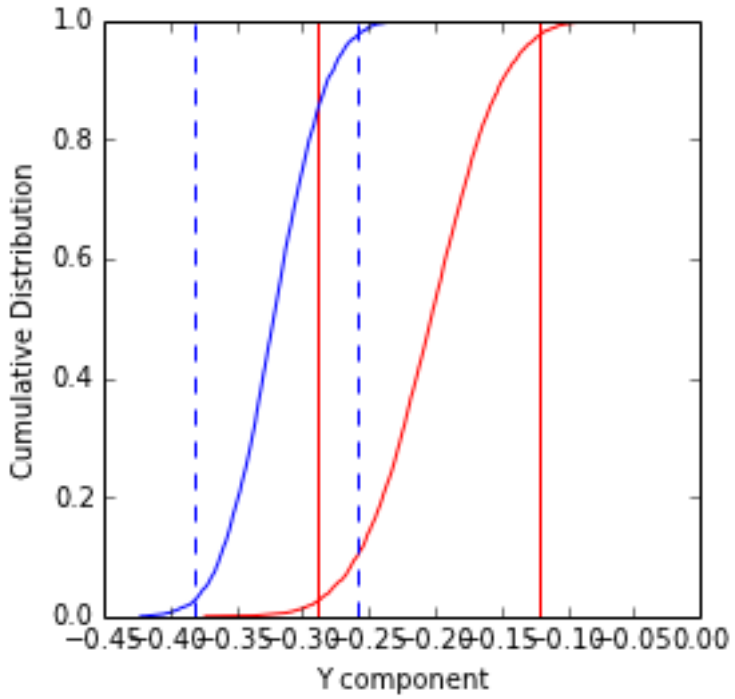
A comparision between VGPs calculated from Halls, 1974 data and data from the lower third of the Simpson Island stratigraphy fails Watson's V and bootstrap tests for a common mean indicating that the populations of directions do not share a common mean at the 95% confidence level.

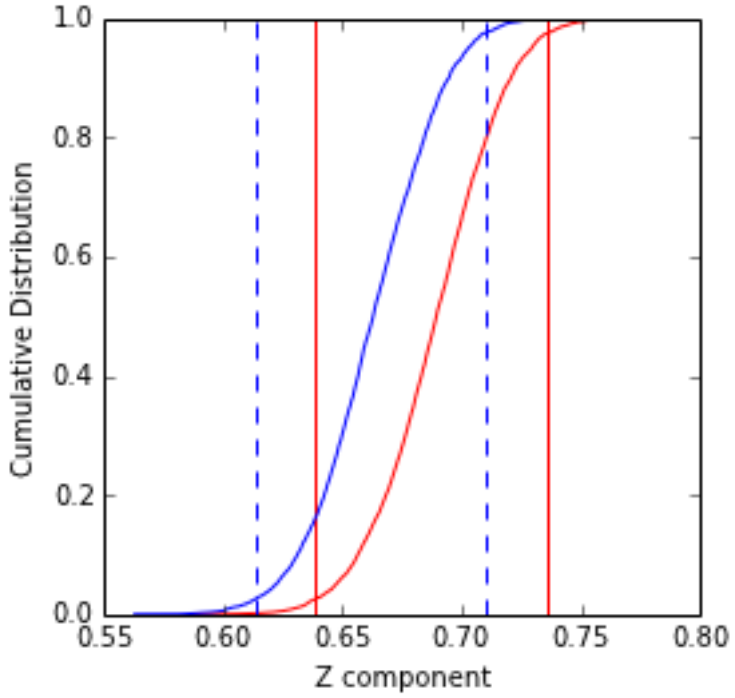
Common mean tests between Halls (1974) data and the data from the upper third of the Simpson Island stratigraphy

