CS168 Spring Assignment 4

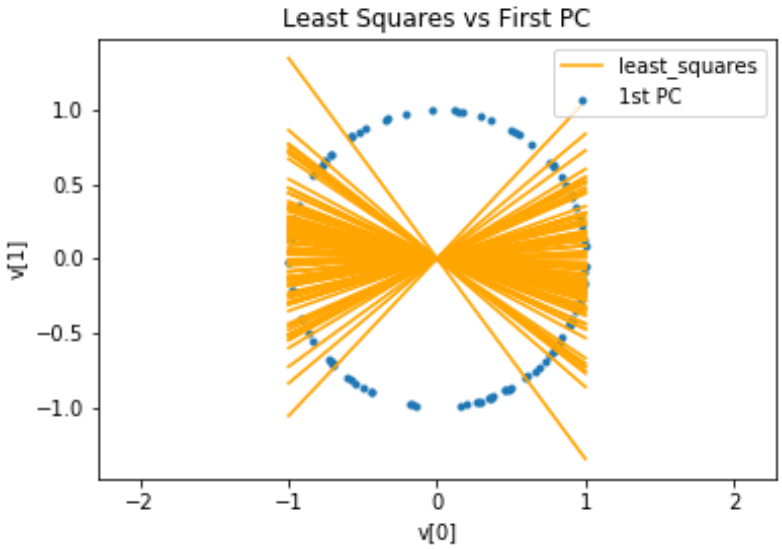
SUNet ID(s): sasm
Name(s): Adam Stanford-Moore
Collaborators: None

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

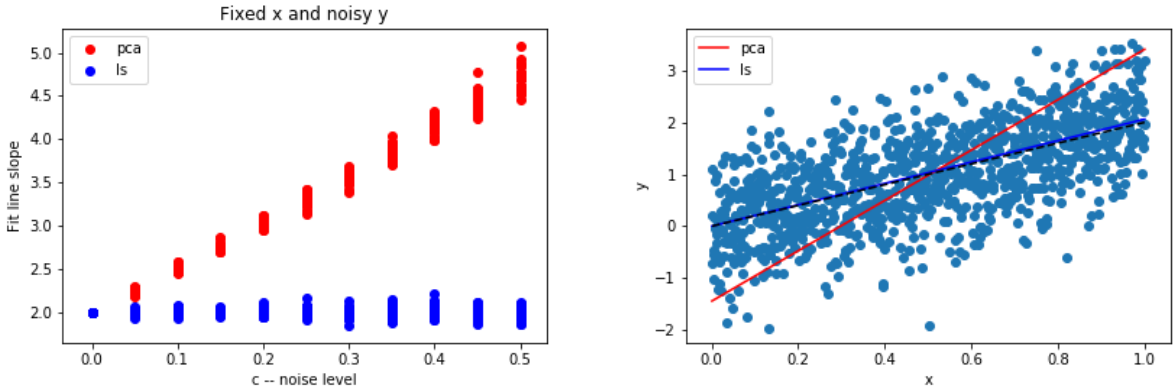
Part 1

(a)

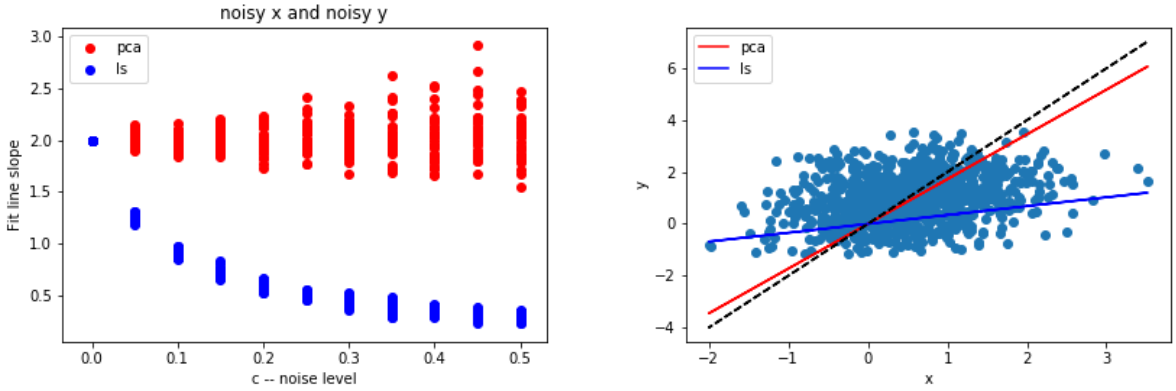
(b) The first principle component is a uniformly random vector over the circle while the least squares vector is preferably more horizontal than vertical, to minimize vertical distance from points to line



(c) It seems that least squares fits better for fixed x.



(d) It seems that PCA fits better for noisy x and y.



(e) (i) PCA does poorly with noise only in Y since it doesn't take into account the fact that x is fixed and tries to fit the data freely. PCA minimizes the perpendicular distance of points to the principle component, independent of coordinate axis.

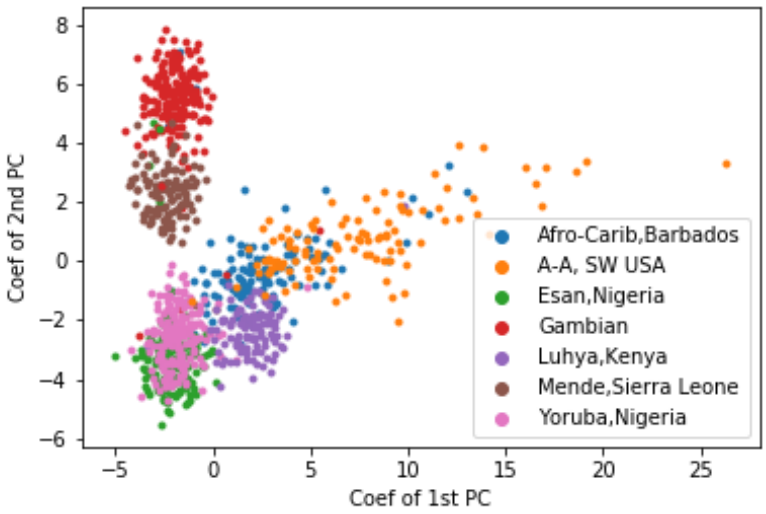
(ii) PCA does well with noise in both X and Y since it fits the data freely in both x and y with no preference to there being an independent and dependent variable (least squares requires there to be a dependent y and independent x).

(iii) Least squares assumes fixed x values and minimizes the distance from x to the prediction a*x. If x is noisy in addition to y, then the the vertical distance to the line is less physical and the data is spread much more in the horizontal direction.

Part 2

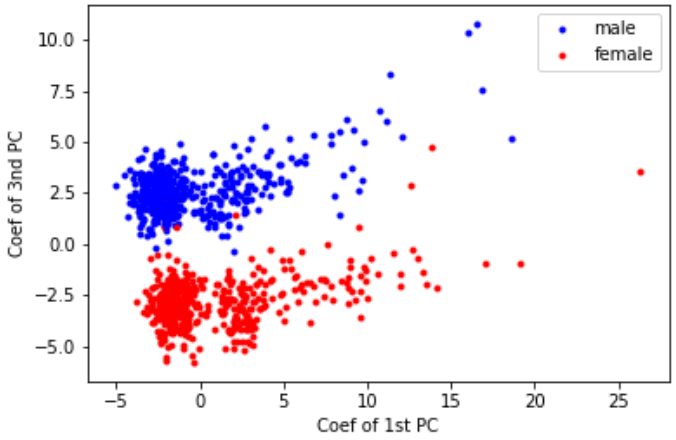
(a) The 995 datapoints (people) exist in $R^1 0101$, so the each principle component would also be a vector in $R^1 0101$.

(b) Projections of genetic data onto top two principle components.



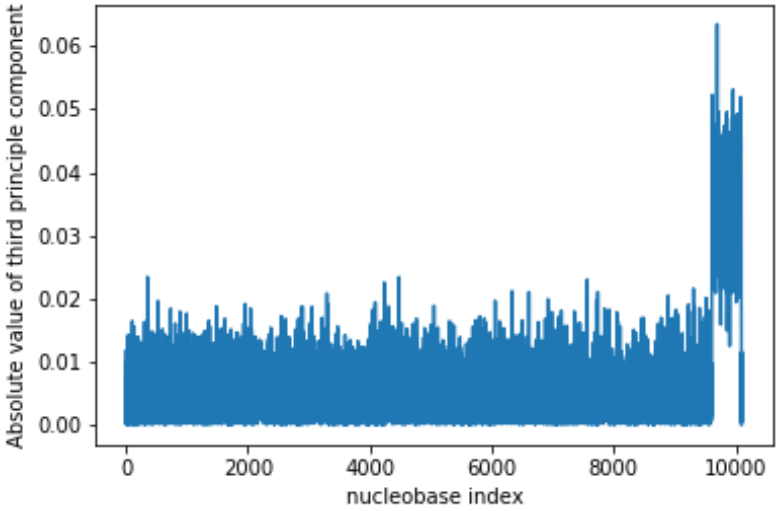
(c) Gambia (GWD) and Sierra Leone (MSL) are close together. Esan and Yorubi in Nigeria are also close to each other. Also we see that African Caribbeans and African Americans have much more genetic diversity, likely as their ancestors were brought from different regions of western Africa through the slave trade.

(d) The third component represents the gender of the individual!



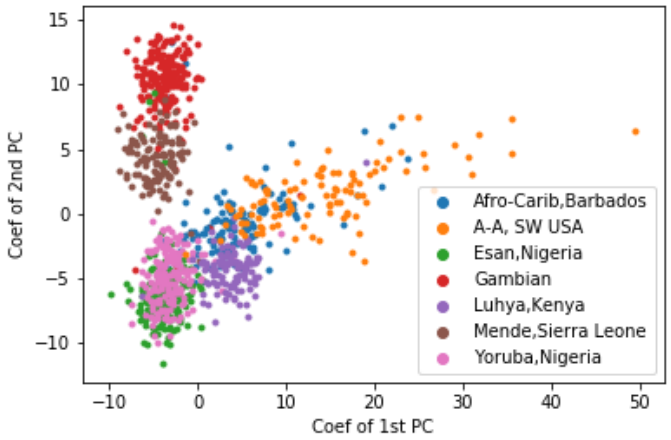
(e) The third component represents the gender of the individual!

(f) Looks like the Y chromosome is in the location from 9600 to 10100. This region is most important in differentiating males and females!

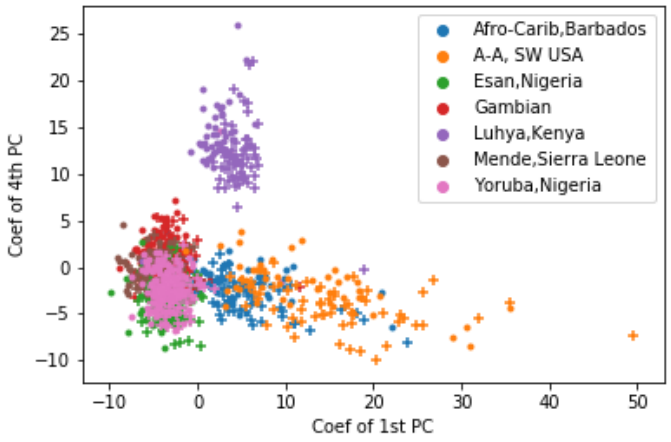


(g) Instead of creating a binary array we can map each letter to a number, for instance 'A' to 1, 'C' to 2, 'G' to 3 and 'T' to 4. Now we represent the dataset exactly. Alternatively we could represent each person a 10101*4 dimensional array with each base represented as a 4 dimensional one-hot encoding.

(h) Unfortunately, with this more complicated representation, not much changed.



(i) The 4th component seems to distinguish Kenyans from the rest of the group. Maybe Kenyans have a particular gene of some sort that the West Africans do not.



pset4_code

May 4, 2020

[15]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

[78]:
```
def pca_recover(X,Y):
    data = np.vstack((X,Y)).T
    pca = PCA(n_components=2)
    principalComponents = pca.fit(data).components_
    first_PC = principalComponents[0]
    return first_PC[1]/first_PC[0]
def ls_recover(X,Y):
    X_bar = np.mean(X)
    Y_bar = np.mean(Y)
    return (X - X_bar) @ (Y - Y_bar) / np.linalg.norm(X-X_bar)**2
```

[79]:
```
X = np.arange(0.001,1,.001)
Y = 2*X
print(pca_recover(X,Y))
print(ls_recover(X,Y))
```

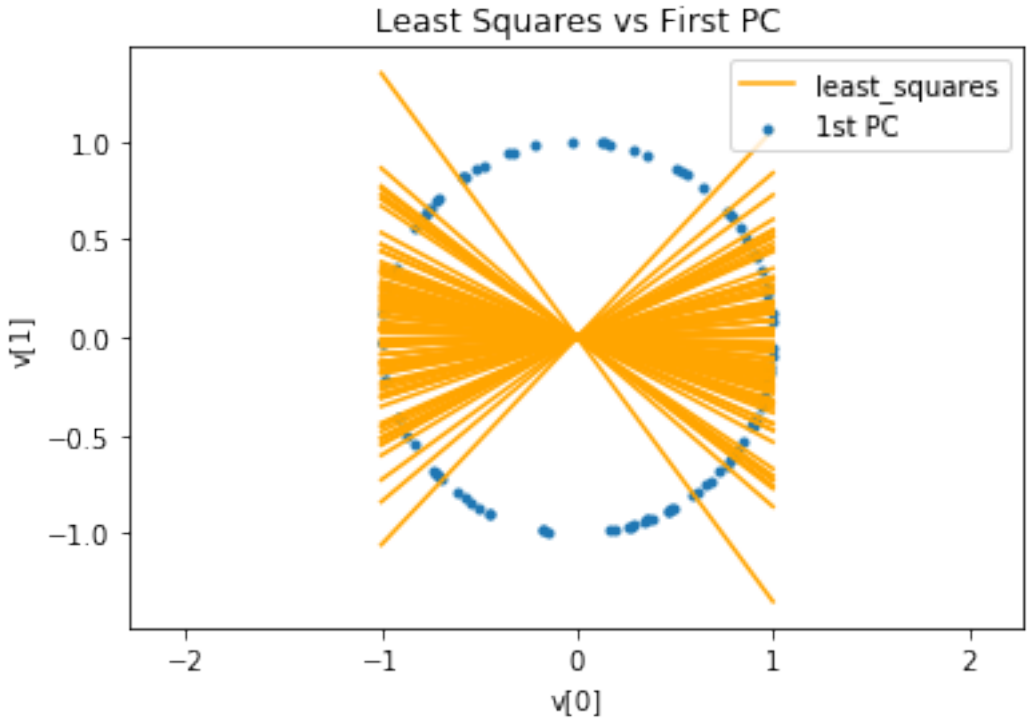
```
2.0
2.0
```

0.1 part b

[70]:
```
x,y = [],[]

for i in range(100):
    X = np.random.rand(10)
    Y = np.random.rand(10)
    v = pca_recover(X,Y)
    plt.plot([-1,1],np.array([-1,1])*ls_recover(X,Y),color='orange')
    x.append(v[0])
    y.append(v[1])
plt.plot([],[],color='orange',label='least_squares')
plt.scatter(x,y,marker='.',label='1st PC')
plt.legend()
plt.gca().axis('equal')
```

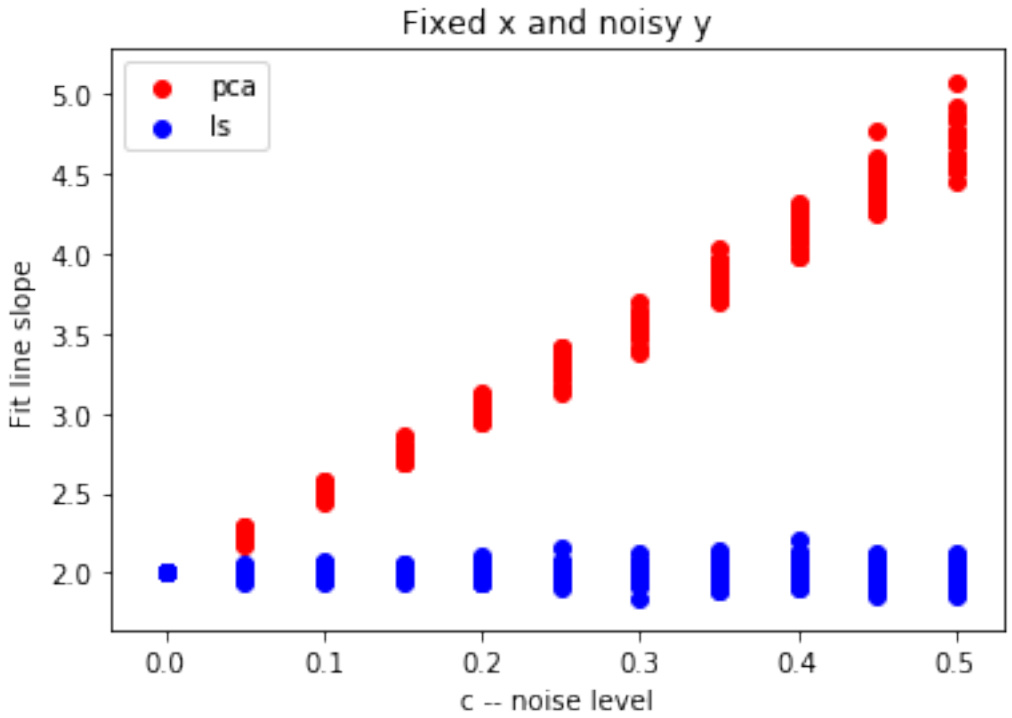
```
plt.title("Least Squares vs First PC")
plt.xlabel("v[0]")
plt.ylabel("v[1]")
plt.savefig('1b.png')
plt.show()
```



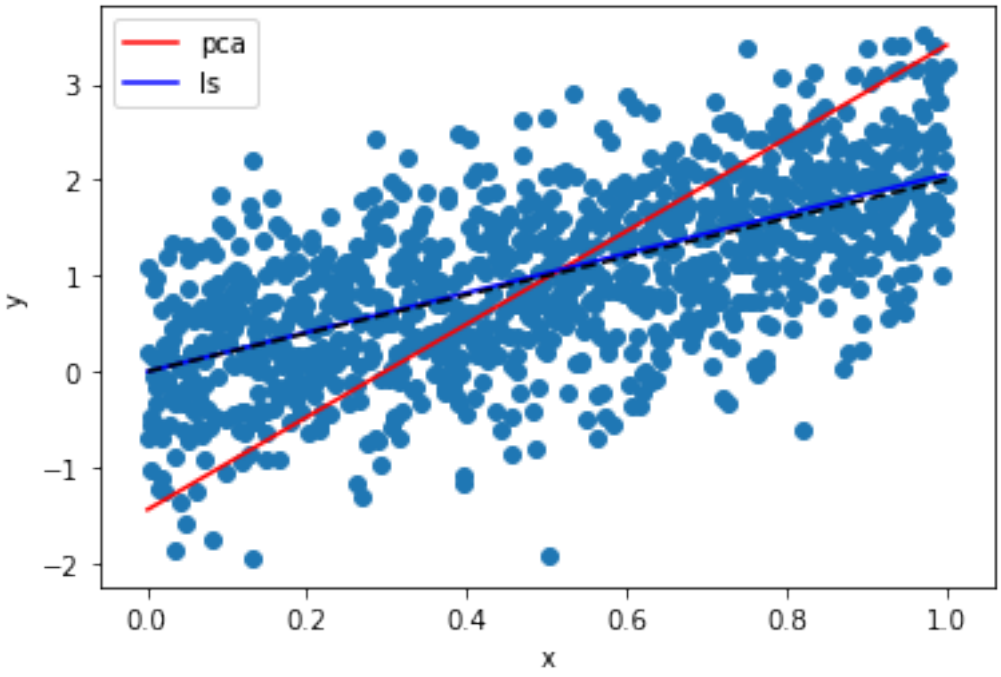
0.2 part c

[86]:
```
X = np.linspace(0.001,1,1000)
cs = np.arange(0,0.55,0.05)
num_iter = 30
for c in cs:
    pca = []
    ls = []
    for iter in range(num_iter):
        Y = 2*X + np.random.randn(1000)*np.sqrt(c)
        pca.append(pca_recover(X,Y))
        ls.append(ls_recover(X,Y))
    plt.scatter([c]*num_iter, pca,color='r')
    plt.scatter([c]*num_iter, ls,color='b')
plt.scatter([], [],color='r',label='pca')
plt.scatter([], [],color='b',label='ls')
plt.legend()
```

```
plt.title('Fixed x and noisy y')
plt.xlabel('c -- noise level')
plt.ylabel('Fit line slope')
plt.savefig('1c.png')
plt.show()
```

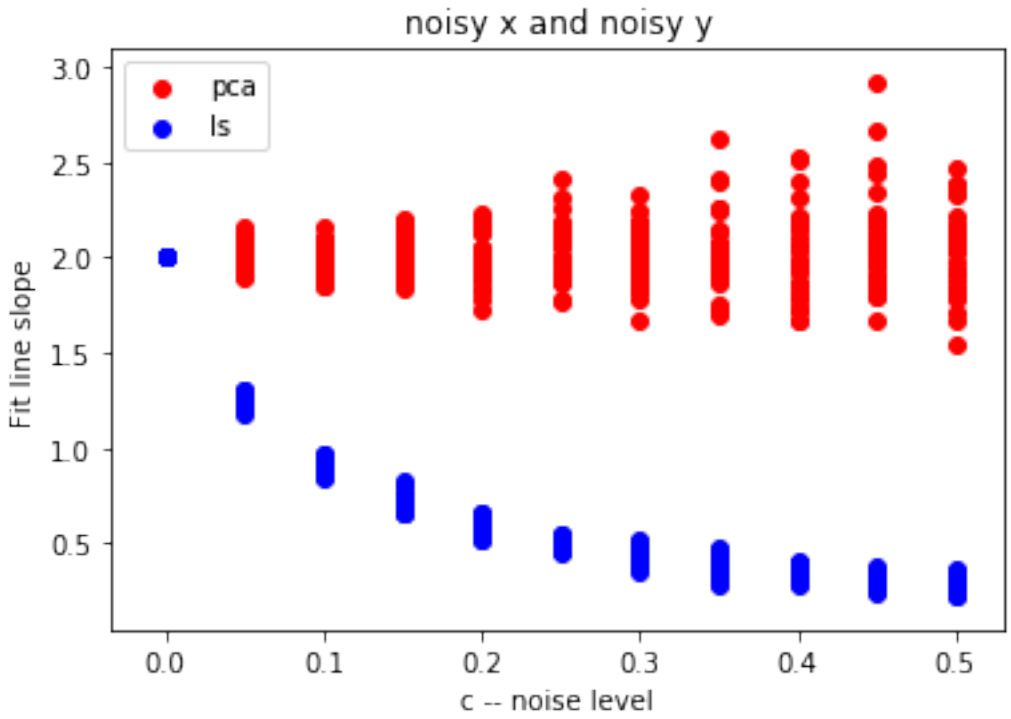


[116]:
```
c = 0.5
X = np.linspace(0.001,1,1000)
Y = 2*X + np.random.randn(1000)*np.sqrt(c)
plt.scatter(X,Y)
mu_x = np.mean(X)
mu_y = np.mean(Y)
plt.plot(X,pca_recover(X,Y)*(X-mu_x) + mu_y,label='pca',color='r')
plt.plot(X,ls_recover(X,Y)*X,label='ls',color='b')
plt.plot(X,2*X,color='k',linestyle='--')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('1c_iter.png')
plt.show()
```

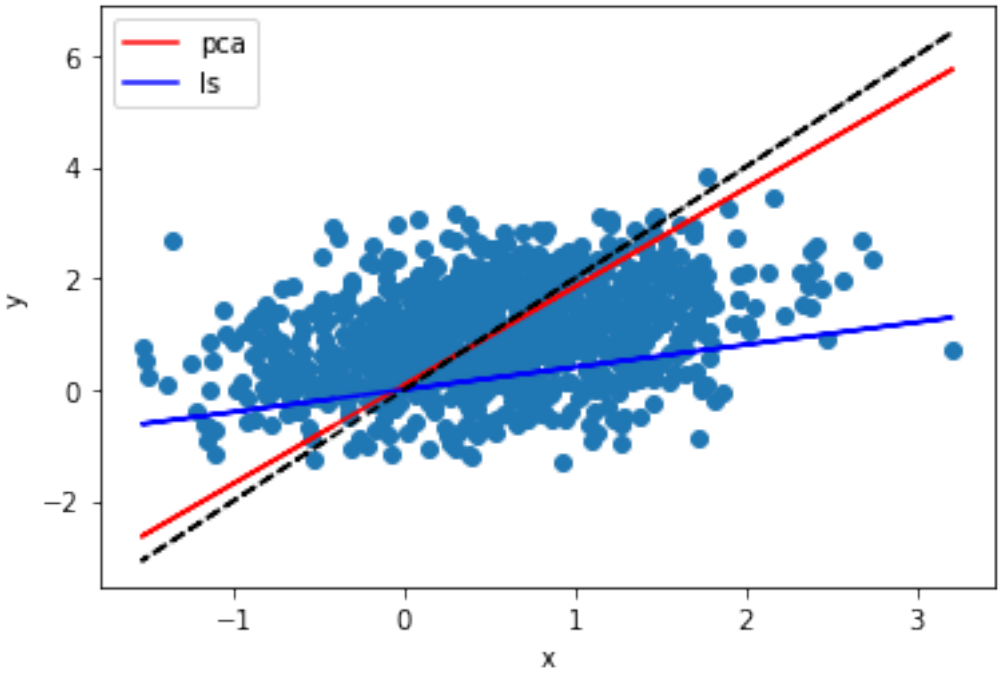


0.3 part d

[87]:
```
cs = np.arange(0,0.55,0.05)
num_iter = 30
for c in cs:
    pca = []
    ls = []
    for iter in range(num_iter):
        x = np.linspace(0.001,1,1000)
        X = x + np.random.randn(1000)*np.sqrt(c)
        Y = 2*x + np.random.randn(1000)*np.sqrt(c)
        pca.append(pca_recover(X,Y))
        ls.append(ls_recover(X,Y))
    plt.scatter([c]*num_iter, pca,color='r')
    plt.scatter([c]*num_iter, ls,color='b')
plt.scatter([], [],color='r',label='pca')
plt.scatter([], [],color='b',label='ls')
plt.legend()
plt.title('noisy x and noisy y')
plt.xlabel('c -- noise level')
plt.ylabel('Fit line slope')
plt.savefig('1d.png')
plt.show()
```



[118]:
```
c = 0.5
x = np.linspace(0.001,1,1000)
X = x + np.random.randn(1000)*np.sqrt(c)
Y = 2*x + np.random.randn(1000)*np.sqrt(c)
plt.scatter(X,Y)
mu_x = np.mean(X)
mu_y = np.mean(Y)
plt.plot(X,pca_recover(X,Y)*(X-mu_x) + mu_y,label='pca',color='r')
plt.plot(X,ls_recover(X,Y)*X,label='ls',color='b')
plt.plot(X,2*X,color='k',linestyle='--')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('1d_iter.png')
plt.show()
```



0.4 part 2

[120]:
```
genomes = np.loadtxt('p4dataset2020.txt',dtype='str')
```

[121]:
```
genomes.shape
```

[121]:
```
(995, 10104)
```

[122]:
```
genomes
```

[122]:
```
array([['HG01879', '1', 'ACB', …, 'T', 'G', 'A'],
       ['HG01880', '2', 'ACB', …, 'T', 'G', 'G'],
       ['HG01881', '2', 'ACB', …, 'T', 'G', 'G'],
       …,
       ['NA20364', '2', 'ASW', …, 'T', 'G', 'G'],
       ['NA20412', '2', 'ASW', …, 'T', 'G', 'G'],
       ['NA20413', '1', 'ASW', …, 'T', 'G', 'G']], dtype='<U7')
```

[127]:
```
from scipy import stats
modes = stats.mode(genomes[:,3:])[0]
```

[137]:
```
X = (genomes[:,3:] != modes).astype(float)
```

[ ]:
```

```

0.5 part b

[147]:
```
pca = PCA(n_components=2)
projected = pca.fit_transform(X)
```

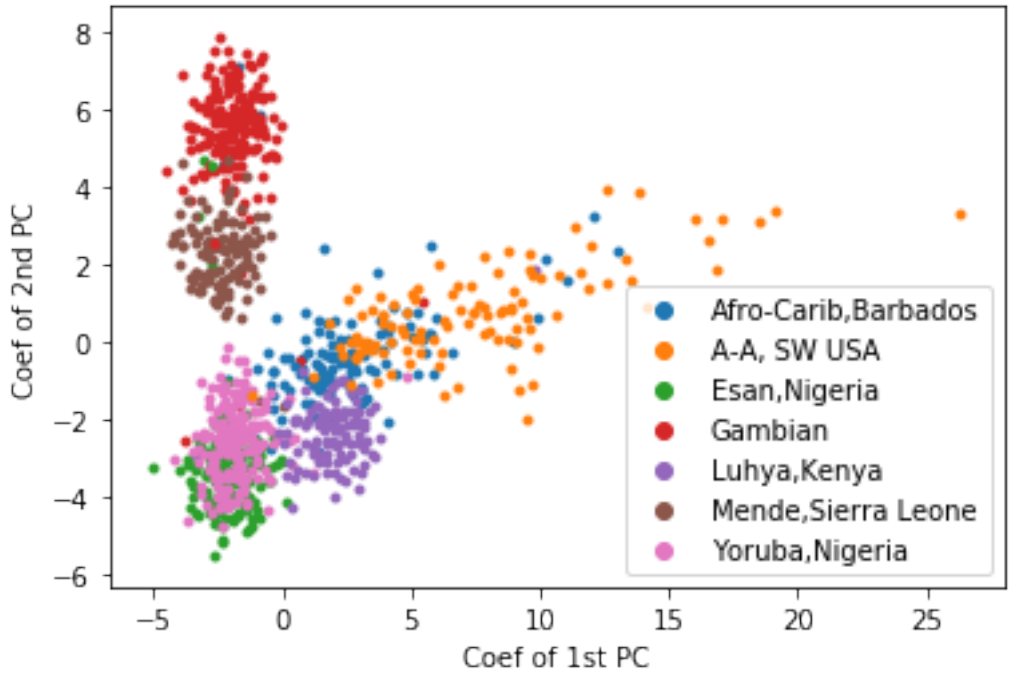
[169]:
```
region_codes = np.unique(genomes[:,2]).tolist()
colors = ['C%d'%i for i in range(len(region_codes))]
```

[187]:
```
regions = ['Afro-Carib,Barbados', 'A-A, SW USA',␣
↪'Esan,Nigeria','Gambian','Luhya,Kenya','Mende,Sierra Leone',␣
↪'Yoruba,Nigeria' ]
```

[188]:
```
for a,b in zip(regions,region_codes):
    print("%s : %s" %(b,a))
```

```
ACB : Afro-Carib,Barbados
ASW : A-A, SW USA
ESN : Esan,Nigeria
GWD : Gambian
LWK : Luhya,Kenya
MSL : Mende,Sierra Leone
YRI : Yoruba,Nigeria
```

[189]:
```
for i in range(projected.shape[0]):
    ind = region_codes.index(region)
    region = genomes[i,2]
    color = colors[ind]
    plt.scatter(projected[i,0],projected[i,1],color=color,marker='.')
for region in region_codes:
    ind = region_codes.index(region)
    color = colors[ind]
    plt.scatter([],[],color=color,label=regions[ind])
#plt.xlim([-7,50])
plt.legend()
plt.xlabel("Coef of 1st PC")
plt.ylabel("Coef of 2nd PC")
plt.savefig('2b')
```



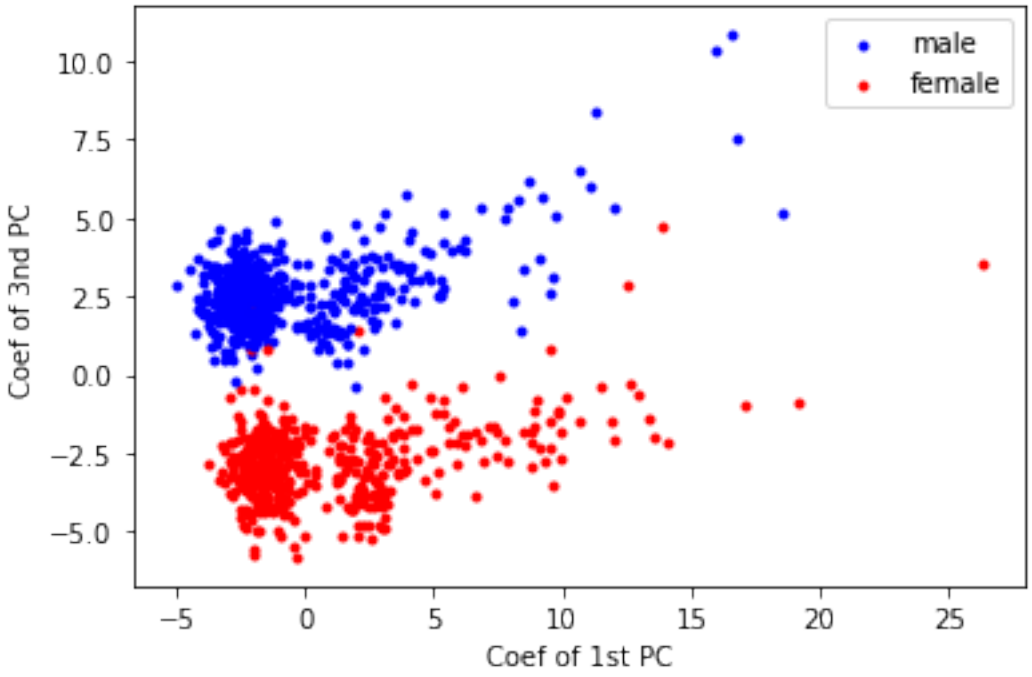
[ ]:

0.6 part d

[190]:
```
pca = PCA(n_components=3)
projected = pca.fit_transform(X)
```

[199]:
```
for i in range(projected.shape[0]):
    ind = region_codes.index(region)
    region = genomes[i,2]
    #color = colors[ind]
    sex = 'male' if genomes[i,1] == '1' else 'female'
    color = 'r' if sex=='female' else 'b'
    marker = '.' #if genomes[i,1] == '1' else '+'
    plt.scatter(projected[i,0],projected[i,2],color=color,marker=marker)
plt.scatter([],[],color='b',marker='.',label='male')
plt.scatter([],[],color='r',marker='.',label='female')
#for region in region_codes:
#    ind = region_codes.index(region)
#    color = colors[ind]
#    plt.scatter([],[],color=color,label=regions[ind])
#plt.xlim([-7,50])
plt.legend()
plt.xlabel("Coef of 1st PC")
```

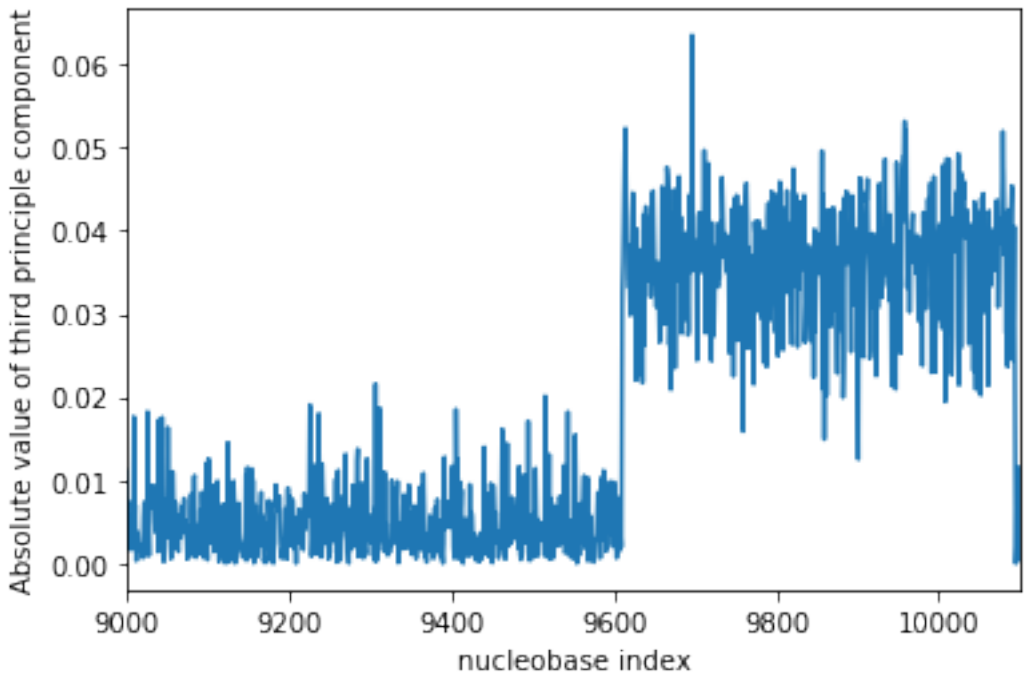
```
plt.ylabel("Coef of 3nd PC")
plt.savefig('2d')
```



0.7 part f

[205]:
```
plt.plot(np.abs(pca.components_[2]))
plt.xlabel('nucleobase index')
plt.ylabel('Absolute value of third principle component')
plt.xlim([9000,10101])
#plt.savefig('2f.png')
```

[205]: (9000, 10101)



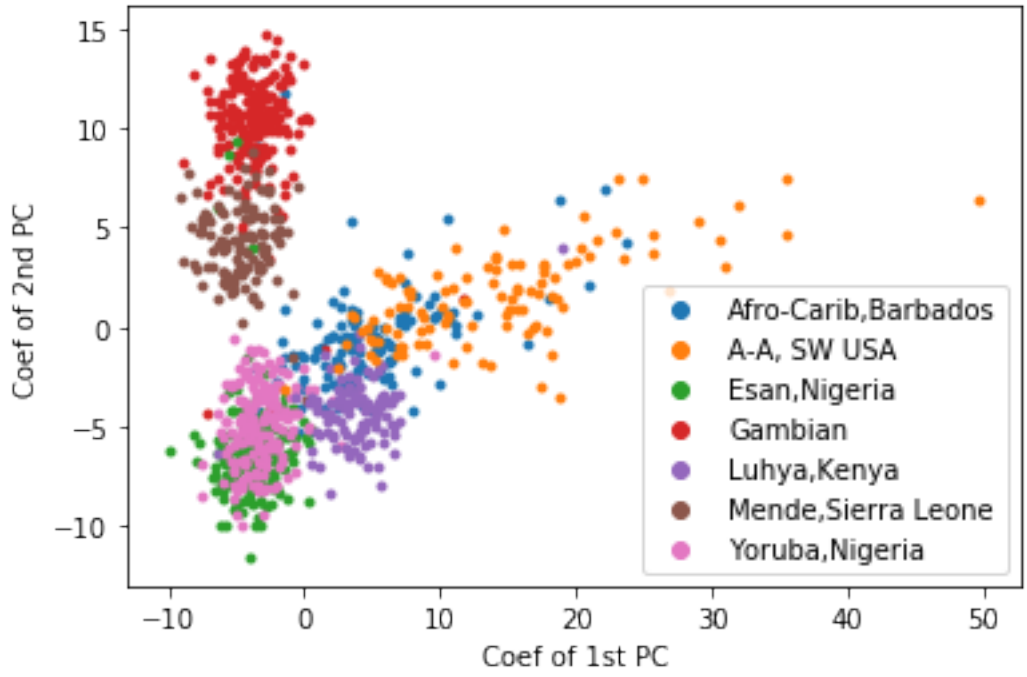
0.8 Bonus

[211]:
```
Y = genomes[:,3:]
Y[Y=='A'] = 1
Y[Y=='C'] = 2
Y[Y=='G'] = 3
Y[Y=='T'] = 4
Y = Y.astype(float)
```

[213]:
```
pca = PCA(n_components=4)
projected = pca.fit_transform(Y)
```

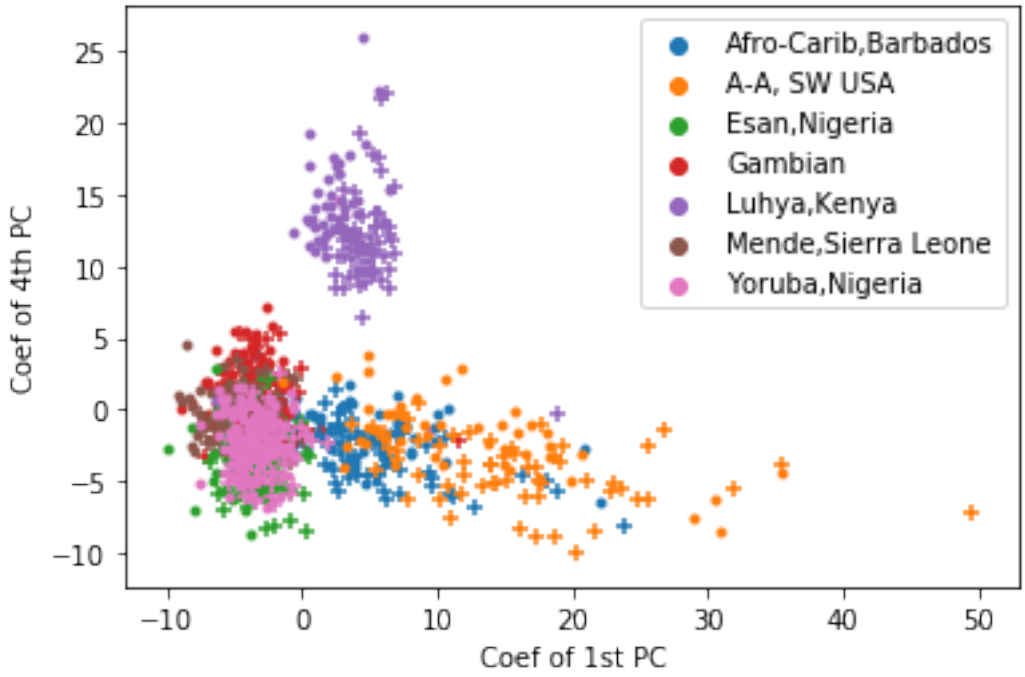
[217]:
```
for i in range(projected.shape[0]):
    ind = region_codes.index(region)
    region = genomes[i,2]
    color = colors[ind]
    plt.scatter(projected[i,0],projected[i,1],color=color,marker='.')
for region in region_codes:
    ind = region_codes.index(region)
    color = colors[ind]
    plt.scatter([],[],color=color,label=regions[ind])
#plt.xlim([-7,50])
plt.legend()
plt.xlabel("Coef of 1st PC")
```

```
plt.ylabel("Coef of 2nd PC")
plt.savefig('2h')
```



0.9 2i

[219]:
```
for i in range(projected.shape[0]):
    ind = region_codes.index(region)
    region = genomes[i,2]
    color = colors[ind]
    sex = 'male' if genomes[i,1] == '1' else 'female'
    marker = '.' if sex=='male' else '+'
    plt.scatter(projected[i,0],projected[i,3],color=color,marker=marker)
for region in region_codes:
    ind = region_codes.index(region)
    color = colors[ind]
    plt.scatter([],[],color=color,label=regions[ind])
#plt.xlim([-7,50])
plt.legend()
plt.xlabel("Coef of 1st PC")
plt.ylabel("Coef of 4th PC")
plt.savefig('2i')
```


Afro-Carib,Barbados
A-A, SW USA
Esan,Nigeria
Gambian
Luhya,Kenya
Mende,Sierra Leone
Yoruba,Nigeria
Coef of 4th PC
Coef of 1st PC
25
20
15
10
5
0
−5
−10
−10
0
10
20
30
40
50

[ ]: