Statistical_Modeling_for_Soccer_Games

Unknown Author

May 6, 2014

Part I

Stat222 Capstone Project: Statistical Modeling for Soccer Games

1 Authors: Shengying Wang and Siwei Tu

Part II

Introduction

2 Background

With the approaching of 2014 World Cup, we are interested in applying our statistical knowledge and skills to analyze some interesting questions related to one of the world's most popular sports, soccer.

3 Questions to answer

How to predicte the detailed results of 2014 World Cup in Brazil?

How to compare the players' actual performance rating to see which players deserve theri reputations?

4 Data Source

Unfortunately we can't find free downable data stes for our analysis. So we looked for web pages with the statistics about players and teams' performances in various season, and use their htlm source code to create our own data by Pyton regular expression and string maniplating techniques.The following is our data source reference. We have totally 83 data files.

Format: **Filename.txt** : **url of the website**

In [1]:

```
#filename.txt : url (html cource code)
#argentina : http://www.ca2011.com/estatistica_porselecao.php?idS=eccbc87e4b5ce2fe2830
#brazil : http://www.ca2011.com/estatistica_porselecao.php?idS=c4ca4238a0b923820dcc509
#brazilplayers13 : http://www.whoscored.com/Regions/31/Tournaments/95/Seasons/3753/Sta
#bundesliga0809 : http://espnfc.com/tables/_/league/ger.1/season/2008/german-bundeslig
#bundesliga0910 : http://espnfc.com/tables/_/league/ger.1/season/2009/german-bundeslig
#bundesliga1011 : http://espnfc.com/tables/_/league/ger.1/season/2010/german-bundeslig
#bundesliga1112 : http://espnfc.com/tables/_/league/ger.1/season/2011/german-bundeslig
#bundesliga1213 : http://espnfc.com/tables/_/league/ger.1/season/2012/german-bundeslig
#bundesligaplayers0910 : http://www.whoscored.com/Regions/81/Tournaments/3/Seasons/190
#bundesligaplayers1011 : http://www.whoscored.com/Regions/81/Tournaments/3/Seasons/252
#bundesligaplayers1112 : http://www.whoscored.com/Regions/81/Tournaments/3/Seasons/294
#bundesligaplayers1213 : http://www.whoscored.com/Regions/81/Tournaments/3/Seasons/342
#bundesligaplayers1314 : http://www.whoscored.com/Regions/81/Tournaments/3/Seasons/386
#championplayers1314 : http://www.whoscored.com/Regions/252/Tournaments/7/Seasons/3859
#chile : http://www.ca2011.com/estatistica_porselecao.php?idS=e4da3b7fbbce2345d7772b06
#colombia : http://www.ca2011.com/estatistica_porselecao.php?idS=1679091c5a880faf6fb5e
#costa : http://www.ca2011.com/estatistica_porselecao.php?idS=9bf31c7ff062936a96d3c8bd
#dutch0809 : http://espnfc.com/tables/_/league/ned.1/season/2008/dutch-eredivisie?cc=5
#dutch0910 : http://espnfc.com/tables/_/league/ned.1/season/2009/dutch-eredivisie?cc=5
#dutch1011 : http://espnfc.com/tables/_/league/ned.1/season/2010/dutch-eredivisie?cc=5
#dutch1112 : http://espnfc.com/tables/_/league/ned.1/season/2011/dutch-eredivisie?cc=5
#dutch1213 : http://espnfc.com/tables/_/league/ned.1/season/2012/dutch-eredivisie?cc=5
#dutchplayers1314 : http://www.whoscored.com/Regions/155/Tournaments/13/Seasons/3851/S
#ecuador : http://www.ca2011.com/estatistica_porselecao.php?idS=8f14e45fceea167a5a36de
#EURO2012 : http://www.uefa.com/uefaeuro/season=2012/statistics/round=15172/teams/inde
#french0809 : http://espnfc.com/tables/_/league/fra.1/season/2008/french-ligue-1?cc=59
#french0910 : http://espnfc.com/tables/_/league/fra.1/season/2009/french-ligue-1?cc=59
#french1011 : http://espnfc.com/tables/_/league/fra.1/season/2010/french-ligue-1?cc=59
#french1112 : http://espnfc.com/tables/_/league/fra.1/season/2011/french-ligue-1?cc=59
#french1213 : http://espnfc.com/tables/_/league/fra.1/season/2012/french-ligue-1?cc=59
#frenchplayers0910 : http://www.whoscored.com/Regions/74/Tournaments/22/Seasons/1839/S
#frenchplayers1011 : http://www.whoscored.com/Regions/74/Tournaments/22/Seasons/2417/S
#frenchplayers1112 : http://www.whoscored.com/Regions/74/Tournaments/22/Seasons/2920/S
#frenchplayers1213 : http://www.whoscored.com/Regions/74/Tournaments/22/Seasons/3356/S
#frenchplayers1314 : http://www.whoscored.com/Regions/74/Tournaments/22/Seasons/3836/S
#laliga0809 : http://espnfc.com/tables/_/league/esp.1/season/2008/spanish-primera-divi
#laliga0910 : http://espnfc.com/tables/_/league/esp.1/season/2009/spanish-primera-divi
#laliga1011 : http://espnfc.com/tables/_/league/esp.1/season/2010/spanish-primera-divi
#laliga1112 : http://espnfc.com/tables/_/league/esp.1/season/2011/spanish-primera-divi
#laliga1213 : http://espnfc.com/tables/_/league/esp.1/season/2012/spanish-primera-divi
#laligaplayers0910 : http://www.whoscored.com/Regions/206/Tournaments/4/Seasons/1929/S
#laligaplayers1011 : http://www.whoscored.com/Regions/206/Tournaments/4/Seasons/2596/S
#laligaplayers1112 : http://www.whoscored.com/Regions/206/Tournaments/4/Seasons/3004/S
#laligaplayers1213 : http://www.whoscored.com/Regions/206/Tournaments/4/Seasons/3922/S
#laligaplayers1314 : http://www.whoscored.com/Regions/206/Tournaments/4/Seasons/3922/S
#majorplayers13 : http://www.whoscored.com/Regions/233/Tournaments/85/Seasons/3672/Sta
#mexico : http://www.ca2011.com/estatistica_porselecao.php?idS=45c48cce2e2d7fbdea1afc5
#premier0809 : http://espnfc.com/tables/_/league/eng.1/season/2008/barclays-premier-le
#premier0910 : http://espnfc.com/tables/_/league/eng.1/season/2009/barclays-premier-le
#premier1011 : http://espnfc.com/tables/_/league/eng.1/season/2010/barclays-premier-le
#premier1112 : http://espnfc.com/tables/_/league/eng.1/season/2011/barclays-premier-le
#premier1213 : http://espnfc.com/tables/_/league/eng.1/season/2012/barclays-premier-le
#premierplayers0910 : http://www.whoscored.com/Regions/252/Tournaments/2/Seasons/1849/
#premierplayers1011 : http://www.whoscored.com/Regions/252/Tournaments/2/Seasons/2458/
#premierplayers1112 : http://www.whoscored.com/Regions/252/Tournaments/2/Seasons/2935/
#premierplayers1213 : http://www.whoscored.com/Regions/252/Tournaments/2/Seasons/3389/
#premierplayers1314 : http://www.whoscored.com/Regions/252/Tournaments/2/Seasons/3853/
#russianplayers1314 : http://www.whoscored.com/Regions/182/Tournaments/77/Seasons/3861
#serieA0809 : http://espnfc.com/tables/_/league/ita.1/season/2008/italian-serie-a?cc=5
#serieA0910 : http://espnfc.com/tables/_/league/ita.1/season/2009/italian-serie-a?cc=5
#serieA1011 : http://espnfc.com/tables/_/league/ita.1/season/2010/italian-serie-a?cc=5
#serieA1112 : http://espnfc.com/tables/_/league/ita.1/season/2011/italian-serie-a?cc=5
#serieA1213 : http://espnfc.com/tables/_/league/ita.1/season/2012/italian-serie-a?cc=5
#serieAplayers0910 : http://www.whoscored.com/Regions/108/Tournaments/5/Seasons/1957/S
```

```
#serieAplayers1011 : http://www.whoscored.com/Regions/108/Tournaments/5/Seasons/2626/S
#serieAplayers1112 : http://www.whoscored.com/Regions/108/Tournaments/5/Seasons/3054/S
#serieAplayers1213 : http://www.whoscored.com/Regions/108/Tournaments/5/Seasons/3512/S
#serieAplayers1314 : http://www.whoscored.com/Regions/108/Tournaments/5/Seasons/3978/S
#uefa0304 : http://www.uefa.com/uefachampionsleague/season=2003/statistics/round=1712/
#uefa0405 : http://www.uefa.com/uefachampionsleague/season=2004/statistics/round=1968/
#uefa0506 : http://www.uefa.com/uefachampionsleague/season=2005/statistics/round=2201/
#uefa0607 : http://www.uefa.com/uefachampionsleague/season=2006/statistics/round=2357/
#uefa0708 : http://www.uefa.com/uefachampionsleague/season=2008/statistics/round=15105
#uefa0809 : http://www.uefa.com/uefachampionsleague/season=2009/statistics/round=15276
#uefa0910 : http://www.uefa.com/uefachampionsleague/season=2010/statistics/round=20000
#uefa1011 : http://www.uefa.com/uefachampionsleague/season=2011/statistics/round=20001
#uefa1112 : http://www.uefa.com/uefachampionsleague/season=2012/statistics/round=20002
#uefa1213 : http://www.uefa.com/uefachampionsleague/season=2013/statistics/round=20003
#uruguay : http://www.ca2011.com/estatistica_porselecao.php?idS=c20ad4d76fe97759aa27a0
#worldcup2010att : http://www.fifa.com/tournaments/archive/worldcup/southafrica2010/st
#worldcup2010cor : http://www.fifa.com/tournaments/archive/worldcup/southafrica2010/st
#worldcup2010def : http://www.fifa.com/tournaments/archive/worldcup/southafrica2010/st
#worldcup2010dis [U+FF1A]http://www.fifa.com/tournaments/archive/worldcup/southafrica2
#worldcup2010shot [U+FF1A]http://www.fifa.com/tournaments/archive/worldcup/southafrica
```

4.1 Assumption

In soccer games, we assume the number of goals scored by each team are Poisson Distributions.

In [1]:

Part III

World Cup Analysis

Part IV

Home away effect

In [2]:
```
import re
import numpy as np
from pandas import DataFrame
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
%matplotlib inline
```

Input the European leagues data from season 08/09 to 12/13, including England, Germany, France, Dutch, Italy and Spain.

In [3]:
```
#read data for English league
filename = 'premier0809.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Manchester United')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
```

```
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
premier0809 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','

filename = 'premier0910.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Chelsea')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
premier0910 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','

filename = 'premier1011.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Manchester United')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
premier1011 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','

filename = 'premier1112.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Manchester City')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
premier1112 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','

filename = 'premier1213.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Manchester United')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
premier1213 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','

premier1213hf = premier1213['hf']
premier1213af = premier1213['af']

#compute the different performances for home team and guest team
```

```
premierhf = [np.mean(premier0809['hf']),np.mean(premier0910['hf']),np.mean(premier1011
premieraf = [np.mean(premier0809['af']),np.mean(premier0910['af']),np.mean(premier1011

#read data for German league
filename = 'bundesliga0809.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('VfL Wolfsburg')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
bundesliga0809 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd

filename = 'bundesliga0910.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Bayern Munich')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
bundesliga0910 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd

filename = 'bundesliga1011.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Borussia Dortmund')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
bundesliga1011 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd

filename = 'bundesliga1112.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Borussia Dortmund')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
bundesliga1112 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd

filename = 'bundesliga1213.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Bayern Munich')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
```

```
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
bundesliga1213 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd

bundesligahf = [np.mean(bundesliga0809['hf']),np.mean(bundesliga0910['hf']),np.mean(bu
bundesligaaf = [np.mean(bundesliga0809['af']),np.mean(bundesliga0910['af']),np.mean(bu


#read data for Dutch league
filename = 'dutch0809.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('AZ Alkmaar')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
dutch0809 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','hl

filename = 'dutch0910.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Twente Enschede')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
dutch0910 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','hl

filename = 'dutch1011.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Ajax Amsterdam')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
dutch1011 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','hl

filename = 'dutch1112.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Ajax Amsterdam')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
```

```
index = data8[:,0]
dutch1112 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','hl

filename = 'dutch1213.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Ajax Amsterdam')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(18,19)
index = data8[:,0]
dutch1213 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','hl

dutchhf = [np.mean(dutch0809['hf']),np.mean(dutch0910['hf']),np.mean(dutch1011['hf']),
dutchaf = [np.mean(dutch0809['af']),np.mean(dutch0910['af']),np.mean(dutch1011['af']),


#read data for French league
filename = 'french0809.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Bordeaux')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
french0809 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'french0910.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Marseille')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
french0910 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'french1011.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Lille')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
french1011 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'french1112.txt'
txt = open(filename)
```

```
data1 = txt.read()
data2 = data1.split('Montpellier')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
french1112 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'french1213.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Paris')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
french1213 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

frenchhf = [np.mean(french0809['hf']),np.mean(french0910['hf']),np.mean(french1011['hf
frenchaf = [np.mean(french0809['af']),np.mean(french0910['af']),np.mean(french1011['af


#read data for Spanish league
filename = 'laliga0809.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Barcelona')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
laliga0809 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'laliga0910.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Barcelona')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
laliga0910 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'laliga1011.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Barcelona')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
```

```
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
laliga1011 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'laliga1112.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Real Madrid')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
laliga1112 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'laliga1213.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Barcelona')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
laliga1213 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

laligahf = [np.mean(laliga0809['hf']),np.mean(laliga0910['hf']),np.mean(laliga1011['hf
laligaaf = [np.mean(laliga0809['af']),np.mean(laliga0910['af']),np.mean(laliga1011['af


#read data for Italian league
filename = 'serieA0809.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Internazionale')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
serieA0809 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'serieA0910.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Internazionale')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
```

```
serieA0910 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'serieA1011.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('AC Milan')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
serieA1011 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'serieA1112.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Juventus')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
serieA1112 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

filename = 'serieA1213.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Juventus')
data3 = data2[1].split('tbody')
regex = re.compile('>-*[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = ['1']+data6
data8 = np.array(data7).reshape(20,19)
index = data8[:,0]
serieA1213 = DataFrame(data8,columns =['id','op','ow','od','ol','of','oa','hw','hd','h

serieAhf = [np.mean(serieA0809['hf']),np.mean(serieA0910['hf']),np.mean(serieA1011['hf
serieAaf = [np.mean(serieA0809['af']),np.mean(serieA0910['af']),np.mean(serieA1011['af


#total "home-away" difference for each country
countryhf = [np.mean(dutchhf),np.mean(premierhf),np.mean(serieAhf),np.mean(bundesligah
countryaf = [np.mean(dutchaf),np.mean(premieraf),np.mean(serieAaf),np.mean(bundesligaa
```