In [17]:
```
IPmag.iWatsonV(OslerHalls_VGP,SI_UpperThird_Poles)
IPmag.iBootstrap(OslerHalls_VGP,SI_UpperThird_Poles)
```

```
Results of Watson V test:

Watson's V:           5.4
Critical value of V:  7.0
"Pass": Since V is less than Vcrit, the null hypothesis
that the two populations are drawn from distributions
that share a common mean direction can not be rejected.

M&M1990 classification:

Angle between data set means: 7.0
Critical angle for M&M1990:   7.9
The McFadden and McElhinny (1990) classification for
this test is: 'B'

===============

Here are the results of the bootstrap test for a common mean
```







A comparision between VGPs calculated from Halls, 1974 data (see http://earthref.org/MAGIC/9518) and data from the upper third of the Simpson Island stratigraphy passes Watson's V and bootstrap tests for a common mean.

Combining the data from the upper third of the Simpson Island stratigraphy and the Halls (1974) reversed polarity data into a single pole for the upper portion of the reversed Osler Group stratigraphy

In [18]:
```
upperR_Osler_VGPs=OslerHalls_VGP+SI_UpperThird_Poles

pars_1=pmag.fisher_mean(upperR_Osler_VGPs)

print 'The fisher mean parameters for the pole calculated from upper reversed'
print 'Osler Group VGPs (SI_UpperThird_Poles+Halls, 1974 reversed data) are: '
print str(pars_1)
```

```
The fisher mean parameters for the pole calculated from upper reversed
Osler Group VGPs (SI_UpperThird_Poles+Halls, 1974 reversed data) are:
{'csd': 15.984670336321198, 'k': 25.678087233610714, 'n': 59, 'r':
56.741264780653822, 'alpha95': 3.7227743269632096, 'dec':
201.64704637965517, 'inc': 42.537024291694905}
```

Pole ID

Plong

Plat

A95

N

approximate age

upper Osler Group reversed pole (this work and Halls, 1974)

201.6

42.5

3.7

59

1105±2 Ma

6.3 Paleogeographic work outside of this IPython notebook using the software package GPlates.

For details on working with paleomagnetic data in GPlates check out this tutorial.

The steps necessary to develop the reconstruction using these Osler Volcanic Group poles are detailed below.

(1) Develop .rot file that rotates Laurentia such that the Osler poles are at geographic north. In the text for the .rot file below the Osler_lowerthird pole is assigned to 1110 Ma, the Osler_middlethird pole is assigned to 1107.5 and the Osler_upperthird pole is assigned to 1105 Ma.

The .rot file has the format:

moving plate ID number

age in millions of years

latitude of Euler pole

longitude of Euler pole

angle

fixed plate

comment

In this .rot file, the fixed reference frame is 000 and 001 (which could be split into a relative and absolute reference frame that seperates out TPW), while the plate ID for Laurentia is 199. 001 0.0 0.0 0.0 0.0 000 !
001 3900.0 0.0 0.0 0.0 000 !
199 0.0 0.0 0.0 0.0 001 !
199 1105.0 0.0 115.4 48.4 001 !Put Osler-upperthird at N for Laurentia
199 1107.5 0.0 121.3 47.3 001 !Put Osler-middlethird at N for Laurentia
199 1110.0 0.0 128.7 49.0 001 !Put Osler-lowerthird at N for Laurentia

(2) Open the following feature collections in GPlates: 199-Laurentia nogrid.dat, OslerPoles.gpml and Osler.rot where 199-Laurentia nogrid.dat is the outline of Laurentia in the late Proterozoic and OslerPoles.gpml contains the three calculated Osler paleomagnetic poles. An animation from 1110 to 1105 Ma yields the following reconstruction shown here with a time slice for Osler_lowerthird (red), Osler_middlethird (yellow) and Osler_upperthird (blue).

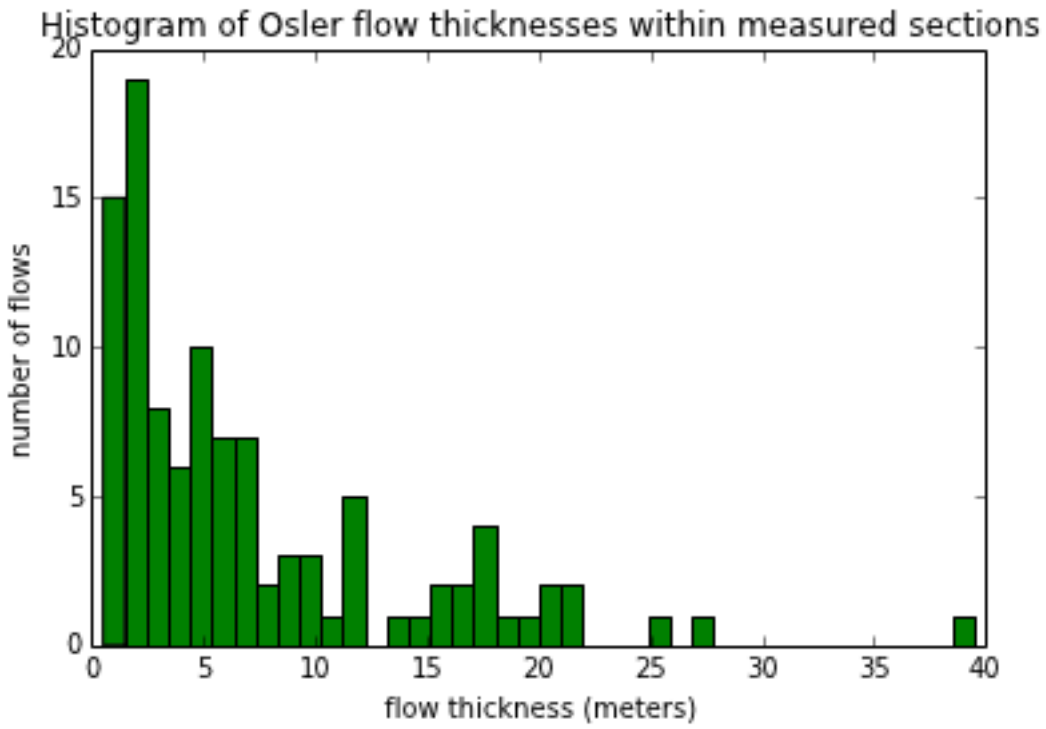
In [19]:
```
from IPython.core.display import SVG
Paleogeo=SVG(filename='Paleogeo.svg')
display(Paleogeo)
```

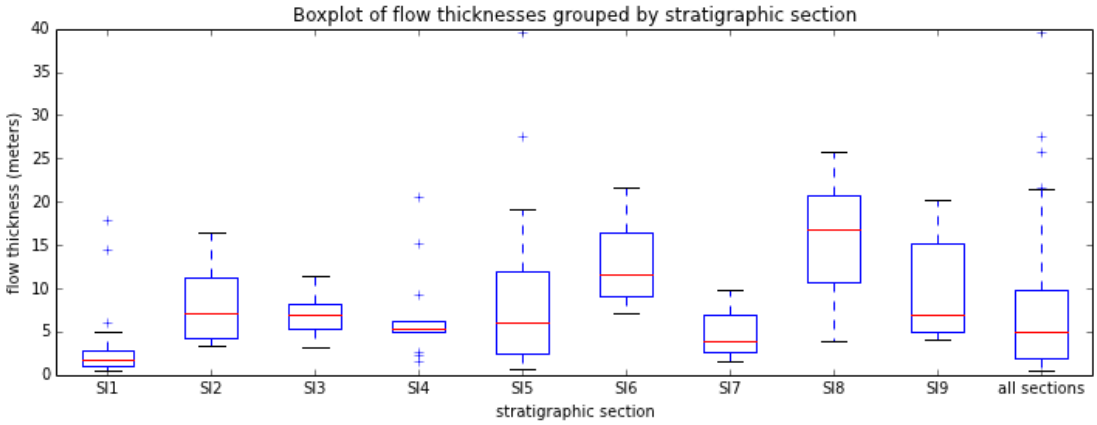
7 Lava flow thickness data

In the section of the text entitled “Osler Volcanic Group lithologies” the median flow thickness is reported from the subset of flows within measured sections where there is sufficient exposure of the flow base, interior and top to determine thickness. In the text it is reported that the median flow thickness is 4.9 meters with a first quartile thickness of 2.0 meters and third quartile thickness of 9.8 meters. The flow thickness data used in this analysis is presented and analyzed below.

In [20]:
```
#flow thicknesses where bottom and top are exposed without cover in between or with mi
SI1_flowthickness = array([1.2, 1.1+0.7,0.2+2.9+0.3,5.1+5.4+0.6+3.4,0.6+1+.3,0.9,0.5,2
SI2_flowthickness = array([0.6+2.8,5.7+2+3.6,1.3+0.7+1.1+1.1,6.2+1,1.8+3.3+11.4])
SI3_flowthickness = array([0.6+0.8+1+.8,1+4.9+1.1,0.3+1.6+4.5+1.2,0.2+1.5+7.2+0.9,0.5+
SI4_flowthickness = array([0.2+1.2+0.9,0.3+1.5+0.4+3.2,14+6.5,0.7+1+3.6,8.2+2.4+4.6,3.
SI5_flowthickness = array([6.6+9.8+7.4+1.4+2.3,10.2+0.8,5.6+2.8+1.4+2.2,0.2+2.8+2.4+0.
SI6_flowthickness = array([13.1+3.3,13.5+3.7,6.5+0.6,6.1+2.1+0.9,0.9+6.6+4.0,7.3+5.6+1
SI7_flowthickness = array([0.4+0.9+1.2+1.4,0.3+1.2,0.5+2.5+5+1.9])
SI8_flowthickness = array([1.4+1.9+0.6,0.5+1+8.8+4+1.1,5+15.8+2.8+2.1,5+4.2,8.4+8.1+5,
SI9_flowthickness = array([5.0+4.2,14.0+1+2.1,2.8+2.1,3.6+12.3+3.6+0.7,4.2+2.3,1.3+1.4

SIall_flowthickness = concatenate((SI1_flowthickness, SI2_flowthickness,
                                   SI3_flowthickness, SI4_flowthickness,
                                   SI5_flowthickness, SI6_flowthickness,
                                   SI7_flowthickness, SI8_flowthickness,
                                   SI9_flowthickness))
print "The number of flows for which thickness estimates could be obtained is:"
print len(SIall_flowthickness)
print "The mean flow thickness is:"
print mean(SIall_flowthickness)
print "The median flow thickness is:"
print median(SIall_flowthickness)

print "The first quartile thickness is:"
print percentile(SIall_flowthickness,25)
print "The third quartile thickness is:"
print percentile(SIall_flowthickness,75)

hist(SIall_flowthickness, 40, facecolor='green')
title('Histogram of Osler flow thicknesses within measured sections')
ylabel('number of flows')
xlabel('flow thickness (meters)')
plt.show()
data=(SI1_flowthickness,SI2_flowthickness,SI3_flowthickness,
      SI4_flowthickness,SI5_flowthickness, SI6_flowthickness,
      SI7_flowthickness,SI8_flowthickness,SI9_flowthickness,SIall_flowthickness)

figure(figsize=(12,4))
title('Boxplot of flow thicknesses grouped by stratigraphic section')
boxplot(data)
labels = ('SI1', 'SI2', 'SI3', 'SI4', 'SI5', 'SI6',
          'SI7', 'SI8', 'SI9', 'all sections')
xticks(range(1,11),labels)
xlabel('stratigraphic section')
ylabel('flow thickness (meters)')
plt.show()
```

```
The number of flows for which thickness estimates could be obtained
is:
105
The mean flow thickness is:
7.16666666667
The median flow thickness is:
4.9
The first quartile thickness is:
2.0
The third quartile thickness is:
9.8
```





8 The IPmag.py library of functions

Some of the functions used in the analysis above come from the imported IPmag.py library that was developed for this data analysis project. The code that is within IPmag.py is shown below so that the PDF output of this IPython notebook can document more of the tools used in the analysis without the reader needing to look into the .py file itself. This library utilizes functions from the pmag.py and pmagplotlib.py libraries of PmagPy version 2.206.

In [21]:
```
import pmag, pmagplotlib
import pylab
import numpy
import matplotlib.pyplot as plt

def iplotDI(DIblock,color='k'):
```

```
    """
    Plot declination, inclination data on a equal area plot

    This function modifies the plotDI function of PmagPy for use in the IPython notebo

    Parameters
    ----------

    DIblock : a DIblock is comprise of a list of unit vectors [dec,inc,1.]
    color : the default color is black. Other colors can be chosen (e.g. 'r')
    """
    # initialize the variables
    X_down,X_up,Y_down,Y_up=[],[],[],[]
    for rec in DIblock:
        Up,Down=0,0
        XY=pmag.dimap(rec[0],rec[1])
        if rec[1] >= 0:
            X_down.append(XY[0])
            Y_down.append(XY[1])
        else:
            X_up.append(XY[0])
            Y_up.append(XY[1])

    if len(X_up)>0:
        pylab.scatter(X_up,Y_up,facecolors='none', edgecolors=color)

    if len(X_down)>0:
        pylab.scatter(X_down,Y_down,facecolors=color, edgecolors=color)

def iBootstrap(Data1,Data2,NumSims=1000):
    """
    Conduct a bootstrap test (Tauxe, 2010) for a common mean on two declination, incli

    This function modifies code from PmagPy for use calculating and plotting bootstrap
    Three plots are generated (one for x, one for y and one for z).
    If the 95 percent confidence bounds for each component overlap each other, the two

    Parameters
    ----------

    Data1 : a list of directional data [dec,inc]
    Data2 : a list of directional data [dec,inc]
    NumSims : number of bootstrap samples (default is 1000)
    """
    counter=0
    BDI1=pmag.di_boot(Data1)
    BDI2=pmag.di_boot(Data2)
    print ""
    print "==============="
    print ""
    print "Here are the results of the bootstrap test for a common mean"
    CDF={'X':1,'Y':2,'Z':3}
    pylab.figure(CDF['X'],figsize=(4,4),dpi=160)
    pylab.figure(CDF['Y'],figsize=(4,4),dpi=160)
    pylab.figure(CDF['Z'],figsize=(4,4),dpi=160)
    pmagplotlib.plotCOM(CDF,BDI1,BDI2,["",""])



def iWatsonV(Data1,Data2,NumSims=1000):
    """
    Conduct a Watson V test for a common mean on two declination, inclination data set

    This function calculates Watson's V statistic from inumpyut files through Monte Ca
    in order to test whether two populations of directional data could have been drawn
    The critical angle between the two sample mean directions and the corresponding Mc
```

```
    Parameters
    ----------

    Data1 : a list of directional data [dec,inc]
    Data2 : a list of directional data [dec,inc]
    NumSims : number of Monte Carlo simulations (default is 1000)
    """
    pars_1=pmag.fisher_mean(Data1)
    pars_2=pmag.fisher_mean(Data2)

    cart_1=pmag.dir2cart([pars_1["dec"],pars_1["inc"],pars_1["r"]])
    cart_2=pmag.dir2cart([pars_2['dec'],pars_2['inc'],pars_2["r"]])
