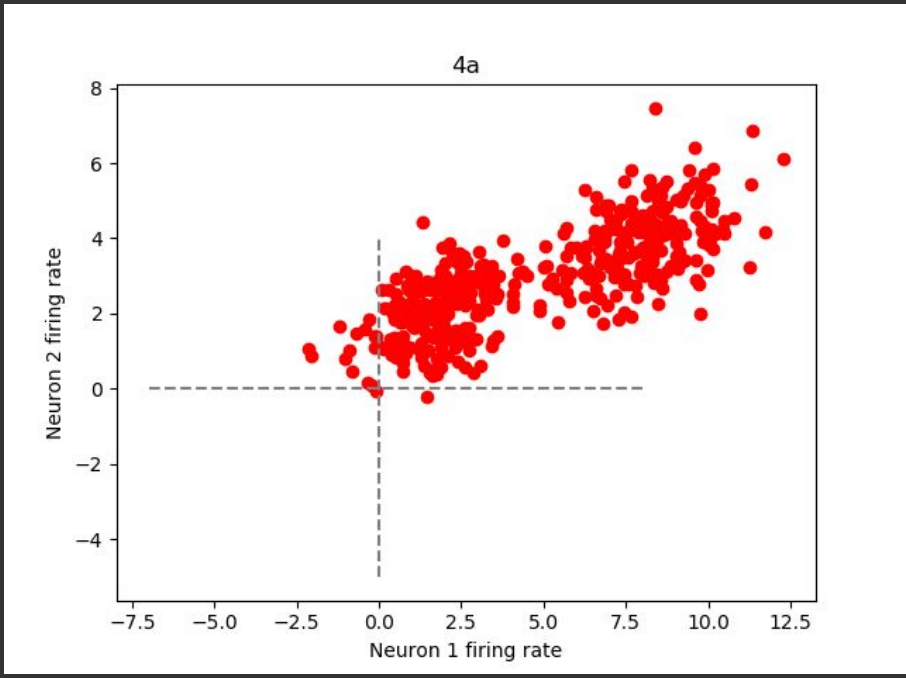
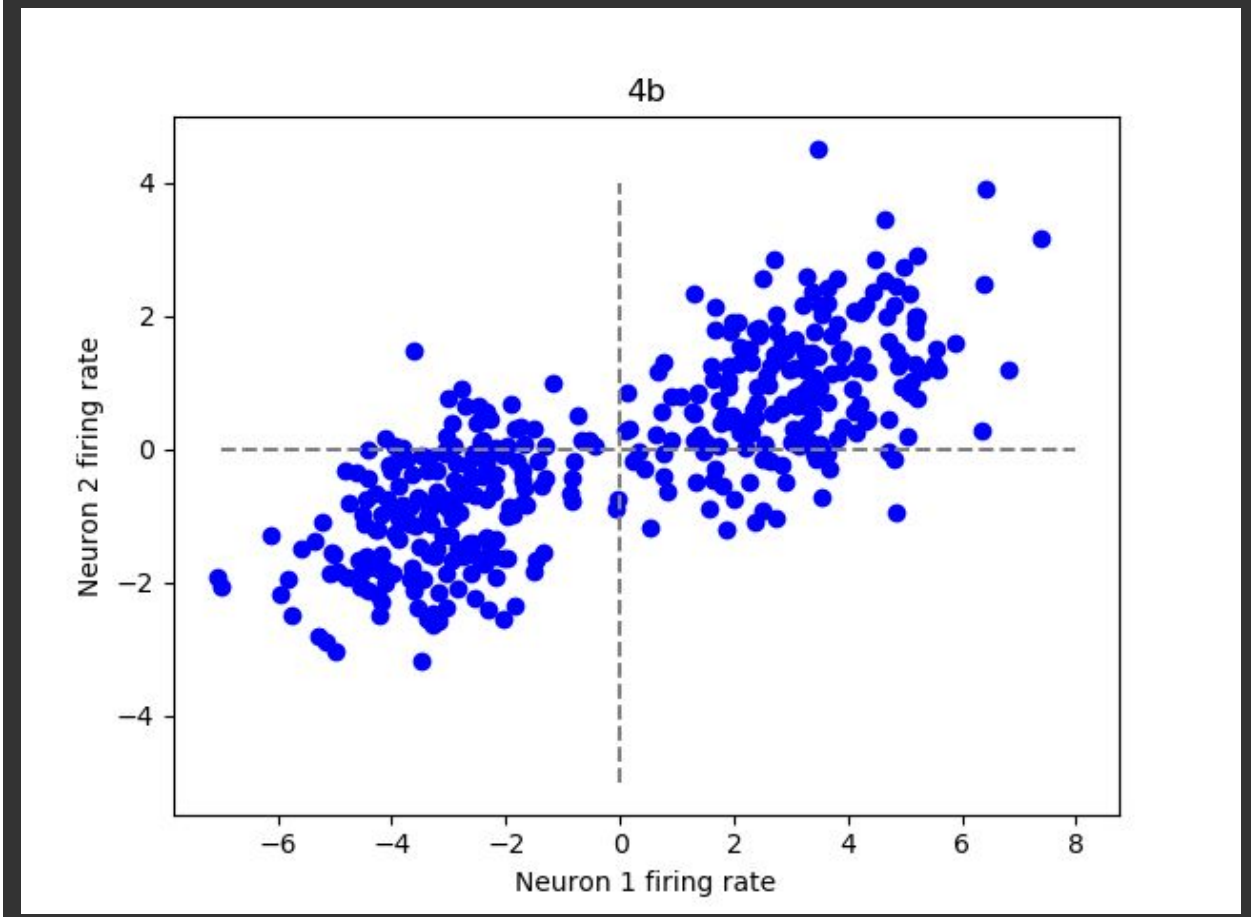
Elin_Ahlstrand_Exercises_6

```
## Set up working environment
include("standard_start.jl")
include("plot_arrow.jl")
include("gsample.jl")

using JLD
      X = load("Ex6datafile.jld")["X"]

## Load data

using PyPlot;
      scatter(X[1,:], X[2,:], color = "red")
      xlabel("Neuron 1 firing rate")
      ylabel("Neuron 2 firing rate")
      vlines(0, -5, 4, linestyle = "--", color = "grey");
      hlines(0, -7, 8, linestyle = "--", color = "grey");
## Scatter plot of raw data
```

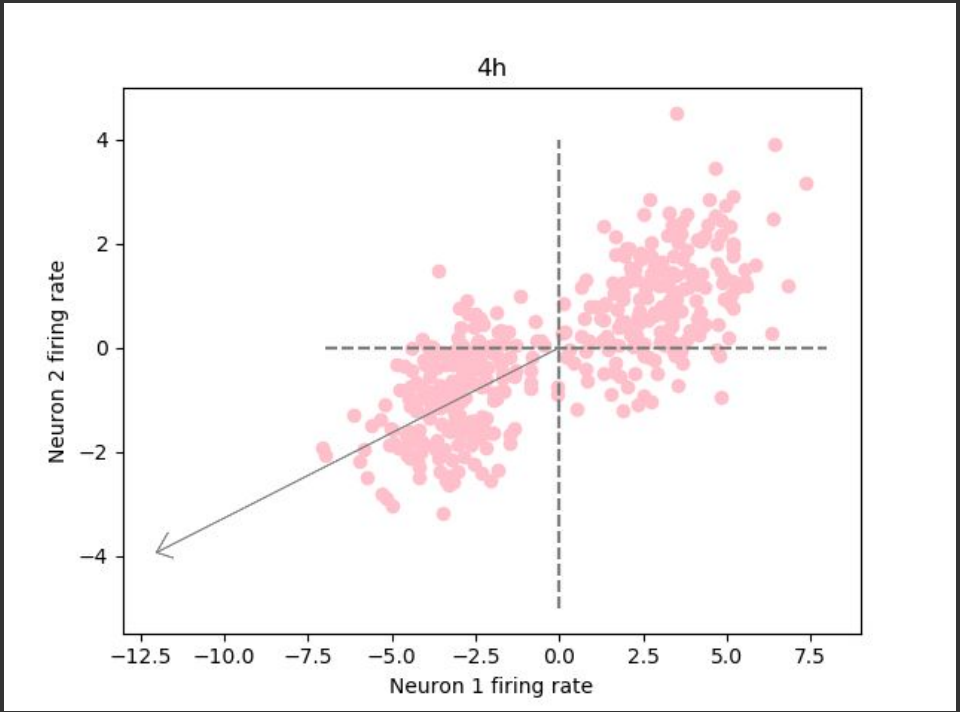


```
## 4b
using Statistics;
using LinearAlgebra;

Xdm1 = X[1,:].-mean(X[1,:]);
Xdm2 = X[2,:].-mean(X[2,:]);
Xdm = [Xdm1 Xdm2]';
      scatter(Xdm[1,:], Xdm[2,:], color = "blue");
      xlabel("Neuron 1 firing rate");
      ylabel("Neuron 2 firing rate");
      vlines(0, -5, 4, linestyle = "--", color = "grey");