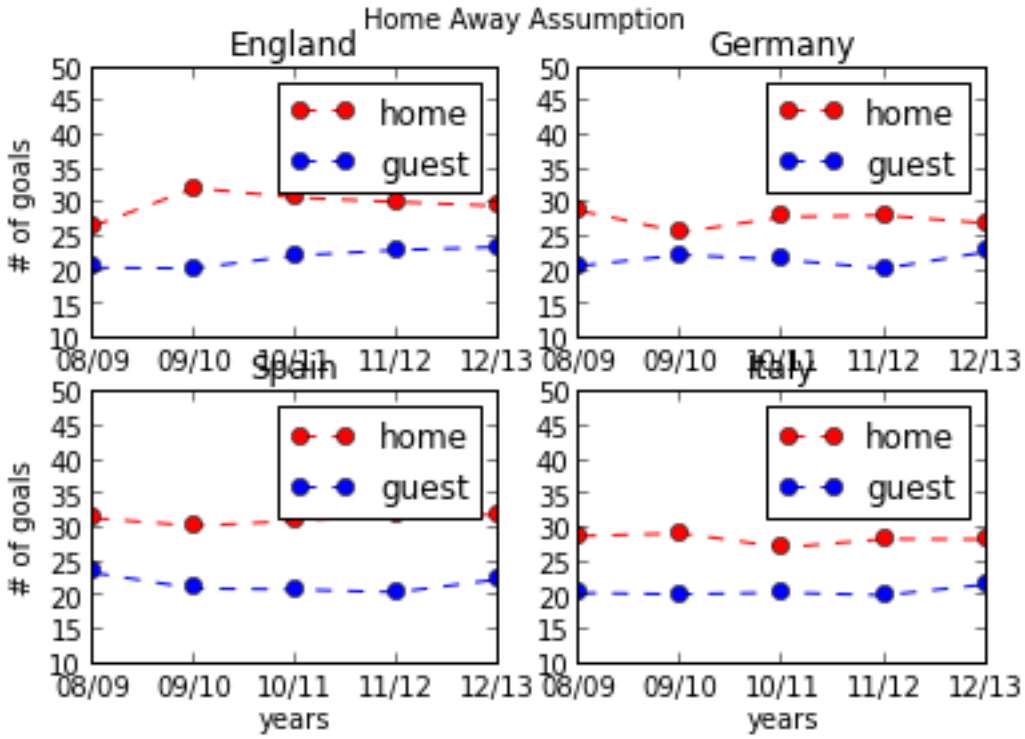
Plot home-away effect for each league.

In [4]:

```
year = [9,10,11,12,13]
fig = plt.figure()

ax1 = fig.add_subplot(2, 2, 1)
ax1.plot(year, premierhf, color = 'r', label = 'home', linestyle='dashed', marker='o')
ax1.plot(year, premieraf, color = 'b', label = 'guest', linestyle='dashed', marker='o'
plt.ylim([10, 50])
ax1.legend(loc='best')
ticks = ax1.set_xticks([9, 10, 11, 12, 13])
labels = ax1.set_xticklabels(['08/09', '09/10', '10/11', '11/12', '12/13'])
ax1.set_ylabel('# of goals')
ax1.set_title('England')
```

```
ax2 = fig.add_subplot(2, 2, 2)
ax2.plot(year, bundesligahf, color = 'r', label = 'home', linestyle='dashed', marker='
ax2.plot(year, bundesligaaf, color = 'b', label = 'guest', linestyle='dashed', marker=
plt.ylim([10, 50])
ax2.legend(loc='best')
ticks = ax2.set_xticks([9, 10, 11, 12, 13])
labels = ax2.set_xticklabels(['08/09', '09/10', '10/11', '11/12', '12/13'])
ax2.set_title('Germany')

ax3 = fig.add_subplot(2, 2, 3)
ax3.plot(year, laligahf, color = 'r', label = 'home', linestyle='dashed', marker='o')
ax3.plot(year, laligaaf, color = 'b', label = 'guest', linestyle='dashed', marker='o')
plt.ylim([10, 50])
ax3.legend(loc='best')
ticks = ax3.set_xticks([9, 10, 11, 12, 13])
labels = ax3.set_xticklabels(['08/09', '09/10', '10/11', '11/12', '12/13'])
ax3.set_title('Spain')
ax3.set_ylabel('# of goals')
ax3.set_xlabel('years')

ax4 = fig.add_subplot(2, 2, 4)
ax4.plot(year, serieAhf, color = 'r', label = 'home', linestyle='dashed', marker='o')
ax4.plot(year, serieAaf, color = 'b', label = 'guest', linestyle='dashed', marker='o')
plt.ylim([10, 50])
ax4.legend(loc='best')
ticks = ax4.set_xticks([9, 10, 11, 12, 13])
labels = ax4.set_xticklabels(['08/09', '09/10', '10/11', '11/12', '12/13'])
ax4.set_title('Italy')
ax4.set_xlabel('years')
plt.suptitle('Home Away Assumption')
<matplotlib.text.Text at 0x505d9d0>
```

Out [4]:



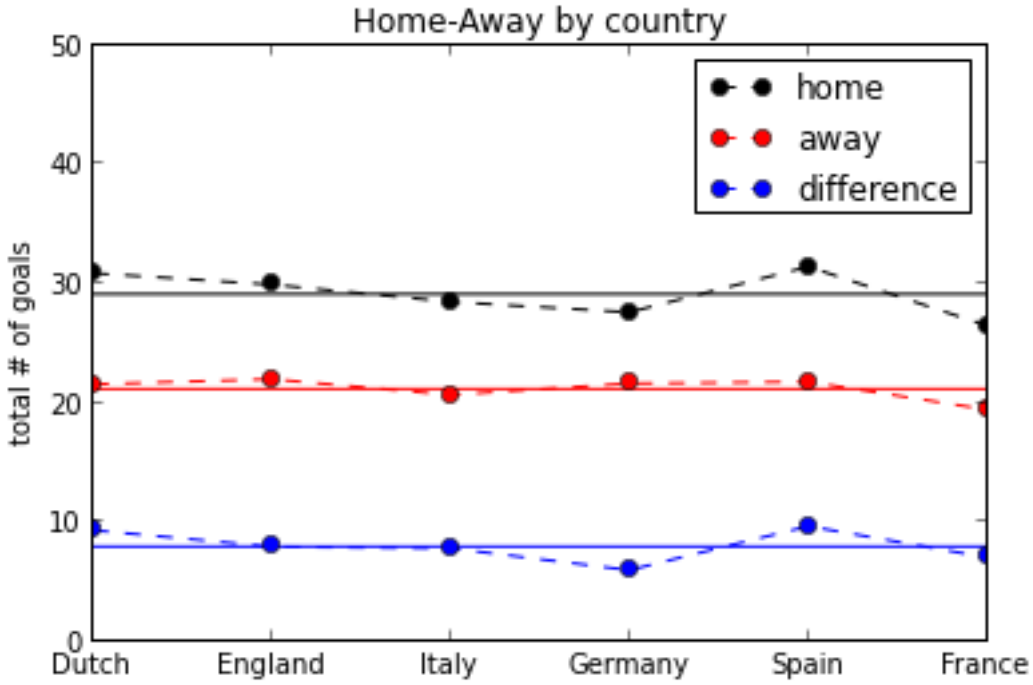
The average number of goals scored by home teas is significantly higher than guest teams. So we can conclude that the home team does have some advantages. The average is 0.22 per game.

In [5]:
```
print (np.mean(countryhf)-np.mean(countryaf))/((38*4+34*2)/6)
```
```
0.220776748971
```

We suspect that whether countries with bigger area may have more significant home advanages. So we plot the home-away effect throught six main leagues with order of area (small to large).

In [6]:
```
fig = plt.figure()
ax1 = fig.add_subplot(1, 1, 1)
ax1.plot([1,2,3,4,5,6], countryhf, color = 'k', label = 'home', linestyle='dashed', ma
ax1.plot([1,2,3,4,5,6], countryaf, color = 'r', label = 'away', linestyle='dashed', ma
ax1.plot([1,2,3,4,5,6], np.array(countryhf) - np.array(countryaf), color = 'b', label
ax1.hlines(np.mean(np.array(countryhf)),1,6, color = 'k')
ax1.hlines(np.mean(np.array(countryaf)),1,6, color = 'r')
ax1.hlines(np.mean(np.array(countryhf) - np.array(countryaf)),1,6, color = 'b')
plt.ylim([0, 50])
ax1.legend(loc='best')
ticks = ax1.set_xticks([1,2,3,4,5,6])
labels = ax1.set_xticklabels(['Dutch','England','Italy','Germany','Spain','France'])
ax1.set_title('Home-Away by country')
ax1.set_ylabel('total # of goals')
```
```
<matplotlib.text.Text at 0x4acbe10>
```

Out [6]:



5 Attact vs Defence

Next, we will deal with the question: which one is more important, offence or defence? We calculate the average points earned by top 5 offensive-ranked teams, which measured by "goals scored". And we calculate the average points earned by top 5 defensive-ranked teams, which measured by "goals against".

In [7]:
```
dutchatt0809 = np.mean(dutch0809.sort('of',ascending = False)['pts'][0:5])
dutchatt0910 = np.mean(dutch0910.sort('of',ascending = False)['pts'][0:5])
dutchatt1011 = np.mean(dutch1011.sort('of',ascending = False)['pts'][0:5])
dutchatt1112 = np.mean(dutch1112.sort('of',ascending = False)['pts'][0:5])
dutchatt1213 = np.mean(dutch1213.sort('of',ascending = False)['pts'][0:5])
dutchatt = [dutchatt0809,dutchatt0910,dutchatt1011,dutchatt1112,dutchatt1213]
```

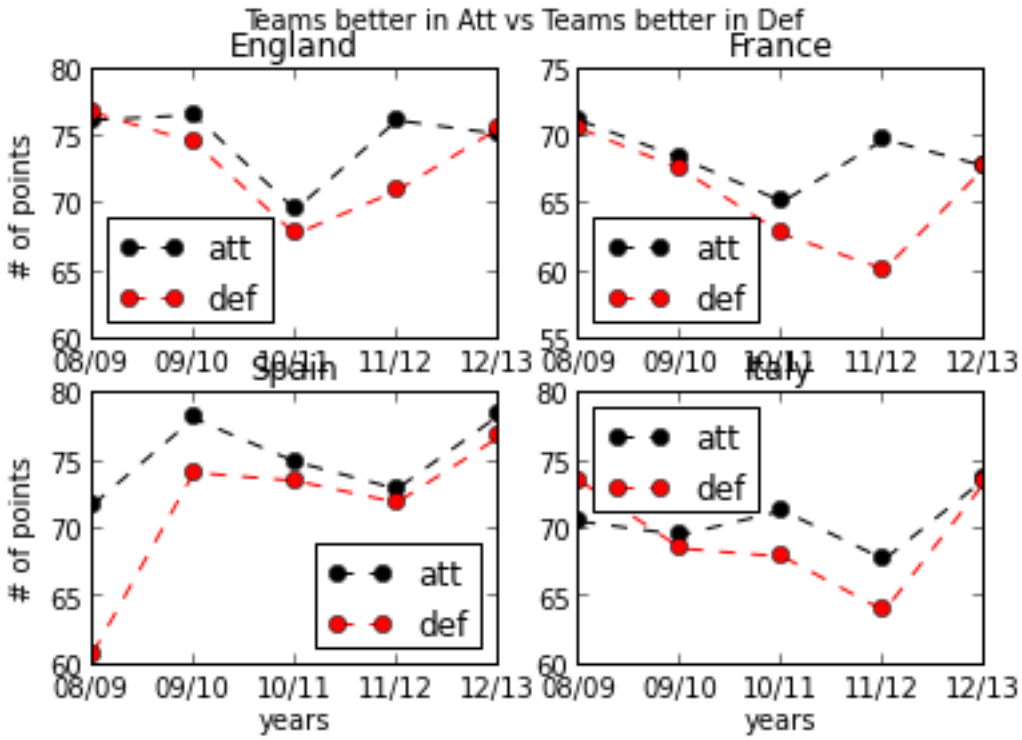
```
dutchdef0809 = np.mean(dutch0809.sort('oa')['pts'][0:5])
dutchdef0910 = np.mean(dutch0910.sort('oa')['pts'][0:5])
dutchdef1011 = np.mean(dutch1011.sort('oa')['pts'][0:5])
dutchdef1112 = np.mean(dutch1112.sort('oa')['pts'][0:5])
dutchdef1213 = np.mean(dutch1213.sort('oa')['pts'][0:5])
dutchdef = [dutchdef0809,dutchdef0910,dutchdef1011,dutchdef1112,dutchdef1213]

premieratt0809 = np.mean(premier0809.sort('of',ascending = False)['pts'][0:5])
premieratt0910 = np.mean(premier0910.sort('of',ascending = False)['pts'][0:5])
premieratt1011 = np.mean(premier1011.sort('of',ascending = False)['pts'][0:5])
premieratt1112 = np.mean(premier1112.sort('of',ascending = False)['pts'][0:5])
premieratt1213 = np.mean(premier1213.sort('of',ascending = False)['pts'][0:5])
premieratt = [premieratt0809,premieratt0910,premieratt1011,premieratt1112,premieratt12

premierdef0809 = np.mean(premier0809.sort('oa')['pts'][0:5])
premierdef0910 = np.mean(premier0910.sort('oa')['pts'][0:5])
premierdef1011 = np.mean(premier1011.sort('oa')['pts'][0:5])
premierdef1112 = np.mean(premier1112.sort('oa')['pts'][0:5])
premierdef1213 = np.mean(premier1213.sort('oa')['pts'][0:5])
premierdef = [premierdef0809,premierdef0910,premierdef1011,premierdef1112,premierdef12


serieAatt0809 = np.mean(serieA0809.sort('of',ascending = False)['pts'][0:5])
serieAatt0910 = np.mean(serieA0910.sort('of',ascending = False)['pts'][0:5])
serieAatt1011 = np.mean(serieA1011.sort('of',ascending = False)['pts'][0:5])
serieAatt1112 = np.mean(serieA1112.sort('of',ascending = False)['pts'][0:5])
serieAatt1213 = np.mean(serieA1213.sort('of',ascending = False)['pts'][0:5])
serieAatt = [serieAatt0809,serieAatt0910,serieAatt1011,serieAatt1112,serieAatt1213]

serieAdef0809 = np.mean(serieA0809.sort('oa')['pts'][0:5])
serieAdef0910 = np.mean(serieA0910.sort('oa')['pts'][0:5])
serieAdef1011 = np.mean(serieA1011.sort('oa')['pts'][0:5])
serieAdef1112 = np.mean(serieA1112.sort('oa')['pts'][0:5])
serieAdef1213 = np.mean(serieA1213.sort('oa')['pts'][0:5])
serieAdef = [serieAdef0809,serieAdef0910,serieAdef1011,serieAdef1112,serieAdef1213]


bundesligaatt0809 = np.mean(bundesliga0809.sort('of',ascending = False)['pts'][0:5])
bundesligaatt0910 = np.mean(bundesliga0910.sort('of',ascending = False)['pts'][0:5])
bundesligaatt1011 = np.mean(bundesliga1011.sort('of',ascending = False)['pts'][0:5])
bundesligaatt1112 = np.mean(bundesliga1112.sort('of',ascending = False)['pts'][0:5])
bundesligaatt1213 = np.mean(bundesliga1213.sort('of',ascending = False)['pts'][0:5])
bundesligaatt = [bundesligaatt0809,bundesligaatt0910,bundesligaatt1011,bundesligaatt11

bundesligadef0809 = np.mean(bundesliga0809.sort('oa')['pts'][0:5])
bundesligadef0910 = np.mean(bundesliga0910.sort('oa')['pts'][0:5])
bundesligadef1011 = np.mean(bundesliga1011.sort('oa')['pts'][0:5])
bundesligadef1112 = np.mean(bundesliga1112.sort('oa')['pts'][0:5])
bundesligadef1213 = np.mean(bundesliga1213.sort('oa')['pts'][0:5])
bundesligadef = [bundesligadef0809,bundesligadef0910,bundesligadef1011,bundesligadef11


laligaatt0809 = np.mean(laliga0809.sort('of',ascending = False)['pts'][0:5])
laligaatt0910 = np.mean(laliga0910.sort('of',ascending = False)['pts'][0:5])
laligaatt1011 = np.mean(laliga1011.sort('of',ascending = False)['pts'][0:5])
laligaatt1112 = np.mean(laliga1112.sort('of',ascending = False)['pts'][0:5])
laligaatt1213 = np.mean(laliga1213.sort('of',ascending = False)['pts'][0:5])
laligaatt = [laligaatt0809,laligaatt0910,laligaatt1011,laligaatt1112,laligaatt1213]

laligadef0809 = np.mean(laliga0809.sort('oa')['pts'][0:5])
laligadef0910 = np.mean(laliga0910.sort('oa')['pts'][0:5])
laligadef1011 = np.mean(laliga1011.sort('oa')['pts'][0:5])
laligadef1112 = np.mean(laliga1112.sort('oa')['pts'][0:5])
laligadef1213 = np.mean(laliga1213.sort('oa')['pts'][0:5])
laligadef = [laligadef0809,laligadef0910,laligadef1011,laligadef1112,laligadef1213]
```

```
frenchatt0809 = np.mean(french0809.sort('of',ascending = False)['pts'][0:5])
frenchatt0910 = np.mean(french0910.sort('of',ascending = False)['pts'][0:5])
frenchatt1011 = np.mean(french1011.sort('of',ascending = False)['pts'][0:5])
frenchatt1112 = np.mean(french1112.sort('of',ascending = False)['pts'][0:5])
frenchatt1213 = np.mean(french1213.sort('of',ascending = False)['pts'][0:5])
frenchatt = [frenchatt0809,frenchatt0910,frenchatt1011,frenchatt1112,frenchatt1213]

frenchdef0809 = np.mean(french0809.sort('oa')['pts'][0:5])
frenchdef0910 = np.mean(french0910.sort('oa')['pts'][0:5])
frenchdef1011 = np.mean(french1011.sort('oa')['pts'][0:5])
frenchdef1112 = np.mean(french1112.sort('oa')['pts'][0:5])
frenchdef1213 = np.mean(french1213.sort('oa')['pts'][0:5])
frenchdef = [frenchdef0809,frenchdef0910,frenchdef1011,frenchdef1112,frenchdef1213]
```