    Sw=pars_1['k']*pars_1['r']+pars_2['k']*pars_2['r'] # k1*r1+k2*r2
    xhat_1=pars_1['k']*cart_1[0]+pars_2['k']*cart_2[0] # k1*x1+k2*x2
    xhat_2=pars_1['k']*cart_1[1]+pars_2['k']*cart_2[1] # k1*y1+k2*y2
    xhat_3=pars_1['k']*cart_1[2]+pars_2['k']*cart_2[2] # k1*z1+k2*z2
    Rw=numpy.sqrt(xhat_1**2+xhat_2**2+xhat_3**2)
    V=2*(Sw-Rw)
    # keep weighted sum for later when determining the "critical angle"
    # let's save it as Sr (notation of McFadden and McElhinny, 1990)
    Sr=Sw

    # do monte carlo simulation of datasets with same kappas as data,
    # but a common mean
    counter=0
    Vp=[] # set of Vs from simulations
    for k in range(NumSims):

    # get a set of N1 fisher distributed vectors with k1,
    # calculate fisher stats
        Dirp=[]
        for i in range(pars_1["n"]):
            Dirp.append(pmag.fshdev(pars_1["k"]))
        pars_p1=pmag.fisher_mean(Dirp)
    # get a set of N2 fisher distributed vectors with k2,
    # calculate fisher stats
        Dirp=[]
        for i in range(pars_2["n"]):
            Dirp.append(pmag.fshdev(pars_2["k"]))
        pars_p2=pmag.fisher_mean(Dirp)
    # get the V for these
        Vk=pmag.vfunc(pars_p1,pars_p2)
        Vp.append(Vk)

    # sort the Vs, get Vcrit (95th percentile one)

    Vp.sort()
    k=int(.95*NumSims)
    Vcrit=Vp[k]

    # equation 18 of McFadden and McElhinny, 1990 calculates the critical
    # value of R (Rwc)

    Rwc=Sr-(Vcrit/2)

    # following equation 19 of McFadden and McElhinny (1990) the critical
    # angle is calculated. If the observed angle (also calculated below)
    # between the data set means exceeds the critical angle the hypothesis
    # of a common mean direction may be rejected at the 95% confidence
    # level. The critical angle is simply a different way to present
    # Watson's V parameter so it makes sense to use the Watson V parameter
    # in comparison with the critical value of V for considering the test
    # results. What calculating the critical angle allows for is the
    # classification of McFadden and McElhinny (1990) to be made
    # for data sets that are consistent with sharing a common mean.

    k1=pars_1['k']
```

```
    k2=pars_2['k']
    R1=pars_1['r']
    R2=pars_2['r']
    critical_angle=numpy.degrees(numpy.arccos(((Rwc**2)-((k1*R1)**2)
                                               -((k2*R2)**2))/
                                              (2*k1*R1*k2*R2)))
    D1=(pars_1['dec'],pars_1['inc'])
    D2=(pars_2['dec'],pars_2['inc'])
    angle=pmag.angle(D1,D2)

    print "Results of Watson V test: "
    print ""
    print "Watson's V:           " '%.1f' %(V)
    print "Critical value of V:  " '%.1f' %(Vcrit)

    if V<Vcrit:
        print '"Pass": Since V is less than Vcrit, the null hypothesis'
        print 'that the two populations are drawn from distributions'
        print 'that share a common mean direction can not be rejected.'
    elif V>Vcrit:
        print '"Fail": Since V is greater than Vcrit, the two means can'
        print 'be distinguished at the 95% confidence level.'
    print ""
    print "M&M1990 classification:"
    print ""
    print "Angle between data set means: " '%.1f'%(angle)
    print "Critical angle for M&M1990:   " '%.1f'%(critical_angle)

    if V>Vcrit:
        print ""
    elif V<Vcrit:
        if critical_angle<5:
            print "The McFadden and McElhinny (1990) classification for"
            print "this test is: 'A'"
        elif critical_angle<10:
            print "The McFadden and McElhinny (1990) classification for"
            print "this test is: 'B'"
        elif critical_angle<20:
            print "The McFadden and McElhinny (1990) classification for"
            print "this test is: 'C'"
        else:
            print "The McFadden and McElhinny (1990) classification for"
            print "this test is: 'INDETERMINATE;"

def lat_from_i(inc):
    """
    Calculate paleolatitude from inclination using the dipole equation
    """
    rad=numpy.pi/180.
    paleo_lat=numpy.arctan( 0.5*numpy.tan(inc*rad))/rad
    return paleo_lat

def shoot(lon, lat, azimuth, maxdist=None):
    """
    This function enables A95 error ellipses to be drawn in basemap around paleomagnet
    (from: http://www.geophysique.be/2011/02/20/matplotlib-basemap-tutorial-09-drawing
    """
    glat1 = lat * numpy.pi / 180.
    glon1 = lon * numpy.pi / 180.
    s = maxdist / 1.852
    faz = azimuth * numpy.pi / 180.

    EPS= 0.00000000005
    if ((numpy.abs(numpy.cos(glat1))<EPS) and not (numpy.abs(numpy.sin(faz))<EPS)):
        alert("Only N-S courses are meaningful, starting at a pole!")

    a=6378.13/1.852
```

```
    f=1/298.257223563
    r = 1 - f
    tu = r * numpy.tan(glat1)
    sf = numpy.sin(faz)
    cf = numpy.cos(faz)
    if (cf==0):
        b=0.
    else:
        b=2. * numpy.arctan2 (tu, cf)

    cu = 1. / numpy.sqrt(1 + tu * tu)
    su = tu * cu
    sa = cu * sf
    c2a = 1 - sa * sa
    x = 1. + numpy.sqrt(1. + c2a * (1. / (r * r) - 1.))
    x = (x - 2.) / x
    c = 1. - x
    c = (x * x / 4. + 1.) / c
    d = (0.375 * x * x - 1.) * x
    tu = s / (r * a * c)
    y = tu
    c = y + 1
    while (numpy.abs (y - c) > EPS):

        sy = numpy.sin(y)
        cy = numpy.cos(y)
        cz = numpy.cos(b + y)
        e = 2. * cz * cz - 1.
        c = y
        x = e * cy
        y = e + e - 1.
        y = (((sy * sy * 4. - 3.) * y * cz * d / 6. + x) *
              d / 4. - cz) * sy * d + tu

    b = cu * cy * cf - su * sy
    c = r * numpy.sqrt(sa * sa + b * b)
    d = su * cy + cu * sy * cf
    glat2 = (numpy.arctan2(d, c) + numpy.pi) % (2*numpy.pi) - numpy.pi
    c = cu * cy - su * sy * cf
    x = numpy.arctan2(sy * sf, c)
    c = ((-3. * c2a + 4.) * f + 4.) * c2a * f / 16.
    d = ((e * cy * c + cz) * sy * c + y) * sa
    glon2 = ((glon1 + x - (1. - c) * d * f + numpy.pi) % (2*numpy.pi)) - numpy.pi

    baz = (numpy.arctan2(sa, b) + numpy.pi) % (2 * numpy.pi)

    glon2 *= 180./numpy.pi
    glat2 *= 180./numpy.pi
    baz *= 180./numpy.pi

    return (glon2, glat2, baz)

def equi(m, centerlon, centerlat, radius, color):
    """
    This function enables A95 error ellipses to be drawn in basemap around paleomagnet
    (from: http://www.geophysique.be/2011/02/20/matplotlib-basemap-tutorial-09-drawing
    """
    glon1 = centerlon
    glat1 = centerlat
    X = []
    Y = []
    for azimuth in range(0, 360):
        glon2, glat2, baz = shoot(glon1, glat1, azimuth, radius)
        X.append(glon2)
        Y.append(glat2)
    X.append(X[0])
    Y.append(Y[0])
```

```
    X,Y = m(X,Y)
    plt.plot(X,Y,color)
```