Python 3 practice

August 17, 2021

1 TYPE OF VARIABLES

Integer

[2]:
```
x=2
```

[3]:
```
x
```

[3]:
```
2
```

[4]:
```
type(x
)
```

[4]:
```
int
```

[5]:
```
type(x)
```

[5]:
```
int
```

Float/Double

[12]:
```
y=2.5
```

[13]:
```
y
```

[13]:
```
2.5
```

[14]:
```
type(y)
```

[14]:
```
float
```

String

[15]:
```
a='Hello 1'
b="Hello 2"
```

[16]:
```
a
```

[16]:
```
'Hello 1'
```

[17]:
```
b
```

[17]:
```
'Hello 2'
```

[18]:
```
type(a)
```

[18]:
```
str
```

[19]:
```
type(b)
```

[19]:
```
str
```

Logical/Bollean

[20]:
```
q1=True
```

[21]:
```
q1
```

[21]:
```
True
```

---

2 Using Variables

[1]:
```
A=10
B=5
```

[2]:
```
#Arthmetics
C=A+B
```

[3]:
```
D=B/A
```

[4]:
```
C
```

[4]:
```
15
```

[8]:
```
print(C) #Print() is different to python 2.7
```

```
15
```

[12]:
```
print(D) #Print() is different to python 2.7
```

```
0.5
```

[10]:
```
type(D)
```

[10]:
```
float
```

[11]:
```
type(C)
```

[11]:
```
int
```

---

[13]:
```
import math
```

[14]:
```
math.sqrt(A)
```

[14]:
```
3.1622776601683795
```

[15]:
```
round(math.sqrt(A))
```

[15]:
```
3
```

---

[17]:
```
greeting='Hello'
name="Bob"
```

[18]:
```
message=greeting+" "+name
```

[21]:
```
print(message)
```

```
Hello Bob
```

2.1 # Loop

While Loop

[1]:
```
# while condition:
#     executable code1
#     executable code2
#     execautable code3

# executable code4
```

[4]:
```
while False:
    print("Hello")
```

[5]:
```
counter=0
while counter<12:
    print(counter)
    counter = counter+1
print("Hello")
```

```
0
1
2
3
4
5
6
7
8
9
10
11
Hello
```

[22]:
```
type(q1)
```

[22]:
```
bool
```

For Loop

[1]:
```
for i in range(5):
    print("Hellow Python")
```

```
Hellow Python
Hellow Python
Hellow Python
Hellow Python
Hellow Python
```

[3]:
```
range(5) #diff in python 2.7
```

[3]:
```
range(0, 5)
```

[6]:
```
list(range(5))
```

[6]:
```
[0, 1, 2, 3, 4]
```

---

[7]:
```
for i in range(5):
    print("Hello Python:", i)
```

```
Hello Python: 0
Hello Python: 1
Hello Python: 2
Hello Python: 3
Hello Python: 4
```

[8]:
```
#Another Way
Mylist=[10,100,1000]
```

[10]:
```
for jj in Mylist:
    print("jj is qual to: ",jj)
```

```
jj is qual to:  10
jj is qual to:  100
jj is qual to:  1000
```

---

if statement

[2]:
```
#-------- -2 ------ -1 -------- 0 -------- 1 ----------- 2
```

[3]:
```
import numpy as np
```

[5]:
```
from numpy.random import randn
```

[39]:
```
randn()
```

[39]:
```
1.5441529390861473
```

[74]:
```
answer=None
x=randn()
if x>1:
    answer="Greater then 1"
print(x)
print(answer)
```

```
0.4596100476452051
None
```

[101]:
```
# else Statement
answer=None
x=randn()
if x>1:
    answer="Greater Then 1"
else:
    answer="less then 1"
print(x)
print(answer)
```

```
1.15697842378555
Greater Then 1
```

[138]:
```
# Nested Statement
answer=None
x=randn()
```

```
if x>1:
    answer="Greater Then 1"
else:
    if x>=-1:
        answer="Between -1 and 1"
    else:
        answer="Less then -1"
print(x)
print(answer)
```

```
-0.5881266604525832
Between -1 and 1
```

[153]:
```
# Chained statements
answer=None
x=randn()
if x>1:
    answer="Greater then 1"
elif x>=-1:
    answer="Between -1 and 1"
else:
    answer="Less than -1"
print(x)
print(answer)
```

```
0.21834026399174536
Between -1 and 1
```

3 List

[1]:
```
MyFirstList=[3,45,56,732]
```

[2]:
```
MyFirstList
```

[2]:
```
[3, 45, 56, 732]
```

[3]:
```
type(MyFirstList)
```

[3]:
```
list
```

[4]:
```
L2=["Hello",24,True,55.3]
```

[5]:
```
L2
```

[5]:
```
['Hello', 24, True, 55.3]
```

[6]:
```
L3=['How are you?',55,MyFirstList]
```

[7]:
```
L3
```

[7]:
```
['How are you?', 55, [3, 45, 56, 732]]
```

---

[8]:
```
range(15) # This is the same as Xrange() in Python2
```

[8]:
```
range(0, 15)
```

[9]:
```
range
```

[9]:
```
range
```

[10]:
```
list(range(15))
```

[10]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

[11]:
```
x={1,2,3,4,5}
```

[12]:
```
x
```

[12]:
```
{1, 2, 3, 4, 5}
```

[13]:
```
y=list(range(8))
```

[14]:
```
y
```

[14]:
```
[0, 1, 2, 3, 4, 5, 6, 7]
```

[16]:
```
z=list(range(1,18))
```

[17]:
```
z
```

[17]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
```

[18]:
```
z1=list(range(100,120))
```

[19]:
```
z1
```

[19]:
```
[100,
 101,
 102,
 103,
 104,
 105,
 106,
 107,
 108,
```

```
 109,
 110,
 111,
 112,
 113,
 114,
 115,
 116,
 117,
 118,
 119]
```

[20]:
```
w=list(range(100,111,2))
```

[21]:
```
w
```

[21]: [100, 102, 104, 106, 108, 110]

[22]:
```
w2=list(range(100,201,10))
```

[23]:
```
w2
```

[23]: [100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200]

[26]:
```
w=['a','b','c','d','e']
```

[27]:
```
w
```

[27]: ['a', 'b', 'c', 'd', 'e']

[28]:
```
w[0]
```

[28]: 'a'

[29]:
```
len(w)
```

[29]: 5

[30]:
```
w[2]='v'
```

[31]:
```
w
```

[31]: ['a', 'b', 'v', 'd', 'e']

[33]:
```
w[4]='z'
```

[34]:
```
w
```

[34]:
```
['a', 'b', 'v', 'd', 'z']
```

4 Slicing

[11]:
```
Litters=['A','B','C','D','E','F','G','H','I','J']
   #      0   1   2   3   4   5   6   7   8   9
   #    -10  -9  -8  -7  -6  -5  -4  -3  -2  -1
```

[4]:
```
Litters
```

[4]:
```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

[7]:
```
Litters[:]
```

[7]:
```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

[8]:
```
Litters[2:]
```

[8]:
```
['C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

[9]:
```
Litters[:7]
```

[9]:
```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

[10]:
```
Litters[2:7]
```

[10]:
```
['C', 'D', 'E', 'F', 'G']
```

[12]:
```
Litters[-8:7]
```

[12]:
```
['C', 'D', 'E', 'F', 'G']
```

[13]:
```
Litters[-8:-3]
```

[13]:
```
['C', 'D', 'E', 'F', 'G']
```

---

[14]:
```
# Advanced Slicing
Litters[2:9:2]
```

[14]:
```
['C', 'E', 'G', 'I']
```

[16]:
```
Litters[::3]
```

[16]:
```
['A', 'D', 'G', 'J']
```

[17]:
```
Litters[::-2]
```

[17]:
```
['J', 'H', 'F', 'D', 'B']
```

[18]:
```
Litters[-9:-4]
```

[18]:
```
['B', 'C', 'D', 'E', 'F']
```

[19]:
```
Litters[-9:-4:-2] #Nothing
```

[19]:
```
[]
```

[20]:
```
Litters[::-1]
```

[20]:
```
['J', 'I', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']
```

[21]:
```
Litters[2:7:1]
```

[21]:
```
['C', 'D', 'E', 'F', 'G']
```

[23]:
```
Litters[2:7:-1] # Nothing
```

[23]:
```
[]
```

[24]:
```
Litters[6:1:-2]
```

[24]:
```
['G', 'E', 'C']
```

---

5 Tuples

[6]:
```
t1=(345,678,435) #Immuatable list There won't change the list
```

[7]:
```
t1
```

[7]:
```
(345, 678, 435)
```

[8]:
```
t1[2]
```

[8]:
```
435
```

---

6 Functions

[1]:
```
range(20,31)
```

[1]:
```
range(20, 31)
```

[2]:
```
list(range(20,31))
```

[2]:
```
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
```

[3]:
```
Mylist1=list(range(20,31))
```

[4]:
```
Mylist1
```

[4]:
```
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
```

[5]:
```
len(Mylist1)
```

[5]:
```
11
```

[6]:
```
type(Mylist1)
```

[6]:
```
list
```

[7]:
```
max(Mylist1)
```

[7]:
```
30
```

[8]:
```
min(Mylist1)
```

[8]:
```
20
```

[1]:
```
max(11,435,7)
```

[1]:
```
435
```

---

7 Packages

[17]:
```
# Step1: Find Package(Optional)----------GITHUB
# Step2: Install Packages-----------CMD(anaconda)-------- conda install␣
 ↪PackageName OR pip install PackageName
# Step3: Import Pakages-------------import scrapy
#    OR
# Import Functions-----------from scrapy.crawler import CrawlerProcess
```

[1]:
```
import scrapy
```

[16]:
```
from scrapy.crawler import CrawlerProcess
```

---

8 Arrays

[1]:
```
l=[123,543,756,768,-1234]
```

[2]:
```
l
```

[2]:
```
[123, 543, 756, 768, -1234]
```

[3]:
```
import numpy as np
```

[4]:
```
a=np.array(l)
```

[5]:
```
a
```

[5]:
```
array([  123,   543,   756,   768, -1234])
```

[6]:
```
b=np.array([12,455,6.4,True,"abc"])
```

[7]:
```
b
```

[7]:
```
array(['12', '455', '6.4', 'True', 'abc'], dtype='<U32')
```

[8]:
```
print(b)
```

```
['12' '455' '6.4' 'True' 'abc']
```

[9]:
```
c=np.array([12,545,6.6,True])
```

[10]:
```
c
```

[10]:
```
array([ 12. , 545. ,   6.6,   1. ])
```

[11]:
```
type(c)
```

[11]:
```
numpy.ndarray
```

[12]:
```
print(c)
```

```
[ 12.  545.    6.6   1. ]
```

[14]:
```
d=np.array([12,545,6.6,7.7])
```

[15]:
```
d
```

[15]:
```
array([ 12. , 545. ,   6.6,   7.7])
```

[16]:
```
e=np.array([12,32,True])
```

[17]:
```
print(e)
```

```
[12 32  1]
```

[27]:
```
l.pop(-1) # Remove index=-1
```

[27]:
```
768
```

[28]:
```
l
```

[28]:
```
[543]
```

[34]:
```
a.mean()
```

[34]:
```
191.2
```

[31]:
```
a
```

[31]:
```
array([  123,   543,   756,   768, -1234])
```

---

[36]:
```
# Sclicing of Arrays
a[2:]
```

[36]:
```
array([  756,   768, -1234])
```

[37]:
```
a[2:4]
```

[37]:
```
array([756, 768])
```

[38]:
```
b=a[2:4]
```

[39]:
```
b
```

[39]:
```
array([756, 768])
```

[40]:
```
b[0]
```

[40]:
```
756
```

[43]:
```
b[:]=111   # It will also change original in a so, be carefully
```

[44]:
```
b
```

[44]:
```
array([111, 111])
```

[45]:
```
a
```

[45]:
```
array([  123,   543,   111,   111, -1234])
```

[46]:
```
c=a.copy() # Copy a and then changes will not effect on a
```

[47]:
```
c
```

[47]:
```
array([  123,   543,   111,   111, -1234])
```

[48]:
```
c[:]=222
```

[49]:
```
c
```

[49]:
```
array([222, 222, 222, 222, 222])
```

[50]:
```
a
```

[50]:
```
array([  123,   543,   111,   111, -1234])
```

---

9 Financial Statment analysis

[2]:
```
revenue = [14574.49, 7606.46, 8611.41, 9175.41, 8058.65, 8105.44, 11496.28,␣
 ↪9766.09, 10305.32, 14379.96, 10713.97, 15433.50]
expenses = [12051.82, 5695.07, 12319.20, 12089.72, 8658.57, 840.20, 3285.73,␣
 ↪5821.12, 6976.93, 16618.61, 10054.37, 3803.96]
```

10 Calculate profit(revenue-expenses)

[12]:
```
profit=[] # OR profit=list([])
for i in range(0,len(revenue)):
    profit.append(revenue[i]-expenses[i])
print(profit)
```

```
[2522.67, 1911.3900000000003, -3707.790000000001, -2914.3099999999995,
-599.9200000000001, 7265.24, 8210.550000000001, 3944.9700000000003,
3328.3899999999994, -2238.6500000000015, 659.5999999999985, 11629.54]
```

11 Calculate tax (profit x 30%)

[15]:
```
tax=[round(i*0.3,2) for i in profit]
print(tax)
```

```
[757, 573, -1112, -874, -180, 2180, 2463, 1183, 999, -672, 198, 3489]
```

12 Calculate profit after tax

[24]:
```
profit_after_tax=[]
for i in range(0,len(profit)):
    profit_after_tax.append(profit[i]-tax[i])
print(profit_after_tax)
```

```
[1765.67, 1338.3900000000003, -2595.790000000001, -2040.3099999999995,
-419.9200000000001, 5085.24, 5747.550000000001, 2761.9700000000003,
2329.3899999999994, -1566.6500000000015, 461.59999999999854, 8140.540000000001]
```

13 Calculate Profit Margin after tax

[26]:
```
profit_margin=list([])
for i in range(0,len(profit)):
    profit_margin.append(profit_after_tax[i]/revenue[i])
profit_margin=[round((i*100),2) for i in profit_margin]
print(profit_margin)
```

```
[12.11, 17.6, -30.14, -22.24, -5.21, 62.74, 49.99, 28.28, 22.6, -10.89, 4.31,
52.75]
```

14 Profit after tax Mean

[29]:
```
mean_PAT=sum(profit_after_tax)/len(profit_after_tax)
print(mean_PAT)
```

```
1750.64
```

15 Good Month

[32]:
```
good_month=[]
for i in range(0,len(profit)):
    good_month.append(profit_after_tax[i]>mean_PAT)
print(good_month)
```

```
[True, False, False, False, False, True, True, True, True, False, False, True]
```

16 Bad Month

[34]:
```
bad_month=[]
for i in range(0,len(profit)):
    bad_month.append(profit_after_tax[i]<mean_PAT)
print(bad_month)
```

```
[False, True, True, True, True, False, False, False, False, True, True, False]
```

17 Best Month

[36]:
```
best_month=[]
for i in range(0,len(profit)):
    best_month.append(profit_after_tax[i]==max(profit_after_tax))
print(best_month)
```

```
[False, False, False, False, False, False, False, False, False, False, False,
True]
```

18 Worst Month

[40]:
```
worst_month=[]
for i in range(0,len(profit)):
    worst_month.append(profit_after_tax[i]==min(profit_after_tax))
print(worst_month)
```

```
[False, False, True, False, False, False, False, False, False, False, False,
False]
```

19 Convert all calculations to units of one thousand Dollars

[44]:
```
revenue_1000= [round(i/1000,2) for i in revenue]
expenses_1000=[round(i/1000,2)for i in expenses]
profit_1000=  [round(i/1000,2) for i in profit]
profit_after_tax_1000=[round(i/1000,2) for i in profit_after_tax]

revenue_1000=[int(i) for i in revenue_1000]
expenses_1000=[int(i) for i in expenses_1000]
profit_1000=[int(i) for i in profit_1000]
profit_after_tax_1000=[int(i) for i in profit_after_tax_1000]
```

[46]:
```
print("Revenue :")
print(revenue_1000)
print("Expenses :")
print(expenses_1000)
print("Profit :")
print(profit_1000)
print("Profit after Tax :")
print(profit_after_tax_1000)
print("Profit Margin :")
print(profit_margin)
print("Good Months :")
print(good_month)
print("Bad Months :")
print(bad_month)
print("Best Month :")
```

```
print(best_month)
print("Worst Month")
print(worst_month)
```

```
Revenue :
[14, 7, 8, 9, 8, 8, 11, 9, 10, 14, 10, 15]
Expenses :
[12, 5, 12, 12, 8, 0, 3, 5, 6, 16, 10, 3]
Profit :
[2, 1, -3, -2, 0, 7, 8, 3, 3, -2, 0, 11]
Profit after Tax :
[1, 1, -2, -2, 0, 5, 5, 2, 2, -1, 0, 8]
Profit Margin :
[12.11, 17.6, -30.14, -22.24, -5.21, 62.74, 49.99, 28.28, 22.6, -10.89, 4.31,
52.75]
Good Months :
[True, False, False, False, False, True, True, True, True, False, False, True]
Bad Months :
[False, True, True, True, True, False, False, False, False, True, True, False]
Best Month :
[False, False, False, False, False, False, False, False, False, False, False,
True]
Worst Month
[False, False, True, False, False, False, False, False, False, False, False,
False]
```

---

20 Basketball Analyze

[4]:
```
#Dear Student,
#
#Welcome to the world of Basketball Data!
#I'm sure you will enjoy this section of the Python Programming course.
#
#Instructions for this dataset:
# Simply copy ALL the lines in this script by pressing
# CTRL+A on Windows or CMND+A on Mac and run the Jupyter cell
# Once you have executed the commands the following objects
# will be created:
# Matrices:
# - Salary
# - Games
# - MinutesPlayed
# - FieldGoals
# - FieldGoalAttempts
# - Points
```

```
# Lists:
# - Players
# - Seasons
# Dictionaries:
# - Sdict
# - Pdict
#We will understand these inside the course.
#
#Sincerely,
#Kirill Eremenko
#www.superdatascience.com

#Copyright: These datasets were prepared using publicly available data.
#           However, theses scripts are subject to Copyright Laws.
#           If you wish to use these Python scripts outside of the Python␣
 ↪Programming Course
#           by Kirill Eremenko, you may do so by referencing www.
 ↪superdatascience.com in your work.

#Comments:
#Seasons are labeled based on the first year in the season
#E.g. the 2012-2013 season is preseneted as simply 2012

#Notes and Corrections to the data:
#Kevin Durant: 2006 - College Data Used
#Kevin Durant: 2005 - Proxied With 2006 Data
#Derrick Rose: 2012 - Did Not Play
#Derrick Rose: 2007 - College Data Used
#Derrick Rose: 2006 - Proxied With 2007 Data
#Derrick Rose: 2005 - Proxied With 2007 Data

#Import numpy
import numpy as np

#Seasons
Seasons =␣
 ↪["2005","2006","2007","2008","2009","2010","2011","2012","2013","2014"]
Sdict = {"2005":0,"2006":1,"2007":2,"2008":3,"2009":4,"2010":5,"2011":6,"2012":
 ↪7,"2013":8,"2014":9}

#Players
Players =␣
 ↪["KobeBryant","JoeJohnson","LeBronJames","CarmeloAnthony","DwightHoward","ChrisBosh","Chris
          "DerrickRose","DwayneWade"]
Pdict = {"KobeBryant":0,"JoeJohnson":1,"LeBronJames":2,"CarmeloAnthony":
 ↪3,"DwightHoward":4,"ChrisBosh":5,"ChrisPaul":6,
```

```
        "KevinDurant":7,"DerrickRose":8,"DwayneWade":9}

#Salaries
KobeBryant_Salary =␣
↪[15946875,17718750,19490625,21262500,23034375,24806250,25244493,27849149,30453805,23500000]
JoeJohnson_Salary =␣
↪[12000000,12744189,13488377,14232567,14976754,16324500,18038573,19752645,21466718,23180790]
LeBronJames_Salary =␣
↪[4621800,5828090,13041250,14410581,15779912,14500000,16022500,17545000,19067500,20644400]
CarmeloAnthony_Salary =␣
↪[3713640,4694041,13041250,14410581,15779912,17149243,18518574,19450000,22407474,22458000]
DwightHoward_Salary =␣
↪[4493160,4806720,6061274,13758000,15202590,16647180,18091770,19536360,20513178,21436271]
ChrisBosh_Salary =␣
↪[3348000,4235220,12455000,14410581,15779912,14500000,16022500,17545000,19067500,20644400]
ChrisPaul_Salary =␣
↪[3144240,3380160,3615960,4574189,13520500,14940153,16359805,17779458,18668431,20068563]
KevinDurant_Salary =␣
↪[0,0,4171200,4484040,4796880,6053663,15506632,16669630,17832627,18995624]
DerrickRose_Salary =␣
↪[0,0,0,4822800,5184480,5546160,6993708,16402500,17632688,18862875]
DwayneWade_Salary =␣
↪[3031920,3841443,13041250,14410581,15779912,14200000,15691000,17182000,18673000,15000000]
#Matrix
Salary = np.array([KobeBryant_Salary, JoeJohnson_Salary, LeBronJames_Salary,␣
↪CarmeloAnthony_Salary, DwightHoward_Salary,
                   ChrisBosh_Salary, ChrisPaul_Salary, KevinDurant_Salary,␣
↪DerrickRose_Salary, DwayneWade_Salary])

#Games
KobeBryant_G = [80,77,82,82,73,82,58,78,6,35]
JoeJohnson_G = [82,57,82,79,76,72,60,72,79,80]
LeBronJames_G = [79,78,75,81,76,79,62,76,77,69]
CarmeloAnthony_G = [80,65,77,66,69,77,55,67,77,40]
DwightHoward_G = [82,82,82,79,82,78,54,76,71,41]
ChrisBosh_G = [70,69,67,77,70,77,57,74,79,44]
ChrisPaul_G = [78,64,80,78,45,80,60,70,62,82]
KevinDurant_G = [35,35,80,74,82,78,66,81,81,27]
DerrickRose_G = [40,40,40,81,78,81,39,0,10,51]
DwayneWade_G = [75,51,51,79,77,76,49,69,54,62]
#Matrix
Games = np.array([KobeBryant_G, JoeJohnson_G, LeBronJames_G, CarmeloAnthony_G,␣
↪DwightHoward_G, ChrisBosh_G, ChrisPaul_G,
                  KevinDurant_G, DerrickRose_G, DwayneWade_G])

#Minutes Played
```

```
KobeBryant_MP = [3277,3140,3192,2960,2835,2779,2232,3013,177,1207]
JoeJohnson_MP = [3340,2359,3343,3124,2886,2554,2127,2642,2575,2791]
LeBronJames_MP = [3361,3190,3027,3054,2966,3063,2326,2877,2902,2493]
CarmeloAnthony_MP = [2941,2486,2806,2277,2634,2751,1876,2482,2982,1428]
DwightHoward_MP = [3021,3023,3088,2821,2843,2935,2070,2722,2396,1223]
ChrisBosh_MP = [2751,2658,2425,2928,2526,2795,2007,2454,2531,1556]
ChrisPaul_MP = [2808,2353,3006,3002,1712,2880,2181,2335,2171,2857]
KevinDurant_MP = [1255,1255,2768,2885,3239,3038,2546,3119,3122,913]
DerrickRose_MP = [1168,1168,1168,3000,2871,3026,1375,0,311,1530]
DwayneWade_MP = [2892,1931,1954,3048,2792,2823,1625,2391,1775,1971]
#Matrix
MinutesPlayed = np.array([KobeBryant_MP, JoeJohnson_MP, LeBronJames_MP,␣
↪CarmeloAnthony_MP, DwightHoward_MP, ChrisBosh_MP,
                          ChrisPaul_MP, KevinDurant_MP, DerrickRose_MP,␣
↪DwayneWade_MP])

#Field Goals
KobeBryant_FG = [978,813,775,800,716,740,574,738,31,266]
JoeJohnson_FG = [632,536,647,620,635,514,423,445,462,446]
LeBronJames_FG = [875,772,794,789,768,758,621,765,767,624]
CarmeloAnthony_FG = [756,691,728,535,688,684,441,669,743,358]
DwightHoward_FG = [468,526,583,560,510,619,416,470,473,251]
ChrisBosh_FG = [549,543,507,615,600,524,393,485,492,343]
ChrisPaul_FG = [407,381,630,631,314,430,425,412,406,568]
KevinDurant_FG = [306,306,587,661,794,711,643,731,849,238]
DerrickRose_FG = [208,208,208,574,672,711,302,0,58,338]
DwayneWade_FG = [699,472,439,854,719,692,416,569,415,509]
#Matrix
FieldGoals  = np.array([KobeBryant_FG, JoeJohnson_FG, LeBronJames_FG,␣
↪CarmeloAnthony_FG, DwightHoward_FG, ChrisBosh_FG,
                        ChrisPaul_FG, KevinDurant_FG, DerrickRose_FG,␣
↪DwayneWade_FG])

#Field Goal Attempts
KobeBryant_FGA = [2173,1757,1690,1712,1569,1639,1336,1595,73,713]
JoeJohnson_FGA = [1395,1139,1497,1420,1386,1161,931,1052,1018,1025]
LeBronJames_FGA = [1823,1621,1642,1613,1528,1485,1169,1354,1353,1279]
CarmeloAnthony_FGA = [1572,1453,1481,1207,1502,1503,1025,1489,1643,806]
DwightHoward_FGA = [881,873,974,979,834,1044,726,813,800,423]
ChrisBosh_FGA = [1087,1094,1027,1263,1158,1056,807,907,953,745]
ChrisPaul_FGA = [947,871,1291,1255,637,928,890,856,870,1170]
KevinDurant_FGA = [647,647,1366,1390,1668,1538,1297,1433,1688,467]
DerrickRose_FGA = [436,436,436,1208,1373,1597,695,0,164,835]
DwayneWade_FGA = [1413,962,937,1739,1511,1384,837,1093,761,1084]
#Matrix
FieldGoalAttempts = np.array([KobeBryant_FGA, JoeJohnson_FGA, LeBronJames_FGA,␣
↪CarmeloAnthony_FGA, DwightHoward_FGA,
```

```
                                ChrisBosh_FGA, ChrisPaul_FGA, KevinDurant_FGA,␣
↪DerrickRose_FGA, DwayneWade_FGA])

#Points
KobeBryant_PTS = [2832,2430,2323,2201,1970,2078,1616,2133,83,782]
JoeJohnson_PTS = [1653,1426,1779,1688,1619,1312,1129,1170,1245,1154]
LeBronJames_PTS = [2478,2132,2250,2304,2258,2111,1683,2036,2089,1743]
CarmeloAnthony_PTS = [2122,1881,1978,1504,1943,1970,1245,1920,2112,966]
DwightHoward_PTS = [1292,1443,1695,1624,1503,1784,1113,1296,1297,646]
ChrisBosh_PTS = [1572,1561,1496,1746,1678,1438,1025,1232,1281,928]
ChrisPaul_PTS = [1258,1104,1684,1781,841,1268,1189,1186,1185,1564]
KevinDurant_PTS = [903,903,1624,1871,2472,2161,1850,2280,2593,686]
DerrickRose_PTS = [597,597,597,1361,1619,2026,852,0,159,904]
DwayneWade_PTS = [2040,1397,1254,2386,2045,1941,1082,1463,1028,1331]
#Matrix
Points = np.array([KobeBryant_PTS, JoeJohnson_PTS, LeBronJames_PTS,␣
↪CarmeloAnthony_PTS, DwightHoward_PTS, ChrisBosh_PTS,
                   ChrisPaul_PTS, KevinDurant_PTS, DerrickRose_PTS,␣
↪DwayneWade_PTS])
```

[5]:
```
print(Salary)
```

```
[[15946875 17718750 19490625 21262500 23034375 24806250 25244493 27849149
  30453805 23500000]
 [12000000 12744189 13488377 14232567 14976754 16324500 18038573 19752645
  21466718 23180790]
 [ 4621800  5828090 13041250 14410581 15779912 14500000 16022500 17545000
  19067500 20644400]
 [ 3713640  4694041 13041250 14410581 15779912 17149243 18518574 19450000
  22407474 22458000]
 [ 4493160  4806720  6061274 13758000 15202590 16647180 18091770 19536360
  20513178 21436271]
 [ 3348000  4235220 12455000 14410581 15779912 14500000 16022500 17545000
  19067500 20644400]
 [ 3144240  3380160  3615960  4574189 13520500 14940153 16359805 17779458
  18668431 20068563]
 [       0        0  4171200  4484040  4796880  6053663 15506632 16669630
  17832627 18995624]
 [       0        0        0  4822800  5184480  5546160  6993708 16402500
  17632688 18862875]
 [ 3031920  3841443 13041250 14410581 15779912 14200000 15691000 17182000
  18673000 15000000]]
```

[6]:
```
print(Points)
```

```
[[2832 2430 2323 2201 1970 2078 1616 2133   83  782]
 [1653 1426 1779 1688 1619 1312 1129 1170 1245 1154]
```

```
[2478 2132 2250 2304 2258 2111 1683 2036 2089 1743]
[2122 1881 1978 1504 1943 1970 1245 1920 2112  966]
[1292 1443 1695 1624 1503 1784 1113 1296 1297  646]
[1572 1561 1496 1746 1678 1438 1025 1232 1281  928]
[1258 1104 1684 1781  841 1268 1189 1186 1185 1564]
[ 903  903 1624 1871 2472 2161 1850 2280 2593  686]
[ 597  597  597 1361 1619 2026  852    0  159  904]
[2040 1397 1254 2386 2045 1941 1082 1463 1028 1331]]
```

[7]:
```
import numpy as np
mydata=np.arange(0,20)
print(mydata)
type(mydata)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

[7]:
```
numpy.ndarray
```

[8]:
```
np.reshape(mydata,(5,4))
```

[8]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

[9]:
```
mart1=np.reshape(mydata,(5,4),order="C")
print(mart1)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

[10]:
```
mart1[2,2]
```

[10]:
```
10
```

[11]:
```
mart2=np.reshape(mydata,(5,4), order="F")
print(mart2)
```

```
[[ 0  5 10 15]
 [ 1  6 11 16]
 [ 2  7 12 17]
 [ 3  8 13 18]
 [ 4  9 14 19]]
```

[12]:
```
mart2[0,2]
```

[12]:
```
10
```

[13]:
```
mydata.reshape((5,4))
```

[13]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

[14]:
```
r1=["I","AM","HAPPY"]
r2=["WHAT","A","DAY"]
r3=[1,2,3]
[r1,r2,r3]
```

[14]:
```
[['I', 'AM', 'HAPPY'], ['WHAT', 'A', 'DAY'], [1, 2, 3]]
```

[15]:
```
np.array([r1,r2,r3])
```

[15]:
```
array([['I', 'AM', 'HAPPY'],
       ['WHAT', 'A', 'DAY'],
       ['1', '2', '3']], dtype='<U11')
```

[16]:
```
print(np.array([r1,r2,r3]))
```

```
[['I' 'AM' 'HAPPY']
 ['WHAT' 'A' 'DAY']
 ['1' '2' '3']]
```

---

21 Dictionaries

[17]:
```
Games
```

[17]:
```
array([[80, 77, 82, 82, 73, 82, 58, 78,  6, 35],
       [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
       [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
       [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
       [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],
       [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
       [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
       [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
       [40, 40, 40, 81, 78, 81, 39,  0, 10, 51],
       [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

[18]:
```
Games[0]
```

[18]:
```
array([80, 77, 82, 82, 73, 82, 58, 78,  6, 35])
```

[19]:
```
Games[2][9]
```

[19]:
```
69
```

[20]:
```
Games[2][-1]
```

[20]:
```
69
```

[21]:
```
Games[2,-1]
```

[21]:
```
69
```

[22]:
```
Games[2,9]
```

[22]:
```
69
```

[23]:
```
Points
```

[23]:
```
array([[2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133,   83,  782],
       [1653, 1426, 1779, 1688, 1619, 1312, 1129, 1170, 1245, 1154],
       [2478, 2132, 2250, 2304, 2258, 2111, 1683, 2036, 2089, 1743],
       [2122, 1881, 1978, 1504, 1943, 1970, 1245, 1920, 2112,  966],
       [1292, 1443, 1695, 1624, 1503, 1784, 1113, 1296, 1297,  646],
       [1572, 1561, 1496, 1746, 1678, 1438, 1025, 1232, 1281,  928],
       [1258, 1104, 1684, 1781,  841, 1268, 1189, 1186, 1185, 1564],
       [ 903,  903, 1624, 1871, 2472, 2161, 1850, 2280, 2593,  686],
       [ 597,  597,  597, 1361, 1619, 2026,  852,    0,  159,  904],
       [2040, 1397, 1254, 2386, 2045, 1941, 1082, 1463, 1028, 1331]])
```

[24]:
```
Points[6]
```

[24]:
```
array([1258, 1104, 1684, 1781,  841, 1268, 1189, 1186, 1185, 1564])
```

[25]:
```
Points[6,1]
```

[25]:
```
1104
```

[26]:
```
dict1={"key1":"val1","key2":"val2","key3":"val3"}
dict1
```

[26]:
```
{'key1': 'val1', 'key2': 'val2', 'key3': 'val3'}
```

[27]:
```
dict1["key1"]
```

[27]:
```
'val1'
```

[28]:
```
dict1["key2"]
```

[28]:
```
'val2'
```

[29]:
```
dict2={"Germany":"I have been here","France":2,"Spain":True}
dict2
```

[29]:
```
{'Germany': 'I have been here', 'France': 2, 'Spain': True}
```

[30]:
```
dict2["France"]
```

[30]:
```
2
```

[31]:
```
print(Pdict)
```

```
{'KobeBryant': 0, 'JoeJohnson': 1, 'LeBronJames': 2, 'CarmeloAnthony': 3,
'DwightHoward': 4, 'ChrisBosh': 5, 'ChrisPaul': 6, 'KevinDurant': 7,
'DerrickRose': 8, 'DwayneWade': 9}
```

[32]:
```
print(Sdict)
```

```
{'2005': 0, '2006': 1, '2007': 2, '2008': 3, '2009': 4, '2010': 5, '2011': 6,
'2012': 7, '2013': 8, '2014': 9}
```

[33]:
```
Pdict["KobeBryant"]
```

[33]:
```
0
```

[34]:
```
Games[0]
```

[34]:
```
array([80, 77, 82, 82, 73, 82, 58, 78,  6, 35])
```

[35]:
```
Points[0]
```

[35]:
```
array([2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133,   83,  782])
```

[36]:
```
Games[Pdict["KobeBryant"]]
```

[36]:
```
array([80, 77, 82, 82, 73, 82, 58, 78,  6, 35])
```

[37]:
```
Points[Pdict["KobeBryant"]]
```

[37]:
```
array([2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133,   83,  782])
```

[38]:
```
print(Salary[Pdict["KobeBryant"]])
```

```
[15946875 17718750 19490625 21262500 23034375 24806250 25244493 27849149
 30453805 23500000]
```

[39]:
```
Games[Pdict["DerrickRose"]]
```

[39]:
```
array([40, 40, 40, 81, 78, 81, 39,  0, 10, 51])
```

[40]:
```
Games[Pdict["DerrickRose"]][7]
```

[40]: 0

[41]:
```
Games[Pdict["DerrickRose"]][Sdict["2012"]]
```

[41]: 0

[42]:
```
Points[Pdict["JoeJohnson"]][Sdict["2010"]]
```

[42]: 1312

[43]:
```
Points[Pdict["JoeJohnson"]]
```

[43]: array([1653, 1426, 1779, 1688, 1619, 1312, 1129, 1170, 1245, 1154])

[44]:
```
Points[Pdict["JoeJohnson"]][Sdict['2014']]
```

[44]: 1154

---

22 Metrice Operations

[45]:
```
print(Pdict["LeBronJames"])
print(Sdict["2010"])
```

```
2
5
```

[46]:
```
Salary[2][5]
```

[46]: 14500000

[47]:
```
Salary[Pdict["LeBronJames"]][Sdict["2010"]]
```

[47]: 14500000

[48]:
```
FieldGoals
```

[48]:
```
array([[978, 813, 775, 800, 716, 740, 574, 738,  31, 266],
       [632, 536, 647, 620, 635, 514, 423, 445, 462, 446],
       [875, 772, 794, 789, 768, 758, 621, 765, 767, 624],
       [756, 691, 728, 535, 688, 684, 441, 669, 743, 358],
       [468, 526, 583, 560, 510, 619, 416, 470, 473, 251],
       [549, 543, 507, 615, 600, 524, 393, 485, 492, 343],
       [407, 381, 630, 631, 314, 430, 425, 412, 406, 568],
       [306, 306, 587, 661, 794, 711, 643, 731, 849, 238],
       [208, 208, 208, 574, 672, 711, 302,   0,  58, 338],
```

```
        [699, 472, 439, 854, 719, 692, 416, 569, 415, 509]])
```

[49]:
```
Games
```

[49]:
```
array([[80, 77, 82, 82, 73, 82, 58, 78,  6, 35],
       [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
       [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
       [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
       [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],
       [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
       [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
       [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
       [40, 40, 40, 81, 78, 81, 39,  0, 10, 51],
       [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

[50]:
```
import warnings
warnings.filterwarnings('ignore')
FieldGoals/Games
```

[50]:
```
array([[12.225     , 10.55844156,  9.45121951,  9.75609756,  9.80821918,
         9.02439024,  9.89655172,  9.46153846,  5.16666667,  7.6       ],
       [ 7.70731707,  9.40350877,  7.8902439 ,  7.84810127,  8.35526316,
         7.13888889,  7.05      ,  6.18055556,  5.84810127,  5.575     ],
       [11.07594937,  9.8974359 , 10.58666667,  9.74074074, 10.10526316,
         9.59493671, 10.01612903, 10.06578947,  9.96103896,  9.04347826],
       [ 9.45      , 10.63076923,  9.45454545,  8.10606061,  9.97101449,
         8.88311688,  8.01818182,  9.98507463,  9.64935065,  8.95      ],
       [ 5.70731707,  6.41463415,  7.1097561 ,  7.08860759,  6.2195122 ,
         7.93589744,  7.7037037 ,  6.18421053,  6.66197183,  6.12195122],
       [ 7.84285714,  7.86956522,  7.56716418,  7.98701299,  8.57142857,
         6.80519481,  6.89473684,  6.55405405,  6.2278481 ,  7.79545455],
       [ 5.21794872,  5.953125  ,  7.875     ,  8.08974359,  6.97777778,
         5.375     ,  7.08333333,  5.88571429,  6.5483871 ,  6.92682927],
       [ 8.74285714,  8.74285714,  7.3375    ,  8.93243243,  9.68292683,
         9.11538462,  9.74242424,  9.02469136, 10.48148148,  8.81481481],
       [ 5.2       ,  5.2       ,  5.2       ,  7.08641975,  8.61538462,
         8.77777778,  7.74358974,         nan,  5.8       ,  6.62745098],
       [ 9.32      ,  9.25490196,  8.60784314, 10.81012658,  9.33766234,
         9.10526316,  8.48979592,  8.24637681,  7.68518519,  8.20967742]])
```

[51]:
```
FieldGoalPerGame=np.matrix.round(FieldGoals/Games)
```

[52]:
```
print(FieldGoalPerGame)
```

```
[[12. 11.  9. 10. 10.  9. 10.  9.  5.  8.]
 [ 8.  9.  8.  8.  8.  7.  7.  6.  6.  6.]
 [11. 10. 11. 10. 10. 10. 10. 10. 10.  9.]
 [ 9. 11.  9.  8. 10.  9.  8. 10. 10.  9.]
```

```
 [ 6.  6.  7.  7.  6.  8.  8.  6.  7.  6.]
 [ 8.  8.  8.  8.  9.  7.  7.  7.  6.  8.]
 [ 5.  6.  8.  8.  7.  5.  7.  6.  7.  7.]
 [ 9.  9.  7.  9. 10.  9. 10.  9. 10.  9.]
 [ 5.  5.  5.  7.  9.  9.  8. nan  6.  7.]
 [ 9.  9.  9. 11.  9.  9.  8.  8.  8.  8.]]
```

[53]:
```
FieldGoalPerGame[Pdict["DerrickRose"]][Sdict["2013"]]
```

[53]:
```
6.0
```

[54]:
```
MinutesPlayed
```

[54]:
```
array([[3277, 3140, 3192, 2960, 2835, 2779, 2232, 3013,  177, 1207],
       [3340, 2359, 3343, 3124, 2886, 2554, 2127, 2642, 2575, 2791],
       [3361, 3190, 3027, 3054, 2966, 3063, 2326, 2877, 2902, 2493],
       [2941, 2486, 2806, 2277, 2634, 2751, 1876, 2482, 2982, 1428],
       [3021, 3023, 3088, 2821, 2843, 2935, 2070, 2722, 2396, 1223],
       [2751, 2658, 2425, 2928, 2526, 2795, 2007, 2454, 2531, 1556],
       [2808, 2353, 3006, 3002, 1712, 2880, 2181, 2335, 2171, 2857],
       [1255, 1255, 2768, 2885, 3239, 3038, 2546, 3119, 3122,  913],
       [1168, 1168, 1168, 3000, 2871, 3026, 1375,    0,  311, 1530],
       [2892, 1931, 1954, 3048, 2792, 2823, 1625, 2391, 1775, 1971]])
```

[55]:
```
MinutesPlayedPerGame=np.matrix.round(MinutesPlayed/Games)
```

[56]:
```
print(MinutesPlayedPerGame)
```

```
[[41. 41. 39. 36. 39. 34. 38. 39. 30. 34.]
 [41. 41. 41. 40. 38. 35. 35. 37. 33. 35.]
 [43. 41. 40. 38. 39. 39. 38. 38. 38. 36.]
 [37. 38. 36. 34. 38. 36. 34. 37. 39. 36.]
 [37. 37. 38. 36. 35. 38. 38. 36. 34. 30.]
 [39. 39. 36. 38. 36. 36. 35. 33. 32. 35.]
 [36. 37. 38. 38. 38. 36. 36. 33. 35. 35.]
 [36. 36. 35. 39. 40. 39. 39. 39. 39. 34.]
 [29. 29. 29. 37. 37. 37. 35. nan 31. 30.]
 [39. 38. 38. 39. 36. 37. 33. 35. 33. 32.]]
```

[57]:
```
AVGSalaryPerGame=np.matrix.round(Salary/Games)
```

[58]:
```
print(AVGSalaryPerGame)
```

```
[[ 199336.  230114.  237691.  259299.  315539.  302515.  435250.  357040.
   5075634.  671429.]
 [ 146341.  223582.  164492.  180159.  197063.  226729.  300643.  274342.
   271731.  289760.]
 [  58504.   74719.  173883.  177908.  207630.  183544.  258427.  230855.
   247630.  299194.]
```

```
[  46420.   72216.  169367.  218342.  228694.  222717.  336701.  290299.
  291006.  561450.]
[  54795.   58619.   73918.  174152.  185397.  213425.  335033.  257057.
  288918.  522836.]
[  47829.   61380.  185896.  187150.  225427.  188312.  281096.  237095.
  241361.  469191.]
[  40311.   52815.   45200.   58643.  300456.  186752.  272663.  253992.
  301104.  244739.]
[      0.       0.   52140.   60595.   58499.   77611.  234949.  205798.
  220156.  703542.]
[      0.       0.       0.   59541.   66468.   68471.  179326.      inf
 1763269.  369860.]
[  40426.   75322.  255711.  182412.  204934.  186842.  320224.  249014.
  345796.  241935.]]
```

[59]:
```
AccuracyPerGame=np.matrix.round(FieldGoals/FieldGoalAttempts,2)*100
```

[60]:
```
print(AccuracyPerGame)
```

```
[[45. 46. 46. 47. 46. 45. 43. 46. 42. 37.]
 [45. 47. 43. 44. 46. 44. 45. 42. 45. 44.]
 [48. 48. 48. 49. 50. 51. 53. 56. 57. 49.]
 [48. 48. 49. 44. 46. 46. 43. 45. 45. 44.]
 [53. 60. 60. 57. 61. 59. 57. 58. 59. 59.]
 [51. 50. 49. 49. 52. 50. 49. 53. 52. 46.]
 [43. 44. 49. 50. 49. 46. 48. 48. 47. 49.]
 [47. 47. 43. 48. 48. 46. 50. 51. 50. 51.]
 [48. 48. 48. 48. 49. 45. 43. nan 35. 40.]
 [49. 49. 47. 49. 48. 50. 50. 52. 55. 47.]]
```

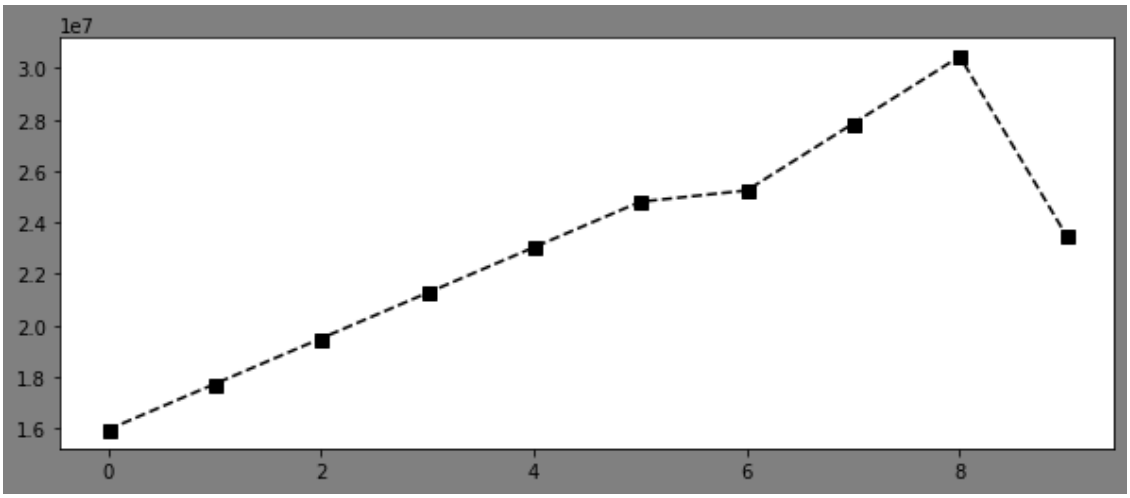
---

23 Virtualization

[61]:
```
import numpy as np
import matplotlib.pyplot as plt
```

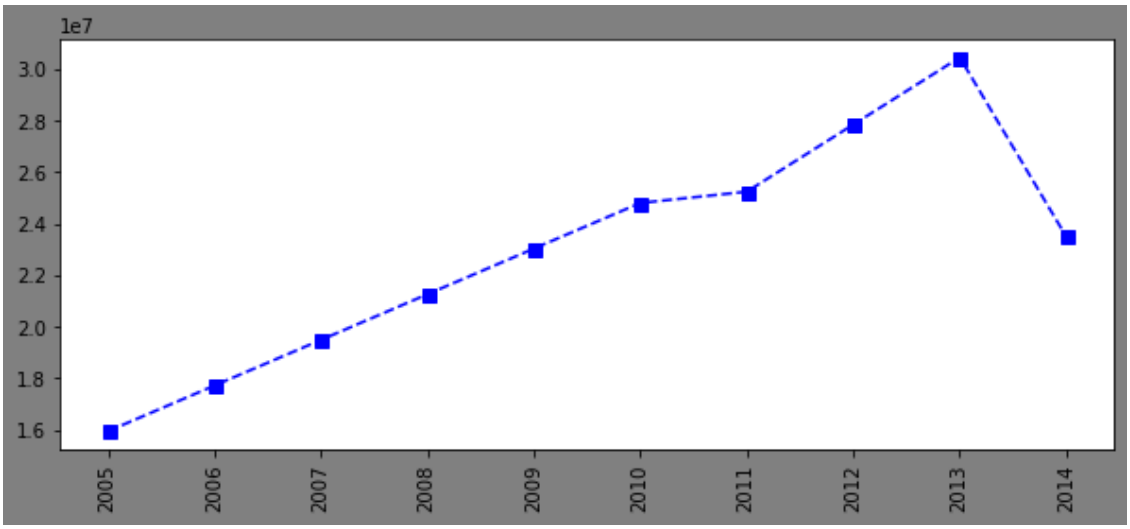
[62]:
```
%matplotlib inline
plt.rcParams['figure.figsize']=10,4
plt.rcParams['figure.facecolor']='Gray'
```

[63]:
```
plt.plot(Salary[0],c='Black',ls='--',marker='s',ms=7,label=Players[0])
plt.show()
```



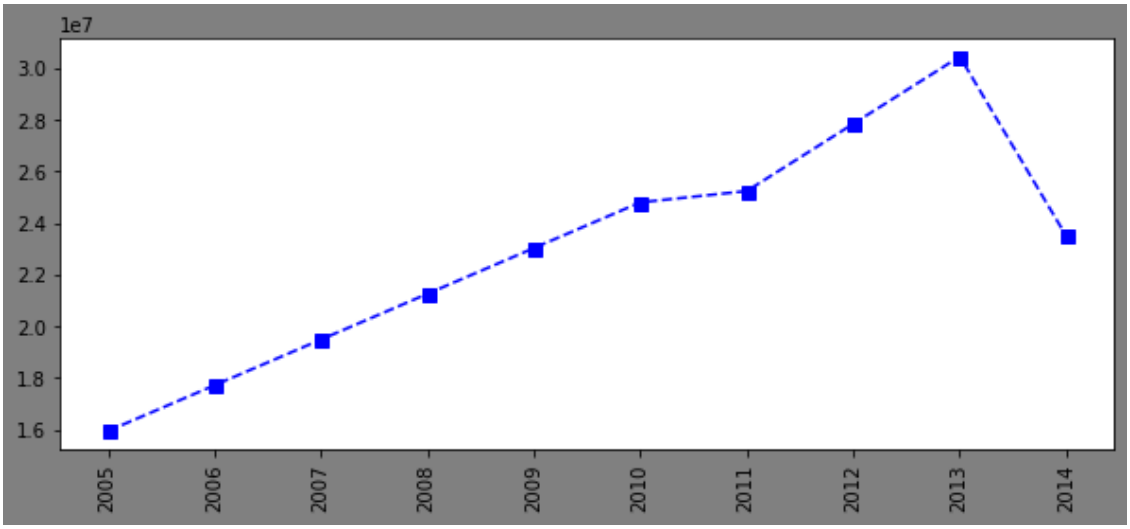
[64]:
```
plt.plot(Salary[0],c="Blue",ls='--',marker='s',ms=7,label=Players[0])
plt.xticks(list(range(0,10)),Seasons,rotation='vertical')
print(Players[0])
plt.show()
```

```
KobeBryant
```

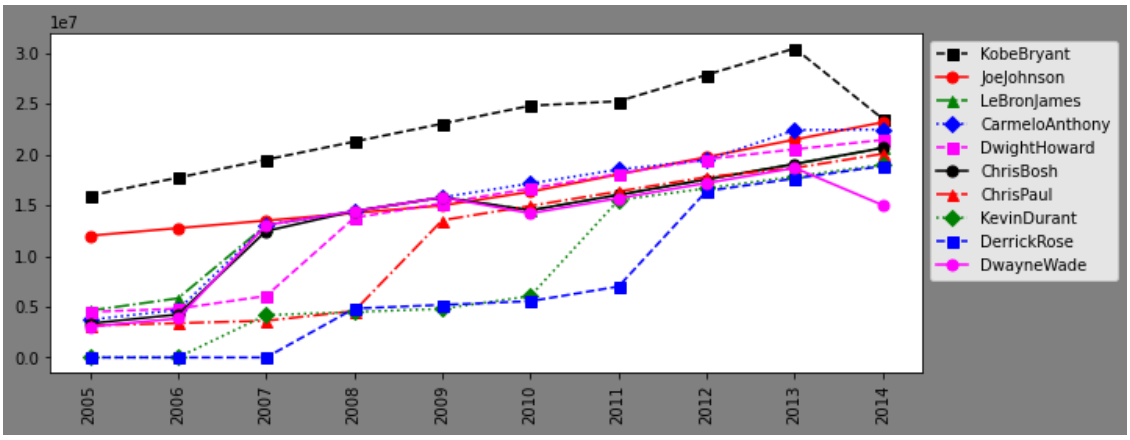


[65]:
```
plt.plot(Salary[0],c="Blue",ls='--',marker='s',ms=7,label=Players[0])
plt.xticks(list(range(0,10)),Seasons,rotation='vertical')
print(Players[0])
plt.show()
```

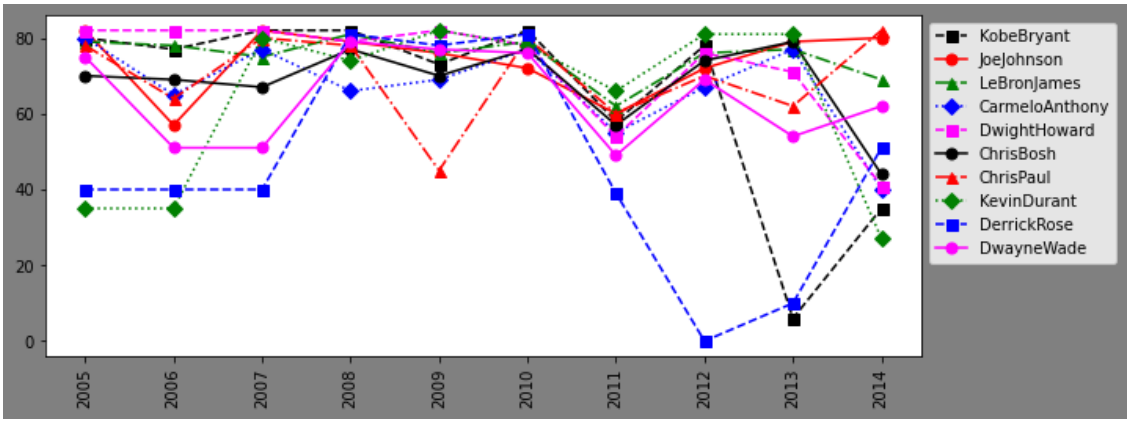
```
KobeBryant
```



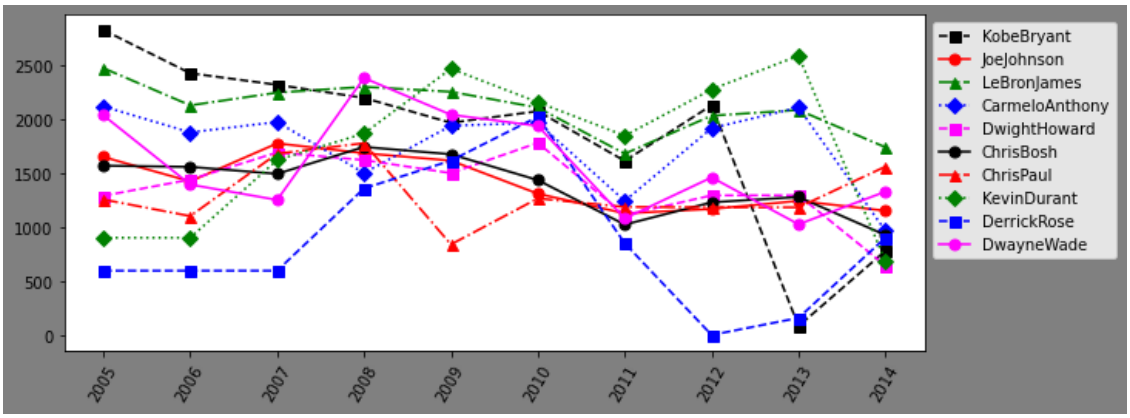
[66]:
```
plt.plot(Salary[0],c="Black",ls='--',marker='s',ms=7,label=Players[0])
plt.plot(Salary[1],c="Red",ls='-',marker='o',ms=7,label=Players[1])
plt.plot(Salary[2],c="Green",ls='-.',marker='^',ms=7,label=Players[2])
plt.plot(Salary[3],c="Blue",ls=':',marker='D',ms=7,label=Players[3])
plt.plot(Salary[4],c="Magenta",ls='--',marker='s',ms=7,label=Players[4])
plt.plot(Salary[5],c="Black",ls='-',marker='o',ms=7,label=Players[5])
plt.plot(Salary[6],c="Red",ls='-.',marker='^',ms=7,label=Players[6])
plt.plot(Salary[7],c="Green",ls=':',marker='D',ms=7,label=Players[7])
plt.plot(Salary[8],c="Blue",ls='--',marker='s',ms=7,label=Players[8])
plt.plot(Salary[9],c="Magenta",ls='-',marker='o',ms=7,label=Players[9])

plt.xticks(list(range(0,10)),Seasons,rotation='vertical')# rotation='vertical'
plt.legend(loc='upper left',bbox_to_anchor=(1,1))
plt.show()
```



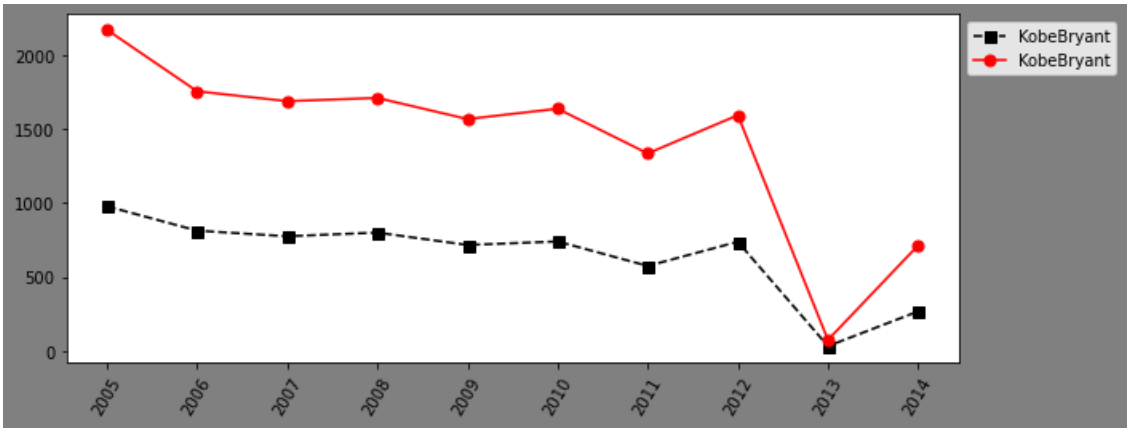
[67]:
```
plt.plot(Games[0],c="Black",ls='--',marker='s',ms=7,label=Players[0])
plt.plot(Games[1],c="Red",ls='-',marker='o',ms=7,label=Players[1])
plt.plot(Games[2],c="Green",ls='-.',marker='^',ms=7,label=Players[2])
plt.plot(Games[3],c="Blue",ls=':',marker='D',ms=7,label=Players[3])
plt.plot(Games[4],c="Magenta",ls='--',marker='s',ms=7,label=Players[4])
plt.plot(Games[5],c="Black",ls='-',marker='o',ms=7,label=Players[5])
plt.plot(Games[6],c="Red",ls='-.',marker='^',ms=7,label=Players[6])
plt.plot(Games[7],c="Green",ls=':',marker='D',ms=7,label=Players[7])
plt.plot(Games[8],c="Blue",ls='--',marker='s',ms=7,label=Players[8])
plt.plot(Games[9],c="Magenta",ls='-',marker='o',ms=7,label=Players[9])

plt.xticks(list(range(0,10)),Seasons,rotation='vertical')# rotation='vertical'
plt.legend(loc='upper left',bbox_to_anchor=(1,1))
plt.show()
```



[68]:
```
plt.plot(Points[0],c="Black",ls='--',marker='s',ms=7,label=Players[0])
plt.plot(Points[1],c="Red",ls='-',marker='o',ms=7,label=Players[1])
plt.plot(Points[2],c="Green",ls='-.',marker='^',ms=7,label=Players[2])
plt.plot(Points[3],c="Blue",ls=':',marker='D',ms=7,label=Players[3])
plt.plot(Points[4],c="Magenta",ls='--',marker='s',ms=7,label=Players[4])
plt.plot(Points[5],c="Black",ls='-',marker='o',ms=7,label=Players[5])
plt.plot(Points[6],c="Red",ls='-.',marker='^',ms=7,label=Players[6])
plt.plot(Points[7],c="Green",ls=':',marker='D',ms=7,label=Players[7])
plt.plot(Points[8],c="Blue",ls='--',marker='s',ms=7,label=Players[8])
plt.plot(Points[9],c="Magenta",ls='-',marker='o',ms=7,label=Players[9])

plt.xticks(list(range(0,10)),Seasons,rotation=60)# rotation='vertical'
plt.legend(loc='upper left',bbox_to_anchor=(1,1))
plt.show()
```



[69]:
```
plt.plot(FieldGoals[0],c="Black",ls='--',marker='s',ms=7,label=Players[0])
plt.plot(FieldGoalAttempts[0],c="Red",ls='-',marker='o',ms=7,label=Players[0])

plt.xticks(list(range(0,10)),Seasons,rotation=60)# rotation='vertical'
plt.legend(loc='upper left',bbox_to_anchor=(1,1))
plt.show()
```

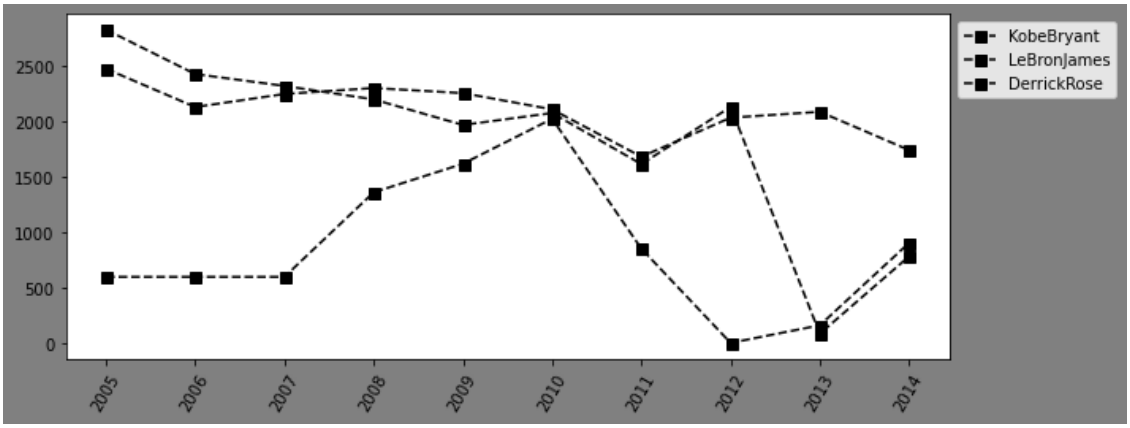


24 Functions Design

[70]:
```
def myplot(PlayersList):
    for name in PlayersList:
        plt.
↪plot(Points[Pdict[name]],c="Black",ls='--',marker='s',ms=7,label=Players[Pdict[name]])

    plt.xticks(list(range(0,10)),Seasons,rotation=60)# rotation='vertical'
    plt.legend(loc='upper left',bbox_to_anchor=(1,1))
```

```
    plt.show()
```

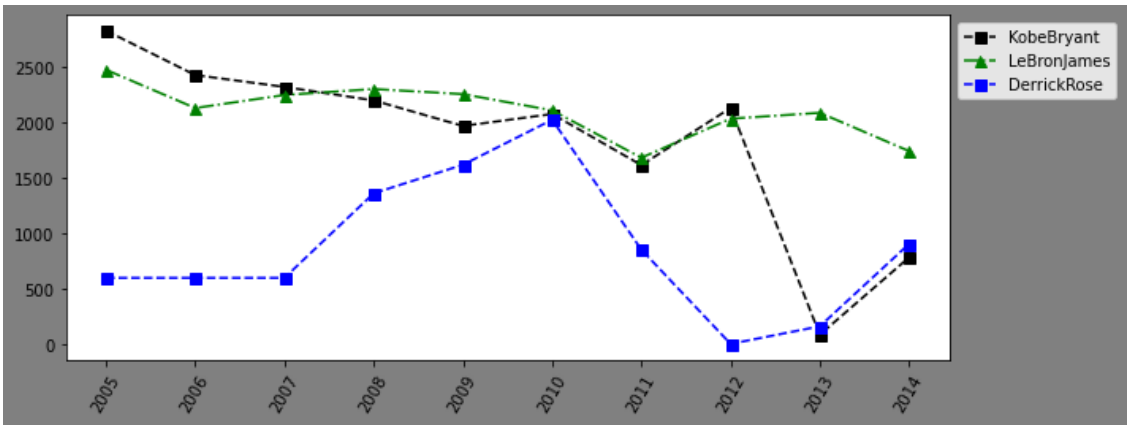
[71]:
```
myplot(["KobeBryant","LeBronJames","DerrickRose"])
```



25 Advanced Functions

[72]:
```
def myplot(PlayersList):
    Col={"KobeBryant":"Black","JoeJohnson":"Red","LeBronJames":
↪"Green","CarmeloAnthony":"Blue","DwightHoward":"Magenta",
         "ChrisBosh":"Black","ChrisPaul":"Red","KevinDurant":
↪"Green","DerrickRose":"Blue","DwayneWade":"Magenta"}

    Mkr={"KobeBryant":"s","JoeJohnson":"o","LeBronJames":"^","CarmeloAnthony":
↪"D","DwightHoward":"s",
         "ChrisBosh":"o","ChrisPaul":"^","KevinDurant":"D","DerrickRose":
↪"s","DwayneWade":"o"}

    Ls={"KobeBryant":'--',"JoeJohnson":'-',"LeBronJames":'-.',"CarmeloAnthony":
↪':',"DwightHoward":'--',
         "ChrisBosh":'-',"ChrisPaul":'-.',"KevinDurant":':',"DerrickRose":
↪'--',"DwayneWade":'-'}

    for name in PlayersList:
        plt.
↪plot(Points[Pdict[name]],c=Col[name],ls=Ls[name],marker=Mkr[name],ms=7,label=Players[Pdict[

    plt.xticks(list(range(0,10)),Seasons,rotation=60)# rotation='vertical'
    plt.legend(loc='upper left',bbox_to_anchor=(1,1))
    plt.show()
```

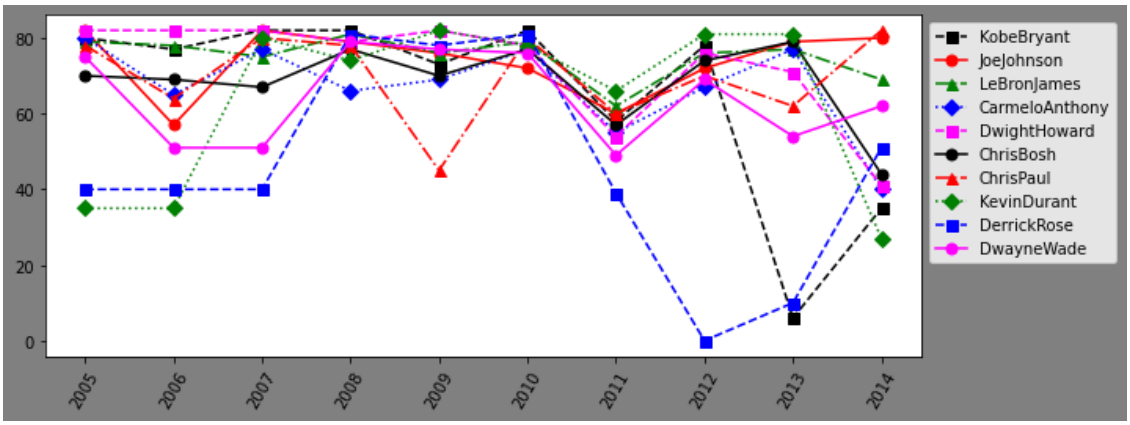
[73]:
```
myplot(["KobeBryant","LeBronJames","DerrickRose"])
```

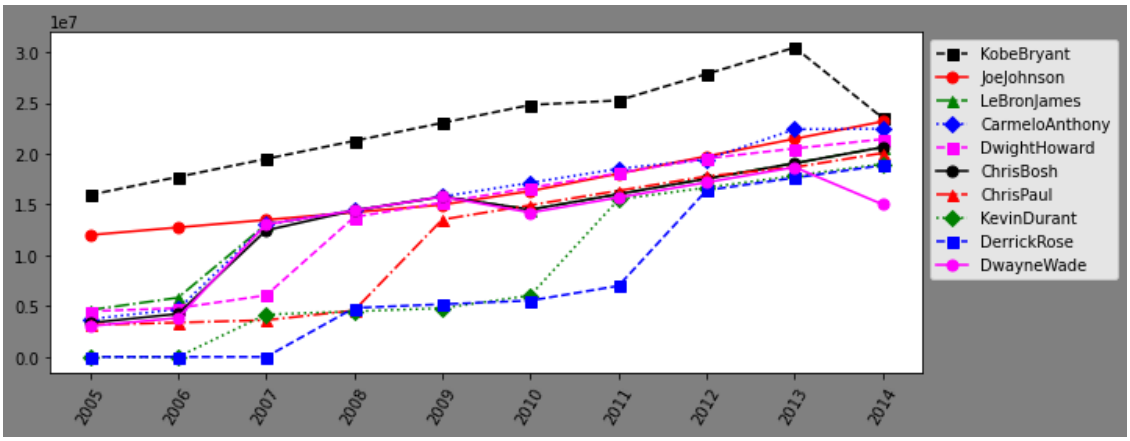


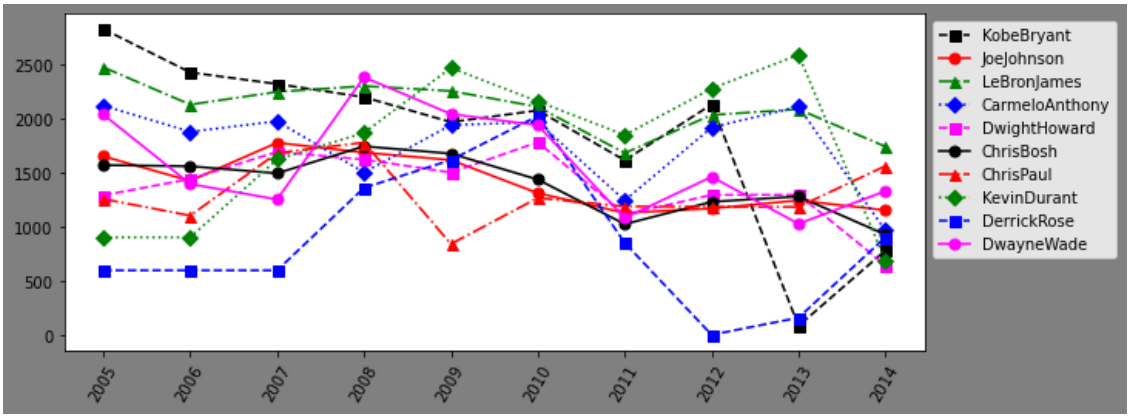
26 Fix up the Input

[74]:
```
def myplot(data, PlayersList=Players):
    Col={"KobeBryant":"Black","JoeJohnson":"Red","LeBronJames":
↪"Green","CarmeloAnthony":"Blue","DwightHoward":"Magenta",
         "ChrisBosh":"Black","ChrisPaul":"Red","KevinDurant":
↪"Green","DerrickRose":"Blue","DwayneWade":"Magenta"}

    Mkr={"KobeBryant":"s","JoeJohnson":"o","LeBronJames":"^","CarmeloAnthony":
↪"D","DwightHoward":"s",
         "ChrisBosh":"o","ChrisPaul":"^","KevinDurant":"D","DerrickRose":
↪"s","DwayneWade":"o"}

    Ls={"KobeBryant":'--',"JoeJohnson":'-',"LeBronJames":'-.',"CarmeloAnthony":
↪':',"DwightHoward":'--',
         "ChrisBosh":'-',"ChrisPaul":'-.',"KevinDurant":':',"DerrickRose":
↪'--',"DwayneWade":'-'}

    for name in PlayersList:
        plt.
↪plot(data[Pdict[name]],c=Col[name],ls=Ls[name],marker=Mkr[name],ms=7,label=Players[Pdict[na

    plt.xticks(list(range(0,10)),Seasons,rotation=60)# rotation='vertical'
    plt.legend(loc='upper left',bbox_to_anchor=(1,1))
    plt.show()
```

[75]:
```
myplot(Games)
myplot(Salary)
myplot(Points)
```

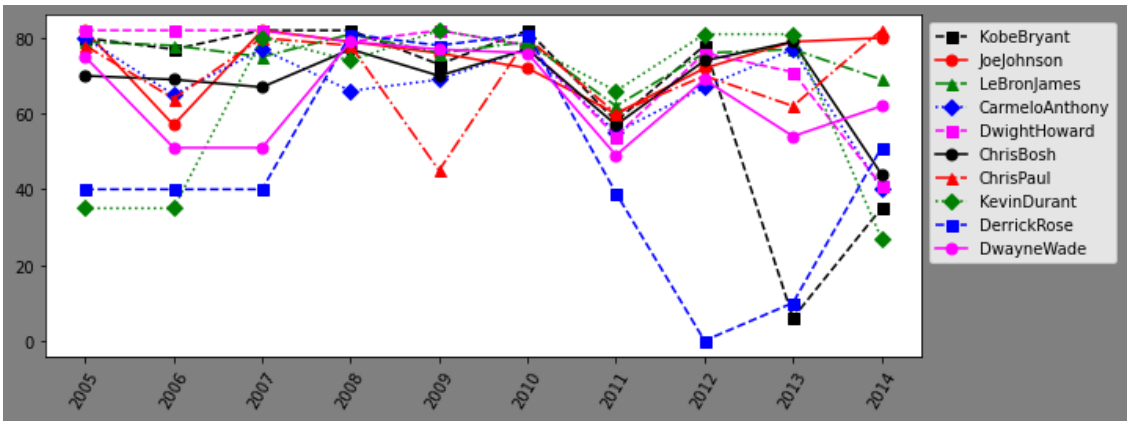

KobeBryant
JoeJohnson
LeBronJames
CarmeloAnthony
DwightHoward
ChrisBosh
ChrisPaul
KevinDurant
DerrickRose
DwayneWade
0
20
40
60
80
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014


1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
KobeBryant
JoeJohnson
LeBronJames
CarmeloAnthony
DwightHoward
ChrisBosh
ChrisPaul
KevinDurant
DerrickRose
DwayneWade
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014


2500
2000
1500
1000
500
0
KobeBryant
JoeJohnson
LeBronJames
CarmeloAnthony
DwightHoward
ChrisBosh
ChrisPaul
KevinDurant
DerrickRose
DwayneWade
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014

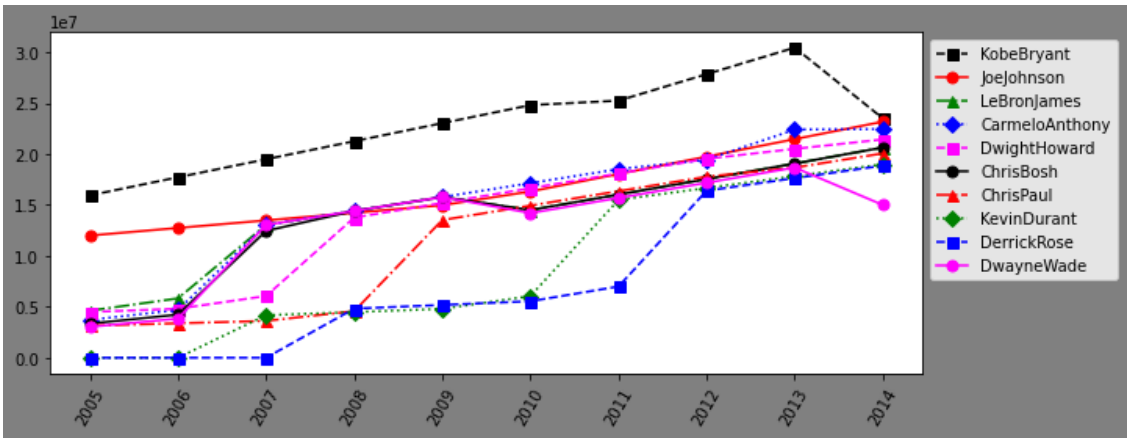
27 Insights

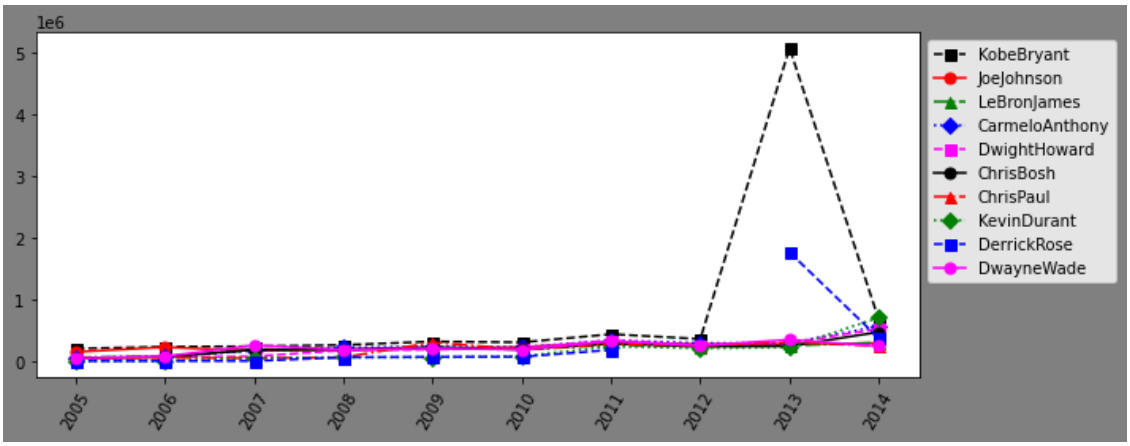
[76]:
```
myplot(Games)
```

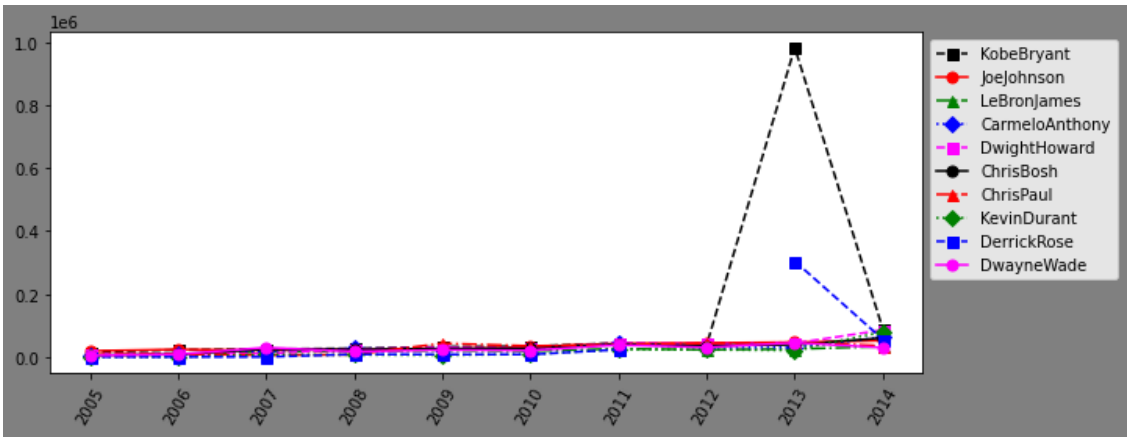


28 Salary

[77]:
```
myplot(Salary)
myplot(Salary/Games)
myplot(Salary/FieldGoals)
```

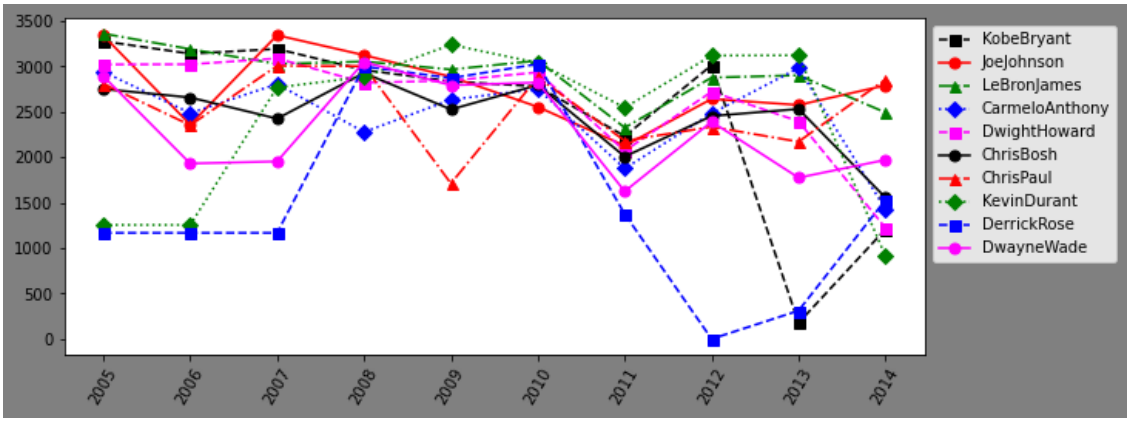


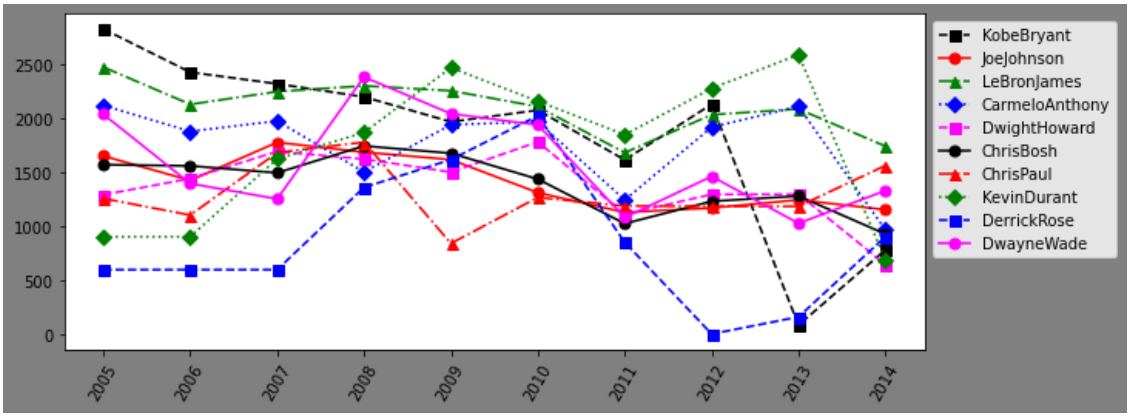




29 In Games Matrics

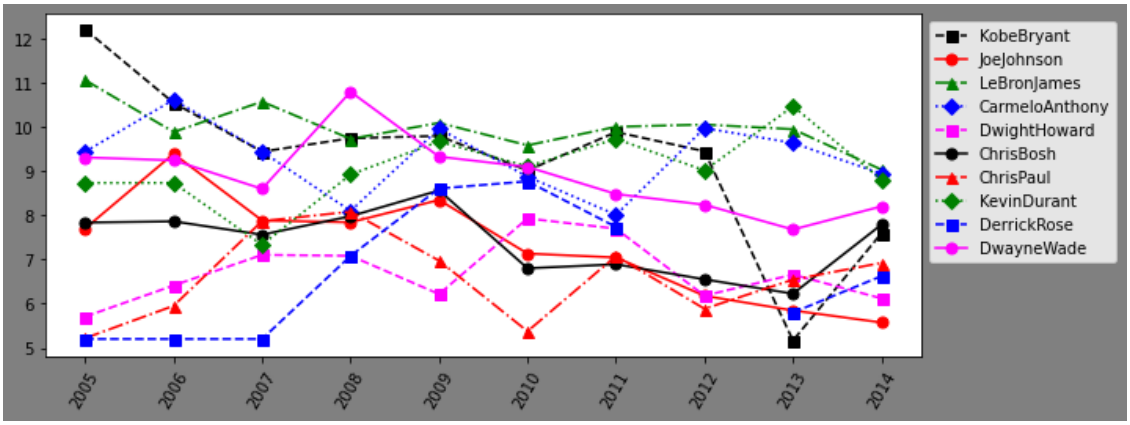
[78]:
```
myplot(MinutesPlayed)
myplot(Points)
```

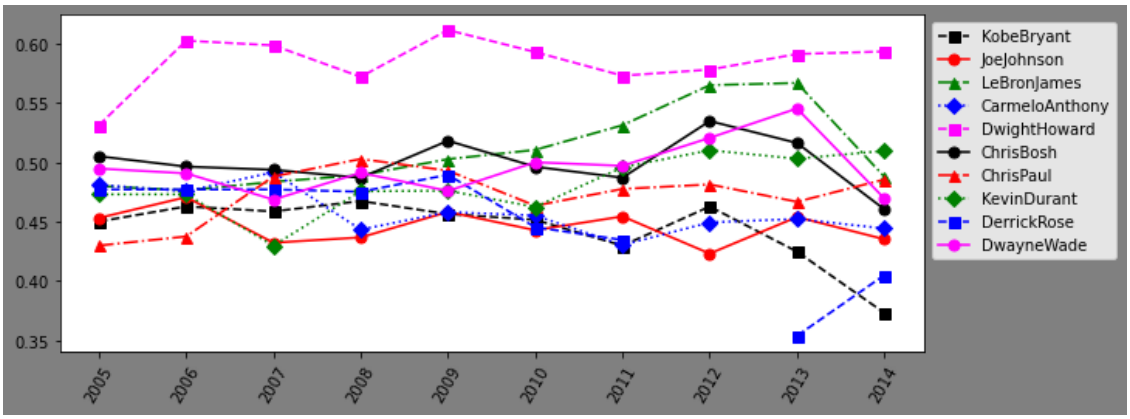


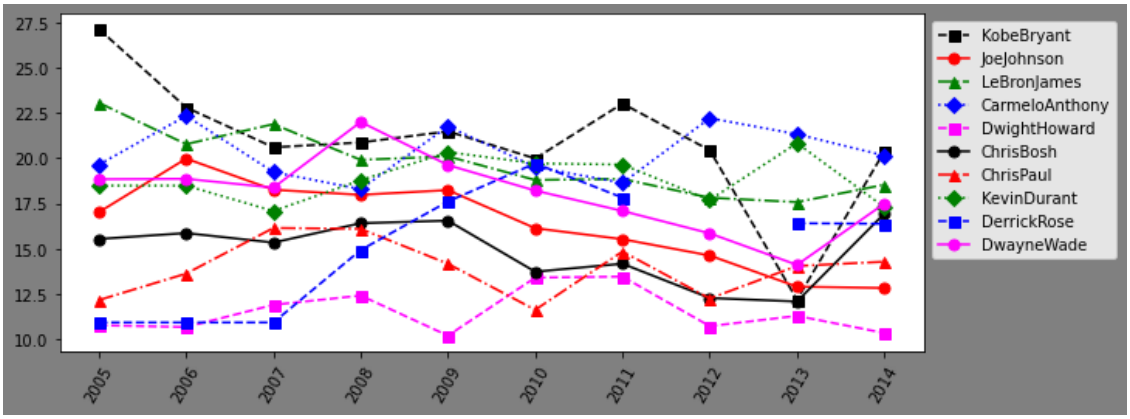


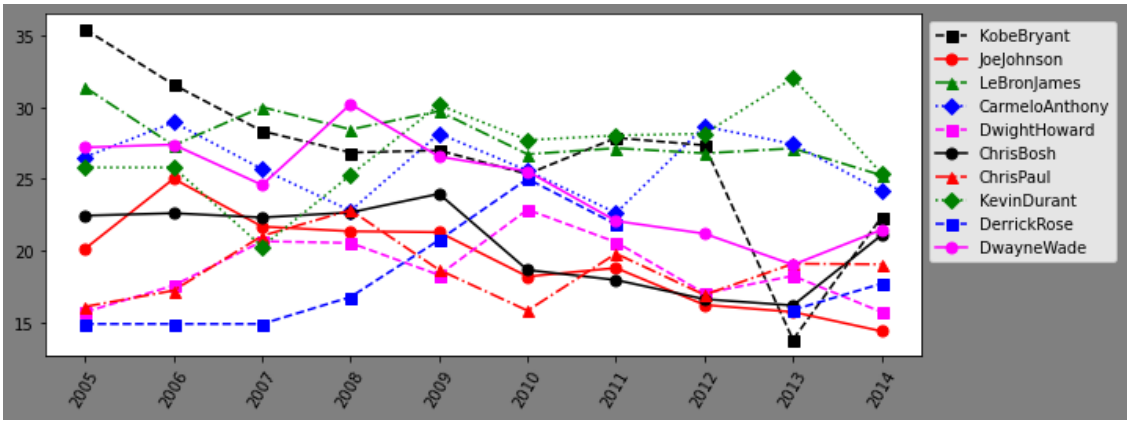
30 In Games Matrics Normalized

[79]:
```
myplot(FieldGoals/Games)
myplot(FieldGoals/FieldGoalAttempts)
myplot(FieldGoalAttempts/Games)
myplot(Points/Games)
```



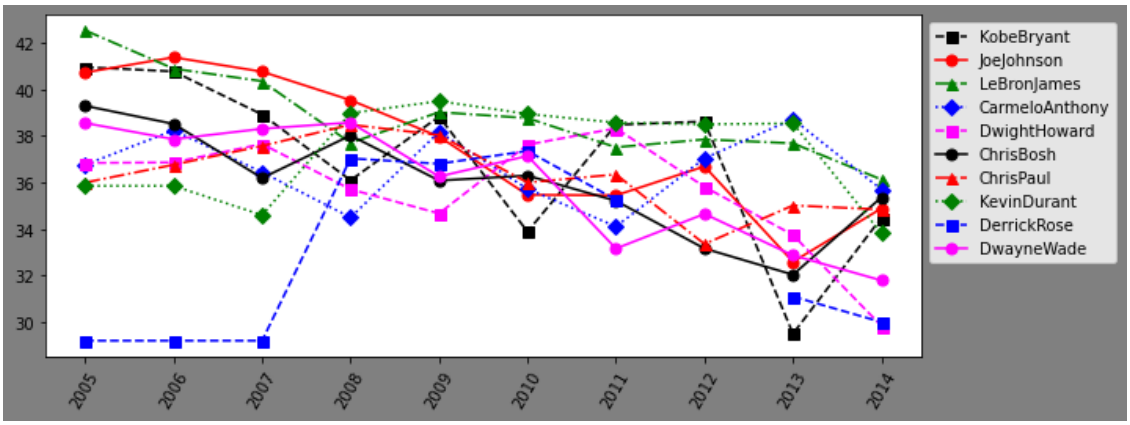

0.60
0.55
0.50
0.45
0.40
0.35
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
KobeBryant
JoeJohnson
LeBronJames
CarmeloAnthony
DwightHoward
ChrisBosh
ChrisPaul
KevinDurant
DerrickRose
DwayneWade

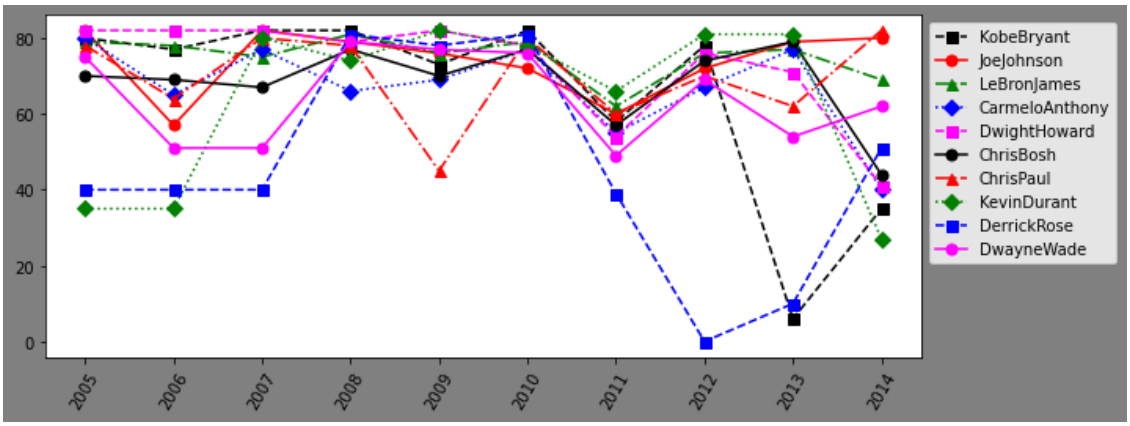

27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
KobeBryant
JoeJohnson
LeBronJames
CarmeloAnthony
DwightHoward
ChrisBosh
ChrisPaul
KevinDurant
DerrickRose
DwayneWade


35
30
25
20
15
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
KobeBryant
JoeJohnson
LeBronJames
CarmeloAnthony
DwightHoward
ChrisBosh
ChrisPaul
KevinDurant
DerrickRose
DwayneWade

31 Interesting Observations

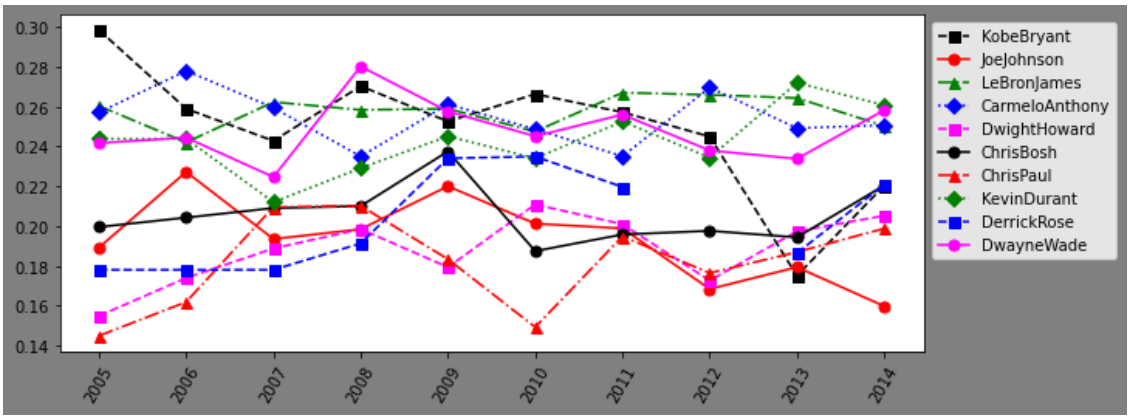
[80]:
```
myplot(MinutesPlayed/Games)
myplot(Games)
```





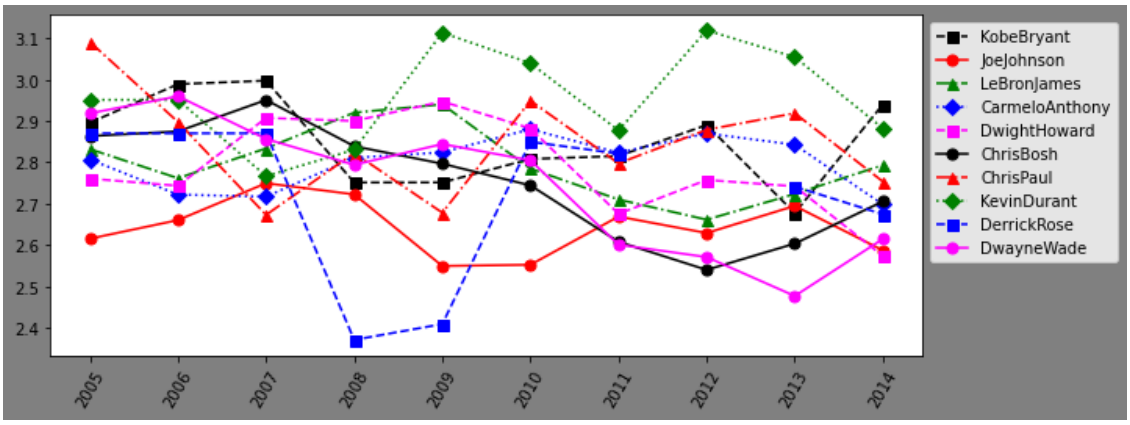
32 Time is Valuable

[81]:
```
myplot(FieldGoals/MinutesPlayed)
```



33 Players Style

[82]:
```
myplot(Points/FieldGoals)
```



---

34 Data Frames setting(working on data)

[4]:
```
import pandas as pd
```

35 Method 1: Specify full path

[9]:
```
stats=pd.read_csv('C:\\Users\\ddaya\\OneDrive\\Documents\\Python programming\\
                   P1-OfficeSupplies.csv')
```

[10]:
```
stats
```

[10]:

	OrderDate	Region	Rep	Item	Units	Unit Price
0	4-Jul-14	East	Richard	Pen Set	62	4.99
1	12-Jul-14	East	Nick	Binder	29	1.99
2	21-Jul-14	Central	Morgan	Pen Set	55	12.49
3	29-Jul-14	East	Susan	Binder	81	19.99
4	7-Aug-14	Central	Matthew	Pen Set	42	23.95
5	15-Aug-14	East	Richard	Pencil	35	4.99
6	24-Aug-14	West	James	Desk	3	275.00
7	1-Sep-14	Central	Smith	Desk	2	125.00
8	10-Sep-14	Central	Bill	Pencil	7	1.29
9	18-Sep-14	East	Richard	Pen Set	16	15.99
10	27-Sep-14	West	James	Pen	76	1.99
11	5-Oct-14	Central	Morgan	Binder	28	8.99
12	14-Oct-14	West	Thomas	Binder	57	19.99
13	22-Oct-14	East	Richard	Pen	64	8.99
14	31-Oct-14	Central	Rachel	Pencil	14	1.29
15	8-Nov-14	East	Susan	Pen	15	19.99
16	17-Nov-14	Central	Alex	Binder	11	4.99
17	25-Nov-14	Central	Matthew	Pen Set	96	4.99
18	4-Dec-14	Central	Alex	Binder	94	19.99
19	12-Dec-14	Central	Smith	Pencil	67	1.29
20	21-Dec-14	Central	Rachel	Binder	28	4.99
21	29-Dec-14	East	Susan	Pen Set	74	15.99
22	6-Jan-15	East	Richard	Pencil	95	1.99
23	15-Jan-15	Central	Bill	Binder	46	8.99
24	23-Jan-15	Central	Matthew	Binder	50	19.99
25	1-Feb-15	Central	Smith	Binder	87	15.00
26	9-Feb-15	Central	Alex	Pencil	36	4.99
27	18-Feb-15	East	Richard	Binder	4	4.99
28	26-Feb-15	Central	Bill	Pen	27	19.99
29	7-Mar-15	West	James	Binder	7	19.99
30	15-Mar-15	West	James	Pencil	56	2.99
31	24-Mar-15	Central	Alex	Pen Set	50	4.99
32	1-Apr-15	East	Richard	Binder	60	4.99
33	10-Apr-15	Central	Rachel	Pencil	66	1.99
34	18-Apr-15	Central	Rachel	Pencil	75	1.99
35	27-Apr-15	East	Nick	Pen	96	4.99
36	5-May-15	Central	Alex	Pencil	90	4.99
37	14-May-15	Central	Bill	Pencil	53	1.29
38	22-May-15	West	Thomas	Pencil	32	1.99
39	31-May-15	Central	Bill	Binder	80	8.99
40	8-Jun-15	East	Richard	Binder	60	8.99
41	17-Jun-15	Central	Matthew	Desk	5	125.00
42	25-Jun-15	Central	Morgan	Pencil	90	4.99

36 Method 2:Change the working Directory

[11]:
```
import os
```

[12]:
```
print(os.getcwd())
```

```
C:\Users\ddaya\Documents\Python Programs
```

[13]:
```
os.chdir('C:\\Users\\ddaya\\OneDrive\\Documents\\Python programming')
```

[14]:
```
print(os.getcwd())
```

```
C:\Users\ddaya\OneDrive\Documents\Python programming
```

[15]:
```
stats=pd.read_csv('P1-OfficeSupplies.csv')
```

[16]:
```
stats
```

[16]:

	OrderDate	Region	Rep	Item	Units	Unit Price
0	4-Jul-14	East	Richard	Pen Set	62	4.99
1	12-Jul-14	East	Nick	Binder	29	1.99
2	21-Jul-14	Central	Morgan	Pen Set	55	12.49
3	29-Jul-14	East	Susan	Binder	81	19.99
4	7-Aug-14	Central	Matthew	Pen Set	42	23.95
5	15-Aug-14	East	Richard	Pencil	35	4.99
6	24-Aug-14	West	James	Desk	3	275.00
7	1-Sep-14	Central	Smith	Desk	2	125.00
8	10-Sep-14	Central	Bill	Pencil	7	1.29
9	18-Sep-14	East	Richard	Pen Set	16	15.99
10	27-Sep-14	West	James	Pen	76	1.99
11	5-Oct-14	Central	Morgan	Binder	28	8.99
12	14-Oct-14	West	Thomas	Binder	57	19.99
13	22-Oct-14	East	Richard	Pen	64	8.99
14	31-Oct-14	Central	Rachel	Pencil	14	1.29
15	8-Nov-14	East	Susan	Pen	15	19.99
16	17-Nov-14	Central	Alex	Binder	11	4.99
17	25-Nov-14	Central	Matthew	Pen Set	96	4.99
18	4-Dec-14	Central	Alex	Binder	94	19.99
19	12-Dec-14	Central	Smith	Pencil	67	1.29
20	21-Dec-14	Central	Rachel	Binder	28	4.99
21	29-Dec-14	East	Susan	Pen Set	74	15.99
22	6-Jan-15	East	Richard	Pencil	95	1.99
23	15-Jan-15	Central	Bill	Binder	46	8.99
24	23-Jan-15	Central	Matthew	Binder	50	19.99
25	1-Feb-15	Central	Smith	Binder	87	15.00
26	9-Feb-15	Central	Alex	Pencil	36	4.99
27	18-Feb-15	East	Richard	Binder	4	4.99
28	26-Feb-15	Central	Bill	Pen	27	19.99

```
29   7-Mar-15     West    James   Binder      7       19.99
30  15-Mar-15     West    James   Pencil     56        2.99
31  24-Mar-15  Central     Alex  Pen Set     50        4.99
32   1-Apr-15     East  Richard   Binder     60        4.99
33  10-Apr-15  Central   Rachel   Pencil     66        1.99
34  18-Apr-15  Central   Rachel   Pencil     75        1.99
35  27-Apr-15     East     Nick      Pen     96        4.99
36   5-May-15  Central     Alex   Pencil     90        4.99
37  14-May-15  Central     Bill   Pencil     53        1.29
38  22-May-15     West   Thomas   Pencil     32        1.99
39  31-May-15  Central     Bill   Binder     80        8.99
40   8-Jun-15     East  Richard   Binder     60        8.99
41  17-Jun-15  Central  Matthew     Desk      5      125.00
42  25-Jun-15  Central   Morgan   Pencil     90        4.99
```

[17]:
```
stats=pd.read_csv('P1-UK-Bank-Customers.csv')
```

[18]:
```
stats
```

[18]:
```
      Customer ID       Name   Surname  Gender  Age            Region  \
0       100000001      Simon     Walsh    Male   21           England
1       400000002    Jasmine    Miller  Female   34  Northern Ireland
2       100000003       Liam     Brown    Male   46           England
3       300000004     Trevor      Parr    Male   32             Wales
4       100000005    Deirdre   Pullman  Female   38           England
…             …          …         …       …  …                 …
4009    200004010        Sam     Lewis    Male   64          Scotland
4010    200004011      Keith    Hughes    Male   52          Scotland
4011    200004012     Hannah  Springer  Female   50          Scotland
4012    200004013  Christian      Reid    Male   51          Scotland
4013    300004014    Stephen       May    Male   33             Wales

     Job Classification Date Joined    Balance
0          White Collar   05.Jan.15  113810.15
1           Blue Collar   06.Jan.15   36919.73
2          White Collar   07.Jan.15  101536.83
3          White Collar   08.Jan.15    1421.52
4           Blue Collar   09.Jan.15   35639.79
…                   …          …          …
4009              Other   30.Dec.15   19711.66
4010        Blue Collar   30.Dec.15   56069.72
4011              Other   30.Dec.15   59477.82
4012        Blue Collar   30.Dec.15     239.45
4013        Blue Collar   30.Dec.15   30293.19

[4014 rows x 9 columns]
```

37 1. Full Data

[19]:
```
stats
```

[19]:
```
      Customer ID       Name   Surname  Gender  Age            Region  \
0       100000001      Simon     Walsh    Male   21           England
1       400000002    Jasmine    Miller  Female   34  Northern Ireland
2       100000003       Liam     Brown    Male   46           England
3       300000004     Trevor      Parr    Male   32             Wales
4       100000005    Deirdre   Pullman  Female   38           England
…             …          …         …       …  …                 …
4009    200004010        Sam     Lewis    Male   64          Scotland
4010    200004011      Keith    Hughes    Male   52          Scotland
4011    200004012     Hannah  Springer  Female   50          Scotland
4012    200004013  Christian      Reid    Male   51          Scotland
4013    300004014    Stephen       May    Male   33             Wales

     Job Classification Date Joined    Balance
0          White Collar   05.Jan.15  113810.15
1           Blue Collar   06.Jan.15   36919.73
2          White Collar   07.Jan.15  101536.83
3          White Collar   08.Jan.15    1421.52
4           Blue Collar   09.Jan.15   35639.79
…                   …          …          …
4009              Other   30.Dec.15   19711.66
4010        Blue Collar   30.Dec.15   56069.72
4011              Other   30.Dec.15   59477.82
4012        Blue Collar   30.Dec.15     239.45
4013        Blue Collar   30.Dec.15   30293.19

[4014 rows x 9 columns]
```

38 2. Number of rows

[20]:
```
len(stats)
```

[20]:
```
4014
```

39 3. See Columns

[21]:
```
stats.columns
```

[21]:
```
Index(['Customer ID', 'Name', 'Surname', 'Gender', 'Age', 'Region',
       'Job Classification', 'Date Joined', 'Balance'],
      dtype='object')
```

40 4. Number of Columns

[22]:
```
len(stats.columns)
```

[22]:
```
9
```

41 5. Top Rows

[23]:
```
stats.head(6) # Remember the brackets
```

[23]:
```
   Customer ID     Name  Surname  Gender  Age            Region  \
0    100000001    Simon    Walsh    Male   21           England
1    400000002  Jasmine   Miller  Female   34  Northern Ireland
2    100000003     Liam    Brown    Male   46           England
3    300000004   Trevor     Parr    Male   32             Wales
4    100000005  Deirdre  Pullman  Female   38           England
5    300000006      Ava  Coleman  Female   30             Wales

  Job Classification Date Joined    Balance
0       White Collar   05.Jan.15  113810.15
1        Blue Collar   06.Jan.15   36919.73
2       White Collar   07.Jan.15  101536.83
3       White Collar   08.Jan.15    1421.52
4        Blue Collar   09.Jan.15   35639.79
5        Blue Collar   09.Jan.15  122443.77
```

42 6. Bottom Rows

[24]:
```
stats.tail() # Or stats.tail(10)
```

[24]:
```
      Customer ID       Name   Surname  Gender  Age    Region  \
4009    200004010        Sam     Lewis    Male   64  Scotland
4010    200004011      Keith    Hughes    Male   52  Scotland
4011    200004012     Hannah  Springer  Female   50  Scotland
4012    200004013  Christian      Reid    Male   51  Scotland
4013    300004014    Stephen       May    Male   33     Wales

     Job Classification Date Joined   Balance
4009              Other   30.Dec.15  19711.66
4010        Blue Collar   30.Dec.15  56069.72
4011              Other   30.Dec.15  59477.82
4012        Blue Collar   30.Dec.15    239.45
4013        Blue Collar   30.Dec.15  30293.19
```

43 7. Information on the columns

[25]:
```
stats.info() #  Like the str function in R
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4014 entries, 0 to 4013
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Customer ID         4014 non-null   int64
 1   Name                4014 non-null   object
 2   Surname             4014 non-null   object
 3   Gender              4014 non-null   object
 4   Age                 4014 non-null   int64
 5   Region              4014 non-null   object
 6   Job Classification  4014 non-null   object
 7   Date Joined         4014 non-null   object
 8   Balance             4014 non-null   float64
dtypes: float64(1), int64(2), object(6)
memory usage: 282.4+ KB
```

[26]:
```
# 8. get stats on the columns
```

[27]:
```
stats.describe() # Like summary() in R
```

[27]:

	Customer ID	Age	Balance
count	4.014000e+03	4014.000000	4014.000000
mean	1.696831e+08	38.611111	39766.448274
std	8.865374e+07	9.819121	29859.489192
min	1.000000e+08	15.000000	11.520000
25%	1.000020e+08	31.000000	16115.367500
50%	1.000038e+08	37.000000	33567.330000
75%	2.000031e+08	45.000000	57533.930000
max	4.000038e+08	64.000000	183467.700000

[28]:
```
stats.describe().transpose()
```

[28]:

	count	mean	std	min	25%	50%	75%	max
Customer ID	4014.0	1.696831e+08	8.865374e+07	1.000000e+08	1.000020e+08	1.000038e+08	2.000031e+08	400003848.0
Age	4014.0	3.861111e+01	9.819121e+00	1.500000e+01	3.100000e+01	3.700000e+01	4.500000e+01	64.0
Balance	4014.0	3.976645e+04	2.985949e+04	1.152000e+01	1.611537e+04	3.356733e+04	5.753393e+04	183467.7

44 Renaming columns of a DataFrame

[29]:
```
stats.columns
```

[29]:
```
Index(['Customer ID', 'Name', 'Surname', 'Gender', 'Age', 'Region',
       'Job Classification', 'Date Joined', 'Balance'],
      dtype='object')
```

[30]:
```
stats.columns=['CustomerID', 'Name', 'Surname', 'Gender', 'Age', 'Region',
               'JobClassification', 'DateJoined', 'Balance']
```

[31]:
```
stats.head()
```

[31]:

	CustomerID	Name	Surname	Gender	Age	Region	JobClassification	DateJoined	Balance
0	100000001	Simon	Walsh	Male	21	England	White Collar	05.Jan.15	113810.15
1	400000002	Jasmine	Miller	Female	34	Northern Ireland	Blue Collar	06.Jan.15	36919.73
2	100000003	Liam	Brown	Male	46	England	White Collar	07.Jan.15	101536.83
3	300000004	Trevor	Parr	Male	32	Wales	White Collar	08.Jan.15	1421.52
4	100000005	Deirdre	Pullman	Female	38	England	Blue Collar	09.Jan.15	35639.79

45 Subsetting Data Frames in Pandas

46 Three Parts Buckle up:

47 - Rows

48 - Columns

49 - Combine the Both

[32]:
```
stats.head()
```

[32]:

	CustomerID	Name	Surname	Gender	Age	Region	JobClassification	DateJoined	Balance
0	100000001	Simon	Walsh	Male	21	England	White Collar	05.Jan.15	113810.15
1	400000002	Jasmine	Miller	Female	34	Northern Ireland	Blue Collar	06.Jan.15	36919.73
2	100000003	Liam	Brown	Male	46	England	White Collar	07.Jan.15	101536.83
3	300000004	Trevor	Parr	Male	32	Wales	White Collar	08.Jan.15	1421.52
4	100000005	Deirdre	Pullman	Female	38	England	Blue Collar	09.Jan.15	35639.79

```
  JobClassification DateJoined    Balance
0      White Collar  05.Jan.15  113810.15
1       Blue Collar  06.Jan.15   36919.73
2      White Collar  07.Jan.15  101536.83
3      White Collar  08.Jan.15    1421.52
4       Blue Collar  09.Jan.15   35639.79
```

50 Part 1. Rows

[33]:
```
stats[21:26]
```

[33]:
```
    CustomerID      Name   Surname  Gender  Age    Region JobClassification  \
21   200000022     Jason    Butler    Male   58  Scotland       Blue Collar
22   300000023   Deirdre  McDonald  Female   41     Wales      White Collar
23   200000024      Carl     Quinn    Male   52  Scotland       Blue Collar
24   100000025  Jennifer    Hughes  Female   38   England      White Collar
25   200000026   Richard    Fraser    Male   55  Scotland       Blue Collar

   DateJoined   Balance
21  18.Jan.15  21252.97
22  18.Jan.15  66785.78
23  19.Jan.15   6580.81
24  20.Jan.15  20505.32
25  21.Jan.15  43249.26
```

[34]:
```
stats[110:120]
```

[34]:
```
     CustomerID     Name    Surname  Gender  Age            Region  \
110   300000111    Sonia  Robertson  Female   25             Wales
111   300000112   Nathan   Paterson    Male   26             Wales
112   400000113      Tim   Hardacre    Male   29  Northern Ireland
113   400000114    Fiona      Mills  Female   18  Northern Ireland
114   400000115     Ruth     Oliver  Female   43  Northern Ireland
115   100000116   Alison   Johnston  Female   36           England
116   100000117      Amy    McGrath  Female   40           England
117   200000118     Adam    McGrath    Male   52          Scotland
118   100000119  Vanessa      Lyman  Female   18           England
119   100000120   Andrea    Dickens  Female   31           England

    JobClassification DateJoined    Balance
110             Other  16.Mar.15   70799.64
111      White Collar  16.Mar.15   20627.65
112      White Collar  16.Mar.15   82229.52
113             Other  16.Mar.15   51171.29
114       Blue Collar  16.Mar.15   37889.38
115             Other  19.Mar.15   21806.30
116             Other  25.Mar.15   12415.50
```

```
117             Other  28.Mar.15   67682.92
118      White Collar  31.Mar.15   33524.41
119      White Collar  31.Mar.15  136370.38
```

[35]:
```
stats[4010:]
```

[35]:
```
      CustomerID       Name   Surname  Gender  Age    Region  \
4010   200004011      Keith    Hughes    Male   52  Scotland
4011   200004012     Hannah  Springer  Female   50  Scotland
4012   200004013  Christian      Reid    Male   51  Scotland
4013   300004014    Stephen       May    Male   33     Wales

     JobClassification DateJoined   Balance
4010       Blue Collar  30.Dec.15  56069.72
4011             Other  30.Dec.15  59477.82
4012       Blue Collar  30.Dec.15    239.45
4013       Blue Collar  30.Dec.15  30293.19
```

[36]:
```
stats[:6] # Same as head()
```

[36]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England
5   300000006      Ava  Coleman  Female   30             Wales

  JobClassification DateJoined    Balance
0      White Collar  05.Jan.15  113810.15
1       Blue Collar  06.Jan.15   36919.73
2      White Collar  07.Jan.15  101536.83
3      White Collar  08.Jan.15    1421.52
4       Blue Collar  09.Jan.15   35639.79
5       Blue Collar  09.Jan.15  122443.77
```

51 Quick Exercise(resfersher)

52 1. Reverse The DataFrame

[37]:
```
stats[::-1]# Or stats[199:100:-1]
```

[37]:
```
      CustomerID       Name   Surname  Gender  Age            Region  \
4013   300004014    Stephen       May    Male   33             Wales
4012   200004013  Christian      Reid    Male   51          Scotland
4011   200004012     Hannah  Springer  Female   50          Scotland
4010   200004011      Keith    Hughes    Male   52          Scotland
```

```
4009   200004010        Sam     Lewis    Male   64          Scotland
…          …        …       …      …   …                …
4      100000005    Deirdre   Pullman  Female   38           England
3      300000004     Trevor      Parr    Male   32             Wales
2      100000003       Liam     Brown    Male   46           England
1      400000002    Jasmine    Miller  Female   34  Northern Ireland
0      100000001      Simon     Walsh    Male   21           England

     JobClassification DateJoined    Balance
4013       Blue Collar  30.Dec.15   30293.19
4012       Blue Collar  30.Dec.15     239.45
4011             Other  30.Dec.15   59477.82
4010       Blue Collar  30.Dec.15   56069.72
4009             Other  30.Dec.15   19711.66
…                …        …        …
4          Blue Collar  09.Jan.15   35639.79
3         White Collar  08.Jan.15    1421.52
2         White Collar  07.Jan.15  101536.83
1          Blue Collar  06.Jan.15   36919.73
0         White Collar  05.Jan.15  113810.15

[4014 rows x 9 columns]
```

53 2. Get only every 20th Rows

[38]:
```
stats[::20]
```

[38]:
```
      CustomerID       Name     Surname  Gender  Age    Region  \
0      100000001      Simon       Walsh    Male   21   England
20     300000021      Boris    Johnston    Male   37     Wales
40     100000041     Edward       Terry    Male   27   England
60     100000061      Kylie      Howard  Female   35   England
80     100000081       Joan    Buckland  Female   36   England
…          …        …          …      …   …        …
3920   100003921      Isaac    Buckland    Male   37   England
3940   200003941     Joshua  Sutherland    Male   59  Scotland
3960   100003961    Leonard       Grant    Male   48   England
3980   200003981  Elizabeth       James  Female   43  Scotland
4000   300004001  Gabrielle      Duncan  Female   34     Wales

     JobClassification DateJoined    Balance
0         White Collar  05.Jan.15  113810.15
20               Other  16.Jan.15   31778.90
40         Blue Collar  01.Feb.15   51412.60
60        White Collar  12.Feb.15    4586.23
80        White Collar  16.Mar.15   59935.75
```

```
…                 …         …         …
3920      White Collar  24.Dec.15   35743.13
3940             Other  25.Dec.15    2114.65
3960             Other  27.Dec.15   72061.71
3980             Other  28.Dec.15   19695.66
4000      White Collar  29.Dec.15   92083.79

[201 rows x 9 columns]
```

54 Part 2. Columns

[39]:
```
stats.columns
```

[39]:
```
Index(['CustomerID', 'Name', 'Surname', 'Gender', 'Age', 'Region',
       'JobClassification', 'DateJoined', 'Balance'],
      dtype='object')
```

[40]:
```
stats.head()
```

[40]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance
0      White Collar  05.Jan.15  113810.15
1       Blue Collar  06.Jan.15   36919.73
2      White Collar  07.Jan.15  101536.83
3      White Collar  08.Jan.15    1421.52
4       Blue Collar  09.Jan.15   35639.79
```

[41]:
```
stats['Name']
```

[41]:
```
0          Simon
1         Jasmine
2            Liam
3          Trevor
4         Deirdre
          …
4009          Sam
4010        Keith
4011       Hannah
4012    Christian
4013      Stephen
Name: Name, Length: 4014, dtype: object
```

[42]:
```
stats['Name'].head()
```

[42]:
```
0      Simon
1    Jasmine
2       Liam
3     Trevor
4    Deirdre
Name: Name, dtype: object
```

[43]:
```
stats[['Name','Surname']].head() # In R you would be passing a vector:
↪c('Name','Surname')
```

[43]:
```
       Name  Surname
0     Simon    Walsh
1  Jasmine   Miller
2     Liam    Brown
3   Trevor     Parr
4  Deirdre  Pullman
```

55 Quick Access requires the name to be one word(or Column)

[44]:
```
stats.Name.head() # or stats.Name
```

[44]:
```
0      Simon
1    Jasmine
2       Liam
3     Trevor
4    Deirdre
Name: Name, dtype: object
```

56 Part 3. Combining Both

[45]:
```
stats[4:8][['Name','Surname']]
```

[45]:
```
       Name  Surname
4  Deirdre  Pullman
5      Ava  Coleman
6  Dorothy  Thomson
7     Lisa     Knox
```

[46]:
```
stats[['Name','Surname']][4:8]  # df2=stats[['Name','Surname']]
                                # df2[4:8]
```

[46]:
```
       Name  Surname
4  Deirdre  Pullman
5      Ava  Coleman
```

```
6  Dorothy  Thomson
7     Lisa     Knox
```

[47]:
```
df2=stats[['Name','Surname']]
df2[4:8]
```

[47]:
```
       Name  Surname
4  Deirdre  Pullman
5      Ava  Coleman
6  Dorothy  Thomson
7     Lisa     Knox
```

57 Basic Operations with DataFrames

58 Mathematical Operation:

[48]:
```
stats.head()
```

[48]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance
0      White Collar  05.Jan.15  113810.15
1       Blue Collar  06.Jan.15   36919.73
2      White Collar  07.Jan.15  101536.83
3      White Collar  08.Jan.15    1421.52
4       Blue Collar  09.Jan.15   35639.79
```

[49]:
```
Result=stats['Balance*2']=stats.Balance*2
Result.head()
```

[49]:
```
0    227620.30
1     73839.46
2    203073.66
3      2843.04
4     71279.58
Name: Balance, dtype: float64
```

59 Add Column:

[50]:
```
stats['Balance*2']=stats.Balance*2
```

[51]:
```
stats.head()
```

[51]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance  Balance*2
0      White Collar  05.Jan.15  113810.15  227620.30
1       Blue Collar  06.Jan.15   36919.73   73839.46
2      White Collar  07.Jan.15  101536.83  203073.66
3      White Collar  08.Jan.15    1421.52    2843.04
4       Blue Collar  09.Jan.15   35639.79   71279.58
```

[52]:
```
# comparison to R
stats.['xyz']=[1,2,3,4,5]# Error No Recycling option
```

```
  File "<ipython-input-52-70bd73c3ba16>", line 2
    stats.['xyz']=[1,2,3,4,5]# Error No Recycling option
          ^
SyntaxError: invalid syntax
```

60 Removing a column

[53]:
```
stats.head()
```

[53]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance  Balance*2
0      White Collar  05.Jan.15  113810.15  227620.30
1       Blue Collar  06.Jan.15   36919.73   73839.46
2      White Collar  07.Jan.15  101536.83  203073.66
3      White Collar  08.Jan.15    1421.52    2843.04
4       Blue Collar  09.Jan.15   35639.79   71279.58
```

[54]:
```
stats.drop('Balance*2',1).head() # 1 is vertical and 0 is Horiz.
```

[54]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance
0      White Collar  05.Jan.15  113810.15
1       Blue Collar  06.Jan.15   36919.73
2      White Collar  07.Jan.15  101536.83
3      White Collar  08.Jan.15    1421.52
4       Blue Collar  09.Jan.15   35639.79
```

[55]:
```
stats.head()
```

[55]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance  Balance*2
0      White Collar  05.Jan.15  113810.15  227620.30
1       Blue Collar  06.Jan.15   36919.73   73839.46
2      White Collar  07.Jan.15  101536.83  203073.66
3      White Collar  08.Jan.15    1421.52    2843.04
4       Blue Collar  09.Jan.15   35639.79   71279.58
```

[56]:
```
stats=stats.drop('Balance*2',1) # Drop Permanently
```

[57]:
```
stats.head()
```

[57]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance
0      White Collar  05.Jan.15  113810.15
1       Blue Collar  06.Jan.15   36919.73
2      White Collar  07.Jan.15  101536.83
3      White Collar  08.Jan.15    1421.52
4       Blue Collar  09.Jan.15   35639.79
```

61 Filtering DataFrames

62 Filtering is about Rows

[58]:
```
stats.Age>70
```

[58]:
```
0       False
1       False
2       False
3       False
4       False
        …
4009    False
4010    False
4011    False
4012    False
4013    False
Name: Age, Length: 4014, dtype: bool
```

[59]:
```
Filter=stats.Age>63
```

[60]:
```
Filter
```

[60]:
```
0       False
1       False
2       False
3       False
4       False
        …
4009     True
4010    False
4011    False
4012    False
4013    False
Name: Age, Length: 4014, dtype: bool
```

[61]:
```
stats[Filter] # Conceptually this is just like R
```

[61]:

	CustomerID	Name	Surname	Gender	Age	Region	\
631	200000632	Nicholas	Allan	Male	64	Scotland	
841	200000842	Matt	Manning	Male	64	Scotland	
1498	200001499	Cameron	Ellison	Male	64	Scotland	
1586	200001587	Jake	Ellison	Male	64	Scotland	
1608	200001609	Yvonne	Dickens	Female	64	Scotland	
1639	200001640	Christopher	Underwood	Male	64	Scotland	
2040	200002041	Abigail	Fraser	Female	64	Scotland	
2055	200002056	Joshua	Carr	Male	64	Scotland	

```
2310   200002311        Kevin     Howard    Male   64  Scotland
2352   200002353      Leonard      Lyman    Male   64  Scotland
2752   200002753          Ian     Hunter    Male   64  Scotland
2772   200002773         Alan     Watson    Male   64  Scotland
3676   200003677      Anthony      Lewis    Male   64  Scotland
4008   200004009       Alison      Quinn  Female   64  Scotland
4009   200004010          Sam      Lewis    Male   64  Scotland

     JobClassification DateJoined    Balance
631         Blue Collar  22.May.15   15909.96
841         Blue Collar  10.Jun.15    3775.44
1498        Blue Collar  03.Aug.15   71106.33
1586              Other  11.Aug.15   14242.57
1608              Other  13.Aug.15   23522.36
1639        Blue Collar  16.Aug.15   47115.07
2040        Blue Collar  13.Sep.15   24729.53
2055              Other  14.Sep.15   14558.13
2310        Blue Collar  26.Sep.15   10325.52
2352        Blue Collar  28.Sep.15  139415.88
2752        Blue Collar  23.Oct.15   92921.99
2772        Blue Collar  24.Oct.15   24994.57
3676        Blue Collar  12.Dec.15   48456.48
4008              Other  30.Dec.15   73503.90
4009              Other  30.Dec.15   19711.66
```

63 Let‘s use in Practice now

[62]:
```
Filter2=stats.Balance<5000
```

[63]:
```
Filter2
```

[63]:
```
0       False
1       False
2       False
3        True
4       False
        …
4009    False
4010    False
4011    False
4012     True
4013    False
Name: Balance, Length: 4014, dtype: bool
```

[64]:
```
stats[Filter2]
```

[64]:

	CustomerID	Name	Surname	Gender	Age	Region	\
3	300000004	Trevor	Parr	Male	32	Wales	
14	300000015	Madeleine	Marshall	Female	36	Wales	
15	100000016	Nicholas	Newman	Male	42	England	
17	200000018	Samantha	Coleman	Female	42	Scotland	
26	400000027	Rachel	McGrath	Female	37	Northern Ireland	
…	…	…	…	…	…	…	
3947	100003948	Owen	Baker	Male	18	England	
3955	300003956	Jane	Duncan	Female	34	Wales	
3987	100003988	Theresa	Forsyth	Female	30	England	
4006	100004007	Rachel	Davies	Female	34	England	
4012	200004013	Christian	Reid	Male	51	Scotland	

	JobClassification	DateJoined	Balance
3	White Collar	08.Jan.15	1421.52
14	Other	12.Jan.15	2846.03
15	White Collar	14.Jan.15	2116.85
17	Other	14.Jan.15	3801.69
26	White Collar	23.Jan.15	3967.20
…	…	…	…
3947	Blue Collar	26.Dec.15	3858.90
3955	Blue Collar	26.Dec.15	4478.46
3987	White Collar	29.Dec.15	4570.98
4006	Blue Collar	30.Dec.15	4561.22
4012	Blue Collar	30.Dec.15	239.45

[324 rows x 9 columns]

[65]:
```
stats[stats.Balance<100]
```

[65]:

	CustomerID	Name	Surname	Gender	Age	Region	\
74	400000075	Olivia	Dowd	Female	30	Northern Ireland	
774	200000775	Warren	Roberts	Male	37	Scotland	
1319	100001320	Jane	King	Female	38	England	
2045	400002046	Megan	Hart	Female	19	Northern Ireland	
3467	300003468	Stewart	Johnston	Male	41	Wales	
3496	300003497	Rebecca	Howard	Female	30	Wales	
3884	100003885	Abigail	Mills	Female	29	England	

	JobClassification	DateJoined	Balance
74	White Collar	12.Feb.15	21.03
774	Blue Collar	01.Jun.15	69.01
1319	White Collar	22.Jul.15	11.52
2045	Blue Collar	13.Sep.15	69.78
3467	Blue Collar	30.Nov.15	77.46
3496	Blue Collar	02.Dec.15	96.26
3884	White Collar	23.Dec.15	98.68

64 More then one filters

[66]:
```
stats[Filter & Filter2]
```

[66]:
```
     CustomerID  Name  Surname Gender  Age    Region JobClassification  \
841   200000842  Matt  Manning   Male   64  Scotland       Blue Collar

    DateJoined  Balance
841  10.Jun.15  3775.44
```

[67]:
```
stats[(stats.Age>63) & (stats.Balance<5000)] # Same as stats[Filter & Filter2]
```

[67]:
```
     CustomerID  Name  Surname Gender  Age    Region JobClassification  \
841   200000842  Matt  Manning   Male   64  Scotland       Blue Collar

    DateJoined  Balance
841  10.Jun.15  3775.44
```

65 AnotherOne:

[68]:
```
stats[stats.Region=='England']
```

[68]:
```
      CustomerID      Name    Surname  Gender  Age   Region JobClassification  \
0      100000001     Simon      Walsh    Male   21  England      White Collar
2      100000003      Liam      Brown    Male   46  England      White Collar
4      100000005   Deirdre    Pullman  Female   38  England       Blue Collar
6      100000007   Dorothy    Thomson  Female   34  England       Blue Collar
9      100000010   Dominic       Parr    Male   42  England      White Collar
…            …       …         …      …   …       …                …
4003   100004004      Jane   Hemmings  Female   28  England       Blue Collar
4004   100004005      John   Hamilton    Male   45  England      White Collar
4005   100004006  Kimberly       Gray  Female   44  England             Other
4006   100004007    Rachel     Davies  Female   34  England       Blue Collar
4007   100004008       Sam  Sanderson    Male   28  England       Blue Collar

     DateJoined    Balance
0     05.Jan.15  113810.15
2     07.Jan.15  101536.83
4     09.Jan.15   35639.79
6     11.Jan.15   42879.84
9     12.Jan.15   10912.45
…           …          …
4003  30.Dec.15   68518.55
4004  30.Dec.15    8435.91
4005  30.Dec.15   64470.77
4006  30.Dec.15    4561.22
4007  30.Dec.15   42128.29
```

```
[2159 rows x 9 columns]
```

[69]:
```
# How to get unique()
stats.Region.unique()
```

[69]:
```
array(['England', 'Northern Ireland', 'Wales', 'Scotland'], dtype=object)
```

66 Quick Exercise:

67 Find out everything about Matt Manning

[70]:
```
stats[(stats.Name=='Matt')&(stats.Surname=='Manning')]
```

[70]:
```
     CustomerID  Name  Surname Gender  Age    Region JobClassification  \
841   200000842  Matt  Manning   Male   64  Scotland       Blue Collar

    DateJoined  Balance
841  10.Jun.15  3775.44
```

68 Accessing Individual Elements

69 1) .at for lables Important: even integers are treated as labels

70 2) .iat for interger location

[71]:
```
stats.head()
```

[71]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance
0      White Collar  05.Jan.15  113810.15
1       Blue Collar  06.Jan.15   36919.73
2      White Collar  07.Jan.15  101536.83
3      White Collar  08.Jan.15    1421.52
4       Blue Collar  09.Jan.15   35639.79
```

[72]:
```
stats.iat[2,1]
```

[72]:
```
'Liam'
```

[73]:
```
stats.at[2,'Name']
```

[73]:
```
'Liam'
```

71 Why we need?

[74]:
```
sub10=stats[::1000]
```

[75]:
```
sub10
```

[75]:
```
      CustomerID        Name  Surname  Gender  Age            Region  \
0      100000001       Simon    Walsh    Male   21           England
1000   400001001       Grace   Duncan  Female   31  Northern Ireland
2000   100002001  Bernadette      Ince  Female   43           England
3000   100003001        Matt  Abraham    Male   47           England
4000   300004001   Gabrielle   Duncan  Female   34             Wales

     JobClassification DateJoined    Balance
0         White Collar  05.Jan.15  113810.15
1000             Other  26.Jun.15   32162.34
2000      White Collar  11.Sep.15   57739.46
3000       Blue Collar  03.Nov.15    8576.46
4000      White Collar  29.Dec.15   92083.79
```

[76]:
```
sub10.iat[1,0] # It's counting 0 axis
```

[76]:
```
400001001
```

[77]:
```
sub10.at[1000,'CustomerID'] # It's sees index
```

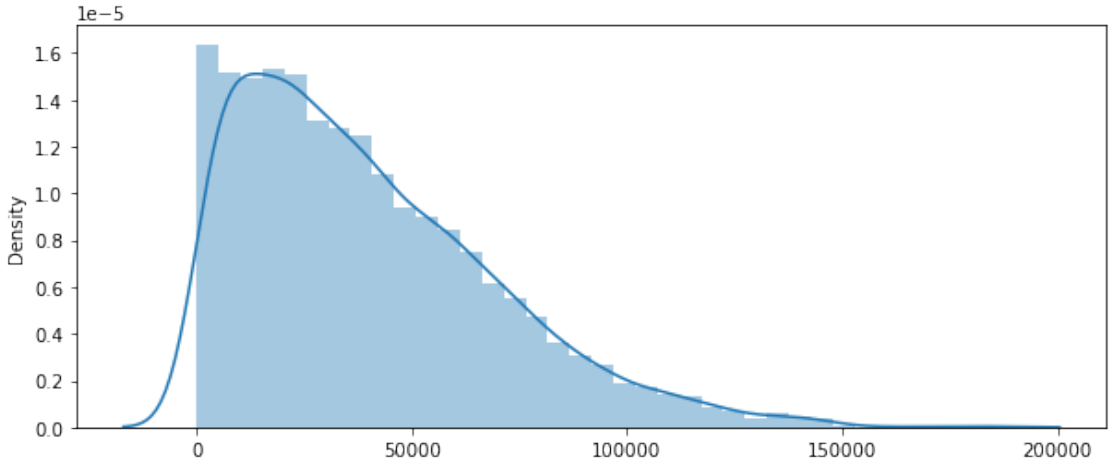
[77]:
```
400001001
```

72 Introduction to Seaborn

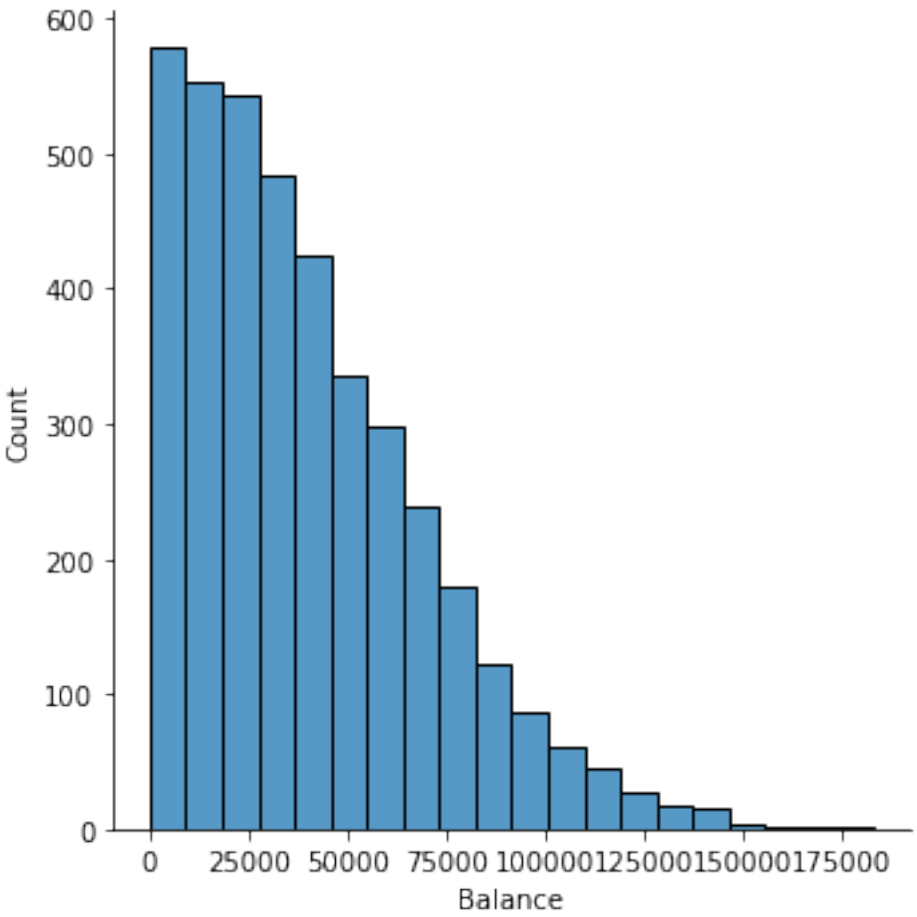
[78]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
plt.rcParams['figure.figsize']=10,4
```

73 Distribution:

[79]:
```
Vis1=sns.distplot([stats.Balance])
```



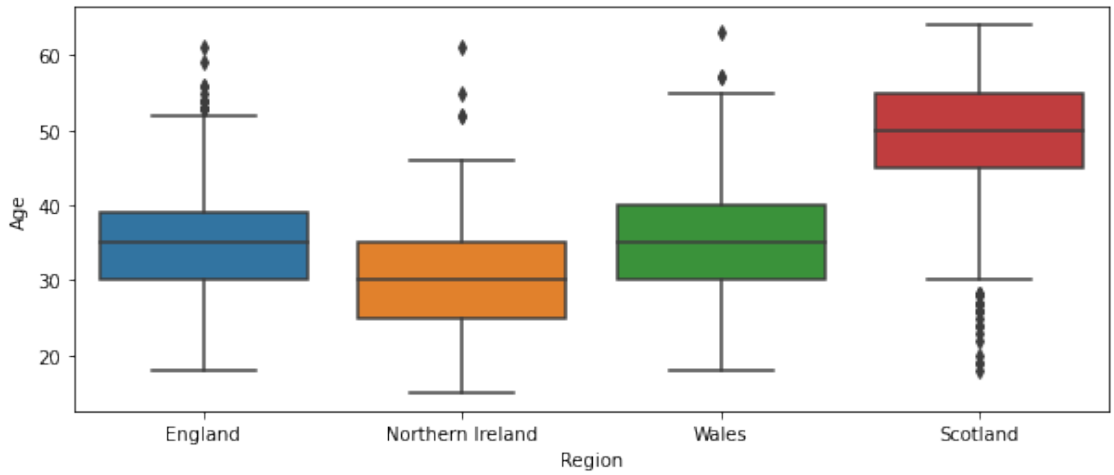
[80]:
```
Vis1=sns.displot(stats["Balance"],bins=20)
```



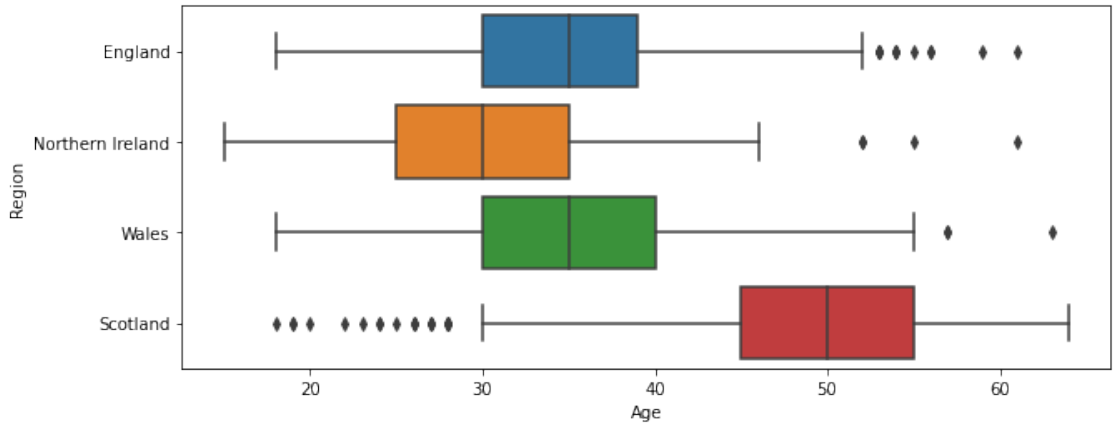
74 BoxPlots:

Google: Seaborn gallery

[81]:
```
vis2=sns.boxplot(data=stats,x="Region",y="Age")
```



[82]:
```
vis3=sns.boxplot(data=stats,x="Age",y="Region")
```

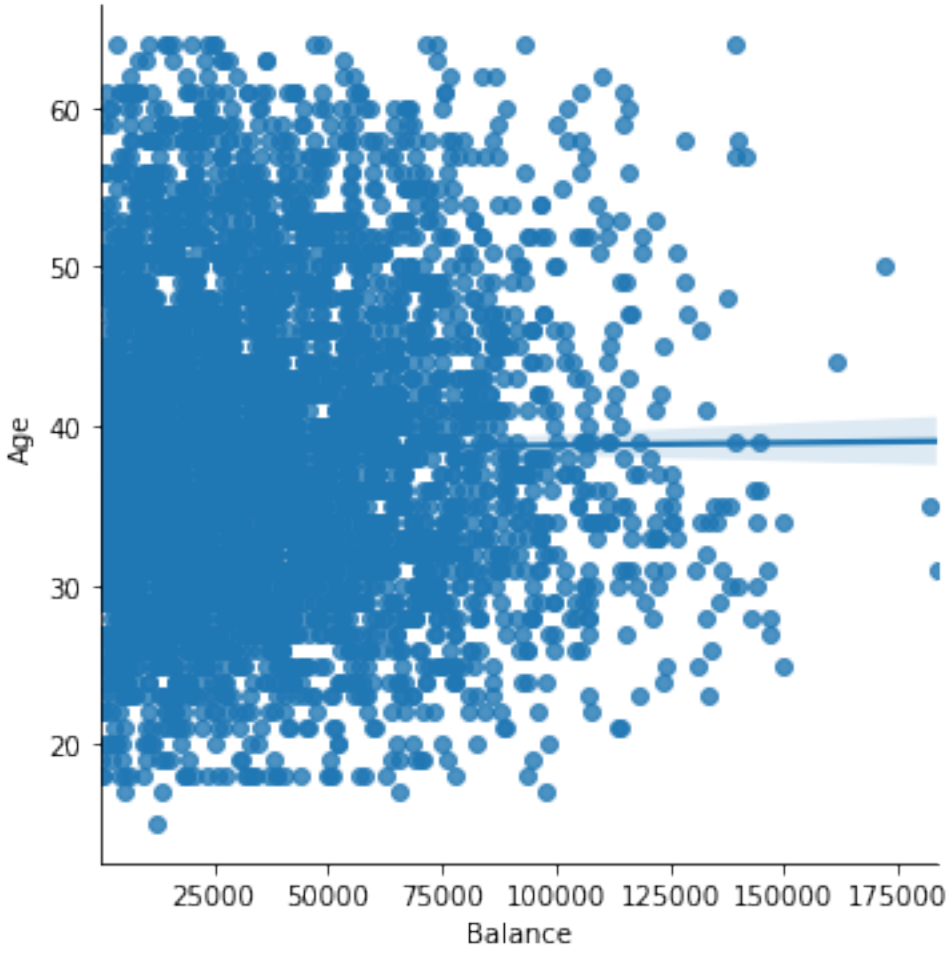


[83]:
```
stats.head()
```

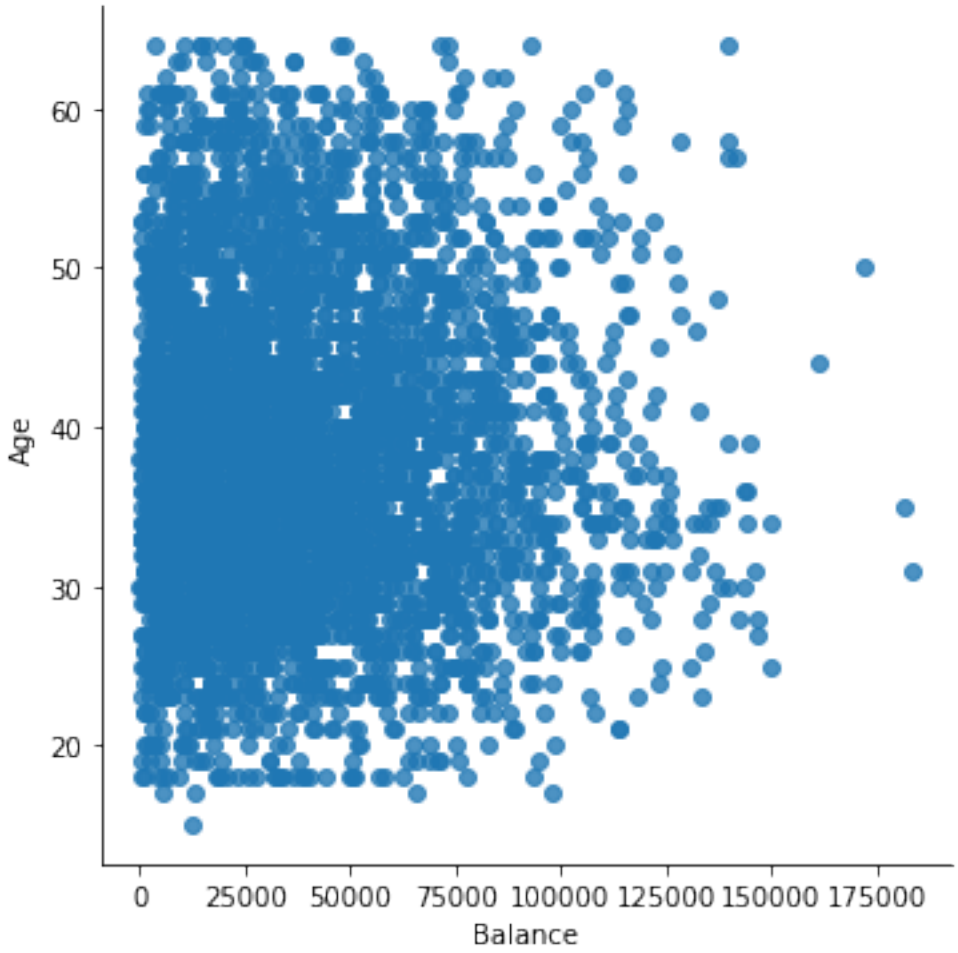
[83]:
```
   CustomerID     Name  Surname  Gender  Age            Region  \
0   100000001    Simon    Walsh    Male   21           England
1   400000002  Jasmine   Miller  Female   34  Northern Ireland
2   100000003     Liam    Brown    Male   46           England
3   300000004   Trevor     Parr    Male   32             Wales
4   100000005  Deirdre  Pullman  Female   38           England

  JobClassification DateJoined    Balance
0      White Collar  05.Jan.15  113810.15
1       Blue Collar  06.Jan.15   36919.73
2      White Collar  07.Jan.15  101536.83
```

```
3      White Collar  08.Jan.15    1421.52
4       Blue Collar  09.Jan.15   35639.79
```

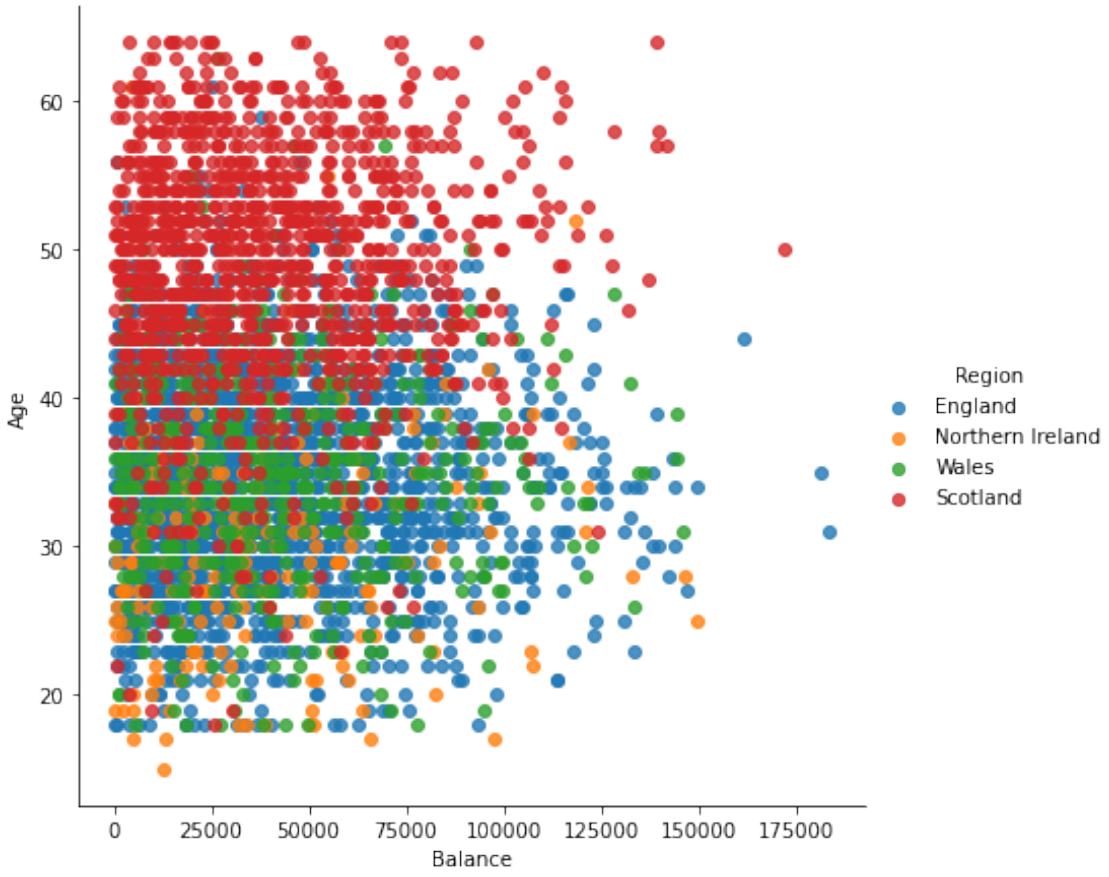
[84]:
```
vis4=sns.lmplot(data=stats,x='Balance',y='Age')
# Or vis4=sns.lmplot,x='Balance',y='Age',data=stats)
```



[85]:
```
vis4=sns.lmplot(data=stats,x='Balance',y='Age',fit_reg=False)
```



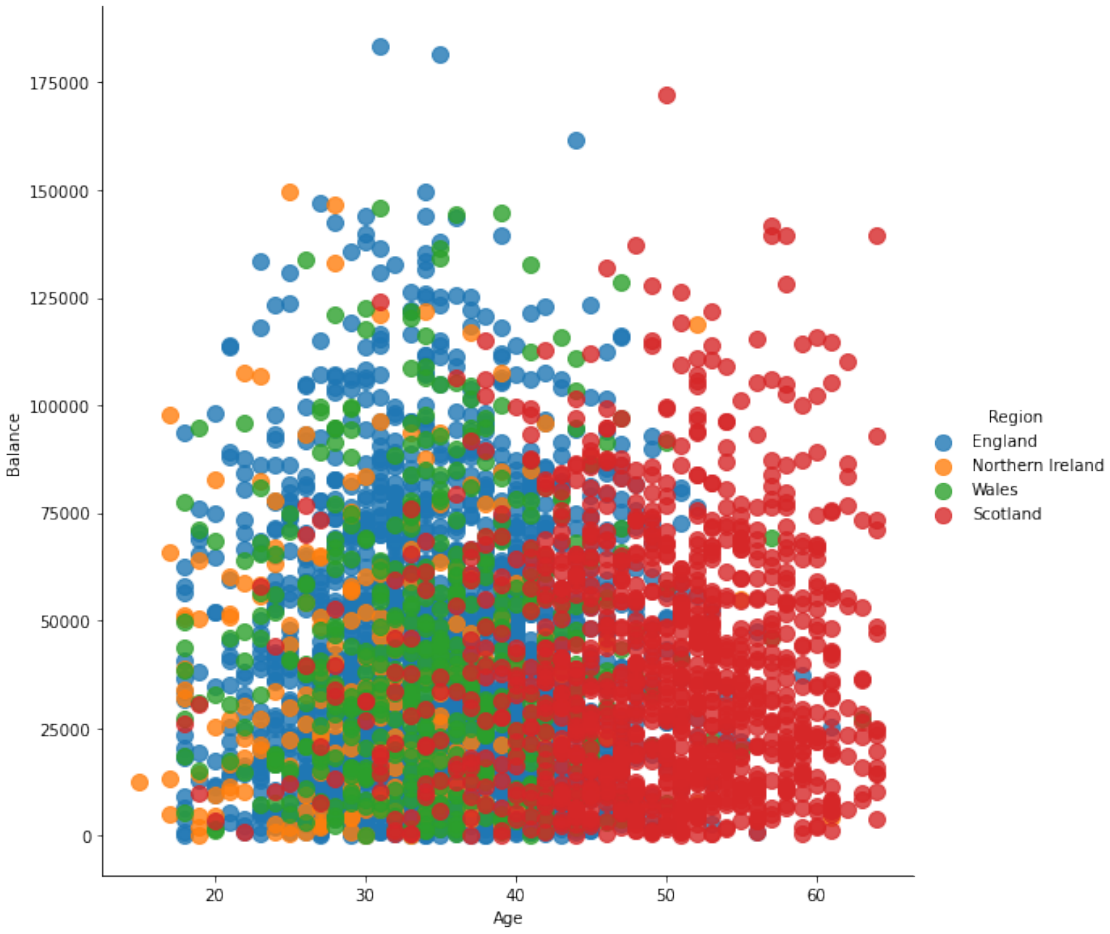
[86]:
```
vis4=sns.
 ↪lmplot(data=stats,x='Balance',y='Age',fit_reg=False,hue='Region',size=6)
#hue=color
```



75 Marker Size

[87]:
```
vis4=sns.lmplot(data=stats,x='Balance',y='Age',fit_reg=False,hue='Region',\
                size=6,scatter_kws={'s':100})
```

[88]:
```
vis4=sns.lmplot(data=stats,x='Age',y='Balance',fit_reg=False,hue='Region',\
                size=8,scatter_kws={'s':100})
```



---

76 Movie-Ratings Analysis

[2]:
```
import pandas as pd
import os
```

[3]:
```
os.getcwd()
```

[3]:
```
'C:\\Users\\ddaya\\Documents\\Python Programs'
```

[4]:
```
os.chdir('C:\\Users\\ddaya\\OneDrive\\Documents\\Python programming')
```

[5]:
```
movies=pd.read_csv('Movie-Ratings.csv')
```

[6]:
```
movies
```

[6]:
```
                     Film      Genre  Rotten Tomatoes Ratings %  \
0    (500) Days of Summer     Comedy                         87
1             10,000 B.C.  Adventure                          9
2               12 Rounds     Action                         30
3               127 Hours  Adventure                         93
4                17 Again     Comedy                         55
..                    …        …                          …
554         Your Highness     Comedy                         26
555       Youth in Revolt     Comedy                         68
556                Zodiac   Thriller                         89
557            Zombieland     Action                         90
558             Zookeeper     Comedy                         14

     Audience Ratings %  Budget (million $)  Year of release
0                    81                   8             2009
1                    44                 105             2008
2                    52                  20             2009
3                    84                  18             2010
4                    70                  20             2009
..                  …                 …              …
554                  36                  50             2011
555                  52                  18             2009
556                  73                  65             2007
557                  87                  24             2009
558                  42                  80             2011

[559 rows x 6 columns]
```

[7]:
```
len(movies)
```

[7]:
```
559
```

[8]:
```
movies.head()
```

[8]:
```
                   Film      Genre  Rotten Tomatoes Ratings %  \
0  (500) Days of Summer     Comedy                         87
1           10,000 B.C.  Adventure                          9
2             12 Rounds     Action                         30
3             127 Hours  Adventure                         93
4              17 Again     Comedy                         55

   Audience Ratings %  Budget (million $)  Year of release
0                  81                   8             2009
1                  44                 105             2008
2                  52                  20             2009
3                  84                  18             2010
4                  70                  20             2009
```

[9]:
```
movies.columns
```

[9]:
```
Index(['Film', 'Genre', 'Rotten Tomatoes Ratings %', 'Audience Ratings %',
       'Budget (million $)', 'Year of release'],
      dtype='object')
```

[10]:
```
movies.columns=['Film', 'Genre', 'CriticRatings', 'AudienceRatings',
       'Budget(million $)', 'Year']
```

[11]:
```
movies.head()
```

[11]:

	Film	Genre	CriticRatings	AudienceRatings	Budget(million $)	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

[12]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Film               559 non-null    object
 1   Genre              559 non-null    object
 2   CriticRatings      559 non-null    int64 
 3   AudienceRatings    559 non-null    int64 
 4   Budget(million $)  559 non-null    int64 
 5   Year               559 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 26.3+ KB
```

[13]:
```
movies.describe()  # it's worng as year also calculate
```

[13]:

	CriticRatings	AudienceRatings	Budget(million $)	Year
count	559.000000	559.000000	559.000000	559.000000
mean	47.309481	58.744186	50.236136	2009.152057
std	26.413091	16.826887	48.731817	1.362632
min	0.000000	0.000000	0.000000	2007.000000

```
25%        25.000000        47.000000          20.000000  2008.000000
50%        46.000000        58.000000          35.000000  2009.000000
75%        70.000000        72.000000          65.000000  2010.000000
max        97.000000        96.000000         300.000000  2011.000000
```

[14]:
```
movies.Film=movies.Film.astype('category')
```

[15]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Film               559 non-null    category
 1   Genre              559 non-null    object
 2   CriticRatings      559 non-null    int64
 3   AudienceRatings    559 non-null    int64
 4   Budget(million $)  559 non-null    int64
 5   Year               559 non-null    int64
dtypes: category(1), int64(4), object(1)
memory usage: 43.6+ KB
```

[16]:
```
movies.Genre=movies.Genre.astype('category')
movies.Year=movies.Year.astype('category')
```

[17]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 559 entries, 0 to 558
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Film               559 non-null    category
 1   Genre              559 non-null    category
 2   CriticRatings      559 non-null    int64
 3   AudienceRatings    559 non-null    int64
 4   Budget(million $)  559 non-null    int64
 5   Year               559 non-null    category
dtypes: category(3), int64(3)
memory usage: 36.5 KB
```

[18]:
```
movies.describe()
```

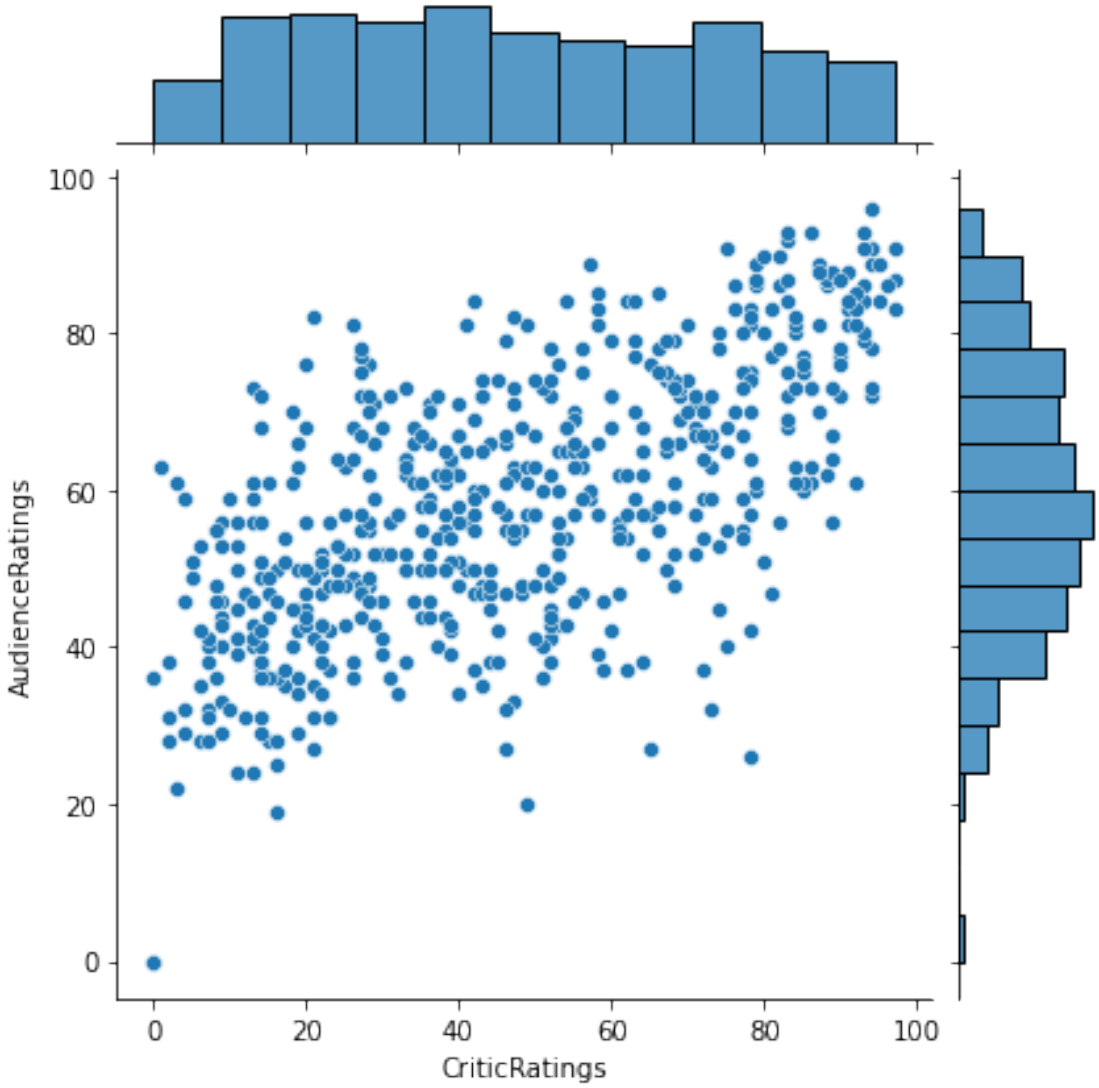
[18]:
```
       CriticRatings  AudienceRatings  Budget(million $)
count     559.000000       559.000000         559.000000
mean       47.309481        58.744186          50.236136
std        26.413091        16.826887          48.731817
```

```
min          0.000000          0.000000          0.000000
25%         25.000000         47.000000         20.000000
50%         46.000000         58.000000         35.000000
75%         70.000000         72.000000         65.000000
max         97.000000         96.000000        300.000000
```

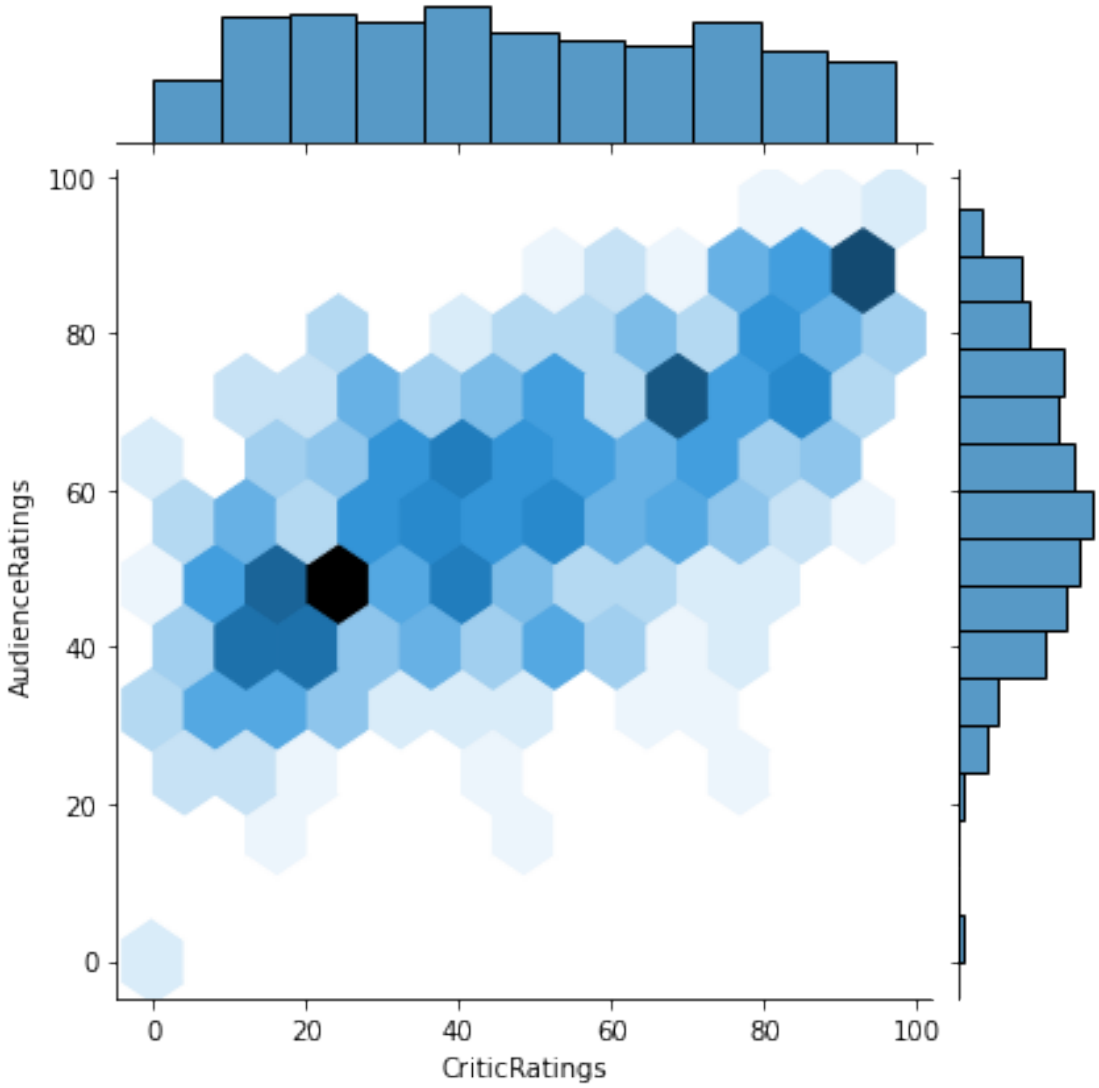
76.1 # Jointplots

[19]:
```
import matplotlib as plt
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

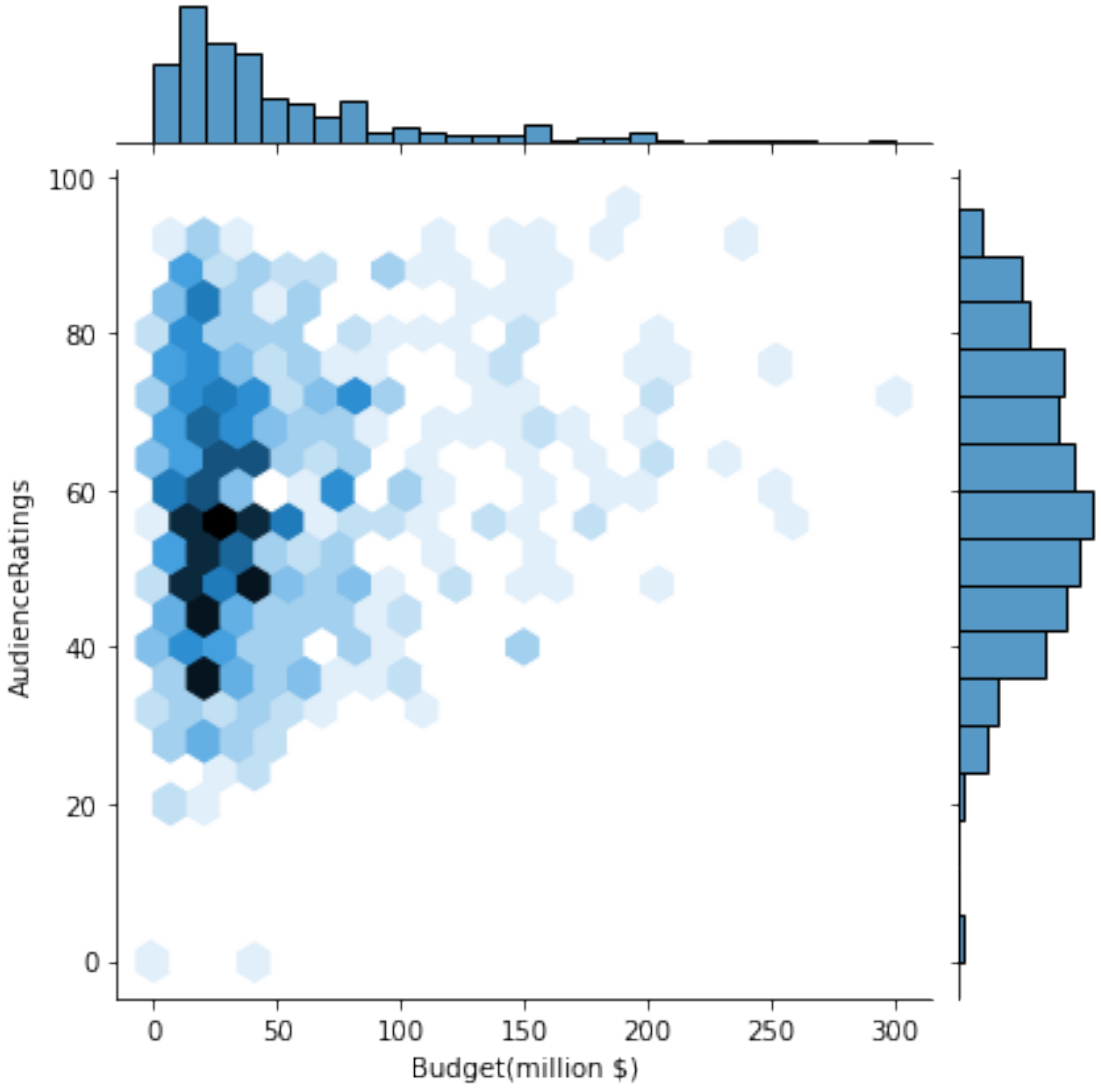
[20]:
```
j=sns.jointplot(data=movies, x='CriticRatings',y='AudienceRatings')
```



[21]:
```
j=sns.jointplot(data=movies, x='CriticRatings',y='AudienceRatings',kind='hex')
```



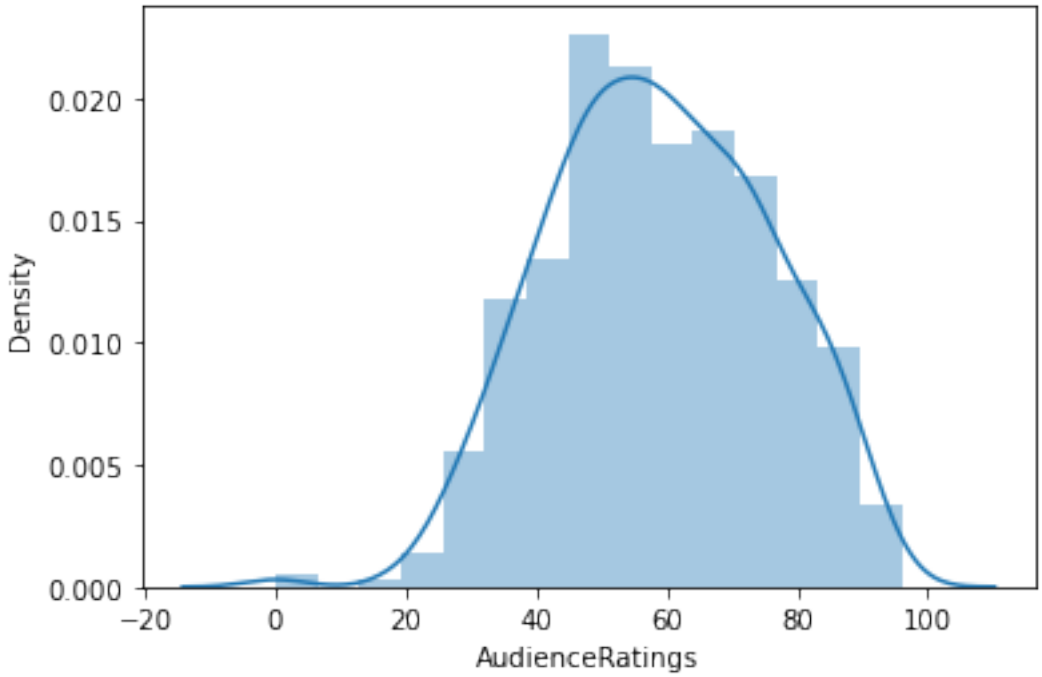
[22]:
```
j=sns.jointplot(data=movies, x='Budget(million␣
↪$)',y='AudienceRatings',kind='hex')
```



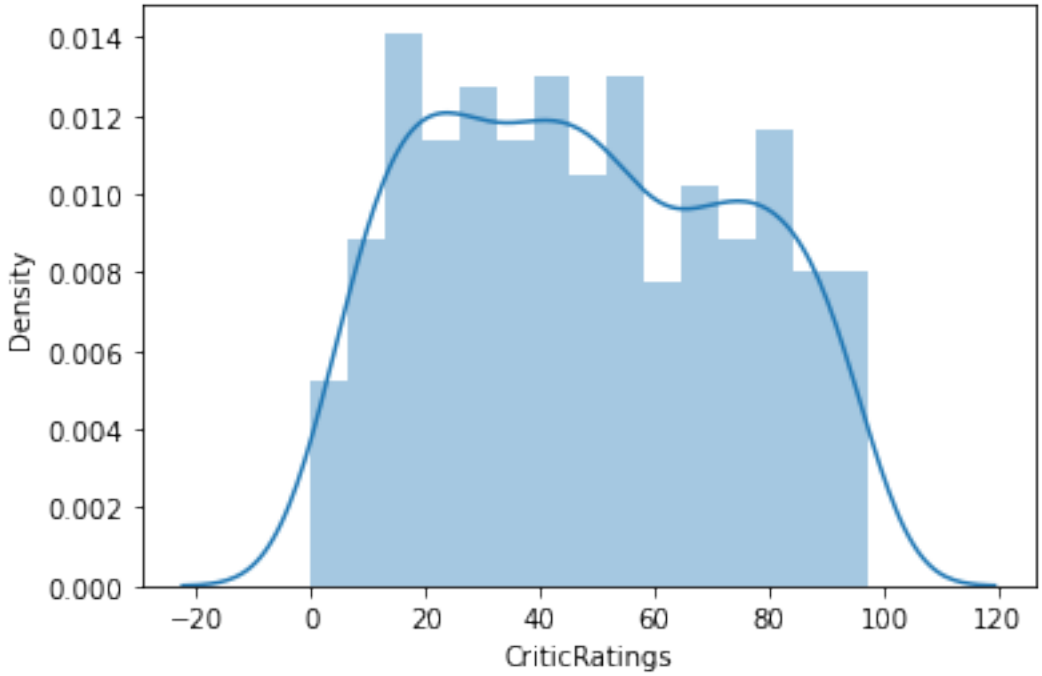
[23]:
```
# Chart 1
```

77 Histograms

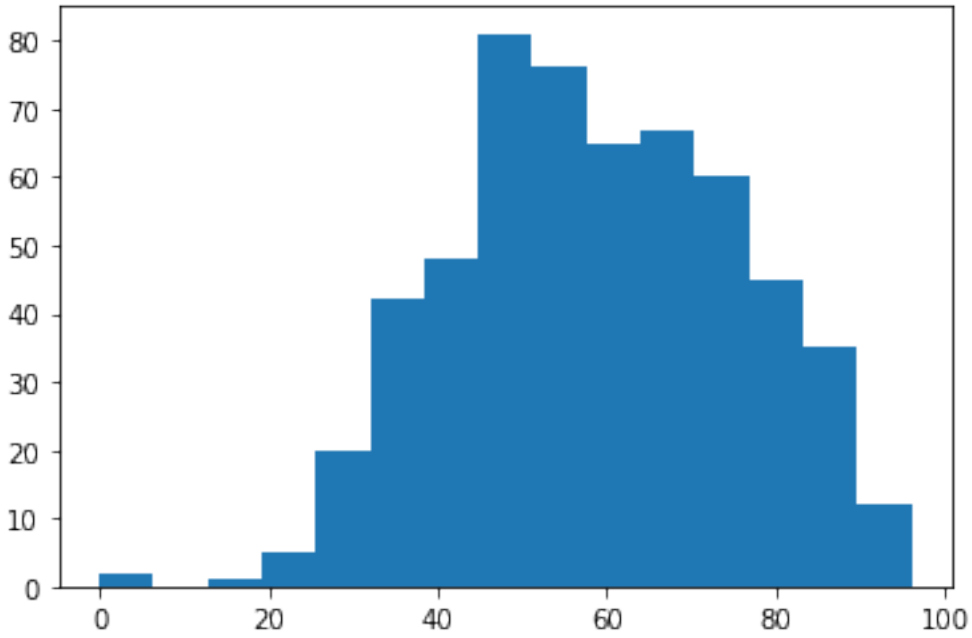
[24]:
```
m1=sns.distplot(movies.AudienceRatings,bins=15)
```



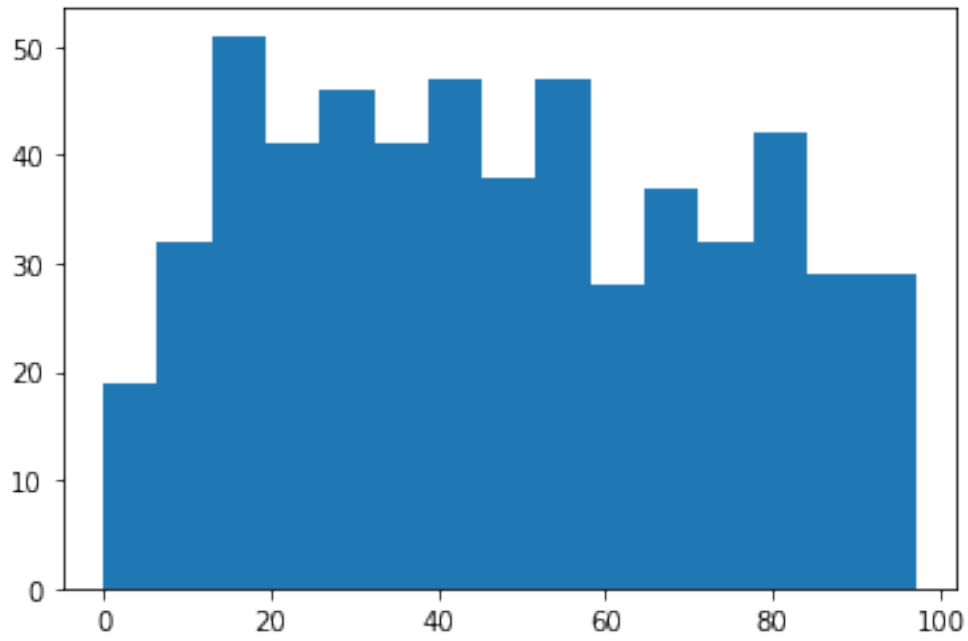
[25]:
```
m2=sns.distplot(movies.CriticRatings,bins=15)
```



[26]:
```
# chart 2
n1 = plt.hist(movies.AudienceRatings,bins=15)
```



[27]:
```
n2 = plt.hist(movies.CriticRatings,bins=15)
```



78 Stacked Histograms

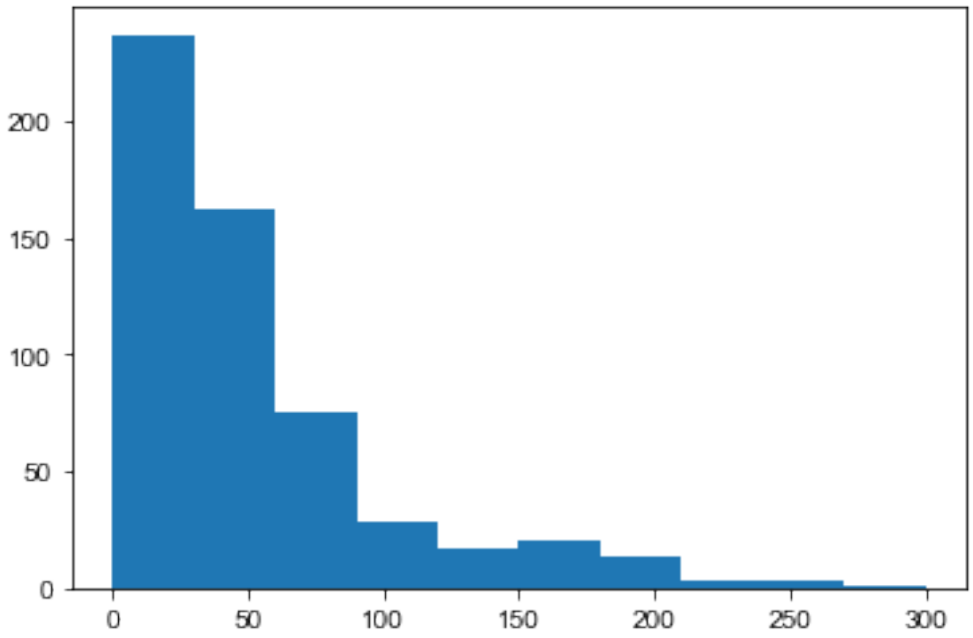
[28]:
```
movies.columns=['Film', 'Genre', 'CriticRatings', 'AudienceRatings',
       'BudgetMillion', 'Year']
```

[29]:
```
movies.head()
```

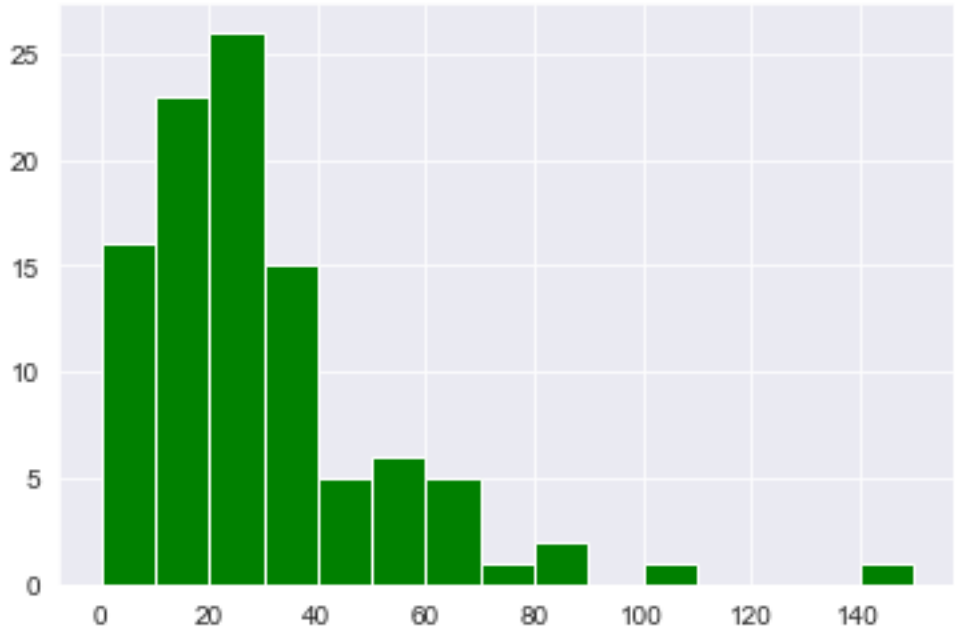
[29]:

	Film	Genre	CriticRatings	AudienceRatings	BudgetMillion	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44	105	2008
2	12 Rounds	Action	30	52	20	2009
3	127 Hours	Adventure	93	84	18	2010
4	17 Again	Comedy	55	70	20	2009

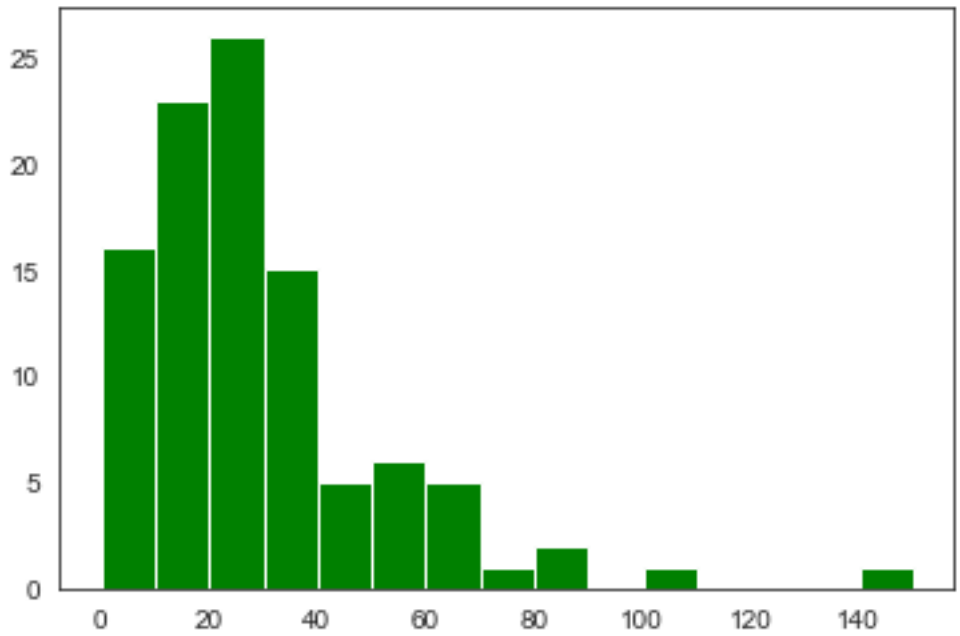
[30]:
```
movies[movies.Genre=='Comedy'] # Filter
plt.hist(movies.BudgetMillion)
sns.set_style("darkgrid")
plt.show()
```



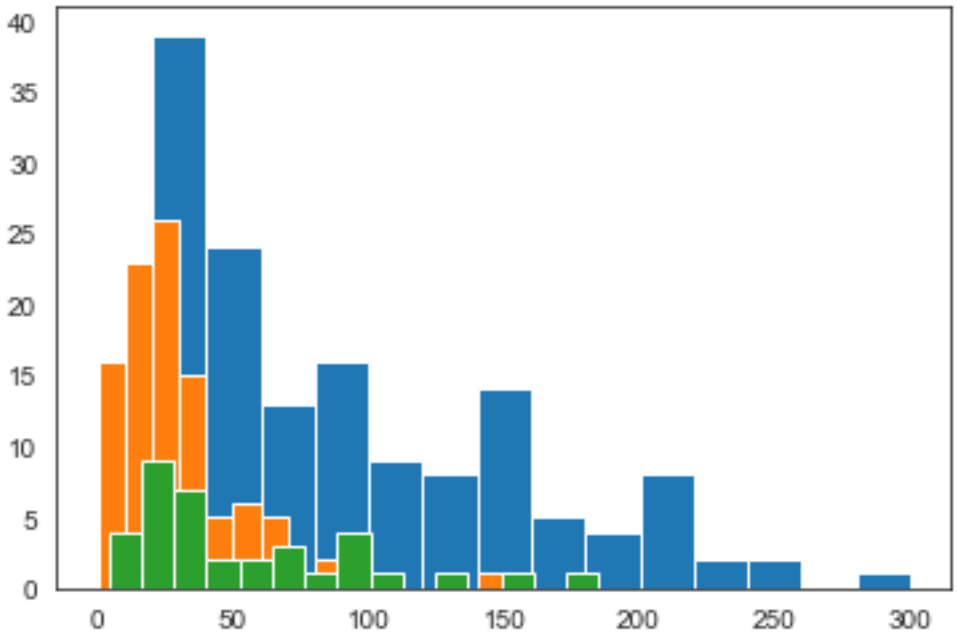
[31]:
```
plt.hist(movies[movies.Genre=='Drama'].BudgetMillion,bins=15,color='Green')
plt.show()
```



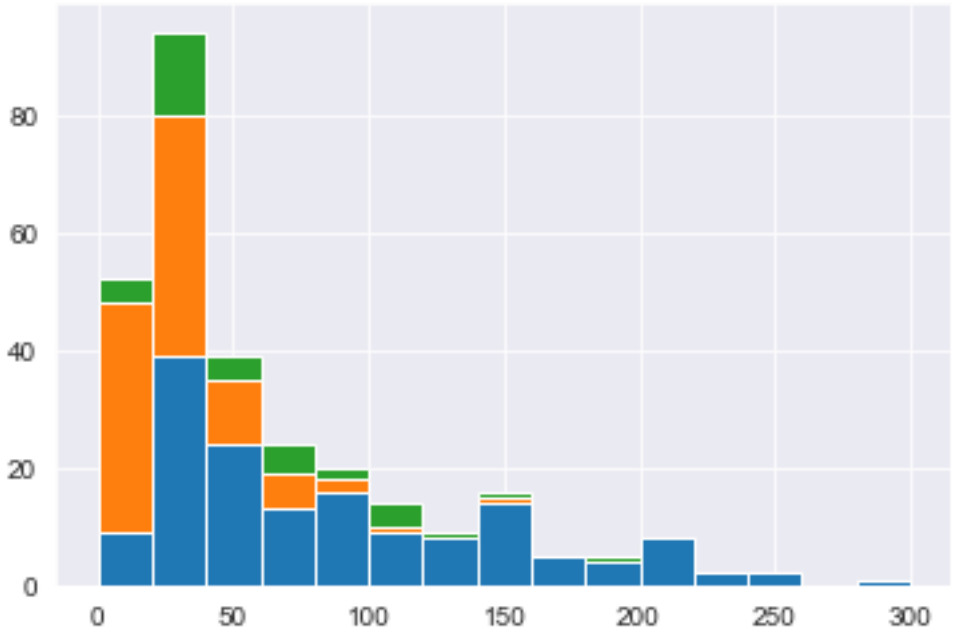
[32]:
```
sns.set_style("white")
plt.hist(movies[movies.Genre=='Drama'].BudgetMillion,bins=15,color='Green')
plt.show()
```



[33]:
```
plt.hist(movies[movies.Genre=='Action'].BudgetMillion,bins=15)
plt.hist(movies[movies.Genre=='Drama'].BudgetMillion,bins=15)
plt.hist(movies[movies.Genre=='Thriller'].BudgetMillion,bins=15)
sns.set_style("darkgrid")
plt.show()
```



[34]:
```
# OR

plt.hist([movies[movies.Genre=='Action'].BudgetMillion,
movies[movies.Genre=='Drama'].BudgetMillion,
movies[movies.Genre=='Thriller'].BudgetMillion],bins=15,stacked=True)
plt.show()
```



[35]:
```
# OR

movies.Genre.cat.categories
```

[35]:
```
Index(['Action', 'Adventure', 'Comedy', 'Drama', 'Horror', 'Romance',
       'Thriller'],
      dtype='object')
```

[36]:
```
for gen in movies.Genre.cat.categories:
    print(gen)
```

```
Action
Adventure
Comedy
Drama
Horror
Romance
Thriller
```

[37]:
```
list1=list([])
for gen in movies.Genre.cat.categories:
    list1.append(movies[movies.Genre==gen].BudgetMillion)
print(list1)
```

```
[2       20
5      200
15      35
```

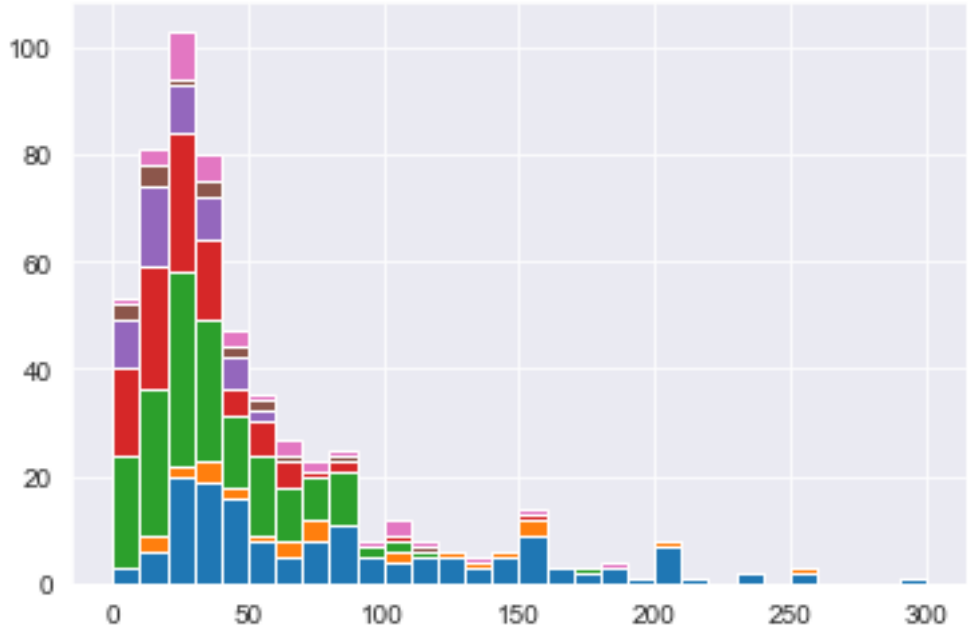
```
29      20
30      20
      …
531    130
542     35
546    150
547    160
557     24
Name: BudgetMillion, Length: 154, dtype: int64, 1      105
3       18
19     200
21      45
24      40
32      78
46      20
65      38
68     140
130     73
165     12
166    125
167    250
168    150
176     36
178    150
192     70
193     60
241     60
272     37
341     19
363     70
386    130
401    155
459     59
463     25
506     38
540    100
548     60
Name: BudgetMillion, dtype: int64, 0       8
4      20
6      30
8      28
9       8
       ..
552    80
553    22
554    50
555    18
558    80
```

```
Name: BudgetMillion, Length: 172, dtype: int64, 10     30
11     20
13      7
18      8
23     20
       ..
529    66
532    38
534    21
541    15
545     2
Name: BudgetMillion, Length: 101, dtype: int64, 7      32
12     35
20     40
28      5
59     26
88     10
97     25
100    30
103    50
109    20
126    40
135    19
137    30
160    20
161    15
175    10
194     2
246    35
259    25
285    20
286    30
292     1
293     3
294     5
311    18
315    12
321    42
322     4
332    10
333    11
335    40
343    25
349     8
355    13
373    50
404    20
414    12
```

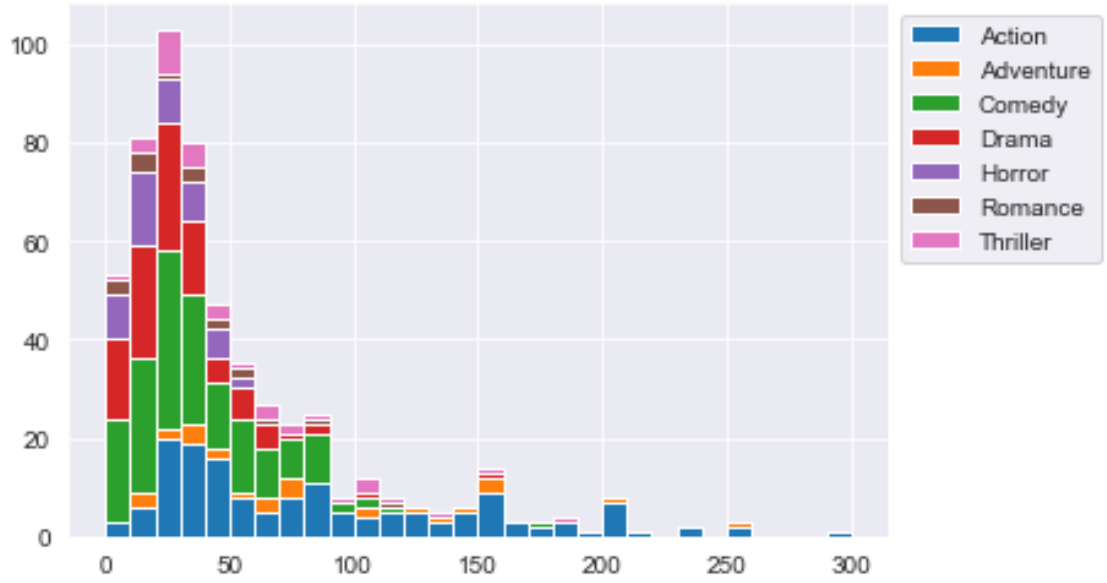
```
416    40
426     5
429    15
453    18
461    40
462    37
464    16
465    25
475     9
478    38
486    16
521    10
Name: BudgetMillion, dtype: int64, 16      45
42      17
78      50
108     60
136     35
201      0
208     80
244     17
250     20
255     40
266     56
284     15
290     30
354     35
507    110
510     15
524      5
525      2
Name: BudgetMillion, dtype: int64, 25     100
72      60
95      20
105     15
179    150
180     60
189     40
225     27
237      4
243     25
253     20
261     20
263    130
267     70
282     85
358     32
385     51
389     20
```

```
394    110
406    185
407    100
408     20
419     90
424     48
432     13
471     15
481    100
491     35
494     21
498     22
503     35
513     30
515     35
519     75
522     40
556     65
Name: BudgetMillion, dtype: int64]
```

[38]:
```
list1=list([])
mylabel=list([])
for gen in movies.Genre.cat.categories:
    list1.append(movies[movies.Genre==gen].BudgetMillion)
h=plt.hist(list1, bins=30,stacked=True,rwidth=1)
```



[39]:
```
list1=list([])
mylabel=list([])
for gen in movies.Genre.cat.categories:
    list1.append(movies[movies.Genre==gen].BudgetMillion)
    mylabel.append(gen)
h=plt.hist(list1, bins=30,stacked=True,rwidth=1,label=mylabel)
plt.legend()
plt.legend(loc='upper left',bbox_to_anchor=(1,1))
plt.show()
```



[40]:
```
# <<< chart 4
```

---

79 KDE Plot

[41]:
```
movies.head()
```

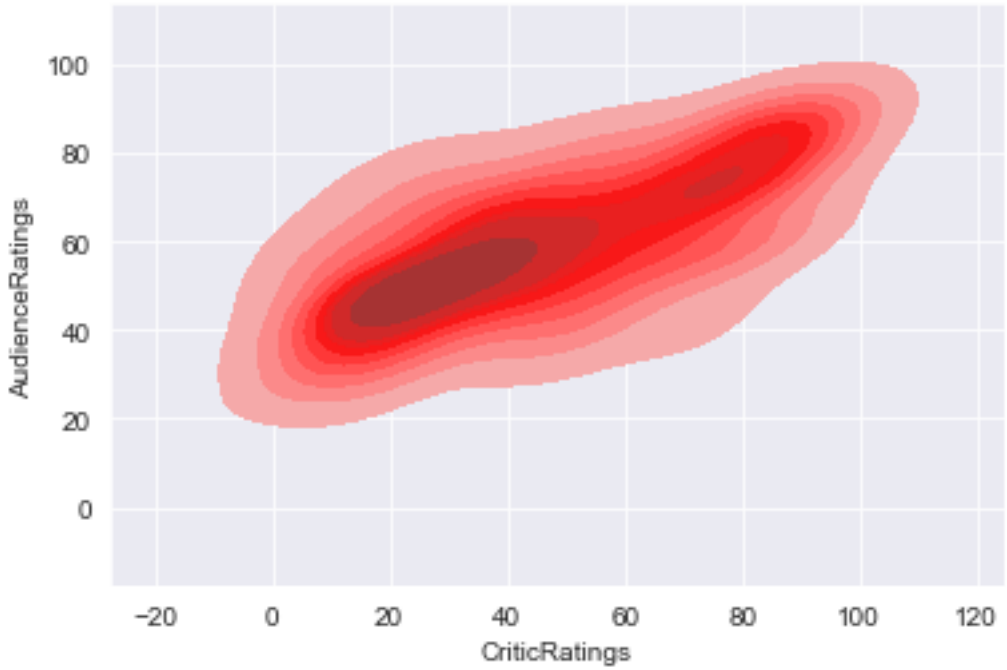
[41]:

	Film	Genre	CriticRatings	AudienceRatings	BudgetMillion	Year
0	(500) Days of Summer	Comedy	87	81	8	2009
1	10,000 B.C.	Adventure	9	44		
2	12 Rounds	Action	30	52		
3	127 Hours	Adventure	93	84		
4	17 Again	Comedy	55	70		

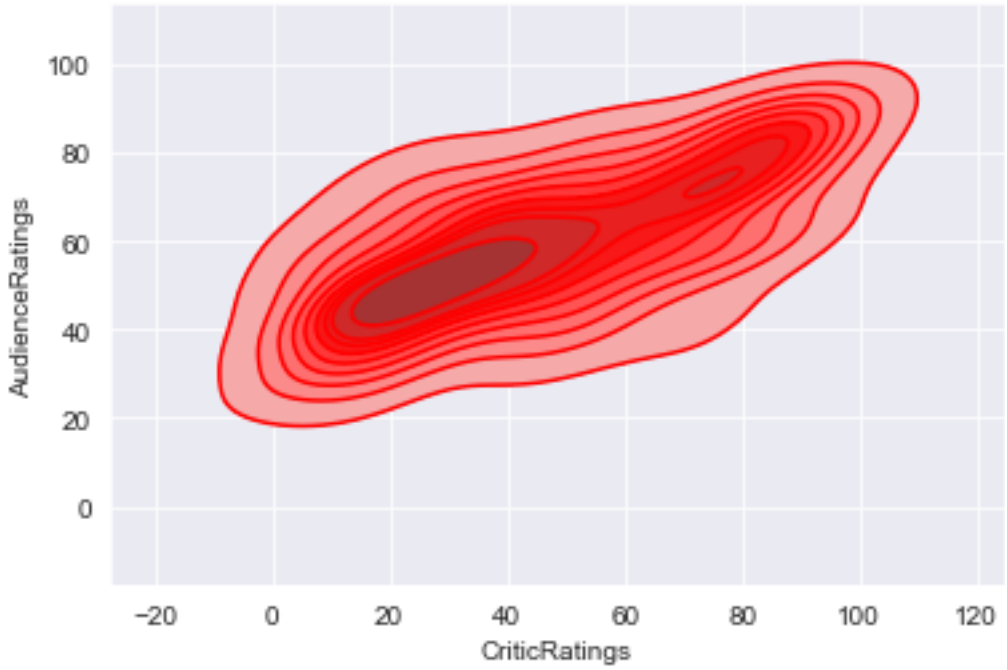
```
1              105  2008
2               20  2009
3               18  2010
4               20  2009
```

[42]:
```
sns.
 ↪lmplot(data=movies,x='CriticRatings',y='AudienceRatings',fit_reg=False,hue='Genre',size=6,a
plt.show()
```

[43]:
```
k1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               shade=True,shade_lowest=False,color='Red')
```



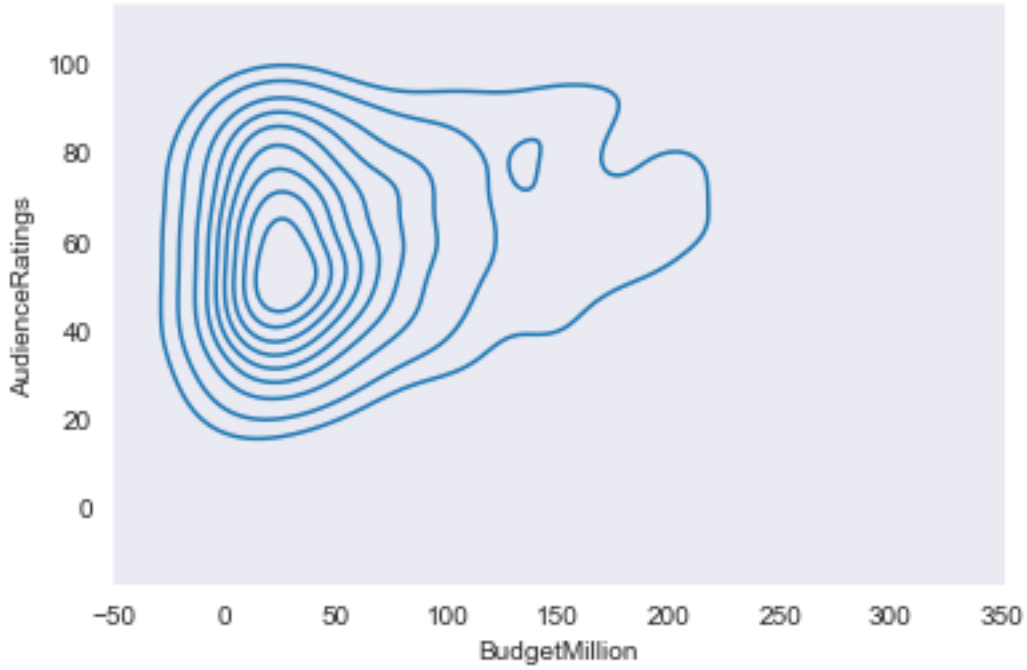
[44]:
```
k1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               shade=True,shade_lowest=False,color='Red')

k1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               color='Red')
```



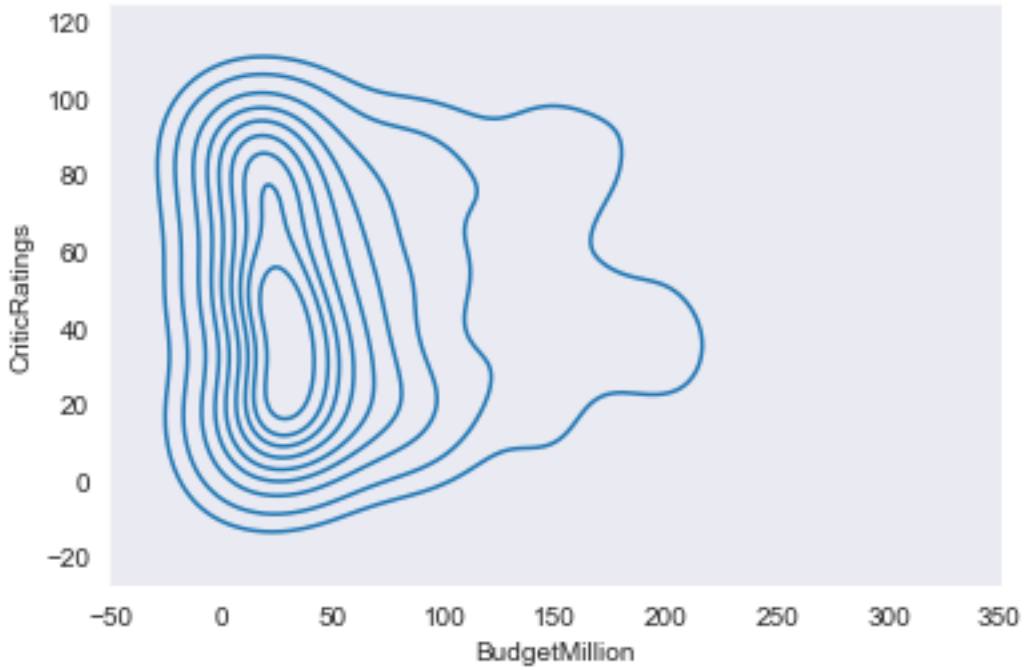
80 working with Subplots()

[45]:
```
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

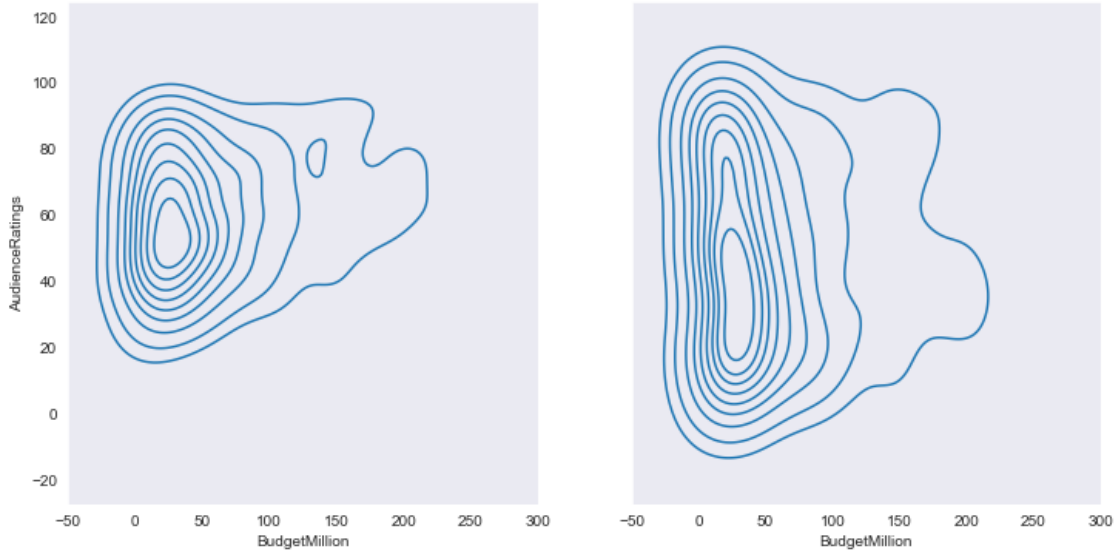
[46]:
```
sns.set_style('dark')
k1=sns.kdeplot(movies.BudgetMillion,movies.AudienceRatings)
plt.show()
```



[47]:
```
k2=sns.kdeplot(movies.BudgetMillion,movies.CriticRatings)
plt.show()
```

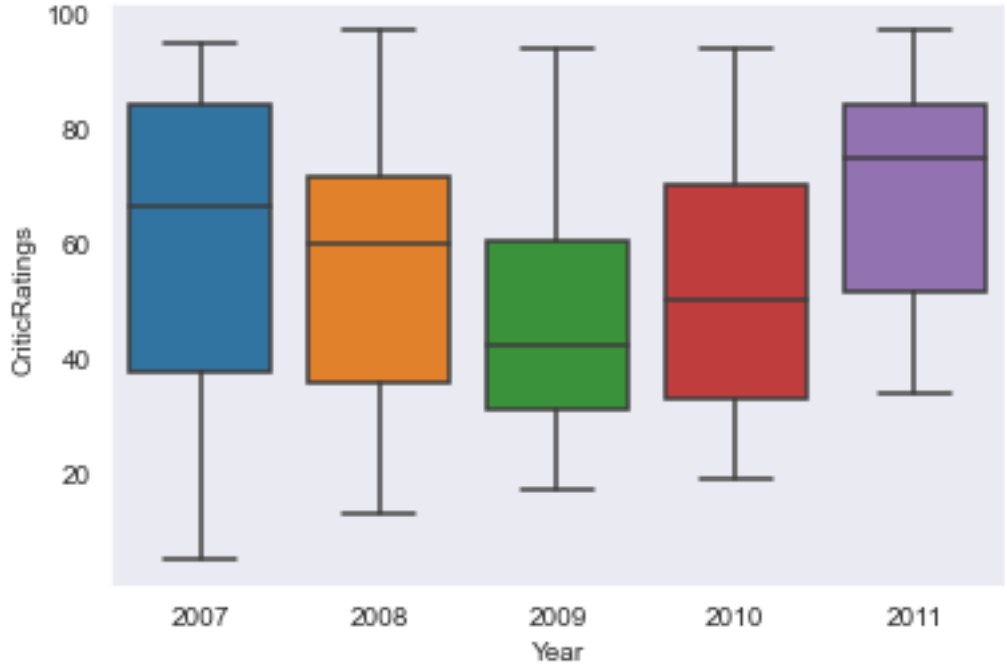


[48]:
```
f, axes=plt.subplots(1,2, figsize=(12,6),sharex=True,sharey=True)
k1=sns.kdeplot(movies.BudgetMillion,movies.AudienceRatings,ax=axes[0])
k2=sns.kdeplot(movies.BudgetMillion,movies.CriticRatings,ax=axes[1])
k1.set(xlim=(-50,300))
plt.show()
```

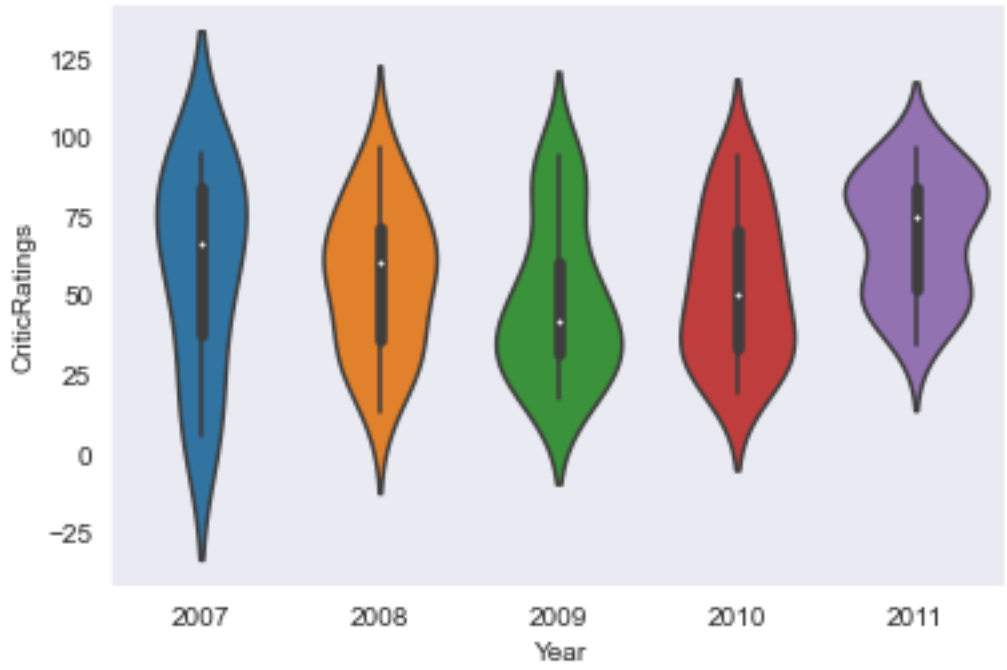


81 Violinplots Vs Boxplots

[49]:
```
w=sns.boxplot(data=movies[movies.Genre=='Drama'],x='Year',y='CriticRatings')
# w=sns.boxplot(data=movies,x='Genre',y='CriticRatings')
```

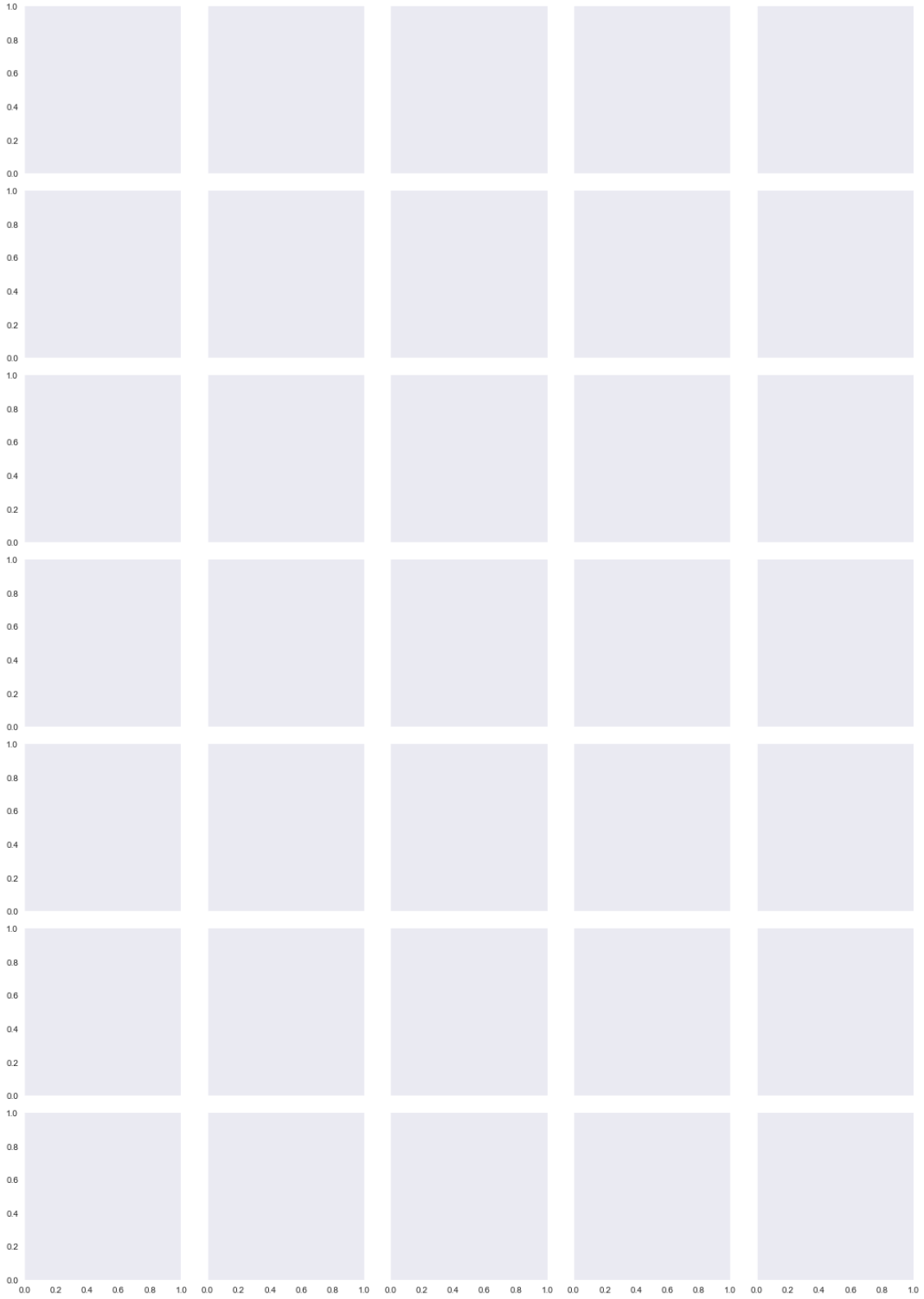


[50]:
```
z=sns.violinplot(data=movies[movies.Genre=='Drama'],x='Year',y='CriticRatings')
# z=sns.violinplot(data=movies,x='Genre',y='CriticRatings')
```

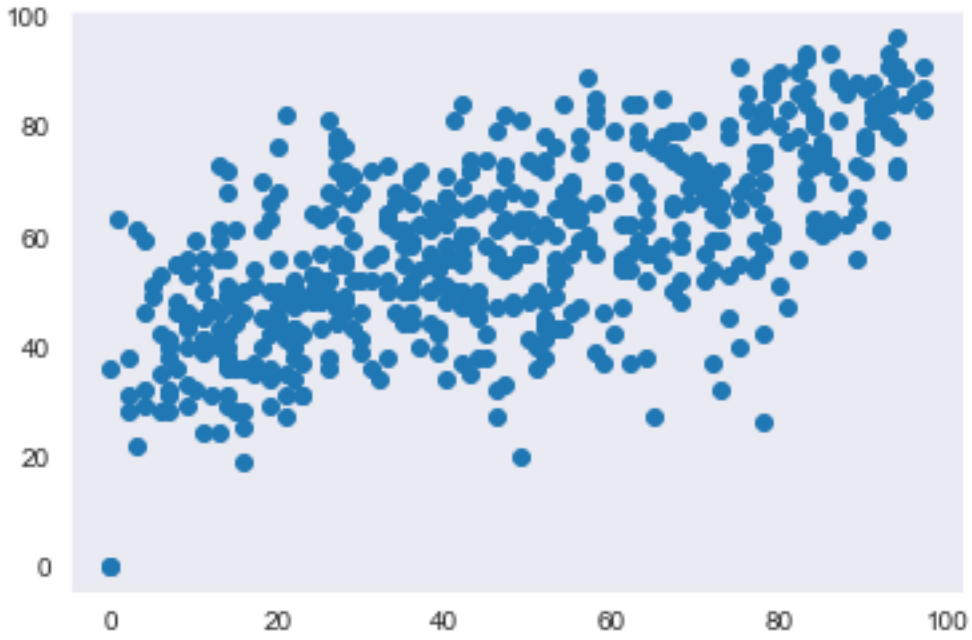


82 Creating a Facet grid

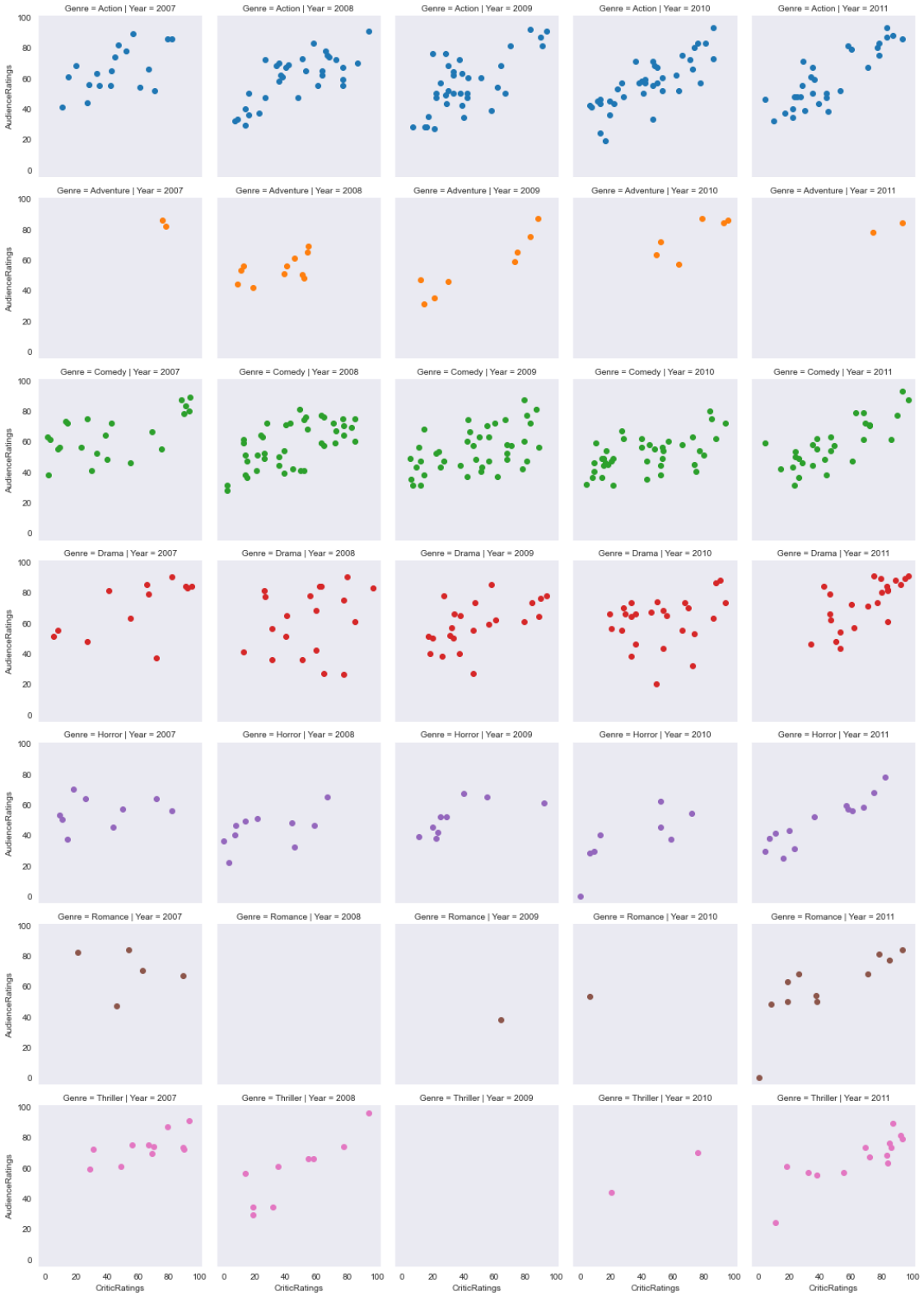
[51]:
```
# g=sns.FacetGrid(movies,row='Genre',hue='Genre')
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')
```


0.0
0.2
0.4
0.6
0.8
1.0

[52]:
```
# g=g.map()
plt.scatter(movies.CriticRatings,movies.AudienceRatings)
plt.show()
```



[53]:
```
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')
g=g.map(plt.scatter,'CriticRatings','AudienceRatings')
plt.show()
```

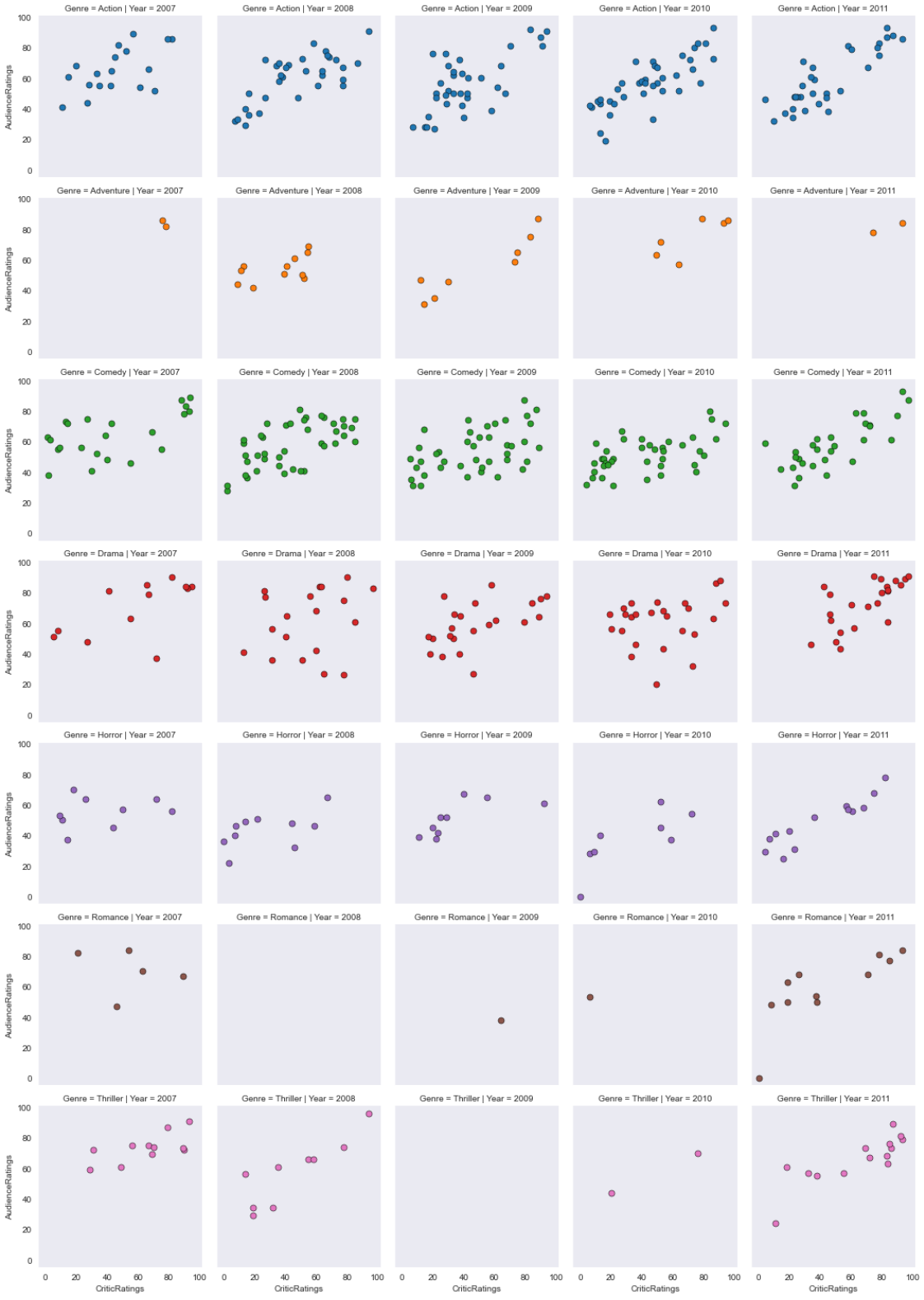

Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007
Genre = Comedy | Year = 2008
Genre = Comedy | Year = 2009
Genre = Comedy | Year = 2010
Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007
Genre = Drama | Year = 2008
Genre = Drama | Year = 2009
Genre = Drama | Year = 2010
Genre = Drama | Year = 2011
Genre = Horror | Year = 2007
Genre = Horror | Year = 2008
Genre = Horror | Year = 2009
Genre = Horror | Year = 2010
Genre = Horror | Year = 2011
Genre = Romance | Year = 2007
Genre = Romance | Year = 2008
Genre = Romance | Year = 2009
Genre = Romance | Year = 2010
Genre = Romance | Year = 2011
Genre = Thriller | Year = 2007
Genre = Thriller | Year = 2008
Genre = Thriller | Year = 2009
Genre = Thriller | Year = 2010
Genre = Thriller | Year = 2011
AudienceRatings
CriticRatings
0
20
40
60
80
100

83 We can papulate with any kind of chart. (Ex. Histograms)

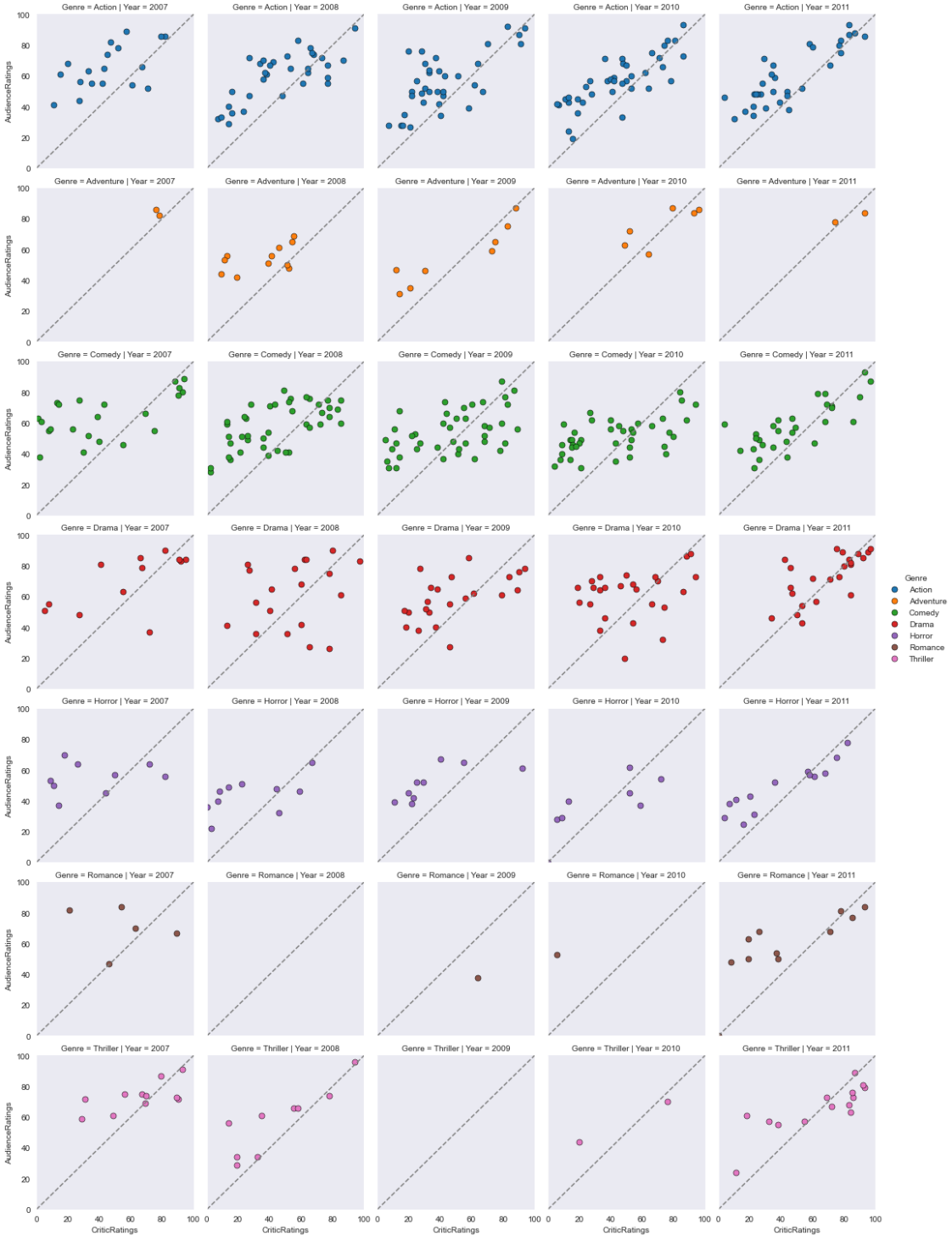
[54]:
```
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')
g=g.map(plt.hist,'BudgetMillion')
plt.show()
```


Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007
Genre = Comedy | Year = 2008
Genre = Comedy | Year = 2009
Genre = Comedy | Year = 2010
Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007
Genre = Drama | Year = 2008
Genre = Drama | Year = 2009
Genre = Drama | Year = 2010
Genre = Drama | Year = 2011
Genre = Horror | Year = 2007
Genre = Horror | Year = 2008
Genre = Horror | Year = 2009
Genre = Horror | Year = 2010
Genre = Horror | Year = 2011
Genre = Romance | Year = 2007
Genre = Romance | Year = 2008
Genre = Romance | Year = 2009
Genre = Romance | Year = 2010
Genre = Romance | Year = 2011
Genre = Thriller | Year = 2007
Genre = Thriller | Year = 2008
Genre = Thriller | Year = 2009
Genre = Thriller | Year = 2010
Genre = Thriller | Year = 2011
0 2 4 6 8 10 12 14 16
0 100 200 300
BudgetMillion

[55]:
```
# Back on
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')
kws=dict(s=50, linewidth=0.5,edgecolor='black')
g=g.map(plt.scatter,'CriticRatings','AudienceRatings',**kws)
plt.show()
```


Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007
Genre = Comedy | Year = 2008
Genre = Comedy | Year = 2009
Genre = Comedy | Year = 2010
Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007
Genre = Drama | Year = 2008
Genre = Drama | Year = 2009
Genre = Drama | Year = 2010
Genre = Drama | Year = 2011
Genre = Horror | Year = 2007
Genre = Horror | Year = 2008
Genre = Horror | Year = 2009
Genre = Horror | Year = 2010
Genre = Horror | Year = 2011
Genre = Romance | Year = 2007
Genre = Romance | Year = 2008
Genre = Romance | Year = 2009
Genre = Romance | Year = 2010
Genre = Romance | Year = 2011
Genre = Thriller | Year = 2007
Genre = Thriller | Year = 2008
Genre = Thriller | Year = 2009
Genre = Thriller | Year = 2010
Genre = Thriller | Year = 2011
AudienceRatings
CriticRatings
0
20
40
60
80
100

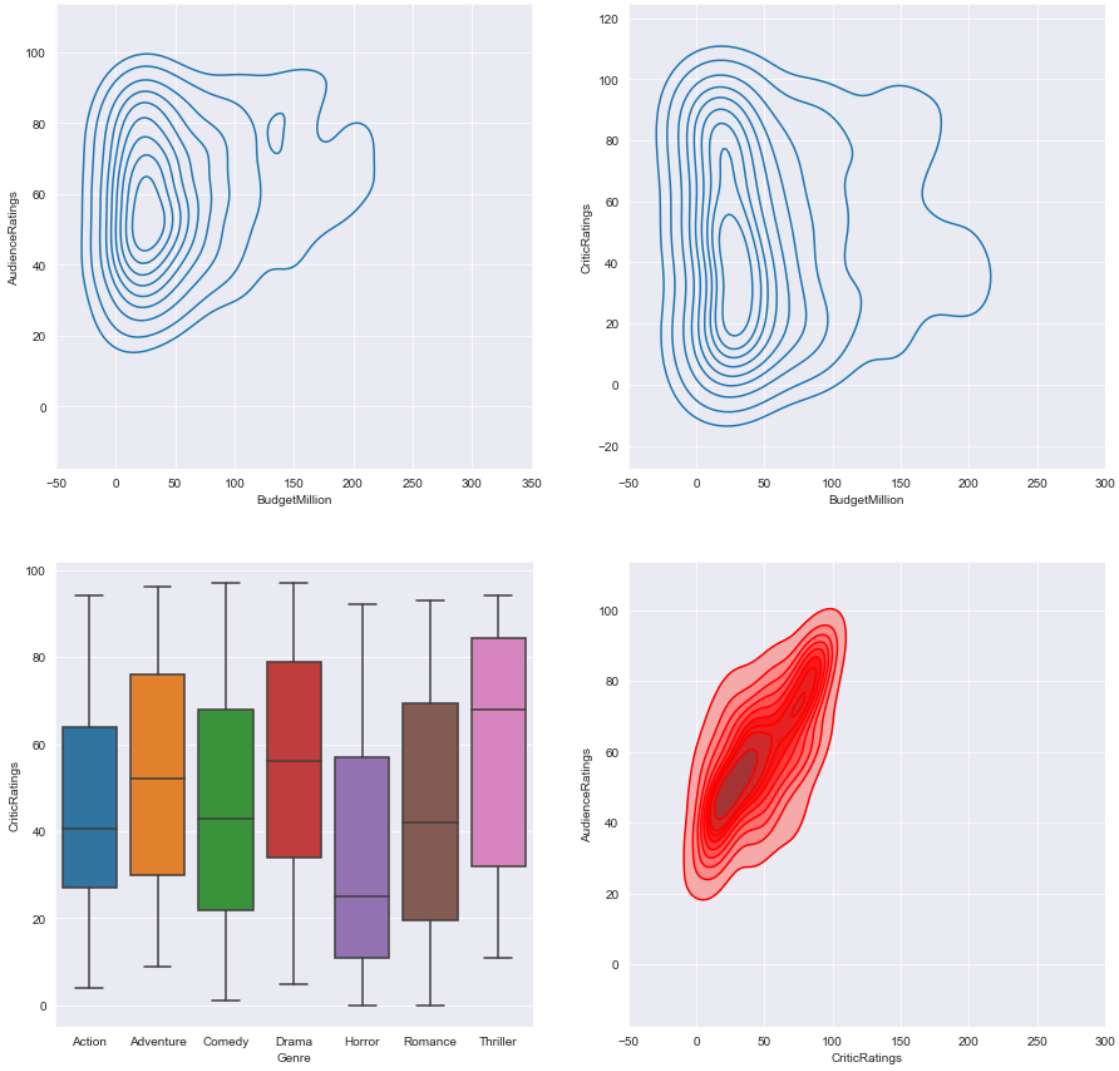
[56]:
```
g=sns.FacetGrid(movies,row='Genre',col='Year',hue='Genre')
kws=dict(s=50, linewidth=0.5,edgecolor='black')
g=g.map(plt.scatter,'CriticRatings','AudienceRatings',**kws)
g.set(xlim=(0,100),ylim=(0,100))
for ax in g.axes.flat:
    ax.plot((0,100),(0,100),c='gray',ls='--')
g.add_legend()
plt.show()
```


Genre = Action | Year = 2007
Genre = Action | Year = 2008
Genre = Action | Year = 2009
Genre = Action | Year = 2010
Genre = Action | Year = 2011
Genre = Adventure | Year = 2007
Genre = Adventure | Year = 2008
Genre = Adventure | Year = 2009
Genre = Adventure | Year = 2010
Genre = Adventure | Year = 2011
Genre = Comedy | Year = 2007
Genre = Comedy | Year = 2008
Genre = Comedy | Year = 2009
Genre = Comedy | Year = 2010
Genre = Comedy | Year = 2011
Genre = Drama | Year = 2007
Genre = Drama | Year = 2008
Genre = Drama | Year = 2009
Genre = Drama | Year = 2010
Genre = Drama | Year = 2011
Genre = Horror | Year = 2007
Genre = Horror | Year = 2008
Genre = Horror | Year = 2009
Genre = Horror | Year = 2010
Genre = Horror | Year = 2011
Genre = Romance | Year = 2007
Genre = Romance | Year = 2008
Genre = Romance | Year = 2009
Genre = Romance | Year = 2010
Genre = Romance | Year = 2011
Genre = Thriller | Year = 2007
Genre = Thriller | Year = 2008
Genre = Thriller | Year = 2009
Genre = Thriller | Year = 2010
Genre = Thriller | Year = 2011
AudienceRatings
CriticRatings
0
20
40
60
80
100
Genre
Action
Adventure
Comedy
Drama
Horror
Romance
Thriller

84 Creating a Dashboard

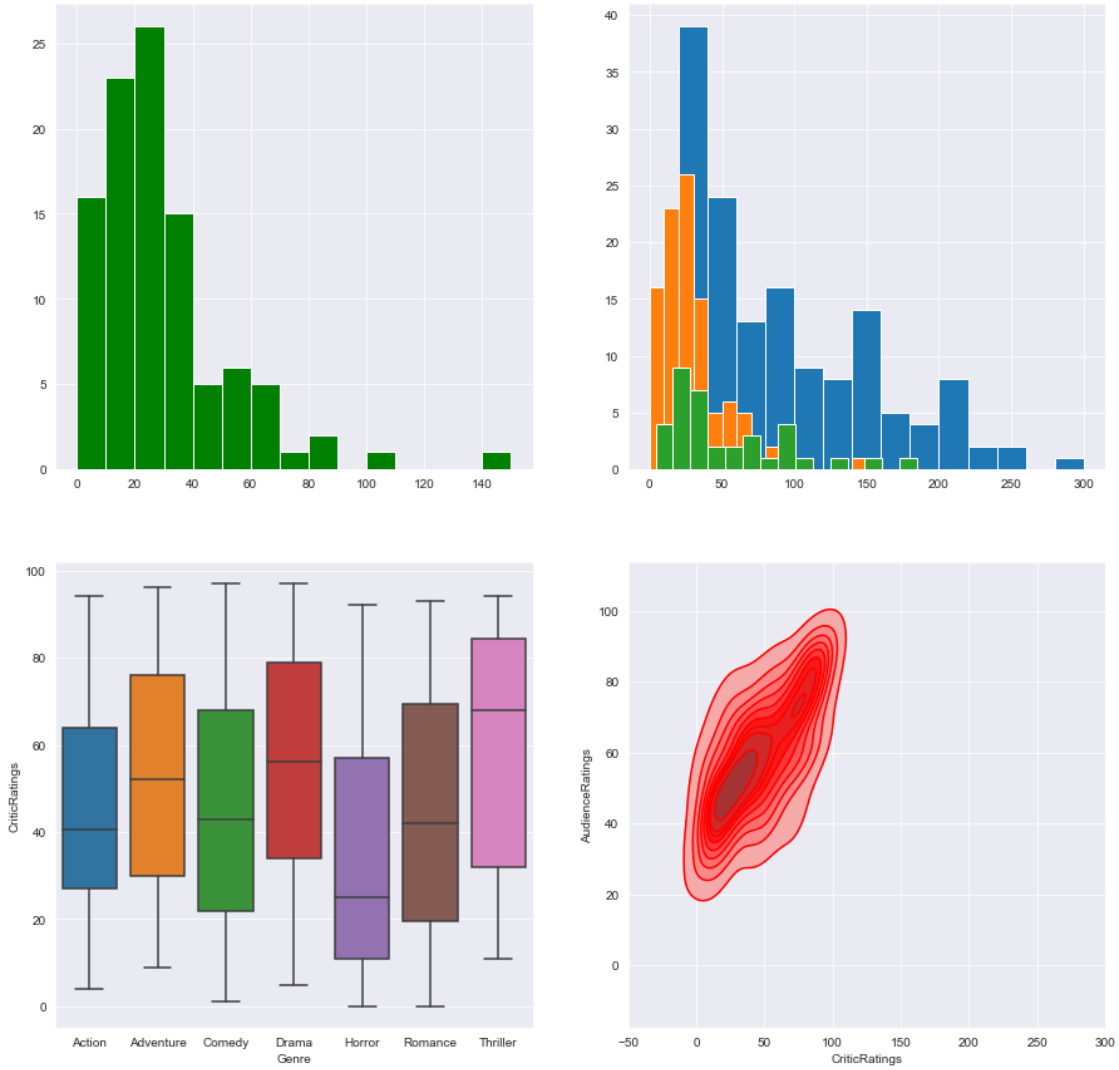
[57]:
```
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

[58]:
```
sns.set_style("darkgrid")
f, axes=plt.subplots(2,2,figsize=(15,15))
k1=sns.kdeplot(movies.BudgetMillion,movies.AudienceRatings,ax=axes[0,0])
k2=sns.kdeplot(movies.BudgetMillion,movies.CriticRatings,ax=axes[0,1])
w=sns.boxplot(data=movies,x='Genre',y='CriticRatings',ax=axes[1,0])

k1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               shade=True,shade_lowest=False,color='Red',ax=axes[1,1])
kw1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               color='Red',ax=axes[1,1])
k1.set(xlim=(-50,300))
k2.set(xlim=(-50,300))
plt.show()
```



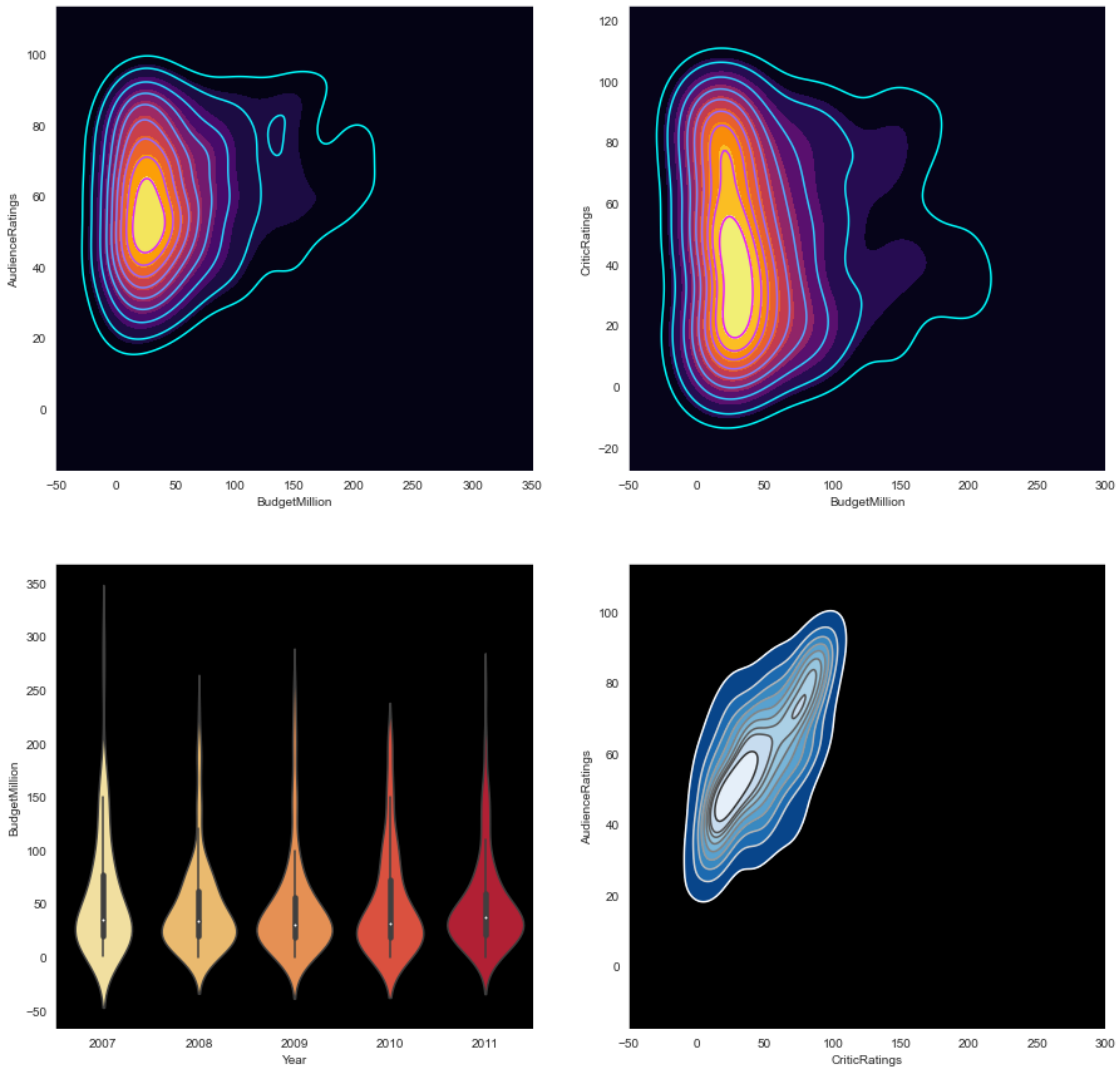
[59]:
```
sns.set_style("darkgrid")
f, axes=plt.subplots(2,2,figsize=(15,15))
#k1=sns.kdeplot(movies.BudgetMillion,movies.AudienceRatings,ax=axes[0,0])
#k2=sns.kdeplot(movies.BudgetMillion,movies.CriticRatings,ax=axes[0,1])
axes[0,0].hist(movies[movies.Genre=='Drama'].
 ↪BudgetMillion,bins=15,color='Green') # as matplotlib

axes[0,1].hist(movies[movies.Genre=='Action'].BudgetMillion,bins=15)
axes[0,1].hist(movies[movies.Genre=='Drama'].BudgetMillion,bins=15)
axes[0,1].hist(movies[movies.Genre=='Thriller'].BudgetMillion,bins=15)


w=sns.boxplot(data=movies,x='Genre',y='CriticRatings',ax=axes[1,0])
```

```
k1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               shade=True,shade_lowest=False,color='Red',ax=axes[1,1])
kw1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               color='Red',ax=axes[1,1])
k1.set(xlim=(-50,300))
k2.set(xlim=(-50,300))
plt.show()
```



[60]:
```
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

[61]:
```
sns.set_style("dark",{"axes.facecolor":"black"}) # white, whitegrid, dark,␣
↪darkgrid, ticks
f, axes=plt.subplots(2,2,figsize=(15,15))
# plot[0,0]
k1=sns.kdeplot(movies.BudgetMillion,movies.AudienceRatings,\
               shade=True,shade_lowest=True,cmap='inferno',ax=axes[0,0])

k1b=sns.kdeplot(movies.BudgetMillion,movies.AudienceRatings,\
               cmap='cool',ax=axes[0,0])
# plot[0,1]
k2=sns.kdeplot(movies.BudgetMillion,movies.CriticRatings,\
                shade=True,shade_lowest=True,cmap='inferno',ax=axes[0,1])
k2b=sns.kdeplot(movies.BudgetMillion,movies.CriticRatings,ax=axes[0,1],
                cmap='cool')
# plot[1,0]
w=sns.violinplot(data=movies,x='Year',y='BudgetMillion',\
               palette='YlOrRd',ax=axes[1,0])
# plot[1,1]
k1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               shade=True,shade_lowest=False,cmap='Blues_r',ax=axes[1,1])
kw1=sns.kdeplot(movies.CriticRatings,movies.AudienceRatings, \
               cmap='gist_gray_r',color='Red',ax=axes[1,1])

k1.set(xlim=(-50,300))
k2.set(xlim=(-50,300))
plt.show()
```



[80]:
```
list1=list([])
mylabel=list([])
for gen in movies.Genre.cat.categories:
    list1.append(movies[movies.Genre==gen].BudgetMillion)
    mylabel.append(gen)

sns.set_style("whitegrid")
fig, ax=plt.subplots()
fig.set_size_inches(11.7,8.27) # A4 size paper
h=plt.hist(list1, bins=30,stacked=True,rwidth=1,label=mylabel)
plt.title("Movie Budget Distribution" ,fontsize=35,␣
 ↪color='darkblue',fontname='console')
plt.ylabel("Number of Movies",fontsize=25,color='Red')
plt.xlabel("Budget",fontsize=25,color='Red')
```

```
plt.yticks(fontsize=25)
plt.xticks(fontsize=25)
plt.legend()
plt.legend(loc='upper left',bbox_to_anchor=(1,1))
plt.show()
```