      hlines(0, -7, 8, linestyle = "--", color = "grey");

## Normalize the data
```

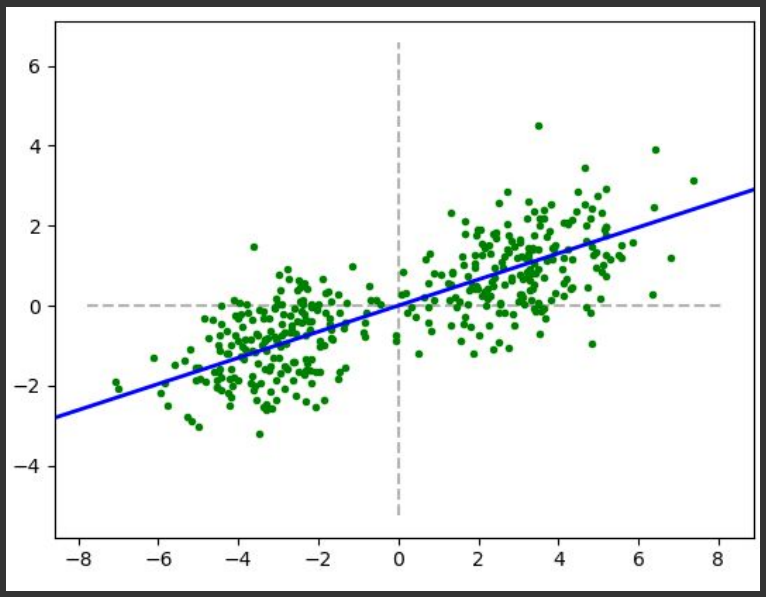


```
## 4c & 4d - find the variance
N1_varXdm = sum(Xdm[1,:].^2)/(length(Xdm[1,:])-1)
N2_varXdm = sum(Xdm[2,:].^2)/(length(Xdm[2,:])-1)
      println("N1_varXdm = ", N1_varXdm, "\nN2_varXdm = ", N2_varXdm);
##
N1_varXdm = 11.491236402336265
N2_varXdm = 1.8392619118715987

## 4e
cv = (sum(Xdm[1,:].*Xdm[2,:]))./(length(Xdm[2,:])-1);
      println("covariance (cv) = ", cv);
##
covariance (cv) = 3.531267031946453

## 4f
Cov =(Xdm*Xdm')/(size(Xdm)[2]-1);
      println("Cov = ", Cov,
      "\n1. N1_var = ", Cov[1, 1],
      "\n2. N2_var = ", Cov[2, 2],
      "\n3. covariance = ", Cov[1,2], " & ", Cov[2,1]);

## Printing Cov matrix & checking results - all is well!
Cov = [11.4912 3.53127; 3.53127 1.83926]
1. N1_var = 11.491236402336261
2. N2_var = 1.8392619118715987
3. covariance = 3.531267031946453 & 3.531267031946453


## 4g
using LinearAlgebra
E = eigen(Cov);
print("\n",E)

Eigen{Float64,Float64,Array{Float64,2},Array{Float64,1}}
([0.685283, 12.6452], [0.310624 -0.950533; -0.950533 -0.310624])

      D = eigvals(E);
      V = eigvecs(E);

## Finding eignevalues & eigenvectors of E

E_dec = V * (inv(V).* D);
```

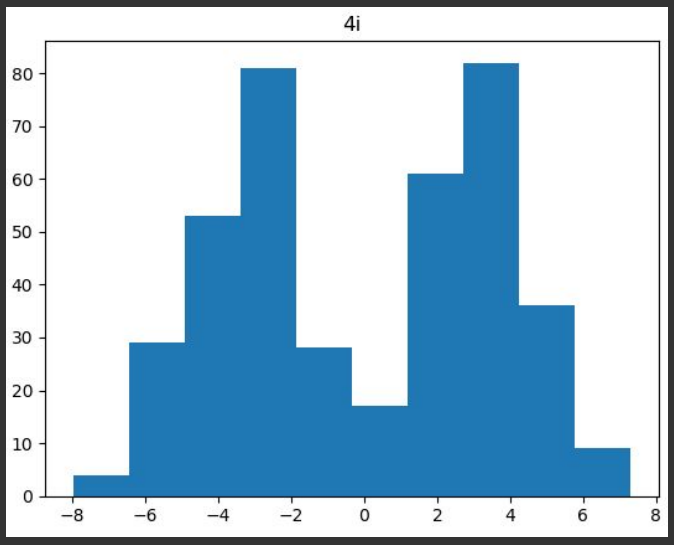
```
      println("E_dec = ", E_dec,"  which is equal to Cov-matrix = "   ,
Cov);

E_dec = [11.4912 3.53127; 3.53127 1.83926]  which is equal to Cov-matrix =
[11.4912 3.53127; 3.53127 1.83926]

## 4g - the eigendecomposition of Cov


PC1 =  maximum(D)*V[:,2];
scatter(Xdm[1,:], Xdm[2,:], color = "pink");
      xlabel("Neuron 1 firing rate");
      ylabel("Neuron 2 firing rate");
      vlines(0, -5, 4, linestyle = "--", color = "grey");
      hlines(0, -7, 8, linestyle = "--", color = "grey");
      plot_arrow([0,0], PC1; hlen=0.04, linewidth=.5,color = "grey")

## 4h
```



```
x_proj1 = projection(Xdm, PC1, plotProjDots=false, plotProjConn=false);
```

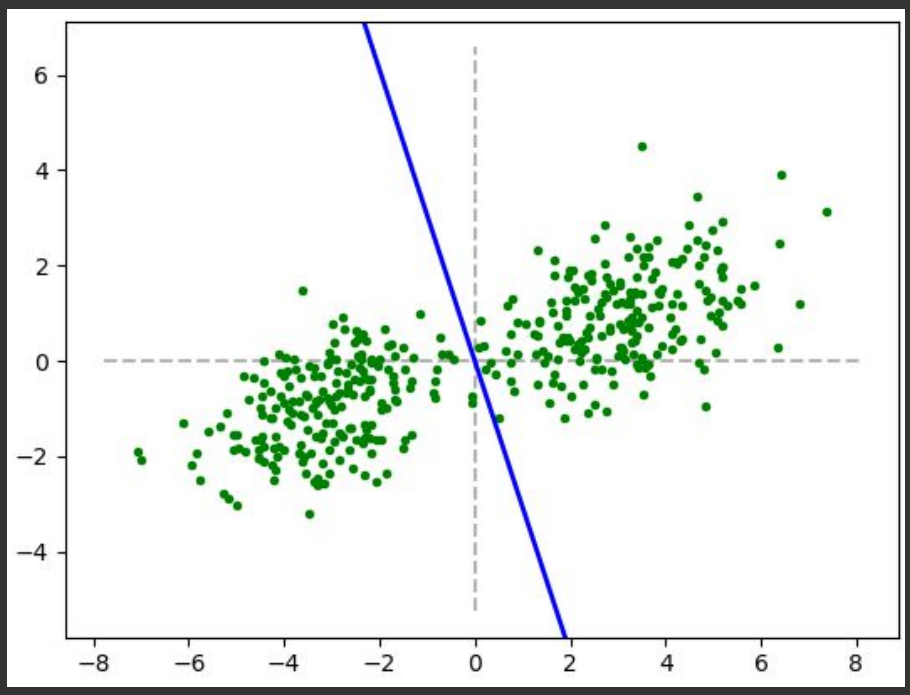



```
hist(x_proj1)
```

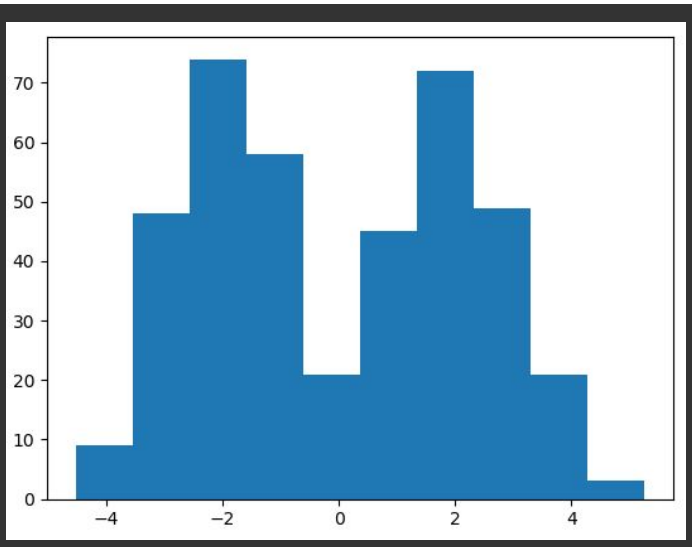



```
## 4i

#= Cannot seem to get the right results with Xdm'*PC1 and I am not sure
why...
Which is why I resorted to using the projection() function from gsample =#

x_proj1_var = sum(x_proj1.^2)/length(x_proj1);
      println(x_proj1_var);
```

```
12.613602651400535
## 4j

PC2 =  minimum(D)*V[:,1];
x_proj2 = projection(Xdm, PC2, plotProjDots=false, plotProjConn=false);
```




```
x_proj2_var = sum(x_proj2.^2)/length(x_proj2);

      println("x_proj1_var >> x_proj2_var   ---->     ", x_proj1_var," >> ",
x_proj2_var)

x_proj1_var >> x_proj2_var   ---->
12.613602651400535 >> 0.6835694170218125



## 4k

x_mean = Xdm'*[1/2; 1/2];
      hist(x_mean)
```




```
## 41
```