Plot att vs def for each league

In [8]:
```
year = [9,10,11,12,13]
fig = plt.figure()

ax1 = fig.add_subplot(2, 2, 1)
ax1.plot(year, premieratt, color = 'k', label = 'att', linestyle='dashed', marker='o')
ax1.plot(year, premierdef, color = 'r', label = 'def', linestyle='dashed', marker='o')
plt.ylim([60, 80])
ax1.legend(loc='best')
ticks = ax1.set_xticks([9, 10, 11, 12, 13])
labels = ax1.set_xticklabels(['08/09', '09/10', '10/11', '11/12', '12/13'])
ax1.set_title('England')
ax1.set_ylabel('# of points')

ax2 = fig.add_subplot(2, 2, 2)
ax2.plot(year, frenchatt, color = 'k', label = 'att', linestyle='dashed', marker='o')
ax2.plot(year, frenchdef, color = 'r', label = 'def', linestyle='dashed', marker='o')
plt.ylim([55, 75])
ax2.legend(loc='best')
ticks = ax2.set_xticks([9, 10, 11, 12, 13])
labels = ax2.set_xticklabels(['08/09', '09/10', '10/11', '11/12', '12/13'])
ax2.set_title('France')


ax3 = fig.add_subplot(2, 2, 3)
ax3.plot(year, laligaatt, color = 'k', label = 'att', linestyle='dashed', marker='o')
ax3.plot(year, laligadef, color = 'r', label = 'def', linestyle='dashed', marker='o')
plt.ylim([60, 80])
ax3.legend(loc='best')
ticks = ax3.set_xticks([9, 10, 11, 12, 13])
labels = ax3.set_xticklabels(['08/09', '09/10', '10/11', '11/12', '12/13'])
ax3.set_title('Spain')
ax3.set_xlabel('years')
ax3.set_ylabel('# of points')

ax4 = fig.add_subplot(2, 2, 4)
ax4.plot(year, serieAatt, color = 'k', label = 'att', linestyle='dashed', marker='o')
ax4.plot(year, serieAdef, color = 'r', label = 'def', linestyle='dashed', marker='o')
plt.ylim([60, 80])
ax4.legend(loc='best')
ticks = ax4.set_xticks([9, 10, 11, 12, 13])
labels = ax4.set_xticklabels(['08/09', '09/10', '10/11', '11/12', '12/13'])
ax4.set_title('Italy')
ax4.set_xlabel('years')

plt.suptitle('Teams better in Att vs Teams better in Def')
```

Out [8]:
```
<matplotlib.text.Text at 0x5a13ad0>
```



From the plots, we can say that attack is slightly more importent than defance because scoring more goals gives the team more points than improving defance to avoid being scored.Also, we are interesting in whether attack is more inportent than defence in UEFA.

In [9]:

```
#read data from html source code from UEFA : http://www.uefa.com/index.html
#in this website, we can get data for UEFA champions league (the highest level tournam
#we download club statistics from the 03/04 to the year 12/13

#the html source code for different year has different patterns, we have to read data
filename = 'uefa0304.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Monaco')
data3 = data2[3].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
#define number of games played by each club
numofgames = [13,13,12,12,10,10,10,10,8,8,8,8,8,8,8,8,6,6,6,6,6,6,6,6,6,6,6,6,6,6]
#take average for each entry in the data frame
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
#include national information of each club
data8['na'] = ['o','o','e','s','e','o','i','s','g','o','i','o','e','s','o','g','o','o'
uefa0304 = data8

filename = 'uefa0405.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Liverpool')
data3 = data2[13].split('table')
```

```
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['e','i','e','o','g','i','i','o','e','s','g','g','e','o','o','s','o','o'
uefa0405 = data8

filename = 'uefa0506.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Arsenal')
data3 = data2[3].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['e','s','i','s','o','i','i','o','o','g','g','e','e','o','o','s','o','s'
uefa0506 = data8

filename = 'uefa0607.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Liverpool')
data3 = data2[13].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['e','i','e','e','g','o','i','s','e','s','o','i','o','o','o','s','o','o'
uefa0607 = data8


filename = 'uefa0708.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Chelsea')
data3 = data2[5].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['e','e','s','e','e','o','i','g','o','i','o','i','o','o','s','s','o','o'
uefa0708 = data8

filename = 'uefa0809.txt'
txt = open(filename)
```

```
data1 = txt.read()
data2 = data1.split('Barcelona')
data3 = data2[11].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['s','e','e','e','g','e','o','s','s','i','i','o','o','s','i','o','o','o'
uefa0809 = data8

filename = 'uefa0910.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Bayern')
data3 = data2[13].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['g','i','s','o','e','o','o','e','e','i','i','o','o','s','s','g','o','s'
uefa0910 = data8

filename = 'uefa1011.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Barcelona')
data3 = data2[11].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['s','e','s','g','e','i','o','e','e','g','o','o','o','i','i','s','o','o'
uefa1011 = data8

filename = 'uefa1112.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Bayern')
data3 = data2[13].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['g','e','s','s','o','o','o','i','e','o','o','i','g','o','i','o','o','o'
uefa1112 = data8
```

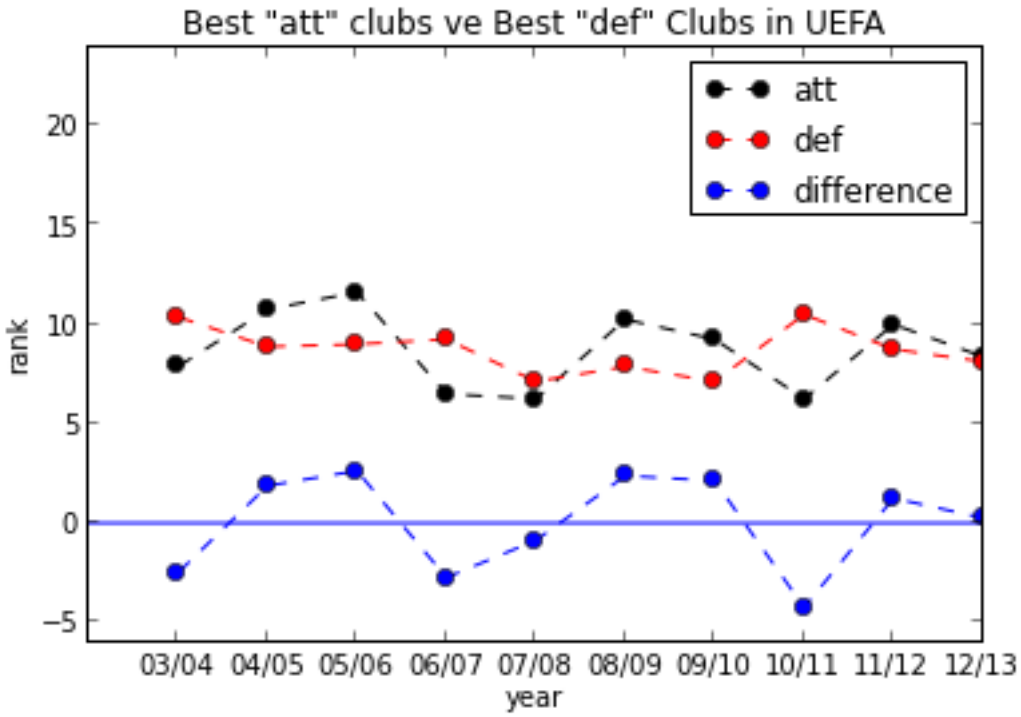
```
filename = 'uefa1213.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Bayern')
data3 = data2[13].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7 = np.array(data6).reshape(32,11)
data8 = DataFrame(data7,columns =['gs','ga','yc','rc','on','off','of','co','fc','posst
for i in range(len(data8.columns)-1):
data8.ix[:,i] = data8.ix[:,i]/np.array(numofgames)
data8['gp'] = numofgames
data8['na'] = ['g','g','s','s','o','i','s','o','e','o','e','i','o','g','o','s','o','o'
uefa1213 = data8
```

In [10]:

```
#compute the different performances by the top 8 offensive teams and top 8 defensive t
years = [uefa0304,uefa0405,uefa0506,uefa0607,uefa0708,uefa0809,uefa0910,uefa1011,uefa1
uefaatt = [np.mean(years[i].sort('gs',ascending = False)[0:8].index) for i in range(le
uefadef = [np.mean(years[i].sort('ga')[0:8].index) for i in range(len(years))]

#plot the att and def for UEFA
fig = plt.figure()
ax1 = fig.add_subplot(1, 1, 1)
ax1.plot([1,2,3,4,5,6,7,8,9,10], uefaatt, color = 'k', label = 'att', linestyle='dashe
ax1.plot([1,2,3,4,5,6,7,8,9,10], uefadef, color = 'r', label = 'def', linestyle='dashe
ax1.plot([1,2,3,4,5,6,7,8,9,10], np.array(uefaatt)-np.array(uefadef), color = 'b', lab
ax1.hlines(0,0,10, color = 'b')
plt.ylim([-6, 24])
ax1.legend(loc='best')
ticks = ax1.set_xticks([1,2,3,4,5,6,7,8,9,10])
labels = ax1.set_xticklabels(['03/04','04/05','05/06','06/07','07/08','08/09','09/10',
ax1.set_title('Best "att" clubs ve Best "def" Clubs in UEFA')
ax1.set_xlabel('year')
ax1.set_ylabel('rank')
```

```
<matplotlib.text.Text at 0x54758d0>
```

Out [10]:



The plot shows that in UEFA, there is not significant difference of preference between attack and defence. They are equal.

6 Clubs performance

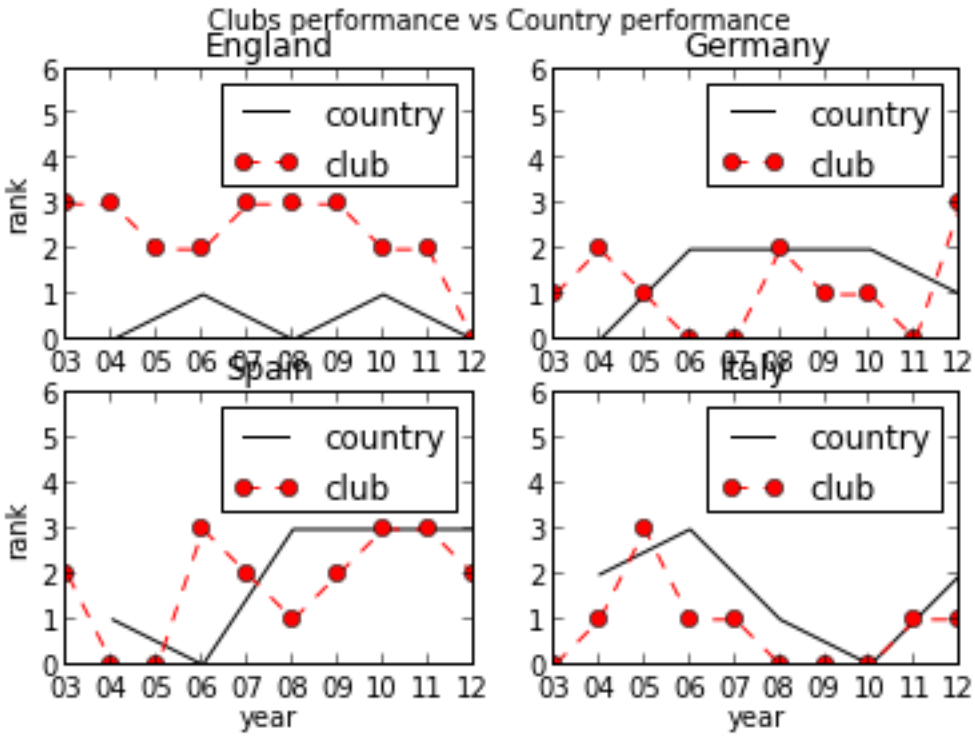
Next, we are going to check whether the clubs' performence in UEFA stage affects the countries' performance in international tounament stage. So we compute the rank for clubs in UEFA for resent 10 years for England, Germany, Italy and Spain, and compare the ranks to the countries' performance in last 10 years to see whether they have some kind of relationships.

In [11]:
```
#compute the performance for each country
#method: take goal difference for each club, then compute the average
#then give the rank for England, Italy, Germany and Spain by club performance
#then compare to the real rank in World Cup and European Cup
g0304 = np.mean(uefa0304[uefa0304['na']=='g']['gs']) - np.mean(uefa0304[uefa0304['na']
g0405 = np.mean(uefa0405[uefa0405['na']=='g']['gs']) - np.mean(uefa0405[uefa0405['na']
g0506 = np.mean(uefa0506[uefa0506['na']=='g']['gs']) - np.mean(uefa0506[uefa0506['na']
g0607 = np.mean(uefa0607[uefa0607['na']=='g']['gs']) - np.mean(uefa0607[uefa0607['na']
g0708 = np.mean(uefa0708[uefa0708['na']=='g']['gs']) - np.mean(uefa0708[uefa0708['na']
g0809 = np.mean(uefa0809[uefa0809['na']=='g']['gs']) - np.mean(uefa0809[uefa0809['na']
g0910 = np.mean(uefa0910[uefa0910['na']=='g']['gs']) - np.mean(uefa0910[uefa0910['na']
g1011 = np.mean(uefa1011[uefa1011['na']=='g']['gs']) - np.mean(uefa1011[uefa1011['na']
g1112 = np.mean(uefa1112[uefa1112['na']=='g']['gs']) - np.mean(uefa1112[uefa1112['na']
g1213 = np.mean(uefa1213[uefa1213['na']=='g']['gs']) - np.mean(uefa1213[uefa1213['na']
uefag = [g0304,g0405,g0506,g0607,g0708,g0809,g0910,g1011,g1112,g1213]

e0304 = np.mean(uefa0304[uefa0304['na']=='e']['gs']) - np.mean(uefa0304[uefa0304['na']
e0405 = np.mean(uefa0405[uefa0405['na']=='e']['gs']) - np.mean(uefa0405[uefa0405['na']
e0506 = np.mean(uefa0506[uefa0506['na']=='e']['gs']) - np.mean(uefa0506[uefa0506['na']
e0607 = np.mean(uefa0607[uefa0607['na']=='e']['gs']) - np.mean(uefa0607[uefa0607['na']
e0708 = np.mean(uefa0708[uefa0708['na']=='e']['gs']) - np.mean(uefa0708[uefa0708['na']
e0809 = np.mean(uefa0809[uefa0809['na']=='e']['gs']) - np.mean(uefa0809[uefa0809['na']
e0910 = np.mean(uefa0910[uefa0910['na']=='e']['gs']) - np.mean(uefa0910[uefa0910['na']
e1011 = np.mean(uefa1011[uefa1011['na']=='e']['gs']) - np.mean(uefa1011[uefa1011['na']
e1112 = np.mean(uefa1112[uefa1112['na']=='e']['gs']) - np.mean(uefa1112[uefa1112['na']
e1213 = np.mean(uefa1213[uefa1213['na']=='e']['gs']) - np.mean(uefa1213[uefa1213['na']
uefae = [e0304,e0405,e0506,e0607,e0708,e0809,e0910,e1011,e1112,e1213]

s0304 = np.mean(uefa0304[uefa0304['na']=='s']['gs']) - np.mean(uefa0304[uefa0304['na']
s0405 = np.mean(uefa0405[uefa0405['na']=='s']['gs']) - np.mean(uefa0405[uefa0405['na']
s0506 = np.mean(uefa0506[uefa0506['na']=='s']['gs']) - np.mean(uefa0506[uefa0506['na']
s0607 = np.mean(uefa0607[uefa0607['na']=='s']['gs']) - np.mean(uefa0607[uefa0607['na']
s0708 = np.mean(uefa0708[uefa0708['na']=='s']['gs']) - np.mean(uefa0708[uefa0708['na']
s0809 = np.mean(uefa0809[uefa0809['na']=='s']['gs']) - np.mean(uefa0809[uefa0809['na']
s0910 = np.mean(uefa0910[uefa0910['na']=='s']['gs']) - np.mean(uefa0910[uefa0910['na']
s1011 = np.mean(uefa1011[uefa1011['na']=='s']['gs']) - np.mean(uefa1011[uefa1011['na']
s1112 = np.mean(uefa1112[uefa1112['na']=='s']['gs']) - np.mean(uefa1112[uefa1112['na']
s1213 = np.mean(uefa1213[uefa1213['na']=='s']['gs']) - np.mean(uefa1213[uefa1213['na']
uefas = [s0304,s0405,s0506,s0607,s0708,s0809,s0910,s1011,s1112,s1213]

i0304 = np.mean(uefa0304[uefa0304['na']=='i']['gs']) - np.mean(uefa0304[uefa0304['na']
i0405 = np.mean(uefa0405[uefa0405['na']=='i']['gs']) - np.mean(uefa0405[uefa0405['na']
i0506 = np.mean(uefa0506[uefa0506['na']=='i']['gs']) - np.mean(uefa0506[uefa0506['na']
i0607 = np.mean(uefa0607[uefa0607['na']=='i']['gs']) - np.mean(uefa0607[uefa0607['na']
i0708 = np.mean(uefa0708[uefa0708['na']=='i']['gs']) - np.mean(uefa0708[uefa0708['na']
i0809 = np.mean(uefa0809[uefa0809['na']=='i']['gs']) - np.mean(uefa0809[uefa0809['na']
i0910 = np.mean(uefa0910[uefa0910['na']=='i']['gs']) - np.mean(uefa0910[uefa0910['na']
i1011 = np.mean(uefa1011[uefa1011['na']=='i']['gs']) - np.mean(uefa1011[uefa1011['na']
i1112 = np.mean(uefa1112[uefa1112['na']=='i']['gs']) - np.mean(uefa1112[uefa1112['na']
i1213 = np.mean(uefa1213[uefa1213['na']=='i']['gs']) - np.mean(uefa1213[uefa1213['na']
uefai = [i0304,i0405,i0506,i0607,i0708,i0809,i0910,i1011,i1112,i1213]

#define index for different years
```

```
ind = ['0304','0405','0506','0607','0708','0809','0910','1011','1112','1213']
uefaclub = DataFrame(np.array([uefae,uefai,uefag,uefas]).reshape(4,10),columns = ind,i

uefaer = []
for i in ind:
uefaer.append(uefaclub.sort(i).transpose().columns.get_loc('e'))

uefair = []
for i in ind:
uefair.append(uefaclub.sort(i).transpose().columns.get_loc('i'))

uefagr = []
for i in ind:
uefagr.append(uefaclub.sort(i).transpose().columns.get_loc('g'))

uefasr = []
for i in ind:
uefasr.append(uefaclub.sort(i).transpose().columns.get_loc('s'))


#rank of country performances from FIFA.com
fifaer = [0,1,0,1,0]
fifair = [2,3,1,0,2]
fifagr = [0,2,2,2,1]
fifasr = [1,0,3,3,3]

#plot rank of club vs rank of country
fig = plt.figure()

ax1 = fig.add_subplot(2, 2, 1)
ax1.plot([2,4,6,8,10], fifaer, color = 'k', label = 'country')
ax1.plot([1,2,3,4,5,6,7,8,9,10], uefaer, color = 'r', label = 'club', linestyle='dashe
plt.ylim([0, 6])
ax1.legend(loc='best')
labels = ax1.set_xticklabels(['03','04','05','06','07','08','09','10','11','12','13'])
ax1.set_title('England')
ax1.set_ylabel('rank')

ax2 = fig.add_subplot(2, 2, 2)
ax2.plot([2,4,6,8,10], fifagr, color = 'k', label = 'country' )
ax2.plot([1,2,3,4,5,6,7,8,9,10], uefagr, color = 'r', label = 'club', linestyle='dashe
plt.ylim([0, 6])
ax2.legend(loc='best')
labels = ax2.set_xticklabels(['03','04','05','06','07','08','09','10','11','12','13'])
ax2.set_title('Germany')

ax3 = fig.add_subplot(2, 2, 3)
ax3.plot([2,4,6,8,10], fifasr, color = 'k', label = 'country')
ax3.plot([1,2,3,4,5,6,7,8,9,10], uefasr, color = 'r', label = 'club', linestyle='dashe
plt.ylim([0, 6])
ax3.legend(loc='best')
labels = ax3.set_xticklabels(['03','04','05','06','07','08','09','10','11','12','13'])
ax3.set_title('Spain')
ax3.set_ylabel('rank')
ax3.set_xlabel('year')

ax4 = fig.add_subplot(2, 2, 4)
ax4.plot([2,4,6,8,10], fifair, color = 'k', label = 'country')
ax4.plot([1,2,3,4,5,6,7,8,9,10], uefair, color = 'r', label = 'club', linestyle='dashe
plt.ylim([0, 6])
ax4.legend(loc='best')
labels = ax4.set_xticklabels(['03','04','05','06','07','08','09','10','11','12','13'])
ax4.set_title('Italy')
ax4.set_xlabel('year')

plt.suptitle('Clubs performance vs Country performance')
```

```
<matplotlib.text.Text at 0x639f3d0>
```

Out [11]:



We can see from the plots that the clubs' performance and countries' performance are not relevent at all. So this factor will not be put into our analysis for predicting the World Cup 2014.

7 Regression model for prediction number of goals scored

By our assumption, the number of goals scored in a single game follows a Poisson distribution with parameter λ. Firstly, we want to estimate $\hat{\lambda}$ by MLE, which is just the mean of goals scored in a series of games. Then we run OLS for $\hat{\lambda}$ as dependent variable and other game statistics as the design matrix. Here, our traing data is from UEFA data sets through resent 10 years. The model will be applied in our predicting process for World Cup 2014.

In [12]:

```
#merge different dataframes to a single one
uefa0304arr = np.array(uefa0304).transpose().reshape(1,416)
uefa0405arr = np.array(uefa0405).transpose().reshape(1,416)
uefa0506arr = np.array(uefa0506).transpose().reshape(1,416)
uefa0607arr = np.array(uefa0607).transpose().reshape(1,416)
uefa0708arr = np.array(uefa0708).transpose().reshape(1,416)
uefa0809arr = np.array(uefa0809).transpose().reshape(1,416)
uefa0910arr = np.array(uefa0910).transpose().reshape(1,416)
uefa1011arr = np.array(uefa1011).transpose().reshape(1,416)
uefa1112arr = np.array(uefa1112).transpose().reshape(1,416)
uefa1213arr = np.array(uefa1213).transpose().reshape(1,416)

a = np.concatenate([uefa0405arr,uefa0506arr,uefa0607arr,uefa0708arr,uefa0809arr,uefa09
data = DataFrame(a[:,0:32].reshape(288,1),columns = ['gs'])
data['ga'] = a[:,32:64].reshape(288,1)
data['yc'] = a[:,64:96].reshape(288,1)
data['rc'] = a[:,96:128].reshape(288,1)
data['on'] = a[:,128:160].reshape(288,1)
data['off'] = a[:,160:192].reshape(288,1)
data['of'] = a[:,192:224].reshape(288,1)
data['co'] = a[:,224:256].reshape(288,1)
```

```
data['fc'] = a[:,256:288].reshape(288,1)
data['posst'] = a[:,288:320].reshape(288,1)
data['poss'] = a[:,320:352].reshape(288,1)
uefa = DataFrame(data,dtype = 'float')
uefa['const'] = np.ones(288)
```

In [13]:

```
#attact model
#a = np.log(uefa.gs)
a = uefa.gs
mod1 = sm.OLS(a, uefa.ix[:,[4,6,7,8,10]]).fit()
# Inspect the results
print mod1.summary()
predictions1 = mod1.predict(uefa.ix[:,[4,6,7,8,10]])
```

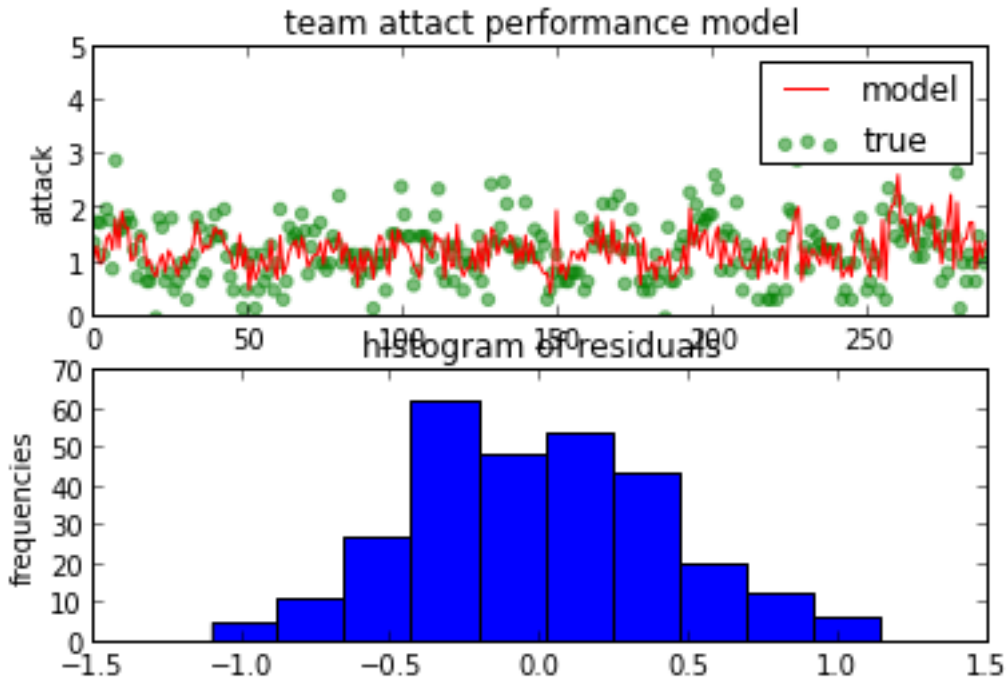
```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                      gs   R-squared:
0.902
Model:                             OLS   Adj. R-squared:
0.900
Method:                  Least Squares   F-statistic:
522.0
Date:                 Mon, 14 Apr 2014   Prob (F-statistic):
1.56e-140
Time:                         08:33:09   Log-Likelihood:
-161.04
No. Observations:                 288   AIC:
332.1
Df Residuals:                     283   BIC:
350.4
Df Model:                           5
==============================================================================
========
                 coef    std err          t      P>|t|      [95.0%
Conf. Int.]
------------------------------------------------------------------------------
--------
on             0.2292      0.019     11.871      0.000          0.191
0.267
of             0.0422      0.024      1.755      0.080         -0.005
0.090
co            -0.0641      0.023     -2.752      0.006         -0.110
-0.018
fc            -0.0124      0.008     -1.629      0.104         -0.027
0.003
poss           0.0087      0.004      2.349      0.020          0.001
0.016
==============================================================================
========
Omnibus:                        2.207   Durbin-Watson:
1.495
Prob(Omnibus):                  0.332   Jarque-Bera (JB):
2.264
Skew:                           0.182   Prob(JB):
0.322
```

```
Kurtosis:                       2.762   Cond. No.
53.4
==============================================================================
========
```

In [14]:
```
#plot the attact model
fig = plt.figure()
ax1 = fig.add_subplot(2, 1, 1)
ax1.plot(predictions1, color = 'r', label = 'model')
ax1.scatter(range(288),a,color = 'g',label = 'true',alpha=0.5)
plt.ylim([0, 5])
plt.xlim([0, 288])
ax1.legend(loc='best')
ax1.set_ylabel('attack')
ax1.set_title('team attact performance model')
ax2 = fig.add_subplot(2, 1, 2)
ax2.hist(a-predictions1)
ax2.set_ylabel('frequencies')
ax2.set_title('histogram of residuals')
```

```
<matplotlib.text.Text at 0x6811090>
```

Out [14]:



Attack model is good since the R^2 is greater than 0.9 and the residuals are bell-shaped.

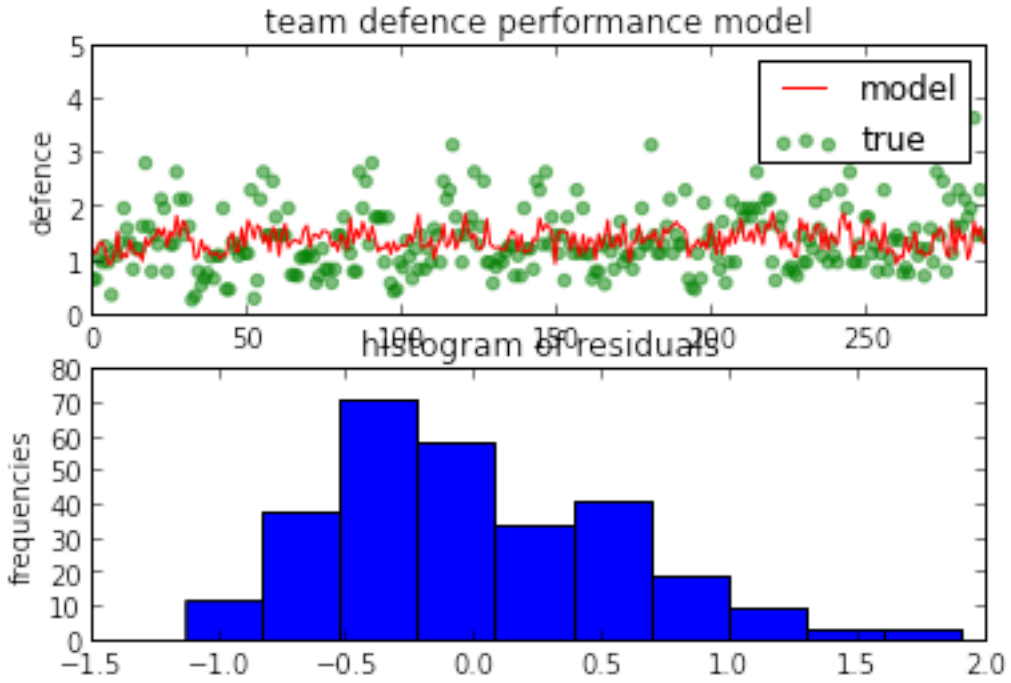
In [15]:
```
#def model
d = uefa.ga
mod2 = sm.OLS(d, uefa.ix[:,[3,4,6,8,10,11]]).fit()
predictions2 = mod2.predict(uefa.ix[:,[3,4,6,8,10,11]])
```

In [16]:
```
#plot defence model
fig = plt.figure()
fig.patch.set_alpha(0.5)
ax1 = fig.add_subplot(2, 1, 1)
ax1.plot(predictions2, color = 'r', label = 'model')
ax1.scatter(range(288),d,color = 'g',label = 'true',alpha=0.5)
plt.ylim([0, 5])
plt.xlim([0, 288])
ax1.legend(loc='best')
ax1.set_ylabel('defence')
```

```
ax1.set_title('team defence performance model')
ax2 = fig.add_subplot(2, 1, 2)
ax2.hist(d-predictions2)
ax2.set_ylabel('frequencies')
ax2.set_title('histogram of residuals')
```

```
<matplotlib.text.Text at 0x6aca990>
```

Out [16]:



Defence model is bad actually because the R^2 is less than 0.2, and the redisuals are skewed. I think the reason is that our game statistics data are only relevent to the attacking performance. The data doesn't contain much more information for defencing performance. That is why our model cannot explain most of the variation.

8 Prediction of World Cup 2014

We need to load data for each team.

In [17]:

```
#load copa america 2011 data by team
#argentina
filename = 'argentina.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Passes')
data3 = data2[5].split('Possession')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7= np.array(data6,dtype = 'float')
argentina = data7

#brazil
filename = 'brazil.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Passes')
```

```
data3 = data2[5].split('Possession')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7= np.array(data6,dtype = 'float')
brazil = data7

#chile
filename = 'chile.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Passes')
data3 = data2[5].split('Possession')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7= np.array(data6,dtype = 'float')
chile = data7

#colombia
filename = 'colombia.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Passes')
data3 = data2[5].split('Possession')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7= np.array(data6,dtype = 'float')
colombia = data7

#costa rica
filename = 'costa.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Passes')
data3 = data2[5].split('Possession')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7= np.array(data6,dtype = 'float')
costa = data7

#ecuador
filename = 'ecuador.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Passes')
data3 = data2[5].split('Possession')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7= np.array(data6,dtype = 'float')
ecuador = data7

#mexico
filename = 'mexico.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Passes')
data3 = data2[5].split('Possession')
```

```
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7= np.array(data6,dtype = 'float')
mexico = data7

#uruguay
filename = 'uruguay.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Passes')
data3 = data2[5].split('Possession')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
data7= np.array(data6,dtype = 'float')
uruguay = data7
```

In [18]:

```
#get world cup 2010 data
#on target
filename = 'worldcup2010shot.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Spain')
data3 = data2[6].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
wolrdcupshot = np.array(data6, dtype = 'float').reshape(32,10)
countrynames = ['spa','uru','ger','gha','arg','net','bra','usa','eng','chi','par','kor
worldontarget = pd.Series(wolrdcupshot[:,2]/wolrdcupshot[:,0],index = countrynames)

#offside
filename = 'worldcup2010att.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Spain')
data3 = data2[6].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
wolrdcupatt = np.array(data6, dtype = 'float').reshape(32,11)
worldoffside = pd.Series(wolrdcupatt[:,7]/wolrdcupatt[:,0],index = countrynames)

#red cards and fouls
filename = 'worldcup2010dis.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Netherlands')
data3 = data2[6].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
wolrdcupdis = np.array(data6, dtype = 'float').reshape(32,7)
worldred = pd.Series(wolrdcupdis[:,2]/wolrdcupatt[:,0],index = countrynames)
worldfoul = pd.Series(wolrdcupdis[:,3]/wolrdcupatt[:,0],index = countrynames)

#corners
filename = 'worldcup2010cor.txt'
txt = open(filename)
data1 = txt.read()
```

```
data2 = data1.split('Spain')
data3 = data2[6].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
wolrdcupcor = np.array(data6, dtype = 'float').reshape(32,10)
worldcor = pd.Series(wolrdcupcor[:,7]/wolrdcupatt[:,0],index = countrynames)

#number of games played
worldgames = pd.Series(wolrdcupshot[:,0],index = countrynames)
```

In [19]:
```
#get EURO 2012 Cup data
filename = 'EURO2012.txt'
txt = open(filename)
data1 = txt.read()
data2 = data1.split('Italy')
data3 = data2[3].split('table')
regex = re.compile('>[0-9]+')
data4 = regex.findall(data3[0])
data5 = ' '+' '.join(data4)
data6 = data5.split(' >')[1:]
numofgames = np.array([6,6,5,5,4,4,4,4,3,3,3,3,3,3,3,3],dtype = 'float')
euronames = ['ita','spa','ger','por','cze','eng','fra','gre','cro','den','net','pol','
euro = np.array(data6, dtype = 'float').reshape(16,8)

#red cards
eurored = pd.Series(euro[:,3]/numofgames,index = euronames)

#on target
euroontarget = pd.Series(euro[:,4]/numofgames,index = euronames)

#offsides
eurooffside = pd.Series(euro[:,5]/numofgames,index = euronames)

#corners
eurocor = pd.Series(euro[:,6]/numofgames,index = euronames)

#fouls
eurofoul = pd.Series(euro[:,7]/numofgames,index = euronames)

#number of games
eurogames = pd.Series(numofgames,index = euronames)
```

We start our prediction.

In [20]:
```
#prediction start
#group A
#Brazil
pred1 = mod1.predict(np.array([worldontarget['bra'],worldoffside['bra'],worldcor['bra'
pred2 = mod2.predict(np.array([worldred['bra'],worldontarget['bra'],worldoffside['bra'
predbra = ((pred1[0]-pred2[0])*worldgames['bra'] + (pred1[1]-pred2[1]))/ (worldgames['
#Cameroon
pred1 = mod1.predict(np.array([worldontarget['cam'],worldoffside['cam'],worldcor['cam'
pred2 = mod2.predict(np.array([worldred['cam'],worldontarget['cam'],worldoffside['cam'
predcam = pred1-pred2
#Mexico
pred1 = mod1.predict(np.array([worldontarget['mex'],worldoffside['mex'],worldcor['mex'
pred2 = mod2.predict(np.array([worldred['mex'],worldontarget['mex'],worldoffside['mex'
predmex = ((pred1[0]-pred2[0])*worldgames['mex'] + (pred1[1]-pred2[1]))/ (worldgames['
#Croatia
pred1 = mod1.predict(np.array([euroontarget['cro'],eurooffside['cro'],eurocor['cro'],e
pred2 = mod2.predict(np.array([eurored['cro'],euroontarget['cro'],eurooffside['cro'],e
predcro = pred1-pred2

#group B
```

```
#Spain
pred1 = mod1.predict(np.array([worldontarget['spa'],worldoffside['spa'],worldcor['spa'
pred2 = mod2.predict(np.array([worldred['spa'],worldontarget['spa'],worldoffside['spa'
predspa = ((pred1[0]-pred2[0])*worldgames['spa'] + (pred1[1]-pred2[1])*eurogames['spa'
#Chile
pred1 = mod1.predict(np.array([worldontarget['chi'],worldoffside['chi'],worldcor['chi'
pred2 = mod2.predict(np.array([worldred['chi'],worldontarget['chi'],worldoffside['chi'
predchi = ((pred1[0]-pred2[0])*worldgames['chi'] + (pred1[1]-pred2[1]))/ (worldgames['
#Australia
pred1 = mod1.predict(np.array([worldontarget['aus'],worldoffside['aus'],worldcor['aus'
pred2 = mod2.predict(np.array([worldred['aus'],worldontarget['aus'],worldoffside['aus'
predaus = pred1-pred2
#Dutch
pred1 = mod1.predict(np.array([worldontarget['net'],worldoffside['net'],worldcor['net'
pred2 = mod2.predict(np.array([worldred['net'],worldontarget['net'],worldoffside['net'
prednet = ((pred1[0]-pred2[0])*worldgames['net'] + (pred1[1]-pred2[1])*eurogames['net'

#group C
#Colombia
pred1 = mod1.predict(np.array([colombia[2],colombia[6],20,colombia[3],50],dtype = 'flo
pred2 = mod2.predict(np.array([3,colombia[2],colombia[6],colombia[3],50,1],dtype = 'fl
predcol = (pred1-pred2)/5

#Cote d'Ivoire
pred1 = mod1.predict(np.array([worldontarget['cot'],worldoffside['cot'],worldcor['cot'
pred2 = mod2.predict(np.array([worldred['cot'],worldontarget['cot'],worldoffside['cot'
predcot = pred1-pred2
#Japan
pred1 = mod1.predict(np.array([worldontarget['jap'],worldoffside['jap'],worldcor['jap'
pred2 = mod2.predict(np.array([worldred['jap'],worldontarget['jap'],worldoffside['jap'
predjap = pred1-pred2
#Greece
pred1 = mod1.predict(np.array([worldontarget['gre'],worldoffside['gre'],worldcor['gre'
pred2 = mod2.predict(np.array([worldred['gre'],worldontarget['gre'],worldoffside['gre'
predgre = ((pred1[0]-pred2[0])*worldgames['gre'] + (pred1[1]-pred2[1])*eurogames['gre'

#group D
#Uruguay
pred1 = mod1.predict(np.array([worldontarget['uru'],worldoffside['uru'],worldcor['uru'
pred2 = mod2.predict(np.array([worldred['uru'],worldontarget['uru'],worldoffside['uru'
preduru = ((pred1[0]-pred2[0])*worldgames['uru'] + (pred1[1]-pred2[1]))/ (worldgames['
#England
pred1 = mod1.predict(np.array([worldontarget['eng'],worldoffside['eng'],worldcor['eng'
pred2 = mod2.predict(np.array([worldred['eng'],worldontarget['eng'],worldoffside['eng'
predeng = ((pred1[0]-pred2[0])*worldgames['eng'] + (pred1[1]-pred2[1])*eurogames['eng'
#Costa Rica
pred1 = mod1.predict(np.array([costa[2],costa[6],15,costa[3],50],dtype = 'float').resh
pred2 = mod2.predict(np.array([0,costa[2],costa[6],costa[3],50,1],dtype = 'float').res
predcos = (pred1-pred2)/3
#Italy
pred1 = mod1.predict(np.array([worldontarget['ita'],worldoffside['ita'],worldcor['ita'
pred2 = mod2.predict(np.array([worldred['ita'],worldontarget['ita'],worldoffside['ita'
predita = ((pred1[0]-pred2[0])*worldgames['ita'] + (pred1[1]-pred2[1])*eurogames['ita'

#group E
#Switzerland
pred1 = mod1.predict(np.array([worldontarget['swi'],worldoffside['swi'],worldcor['swi'
pred2 = mod2.predict(np.array([worldred['swi'],worldontarget['swi'],worldoffside['swi'
predswi = pred1-pred2
#Ecuador
pred1 = mod1.predict(np.array([ecuador[2],ecuador[6],10,ecuador[3],50],dtype = 'float'
pred2 = mod2.predict(np.array([2,ecuador[2],ecuador[6],ecuador[3],50,1],dtype = 'float
predecu = (pred1-pred2)/3
#Honduras
pred1 = mod1.predict(np.array([worldontarget['hon'],worldoffside['hon'],worldcor['hon'
pred2 = mod2.predict(np.array([worldred['hon'],worldontarget['hon'],worldoffside['hon'
```

```
predhon = pred1-pred2
#France
pred1 = mod1.predict(np.array([worldontarget['fra'],worldoffside['fra'],worldcor['fra'
pred2 = mod2.predict(np.array([worldred['fra'],worldontarget['fra'],worldoffside['fra'
predfra = ((pred1[0]-pred2[0])*worldgames['fra'] + (pred1[1]-pred2[1])*eurogames['fra'

#group F
#Argentina
pred1 = mod1.predict(np.array([worldontarget['arg'],worldoffside['arg'],worldcor['arg'
pred2 = mod2.predict(np.array([worldred['arg'],worldontarget['arg'],worldoffside['arg'
predarg = ((pred1[0]-pred2[0])*worldgames['arg'] + (pred1[1]-pred2[1]))/ (worldgames['
#Nigeria
pred1 = mod1.predict(np.array([worldontarget['nig'],worldoffside['nig'],worldcor['nig'
pred2 = mod2.predict(np.array([worldred['nig'],worldontarget['nig'],worldoffside['nig'
prednig = pred1-pred2
#Iran
#data for Iran is missing because Iran didn't show up for any highest level internatio
#so we calculate the attacking factor and defensive factor by its performance for the
#(friendly games are not included)
pred1 = 21.0
pred2 = 18.0
predira = (pred1-pred2)/10
#Bosnia and Herzegovina
#data for Bosnia and Herzegovina is missing because Bosnia and Herzegovina didn't show
#so we calculate the attacking factor and defensive factor by its performance for the
#(friendly games are included)
pred1 = 15.0
pred2 = 11.0
predbos = (pred1-pred2)/10
#group G
#Germany
pred1 = mod1.predict(np.array([worldontarget['ger'],worldoffside['ger'],worldcor['ger'
pred2 = mod2.predict(np.array([worldred['ger'],worldontarget['ger'],worldoffside['ger'
predger = ((pred1[0]-pred2[0])*worldgames['ger'] + (pred1[1]-pred2[1])*eurogames['ger'
#Ghana
pred1 = mod1.predict(np.array([worldontarget['gha'],worldoffside['gha'],worldcor['gha'
pred2 = mod2.predict(np.array([worldred['gha'],worldontarget['gha'],worldoffside['gha'
predgha = pred1-pred2
#United States
pred1 = mod1.predict(np.array([worldontarget['usa'],worldoffside['usa'],worldcor['usa'
pred2 = mod2.predict(np.array([worldred['usa'],worldontarget['usa'],worldoffside['usa'
predusa = pred1-pred2
#Portugal
pred1 = mod1.predict(np.array([worldontarget['por'],worldoffside['por'],worldcor['por'
pred2 = mod2.predict(np.array([worldred['por'],worldontarget['por'],worldoffside['por'
predpor = ((pred1[0]-pred2[0])*worldgames['por'] + (pred1[1]-pred2[1])*eurogames['por'

#group H
#Algeria
pred1 = mod1.predict(np.array([worldontarget['alg'],worldoffside['alg'],worldcor['alg'
pred2 = mod2.predict(np.array([worldred['alg'],worldontarget['alg'],worldoffside['alg'
predalg = pred1-pred2
#Korea
pred1 = mod1.predict(np.array([worldontarget['kor'],worldoffside['kor'],worldcor['kor'
pred2 = mod2.predict(np.array([worldred['kor'],worldontarget['kor'],worldoffside['kor'
predkor = pred1-pred2
#Russia
pred1 = mod1.predict(np.array([euroontarget['rus'],eurooffside['rus'],eurocor['rus'],e
pred2 = mod2.predict(np.array([eurored['rus'],euroontarget['rus'],eurooffside['rus'],e
predrus = pred1-pred2
#belgium
#data for belgium is missing because belgium didn't show up for any highest level tour
#so we calculate the attacking factor and defensive factor by its performance for the
#(friendly games are included)
pred1 = 16.0
pred2 = 12.0
predbel = (pred1-pred2)/10
```

In [21]:
```
#Print the results for every group
#We compare teams by their att factor and def factor and the output is the att - def
#Larger is Better!!
groupA = pd.Series(np.array([predbra,predcro,predmex,predcam]),index = ['Brazil','Croa
groupB = pd.Series(np.array([predspa,prednet,predchi,predaus]),index = ['Spain','Nethe
groupC = pd.Series(np.array([predcol,predgre,predcot,predjap]),index = ['Colombia','Gr
groupD = pd.Series(np.array([preduru,predcos,predeng,predita]),index = ['Uruguay','Cos
groupE = pd.Series(np.array([predswi,predecu,predfra,predhon]),index = ['Switzerland',
groupF = pd.Series(np.array([predarg,predbos,predira,prednig]),index = ['Argentina','B
groupG = pd.Series(np.array([predger,predpor,predgha,predusa]),index = ['Germany','Por
groupH = pd.Series(np.array([predbel,predalg,predrus,predkor]),index = ['Belgium','Alg
print groupA
print groupB
print groupC
print groupD
print groupE
print groupF
print groupG
print groupH
```

```
Brazil     1.242684
Croatia    0.192401
Mexico     0.050443
Cameron    0.379935
dtype: float64
Spain          0.688763
Netherlands    0.105526
Chile          0.433681
Australia     -0.213381
dtype: float64
Colombia          [0.955487736758]
Greece                  -0.3963158
Cote d'Ivoire     [0.522434946535]
Japan              [0.41332028415]
dtype: object
Uruguay       1.034289
Costa Rica    0.626179
England       0.146900
Italy         1.022030
dtype: float64
Switzerland    [-0.130536309989]
Ecuador         [0.533031778393]
France                 0.2939251
Honduras        [-1.24277192666]
dtype: object
Argentina                 1.504878
Bosnia and Herzegovina    0.400000
Iran                      0.300000
Nigeria                  -0.701112
dtype: float64
Germany     0.292132
Portugal    0.081118
Ghana       0.546163
USA         0.480041
dtype: float64
Belgium           0.400000
Algeria          -0.556775
Russia           -0.014848
Korea Republic    0.197414
```

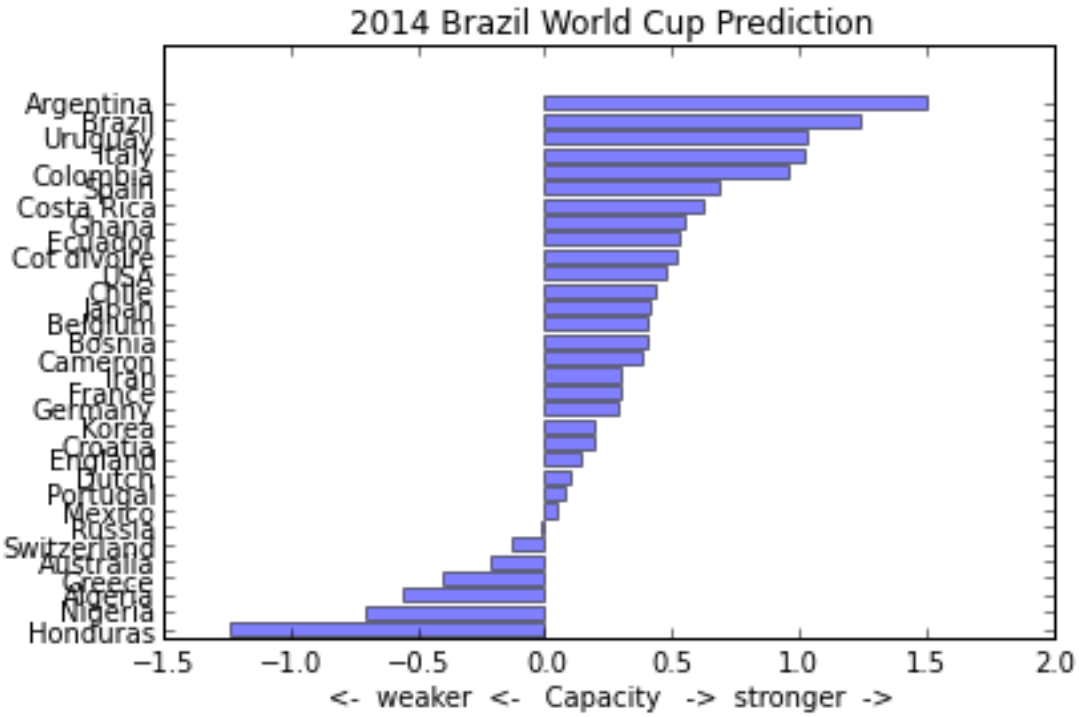
```
dtype: float64
```

Plot the rank by our prediction

In [22]:
```
allgroup = pd.Series(np.array([predbra,predcro,predmex,predcam,predspa,prednet,predchi
from pylab import *

fig = plt.figure()
ax1 = fig.add_subplot(1, 1, 1)
pos = arange(32)+.5    # the bar centers on the y axis
barh(pos,allgroup.order(), align='center',alpha =0.5 )
yticks(pos, allgroup.order().index)
xlabel('<-  weaker  <-   Capacity   ->  stronger  ->')
title('2014 Brazil World Cup Prediction')
```

```
<matplotlib.text.Text at 0x6f42a90>
```

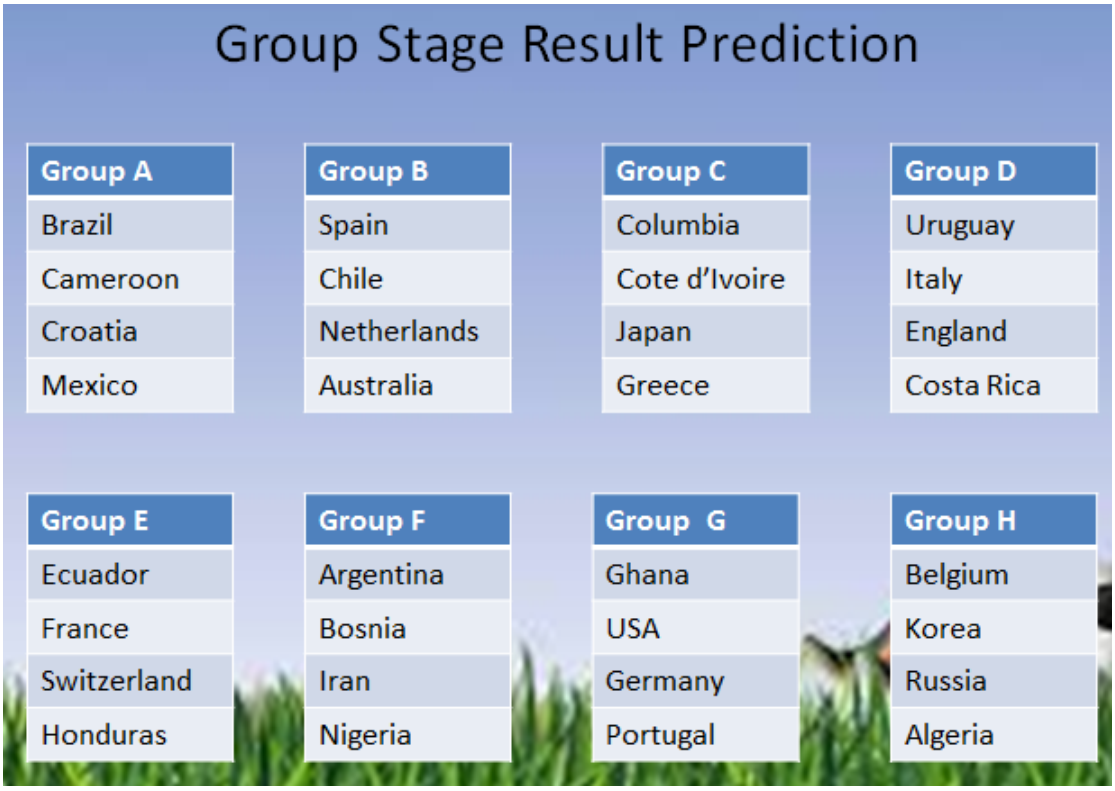
Out [22]:



8.1 The group stage results:

In [23]:
```
from IPython.core.display import Image
Image(filename='group_stage.png')
```

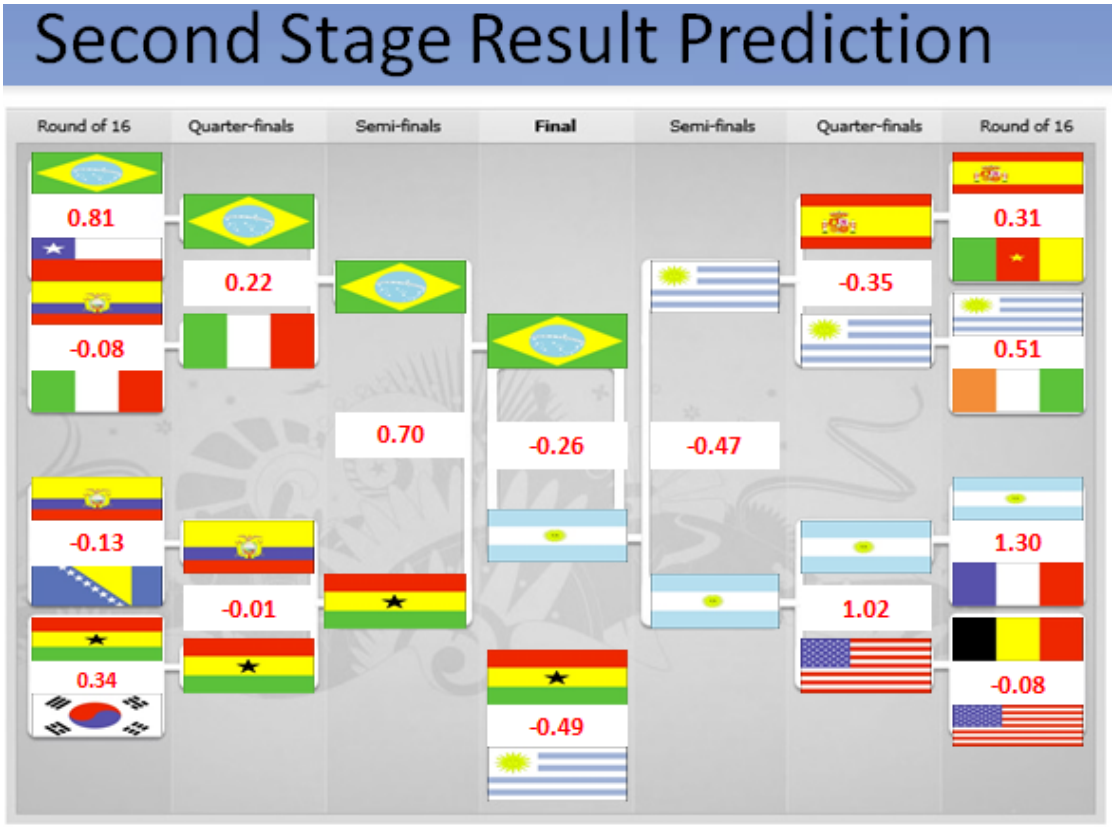
Out [23]:



8.2 The second stage results:

In [24]:
```
from IPython.core.display import Image
Image(filename='second_stage.png')
```

Out [24]:



In our prediction, Brazil and Argenina will show up at the final. And eventually Argentina will win the 2014 World Cup!

8.3 Latest winning odds given by several famous betting companies:

In [25]:
```
from IPython.core.display import Image
Image(filename='bettingodds.png')
```

Out [25]:

Latest 2014 World Cup Odds

	Ladbrokes	PADDYPOWER	bet365	BETVICTOR	BETFRED	William HILL	sky BET
Brazil	3/1	3/1	3/1	3/1	3/1	3/1	3/1
Argentina	9/2	5/1	4/1	5/1	9/2	9/2	5/1
Germany	11/2	11/2	11/2	11/2	11/2	11/2	5/1
Spain	6/1	7/1	7/1	7/1	7/1	7/1	6/1
Belgium	14/1	14/1	14/1	14/1	14/1	14/1	16/1
France	18/1	22/1	20/1	25/1	20/1	20/1	25/1
Italy	22/1	25/1	28/1	28/1	25/1	25/1	20/1
Colombia	22/1	25/1	25/1	25/1	20/1	22/1	25/1
Uruguay	28/1	25/1	25/1	28/1	28/1	28/1	28/1
Holland	20/1	22/1	28/1	25/1	25/1	28/1	25/1
England	33/1	25/1	33/1	33/1	33/1	33/1	28/1

Comparing to the betting odds, we are consistent for the final stage because we all believe that Brazil and Argenita are the best two teams. In our model, we don't trust those teams from Europe. However, the betting companies believe that Spain and Germany may be two good candidates for the winner. Let's see whose results are close in this summer!

Part V

Players Performance Analysus

We would like to see what is the relation between players' statistics and their performance. We use rating to measure a player's performance. The rating is given by authority groups which consist of commentators, newspapers and broadcasts. We assume the relationship between players statistics and their rating for each single game is linear.

9 Linear Model

First of all, we use a wider range of variables.

In [26]:
```
#in this script, we are going to run linear regression on players' statistics to see t
#the html source code for different year has the same pattern, so we can read data at
#read players data
filename = ['premierplayers0910.txt','premierplayers1011.txt','premierplayers1112.txt'
'laligaplayers1112.txt','laligaplayers1213.txt','serieAplayers0910.txt','serieAplayers
'bundesligaplayers0910.txt','bundesligaplayers1011.txt','bundesligaplayers1112.txt','b
'frenchplayers1112.txt','frenchplayers1213.txt','brazilplayers13.txt','championplayers
'russianplayers1314.txt','laligaplayers1314.txt','serieAplayers1314.txt','bundesligapl

columnnames = ['TeamId','PlayerId','Field','GameStarted','SubOn','SubOff','Yellow','Se
'AerialWon','AerialLost','Rating','ManOfTheMatch','TotalTackles','Interceptions','Foul
'ShotsOnTarget','ShotsBlocked','OwnGoals','KeyPasses','Dribbles','WasFouled','Offsides
'TotalLongBalls','AccurateLongBalls','TotalThroughBalls','AccurateThroughBalls','Heigh
data = []
```

```
for name in filename:
txt = open(name)
data1 = txt.read()
data2 = data1.split('DataStore.prime')
data3 = data2[1].split('}]]);')[0]
data4 = data3.split('[{')[1]
regex = re.compile(':[0-9]+\.*[0-9]*')
data5 = regex.findall(data4)
data6 = ' '+' '.join(data5)
data7 = data6.split(' :')[1:]
data.append(data7)

data8 = np.array(data).reshape(600,43)
data9 = DataFrame(data8,columns = columnnames,dtype='float')
data9['const'] = np.ones(600)
#reorder the index
ind = np.random.permutation(range(0,600))
mod1 = sm.OLS(data9.ix[0:600,15], data9.ix[0:600,[5,6,8,9,10,11,13,17,18,19,21,24,26,2
#Inspect the results
print mod1.summary()
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                  Rating   R-squared:
0.596
Model:                            OLS   Adj. R-squared:
0.581
Method:                 Least Squares   F-statistic:
40.59
Date:                Mon, 14 Apr 2014   Prob (F-statistic):
3.53e-99
Time:                        08:33:42   Log-Likelihood:
221.91
No. Observations:                 600   AIC:
-399.8
Df Residuals:                     578   BIC:
-303.1
Df Model:                          21
==============================================================================
=============
                     coef    std err          t      P>|t|
[95.0% Conf. Int.]
------------------------------------------------------------------------------
-------------
SubOff             -0.0118      0.002     -6.727      0.000
-0.015    -0.008
Yellow              0.0016      0.003      0.495      0.621
-0.005     0.008
Red                -0.0081      0.017     -0.481      0.630
-0.041     0.025
Goals               0.0222      0.003      8.726      0.000
0.017     0.027
Assists             0.0153      0.003      5.215      0.000
0.010     0.021
TotalPasses      2.412e-05   1.96e-05      1.233      0.218
-1.43e-05  6.25e-05
```

```
AerialWon           0.0017      0.000      6.063      0.000
0.001     0.002
TotalTackles        0.0008      0.000      1.699      0.090
-0.000     0.002
Interceptions      -0.0004      0.000     -0.835      0.404
-0.001     0.000
Fouls              -0.0021      0.001     -3.058      0.002
-0.003    -0.001
TotalClearances    -0.0001      0.000     -0.893      0.372
-0.000     0.000
ShotsOnTarget       0.0009      0.001      0.831      0.407
-0.001     0.003
OwnGoals           -0.0273      0.028     -0.972      0.331
-0.082     0.028
KeyPasses           0.0006      0.000      1.133      0.258
-0.000     0.002
Dribbles            0.0041      0.000     11.510      0.000
0.003     0.005
WasFouled          -0.0005      0.000     -1.095      0.274
-0.001     0.000
Offsides           -0.0031      0.001     -3.589      0.000
-0.005    -0.001
Turnovers          -0.0042      0.000     -9.842      0.000
-0.005    -0.003
Height             -0.0007      0.001     -0.685      0.494
-0.003     0.001
Weight              0.0011      0.002      0.688      0.492
-0.002     0.004
Age                -0.0007      0.002     -0.331      0.741
-0.005     0.003
const               7.2833      0.121     60.037      0.000
7.045     7.522
==============================================================================
========
Omnibus:                       40.958   Durbin-Watson:
1.114
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
69.924
Skew:                           0.472   Prob(JB):
6.55e-16
Kurtosis:                       4.380   Cond. No.
2.36e+04
==============================================================================
========

Warnings:
[1] The condition number is large, 2.36e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

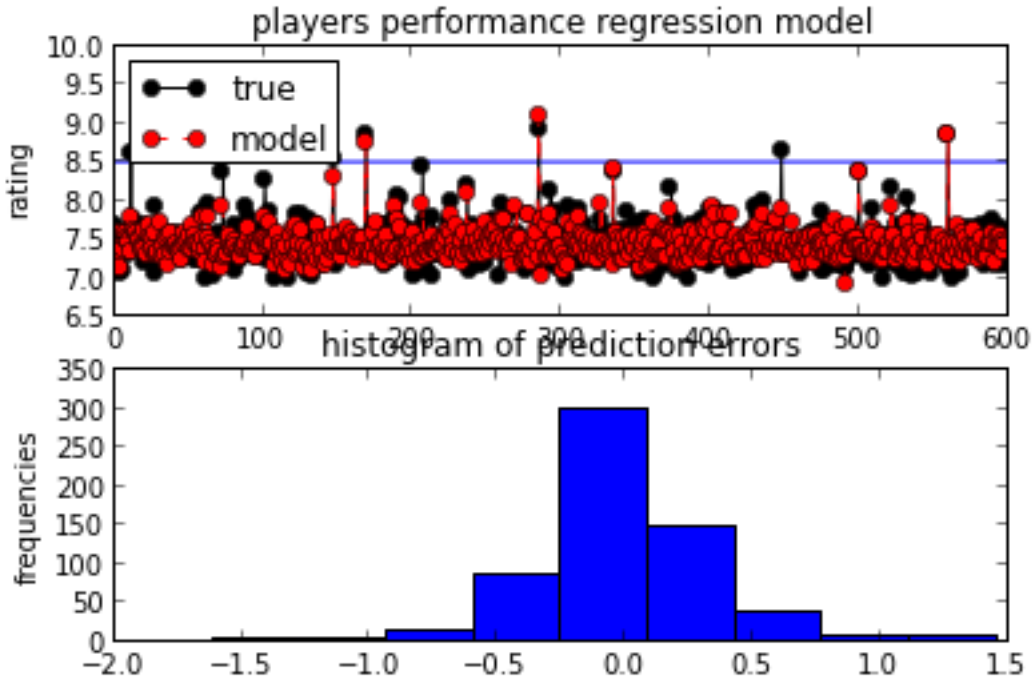
From the results, we can see there exits some multicolinearity. So we take of those highly correlated observations and those which fail to pass the t-test.

In [27]:
```
mod2 = sm.OLS(data9.ix[0:600,15], data9.ix[0:600,[5,9,10,13,17,19,27,28,30,32,43]]).fi
#Inspect the results
print mod2.summary()
predictions = mod2.predict(data9.ix[0:600,[5,9,10,13,17,19,27,28,30,32,43]])
fig = plt.figure()
ax1 = fig.add_subplot(2, 1, 1)
ax1.plot(data9.ix[ind,15], color = 'k', label = 'true', marker='o')
ax1.plot(predictions[ind], color = 'r', label = 'model',linestyle='dashed', marker='o'
ax1.hlines(8.5,0,600, color = 'b')
plt.ylim([6.5, 10])
plt.xlim([0, 600])
ax1.legend(loc='best')
ticks = ax1.set_xticks([0,100,200,300,400,500,600])
ax1.set_ylabel('rating')
ax1.set_title('players performance regression model')
ax2 = fig.add_subplot(2, 1, 2)
ax2.hist(data9.ix[ind,15]-predictions)
ax2.set_ylabel('frequencies')
ax2.set_title('histogram of prediction errors')
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                 Rating   R-squared:
0.590
Model:                            OLS   Adj. R-squared:
0.583
Method:                 Least Squares   F-statistic:
84.87
Date:                Mon, 14 Apr 2014   Prob (F-statistic):
2.94e-107
Time:                        08:33:44   Log-Likelihood:
217.79
No. Observations:                 600   AIC:
-413.6
Df Residuals:                     589   BIC:
-365.2
Df Model:                          10
==============================================================================
==========
                   coef    std err          t      P>|t|      [95.0%
Conf. Int.]
------------------------------------------------------------------------------
----------
SubOff          -0.0117      0.002     -6.969      0.000        -0.015
-0.008
Goals            0.0244      0.001     16.709      0.000         0.022
0.027
Assists          0.0159      0.003      5.476      0.000         0.010
0.022
AerialWon        0.0015      0.000      6.556      0.000         0.001
0.002
TotalTackles     0.0006      0.000      1.819      0.069     -4.77e-05
0.001
Fouls           -0.0019      0.001     -3.337      0.001        -0.003
-0.001
KeyPasses        0.0008      0.000      1.965      0.050      2.57e-07
0.002
Dribbles         0.0041      0.000     13.048      0.000         0.003
```

```
0.005
Offsides         -0.0030       0.001     -3.696       0.000         -0.005
-0.001
Turnovers        -0.0044       0.000    -11.118       0.000         -0.005
-0.004
const             7.2212       0.026    275.718       0.000          7.170
7.273
==============================================================================
========
Omnibus:                       39.394   Durbin-Watson:
1.093
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
64.591
Skew:                           0.470   Prob(JB):
9.42e-15
Kurtosis:                       4.303   Cond. No.
396.
==============================================================================
========
<matplotlib.text.Text at 0x75b3450>
```

Out [27]:



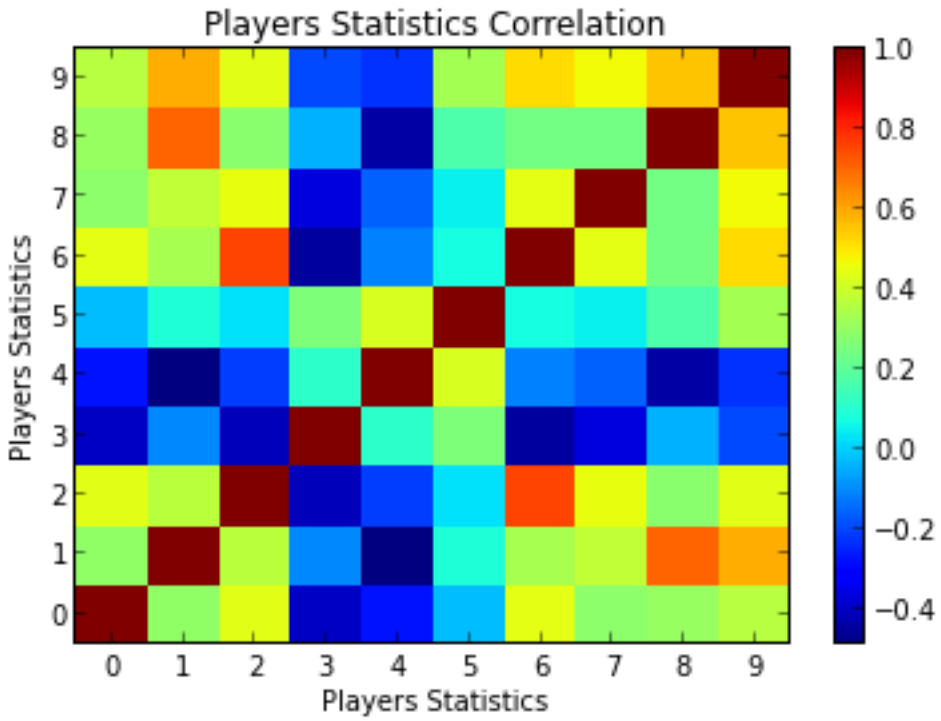
However, there still exits some multicorlinearity

In [28]:

```
fig = plt.figure()
ax1 = fig.add_subplot(1, 1, 1)
R = np.corrcoef(data9.ix[0:600,[5,9,10,13,17,19,27,28,30,32]].transpose())
pcolor(R)
colorbar()
yticks(arange(0.5,10.5),range(0,10))
xticks(arange(0.5,10.5),range(0,10))
xlabel('Players Statistics')
ylabel('Players Statistics')
title('Players Statistics Correlation')
```

Out [28]:

```
<matplotlib.text.Text at 0x7757f90>
```



10 Cross Validation

Since there should be some outliers, we take the leave-one-out cross validation method to do a model selection.

In [29]:

```
val = []
for i in ind:
mod = sm.OLS(data9[data9.index != i].ix[:,15], data9[data9.index != i].ix[:,[5,9,10,13
prediction = mod.predict(data9.ix[i,[5,9,10,13,17,19,27,28,30,32,43]])
val.append((prediction-data9.ix[i,15])**2)

#the one with minimum value is "ind[val.index(min(val))]"
mod = sm.OLS(data9[data9.index != ind[val.index(min(val))]].ix[:,15], data9[data9.inde
print mod.summary()
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                 Rating   R-squared:
0.590
Model:                            OLS   Adj. R-squared:
0.583
Method:                 Least Squares   F-statistic:
84.59
Date:                Mon, 14 Apr 2014   Prob (F-statistic):
5.95e-107
Time:                        08:33:51   Log-Likelihood:
216.93
No. Observations:                 599   AIC:
-411.9
Df Residuals:                     588   BIC:
```

```
-363.5
Df Model:                           10
==============================================================================
==========
                    coef    std err          t      P>|t|      [95.0%
Conf. Int.]
------------------------------------------------------------------------------
----------
SubOff          -0.0117      0.002     -6.958      0.000        -0.015
-0.008
Goals            0.0244      0.001     16.692      0.000         0.022
0.027
Assists          0.0159      0.003      5.471      0.000         0.010
0.022
AerialWon        0.0015      0.000      6.550      0.000         0.001
0.002
TotalTackles     0.0006      0.000      1.817      0.070     -4.84e-05
0.001
Fouls           -0.0019      0.001     -3.334      0.001        -0.003
-0.001
KeyPasses        0.0008      0.000      1.963      0.050      -5.2e-07
0.002
Dribbles         0.0041      0.000     13.023      0.000         0.003
0.005
Offsides        -0.0030      0.001     -3.693      0.000        -0.005
-0.001
Turnovers       -0.0044      0.000    -11.108      0.000        -0.005
-0.004
const            7.2212      0.026    275.186      0.000         7.170
7.273
==============================================================================
========
Omnibus:                       39.171   Durbin-Watson:
1.092
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
63.981
Skew:                           0.470   Prob(JB):
1.28e-14
Kurtosis:                       4.296   Cond. No.
396.
==============================================================================
========
```

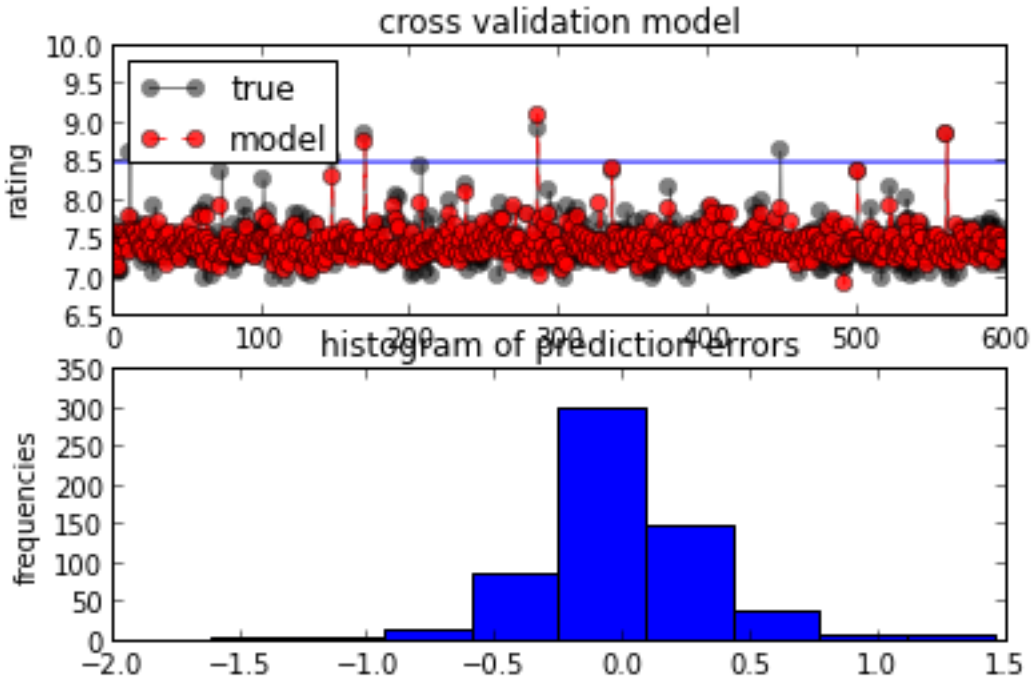
The result shows that it is a little bit better than the previous one. Then we make the same plot again.

In [30]:
```
prediction = mod.predict(data9.ix[:,[5,9,10,13,17,19,27,28,30,32,43]])
fig = plt.figure()
ax1 = fig.add_subplot(2, 1, 1)
ax1.plot(data9.ix[ind,15], color = 'k', label = 'true', marker='o',alpha = 0.5)
ax1.plot(prediction[ind], color = 'r', label = 'model',linestyle='dashed', marker='o',
ax1.hlines(8.5,0,600, color = 'b')
plt.ylim([6.5, 10])
plt.xlim([0, 600])
ax1.legend(loc='best')
ticks = ax1.set_xticks([0,100,200,300,400,500,600])
ax1.set_ylabel('rating')
```

```
ax1.set_title('cross validation model')
ax2 = fig.add_subplot(2, 1, 2)
ax2.hist(data9.ix[ind,15]-prediction)
ax2.set_ylabel('frequencies')
ax2.set_title('histogram of prediction errors')
```

```
<matplotlib.text.Text at 0x7dbc2d0>
```

Out [30]:



Part VI

Test Suites

We are going to use nosetests to test our work (unit test). In our project, we don't have any fancy function. However, we use dataframe a lot to store our data, and sort them to be prepared for calculations. So we need to check whether our data (also dimension) in our dataframe is correct with the one in the web data table.

11 Test for Home-Away analysis

Define the test for all home away table statistics data for each soccer league. First, check the shape of our data frame (or series). Second, record those specific numbers (randomly picked) in the website (data tables) and then compare those numbers in our python dataframe to check whether they are equal (after round them).

In [43]:
```
def test_homeaway():
assert len(bundesligahf) == 5
assert round(bundesligahf[0]) == 29
assert len(bundesligaaf) == 5
assert round(bundesligaaf[0]) == 21
assert len(premierhf) == 5
assert round(premierhf[1]) == 32
assert len(premieraf) == 5
```

```
assert round(premieraf[1]) == 20
assert len(frenchhf) == 5
assert round(frenchhf[2]) == 26
assert len(frenchaf) == 5
assert round(frenchaf[2]) == 19
assert len(laligahf) == 5
assert round(laligahf[3]) == 32
assert len(laligaaf) == 5
assert round(laligaaf[3]) == 21
assert len(serieAhf) == 5
assert round(serieAhf[4]) == 28
assert len(serieAaf) == 5
assert round(serieAaf[4]) == 22
```

12 Test for Attack-Defense analysis

Define the test for all attack-defense statistics data for each soccer league. First, check the shape of our data frame. Second, record those specific numbers (randomly picked) in the website (data tables) and then compare those numbers in our python dataframe to check whether they are equal.

In [44]:
```
def test_attdef():
assert len(bundesligaatt) == 5
assert round(bundesligaatt[0]) == 60
assert len(bundesligadef) == 5
assert round(bundesligadef[0]) == 62
assert len(premieratt) == 5
assert round(premieratt[1]) == 77
assert len(premierdef) == 5
assert round(premierdef[1]) == 75
assert len(frenchatt) == 5
assert round(frenchatt[2]) == 65
assert len(frenchdef) == 5
assert round(frenchdef[2]) == 63
assert len(laligaatt) == 5
assert round(laligaatt[3]) == 73
assert len(laligadef) == 5
assert round(laligadef[3]) == 72
assert len(serieAatt) == 5
assert round(serieAatt[4]) == 74
assert len(serieAdef) == 5
assert round(serieAdef[4]) == 74
```

13 Test for European Champions Leagues data

First, we check for European Champions Leagues data through the season 03/04 to 12/13. We record those specific numbers (randomly picked) in the website (data tables) and then compare those numbers in our python dataframe to check whether they are equal (after round them)

In [45]:
```
def test_uefa0304():
    assert round(uefa0304.ix[0,0]*13) == 27

def test_uefa0405():
    assert round(uefa0405.ix[1,1]*13) == 9

def test_uefa0506():
    assert round(uefa0506.ix[2,2]*12) == 19

def test_uefa0607():
```

```
    assert round(uefa0607.ix[3,3]*12) == 1

def test_uefa0708():
    assert round(uefa0708.ix[4,4]*10) == 53

def test_uefa0809():
    assert round(uefa0809.ix[5,5]*10) == 51

def test_uefa0910():
    assert round(uefa0910.ix[6,6]*10) == 8

def test_uefa1011():
    assert round(uefa1011.ix[7,7]*10) == 41

def test_uefa1112():
    assert round(uefa1112.ix[8,8]*8) == 101

def test_uefa1213():
    assert round(uefa1213.ix[9,9]*8) == 196
```

Define the test for the entire European Champions Leagues dataframe which is used to fit regression model.

In [46]:
```
def test_uefatotal():
assert uefa.shape[0] == 288
assert uefa.shape[1] == 12

assert round(uefa.ix[10,1]) == 2
assert round(uefa.ix[25,2]) == 2
assert round(uefa.ix[38,3]) == 1
assert round(uefa.ix[49,4]) == 6
assert round(uefa.ix[85,5]) == 4
assert round(uefa.ix[123,6]) == 3
assert round(uefa.ix[155,7]) == 6
assert round(uefa.ix[199,8]) == 13
assert round(uefa.ix[223,9]) == 28
```

14 Test for World Cup data

Define the test for World Cup data.

In [47]:
```
def test_worldcup():
assert len(worldontarget) == 32

assert round(worldontarget[3]) == 6
assert round(worldoffside[6]) == 2
assert round(worldfoul[9]) == 21
assert round(worldcor[12]) == 1
assert round(worldgames[15]) == 3
```

15 Test for European Cup data

Define the test for European Cup data.

In [48]:
```
def test_eurocup():
assert len(euroontarget) == 16

assert round(eurogames[0]) == 6
assert round(euroontarget[2]) == 8
assert round(eurooffside[4]) == 2
assert round(eurocor[6]) == 7
assert round(eurofoul[8]) == 20
```

16 Test for Players statistics data

Define the test for all players statistics data. First, check the shape of our data frame. Second, record those specific numbers (randomly picked) in the website (data tables) and then compare those numbers in our players statistics dataframe to check whether they are equal.

In [52]:
```
def test_players():
assert data9.shape[0] == 600
assert data9.shape[1] == 44

assert data9.ix[43,2] == 2
assert data9.ix[88,3] == 20
assert data9.ix[122,5] == 14
assert data9.ix[158,8] == 1
assert data9.ix[182,12] == 1357
assert data9.ix[202,15] == 7.62
assert data9.ix[243,18] == 15
assert data9.ix[288,21] == 2
assert data9.ix[304,24] == 17
assert data9.ix[333,26] == 0
assert data9.ix[365,28] == 73
assert data9.ix[398,33] == 7
assert data9.ix[410,35] == 67
assert data9.ix[437,37] == 8
assert data9.ix[469,31] == 30
assert data9.ix[510,39] == 176
assert data9.ix[548,41] == 9
assert data9.ix[587,42] == 24
```

16.1 Run the nosetests

In [53]:
```
%load_ext ipython_nose
```

```
The ipython_nose extension is already loaded. To reload it, use:
  %reload_ext ipython_nose
```

In [54]:
```
%nose -v -x
```

```
16/16 tests passed
```

Out [54]:

In []: