Homework 1

Rules: You may work together. You may share code. You *must* write up independent solutions with your own conclusions and analysis to turn in. Please also insure that one copy of each piece of code used to get answers is provided to the instructor.

1. Out of the pressure of the classroom, let's explore the modeling exercise we did in class.
   Knowledge: Assume all transcripts human cells transcribe are known.
   Hypothesis and Question: A human cell of type A has different transcript levels from a human cell of type B (*e.g.* one cancerous, other not), (and these transcript levels can reveal the cause of cancer). Which transcripts are differentially expressed?
   Experiment:

   - Materials: one cell of type A, one cell of type B.
   - Randomly sample n mRNA from each cell.
   - Unambiguously identify the transcript of each mRNA.

   The data are counts $\boldsymbol{X}_i = (X_{i1}, X_{i2}, \ldots, X_{im})$, where $i \in \{A, B\}$ indexes the cell and m is the total number of possible transcripts. We will focus on the single gene j, though one could also model the entire multivariate vector. Notice, that the components of the vectors are not independent: as one count, say X_{ij} increases, other counts $X_{ik}, k \neq j$ will tend to decrease because the total number of transcripts in the cell is finite. Fortunately, if m is large and more specifically many members of vector $\boldsymbol{X}_i$ are non-zero, then the components of the vector are nearly independent. Bioinformaticians make use of this fact all the time in RNA-seq experiments; it allows them to handle one gene at a time.

   (a) If cell A has T_A total mRNAs and cell B has T_B total mRNAs, not necessarily equal, and the sample size n is *not* much smaller than T_A (or T_B) and sampling is without replacement, specify a distribution for the data under H_0. Continue to assume this model until directed not to.

   Solution:

   $$X_{ij} \sim H(T_i, \theta_{ij}, n),$$

   where $H(\cdot)$ is the hypergeometric distribution and θ_{ij} is the unknown parameter representing the number of transcript j in cell $i \in \{A, B\}$.

   (b) Hypothesis testing consists of several parts (you can read more about it in the Stat 342 notes):

   - A **model**, which you have already specified.
   - The **null** (H_0) and **alternative** (H_a) hypotheses.
   - A **test statistic**, $T(\boldsymbol{X})$, some function of the data $\boldsymbol{X}$, and a sampling distribution for the $T(\boldsymbol{X})$ under H_0.
   - A **significance level** (α).
   - A **rejection region** (R_R).

The null hypothesis H_0 is a statement about population parameters in your model. When there are no differentially expressed genes in the two cells, what is true about the parameters of the distribution you named, *i.e.* what is H_0?

Solution:

$$H_0 : \frac{\theta_{Aj}}{T_A} = \frac{\theta_{Bj}}{T_B}$$

(c) Since H_0 is a statement about population parameters, a frequently useful test statistic is the maximum likelihood estimator (MLE) of the involved parameters, especially since the MLE is functionally invariant meaning that the MLE of some function of a parameter, $\widehat{f(\theta)}$, is in most cases (see Stat 342 for details), the function applied to the MLE of the parameter: $f(\hat{\theta})$, where $\hat{\cdot}$ is used to indicate the MLE.

(d) Is it possible to obtain MLEs of the parameters in your model? (Don't attempt it, rather do a little Google searching and report what you find.)

Solution: It seems impossible to obtain MLEs of the parameters in the model.

(e) Now suppose $n \ll T_i$ for $i \in \{A, B\}$. While the model you proposed above is still valid, what other model with fewer parameters is now plausible for these data?

Solution: Since $n \ll T_i$, the sampling procedure can be consider as with replacement.

$$X_{ij} \sim B(n, \theta_{ij}),$$

where $B(\cdot)$ is binomial distribution and θ_{ij} is the unknown proportion of transcript j among all transcripts in cell i.

(f) What is H_0 for this model?

Solution: Still,

$$H_0 : \theta_{Aj} = \theta_{Bj}$$

(g) Another convenient test for hypotheses about population parameters is the likelihood ratio test (LRT). It is useful when the null hypothesis imposes constraints on parameters in the more general model of the alternative hypothesis. Read about it in the Stat 342 notes or anywhere else (there are lots of resources; it is a common test) and implement it for this problem, first generically for any data X_{Aj}, X_{Bj}, and then specifically for $n = 100, X_{Aj} = 3$, and $X_{Bj} = 10$.

Solution:

$$H_0 : \theta_{Aj} = \theta_{Bj}$$
$$H_1 : \theta_{Aj} \neq \theta_{Bj}$$
$$\begin{aligned}\lambda &= \frac{\max_{\theta_{Aj}=\theta_{Bj}} L(\Theta; X)}{\max_{\Theta} L(\Theta; X)} \\ &= \frac{\max_{\theta}[\theta^{X_{Aj}+X_{Bj}}(1-\theta)^{2n-X_{Aj}-X_{Bj}}]}{\max_{\theta_{Aj},\theta_{Bj}}[\theta_{Aj}^{X_{Aj}}\theta_{Bj}^{X_{Bj}}(1-\theta_{Aj})^{n-X_{Aj}}(1-\theta_{Bj})^{n-X_{Bj}}]} \\ &= \frac{(\frac{X_{Aj}+X_{Bj}}{2n})^{X_{Aj}+X_{Bj}}(1-\frac{X_{Aj}+X_{Bj}}{2n})^{2n-X_{Aj}-X_{Bj}}}{(\frac{X_{Aj}}{n})^{X_{Aj}}(\frac{X_{Bj}}{n})^{X_{Bj}}(1-\frac{X_{Aj}}{n})^{n-X_{Aj}}(1-\frac{X_{Bj}}{n})^{n-X_{Bj}}}\end{aligned}$$

```
a1 = 13/200;
h0=a1^(13)*(1-a1)^(200-13);
a2 = 3/100;
a3 = 10/100;
h1 = a2^3*(1-a2)^97*a3^10*(1-a3)*90;
log.lambda = log(h0/h1);
pchisq ( -2*log.lambda, df=1, lower.tail=F );
## [1] 1.547767e-08
```

(h) Argue that the Law of Rare Events (or the Poisson limit theorem, according to Wikipedia) applies in this case, and reformulate your model as a Poisson model.

Solution: Since n is large and θ is small, Poisson distribution is a reasonable approximation of the binomial distribution.

$$X_{ij} \sim P(\theta_{ij})$$

(i) Spend about a paragraph explaining what conclusions you can draw when H_0 is rejected. If you can reject H_0 because the p-value (see Stat 342 notes if you need assistance) is very tiny, what can you really conclude? How much closer are you to identifying the cause of this type of cancer? Think both biologically and statistically.

Solution:

If we can reject H_0, we can say it is very unlikely to observe the data we have observed under H_0 and alternatively H_1 is true. However, we are still far from knowing the cause of this type of cancer. The statistical concern is that we have not yet measured the variability between cells. It is plausible that we could identify just as many DE genes by comparing two type A cells because of natural variation between cells. Thus, the DE genes supported by the test

may not only be a consequence of the cancer, but may have nothing to do with cancer at all. Even if we assume the model is true and we identify the real DE genes, i.e., we know which transcripts were differentially transcribed but it still might be the outcome of the cancer instead of the reason.

2. In the next question you will analyze a small example of Hi-C-like data. This question reviews the Sinkhorn-Knopp and related algorithms.

   To account for the bias caused by the propensity for some beads to be more easily detected than others, you learned the Sinkhorn-Knopp (SK) algorithm can transform symmetric $\boldsymbol{C}$ to doubly stochastic $\boldsymbol{T}$, where $\boldsymbol{C} = \boldsymbol{BTB}$ for some diagonal matrix $\boldsymbol{B} = \text{diag}(b_1, \ldots, b_n)$ containing the bead biases b_i. The SK algorithm prescribes alternating between dividing each row entry by the corresponding row sum and dividing each column entry by the corresponding column sum.

   (a) Let $\boldsymbol{x}_0 = \mathbf{1}$, then show that if $\boldsymbol{y}_1 = [\boldsymbol{C}\boldsymbol{x}_0]^{-1}$ (take the multiplicative inverse of each element in the vector $\boldsymbol{C}\boldsymbol{x}_0$ to get $\boldsymbol{y}_1$), the matrix after one iteration of Sinkhorn-Knopp is

   $$\boldsymbol{C}^{(1)} = \text{diag}(\boldsymbol{y}_1)\boldsymbol{C}\text{diag}(\boldsymbol{x}_1),$$

   where $\boldsymbol{x}_1 = [\boldsymbol{C}'\boldsymbol{y}_1]^{-1}$.

   Solution:

   $$\begin{aligned}
   \boldsymbol{C}^{(0.5)} &= [\boldsymbol{C}\boldsymbol{x}_0]^{-1}\boldsymbol{C} = \text{diag}(\boldsymbol{y}_1)\boldsymbol{C} && (1)\\
   \boldsymbol{C}^{(1)} &= \boldsymbol{C}^{(0.5)}\text{diag}(\left[(\boldsymbol{x}_0'\boldsymbol{C}^{0.5})'\right]^{-1}) && (2)\\
   &= \text{diag}(\boldsymbol{y}_1)\boldsymbol{C}\text{diag}([(\boldsymbol{x}_0'\text{diag}(\boldsymbol{y}_1)\boldsymbol{C})']^{-1}) && (3)\\
   &= \text{diag}(\boldsymbol{y}_1)\boldsymbol{C}\text{diag}([(\boldsymbol{y}_1'\boldsymbol{C})']^{-1}) && (4)\\
   &= \text{diag}(\boldsymbol{y}_1)\boldsymbol{C}\text{diag}([\boldsymbol{C}'\boldsymbol{y}_1]^{-1}) && (5)
   \end{aligned}$$

   (b) One can use induction to show (I'm not asking it) that, in general,

   $$\boldsymbol{C}^{(k+1)} = \text{diag}(\boldsymbol{y}_{k+1})\boldsymbol{C}\text{diag}(\boldsymbol{x}_{k+1})$$

   where

   $$\begin{aligned}
   \boldsymbol{y}_{k+1} &= [\boldsymbol{C}\boldsymbol{x}_k]^{-1} && (6)\\
   \boldsymbol{x}_{k+1} &= \left[\boldsymbol{C}'\boldsymbol{y}_{k+1}\right]^{-1} = \left\{\boldsymbol{C}'\left[\boldsymbol{C}\boldsymbol{x}_k\right]^{-1}\right\}^{-1}. && (7)
   \end{aligned}$$

   Let me spend a bit more time showing this, so you can put it to use. At iteration k, we have $\text{diag}(\boldsymbol{y}_k)\boldsymbol{C}\text{diag}(\boldsymbol{x}_k)$ with column sums $\mathbf{1}$. We seek a new matrix $\text{diag}(\boldsymbol{y}_{k+1})$ such that row sums, $\text{diag}(\boldsymbol{y}_{k+1})\boldsymbol{C}\text{diag}(\boldsymbol{x}_k)\mathbf{1} = \mathbf{1}$, are one, but this equation is solved

as $[\text{diag}(\boldsymbol{y}_{k+1})]^{-1} = \boldsymbol{C}\boldsymbol{x}_k$, which yields Eq. 6. Eq. 7 is found similarly. Thus, the Sinkhorn-Knopp algorithm defines a sequence of vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ that converge because the $\boldsymbol{C}^{(k)}$ converge and yields

$$\boldsymbol{T} \approx \text{diag}(\boldsymbol{y}_K)\boldsymbol{C}\text{diag}(\boldsymbol{x}_K)$$

for sufficiently large K. Furthermore, $y_{Kl} = x_{Kl}$ is approximately the inverse bias $\frac{1}{b_l}$ when $\boldsymbol{C}$ is symmetric.

Imakaev2012 propose an algorithm that is very similar to the Sinkhorn-Knopp algorithm.

1. Let n be the number of beads and initialize $k = 0$ and $\boldsymbol{C}^{(k)} = \boldsymbol{C}$.
2. At iteration k, compute the row sums $c_{i\cdot}^{(k)} = \sum_j c_{ij}^{(k)}$ and mean row sum $\bar{c}^{(k)} = \frac{1}{n}\sum_i c_{i\cdot}^{(k)}$. Update $c_{ij}^{(k+1)} = \frac{c_{ij}^{(k)}\left(\bar{c}^{(k)}\right)^2}{c_{i\cdot}^{(k)} c_{\cdot j}^{(k)}}$.
3. Increment k and repeat step 2 until $\boldsymbol{C}^{(k)}$ barely changes any more.

Using the fact that SK converges, prove that this algorithm converges to a constant multiple of a doubly stochastic matrix, $C\boldsymbol{T}$, where C is some constant.

Hint. Write the algorithm as two iterated functions, $f_x(\cdot)$ and $f_y(\cdot)$, as was done above for SK in Eqs. 6–7. The iteration is initialized with $\boldsymbol{x}_0 = \boldsymbol{y}_0 = \boldsymbol{1}$. Show that although the sequences $\{\boldsymbol{x}_0, \boldsymbol{x}_2 = f_x(\boldsymbol{x}_0), \boldsymbol{x}_4 = f_x(\boldsymbol{x}_2), \ldots\}$ and $\{\boldsymbol{y}_0, \boldsymbol{y}_2 = f_y(\boldsymbol{y}_0), \boldsymbol{y}_4 = f_y(\boldsymbol{y}_2), \ldots\}$ obtained from iterating the functions do not converge, their product $\boldsymbol{x}_{2i+2}\boldsymbol{y}_{2i+2} = f_x(\boldsymbol{x}_{2i})f_y(\boldsymbol{y}_{2i})$ does, and so the Imakaev2012 algorithm does too.

(c) Implement the SK algorithm and demonstrate it works on

$$\begin{pmatrix} 3 & 4 & 5 \\ 18 & 5 & 7 \\ 8 & 15 & 9 \end{pmatrix}.$$

Report the doubly stochastic matrix $\boldsymbol{T}$ and left biases b_{l1}, b_{l2}, b_{l3} and right biases b_{r1}, b_{r2}, b_{r3}. Biases for non-symmetric matrices are analogous to biases for symmetric matrices, except a row i may be particularly detectable as a donor, but not as an accepter, when the relationships sampled in the matrix have the concept of donor and acceptor.

Solution:

```r
sinkhorn.knopp.iteration <- function(M) {
  n = dim(M)[1];
  iter = 1;
  x = rep(1, n);
  repeat {
    y = 1/M%*%x;
    x.new = 1/t(M)%*%y;
    if (norm(x.new - x)<1e-6) break;
    iter = iter + 1;
    x = x.new;
  }
  list(D1=as.vector(y), D2=as.vector(x), iter=iter)
}

M = matrix(c(3,4,5,18,5,7,8,15,9), nrow=3, byrow=T);
result=sinkhorn.knopp.iteration(M);
diag(result$D1)
##             [,1]       [,2]       [,3]
## [1,] 0.08164087 0.00000000 0.00000000
## [2,] 0.00000000 0.03458224 0.00000000
## [3,] 0.00000000 0.00000000 0.03077024
diag(result$D2)
##           [,1]     [,2]     [,3]
## [1,] 0.8980171 0.000000 0.000000
## [2,] 0.0000000 1.040552 0.000000
## [3,] 0.0000000 0.000000 1.078502
```

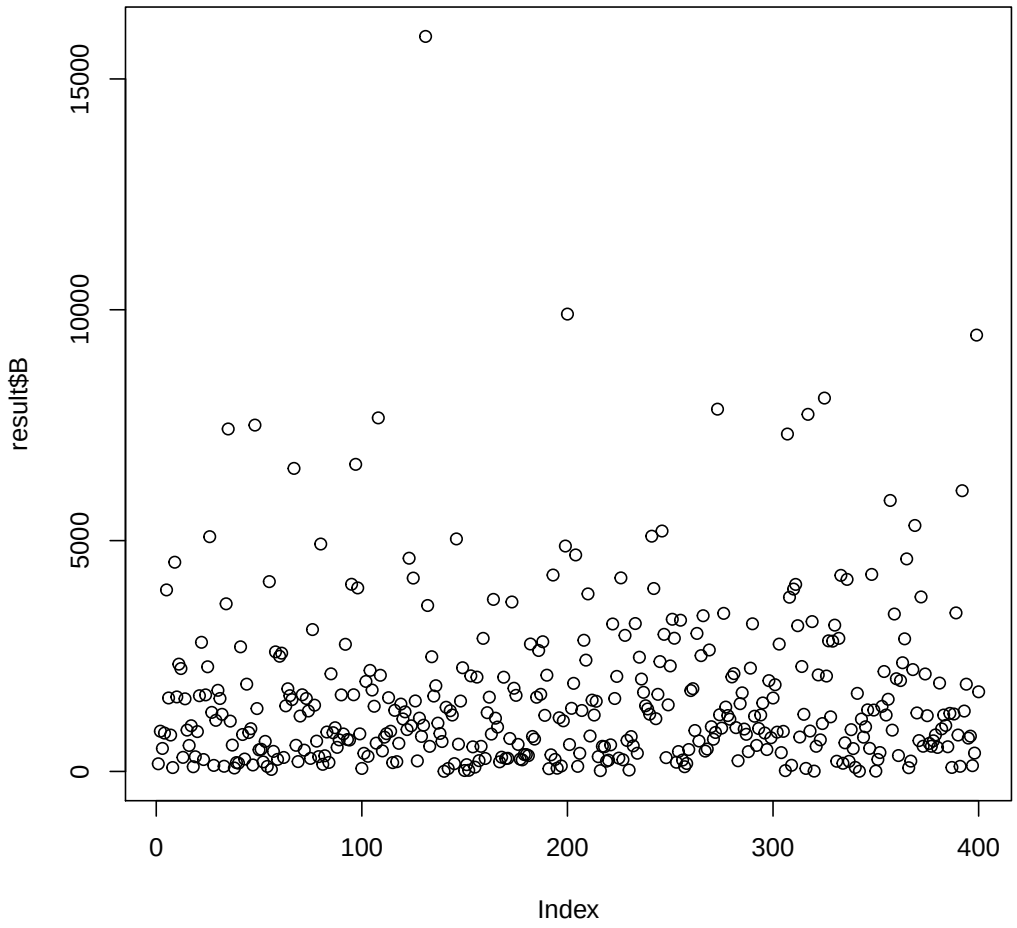
3. Your goal is to figure out the pattern of beads in 2D space when all you observe is the noisy number of times each pair of beads is seen to interact in N total observed interactions.

   Collect the data from Blackboard. These data are counts c_{ij} of the number of times bead i was found to interact with bead j. The data are the lower triangular form of a symmetric matrix given in column order; the diagonals are assumed zero. Unlike the Hi-C data, where the beads are ordered by their position along the genome, these data are not naturally ordered, except spatially, but you will not know the spatial relationships until you estimate them. As a consequence, there are no natural neighbors and hence all beads can potentially interact with each other, unlike in Hi-C data, where interactions between neighboring beads along the genome were dropped from the data.

   (a) Report your estimate of $\boldsymbol{T}$. Which 10 beads have the biggest bias and are least detectable?

Solution:

```
setwd("/home/ruolin/Dropbox/2016BCB568/homework")
beads= scan("counts.Rtxt")
C = diag(400)
C[lower.tri(C, diag=F)] = beads
diag(C) = 0
C = t(C) + C

Imakaev.iteration <- function(M) {
  iter = 1;
  B.aggregate = rep(1, dim(M)[1]);

  repeat {
    k = mean(rowSums(M));
    delta_B  = rowSums(M)/sqrt(k);
    ## NOTE here is the square root of k
    ## Imakaev does not take the squre root but I did. If we
    ## donot take square root, T is not double stochastic matirx.

    B.aggregate = B.aggregate * delta_B;
    M.new = diag(as.vector(1/delta_B))  %*% M %*%
          diag(as.vector(1/delta_B));

    if(var(delta_B) < 1e-10) break;
    iter = iter + 1;
    M = M.new;
  }
  list(B=B.aggregate, iter=iter);
}

result = Imakaev.iteration(C)

T= diag(1/result$B)%*%C%*%diag(1/result$B)
## T is a double stochastic matirx but too large to print
## The bias is very large in the order of 1e3.
plot(result$B)
```

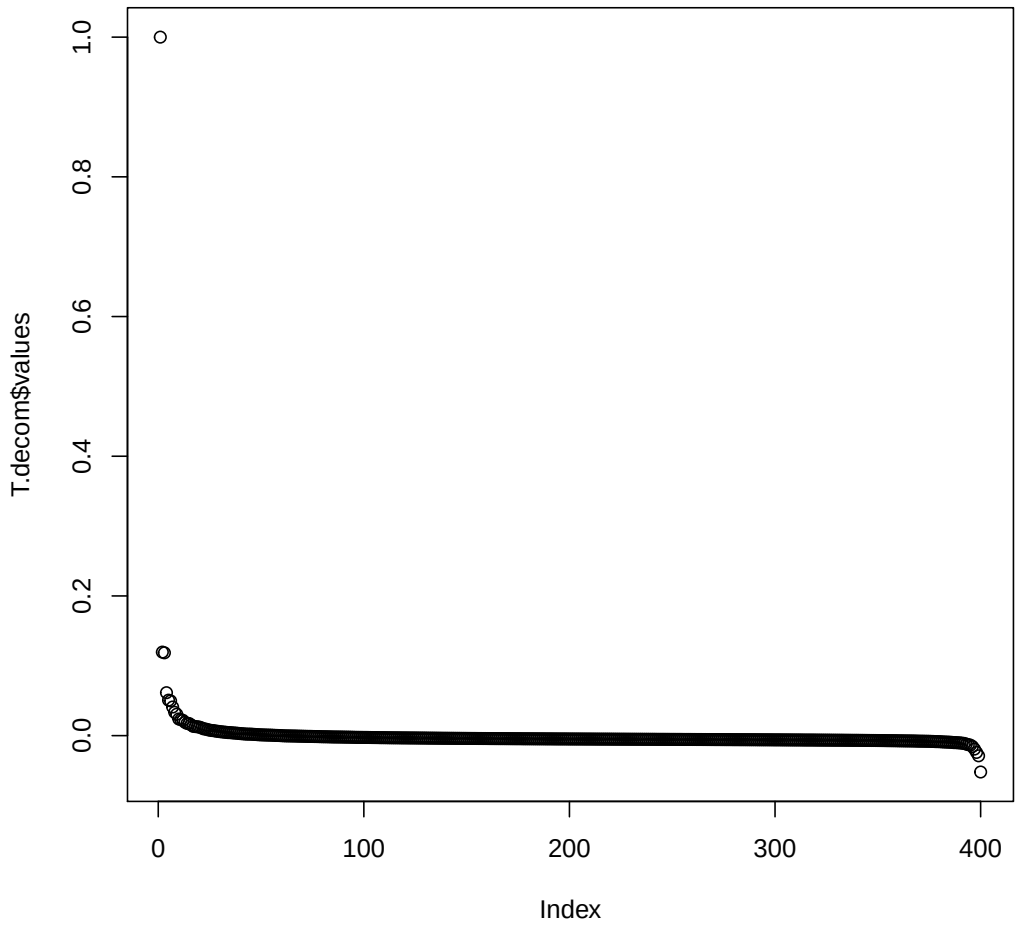



```
### The 10 beads with largest bias
order(result$B, decreasing=TRUE)[1:10]
##  [1] 131 200 399 325 273 317 108  48  35 307
```

(b) Now attempt denoising the data by using the eigenvector expansion for the top K eigenvalues. (Please note, I believe Imakaev is wrong to add back in the mean–it is not clear what mean is meant: use the equation for $\boldsymbol{Z}$ in the PCA notes. Also note that $\boldsymbol{T}$ is not necessarily a positive semi-definite matrix, so some eigenvalues may be negative.) There is plenty of noise in these data, so many small eigenvalues represent noise signal. Hopefully the true signal resides in the first few eigenvalues. Plot the eigenvalues in decreasing order, and see if you can identify a good choice for K such that larger eigenvalues carry signal and smaller eigenvalues probably carry noise.

Solution:

```
T.decom = eigen(T)
plot(T.decom$values)
```

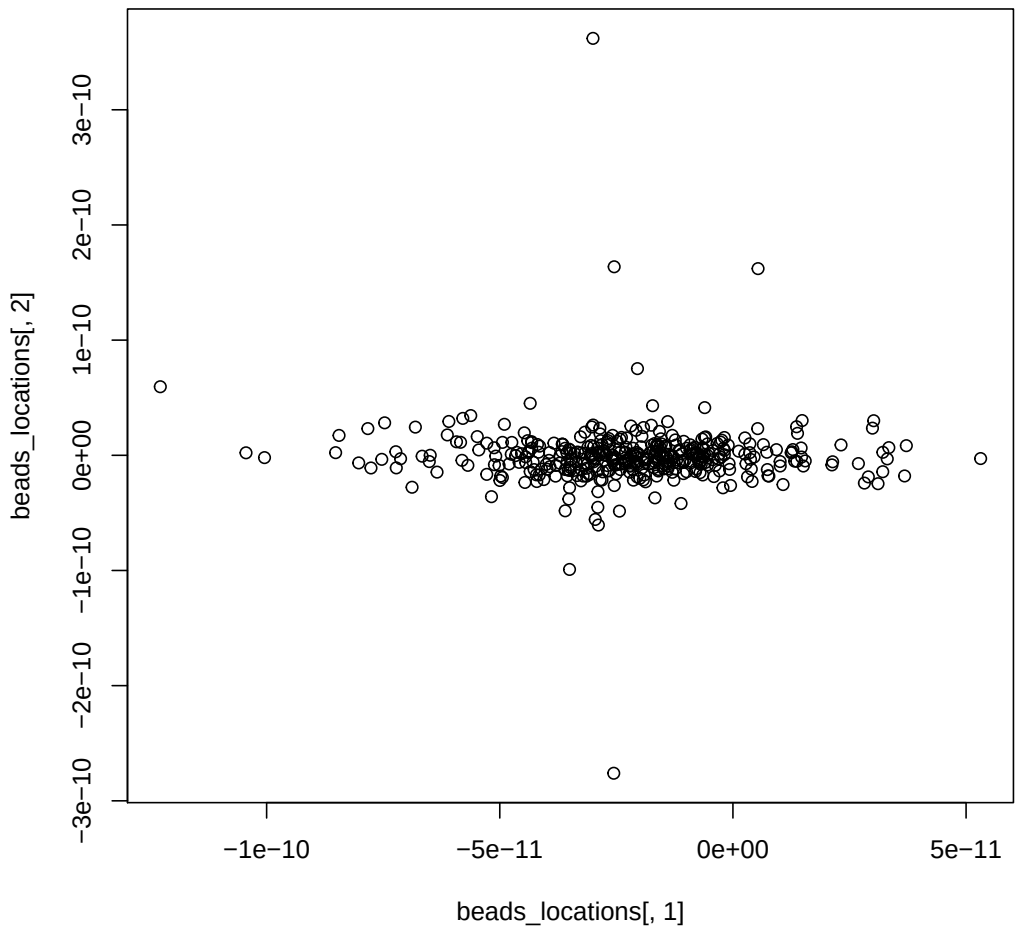



```
## It looks like K=3 might be a good choice since
## the largest three eigen values stand out.
```

(c) Implement metric MDS with Euclidean distances minimizing the sum of squared differences to estimate $\boldsymbol{X}$, the locations of the beads in 2D. In `R`, the function `cmdscale()` does the job. Plot your results for $K = 3$.

Solution:

```r
k=3
T.noised_removed= T.decom$vectors[,1:k] %*%
  diag(T.decom$values[1:k]) %*% t(T.decom$vectors[,1:k])

beads_locations = cmdscale(T.noised_removed, k=2)

plot(beads_locations[,1], beads_locations[,2])
```

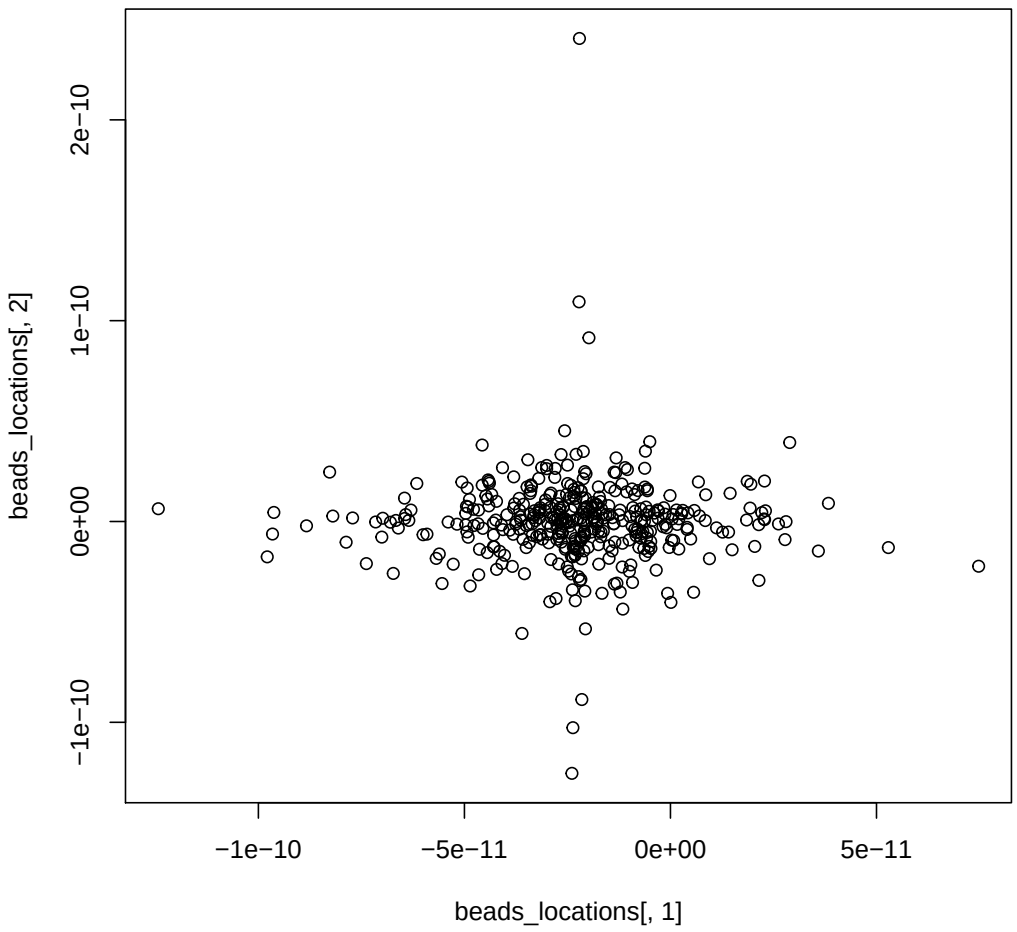



```r
## MDS location of beads in 2D. I can't make sense of the labels
##on x-axis and y-axis
```

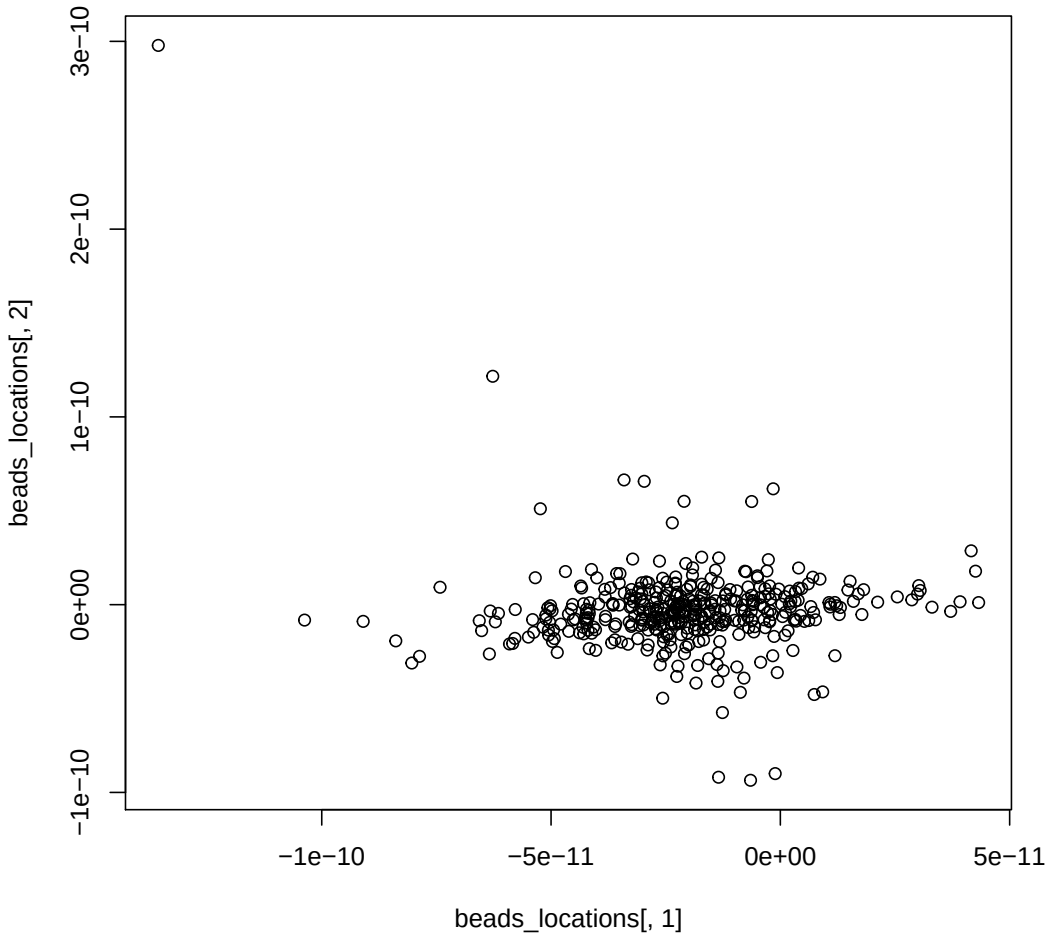
(d) What is the effect of K on your results? Can you guess the true arrangement of the beads?

Solution:

```
k=2
T.noised_removed= T.decom$vectors[,1:k] %*%
  diag(T.decom$values[1:k]) %*% t(T.decom$vectors[,1:k])

beads_locations = cmdscale(T.noised_removed, k=2)

plot(beads_locations[,1], beads_locations[,2])
```



```r
## MDS location of beads in 2D when K=2

k=10
T.noised_removed= T.decom$vectors[,1:k] %*%
  diag(T.decom$values[1:k]) %*% t(T.decom$vectors[,1:k])

beads_locations = cmdscale(T.noised_removed, k=2)

plot(beads_locations[,1], beads_locations[,2])
```




```r
## MDS location of beads in 2D when K=10
```

(e) What determines the error in the placement of beads?

4. Consider the following data relating the genotype at a SNP locus of $n = 2000$ individuals

with their disease phenotype. Perform a test to determine if there is association of the phenotype with the locus. Justify your test. Draw a conclusion.

		Genotype			
		AA	AB	BB	Total
Disease	control	143	17	705	865
	case	132	842	161	1135
	total	275	859	866	2000

5. In this question we will analyze a more realistic (in size) genomic dataset. We will still avoid real data because real data have a pesky problem of missing data. Please find the data on the website, one file of genomic data and another of the (disease) phenotype.

   (a) Perform an Cochran-Armitage test on the sixth locus using aggregated data as in question 5. Why does it produce a more significant p-value than a chi-square test of the same data? (**Note:** In one version of the posted notes, I included some old notes that use `R` function `independence_test()` presumably to perform Cochran-Armitage, but I no longer recall the precise connection between Cochran-Armitage and this function in `R`. Instead, use the derivations in the notes. The derivations force you to remember the model and assumptions, which you need to be aware of whenever applying a test to data.)

   Solution:

```r
library(plyr)
gwas= read.table("gwas.Rtxt")
phenotype = scan("y.Rtxt")

dat = rbind(phenotype, gwas)
#dim(dat)
#unlist(dat[7, dat[1,] == 0])
 negative = table(unlist(dat[7, dat[1,] == 0]))
 positive = table(unlist(dat[7, dat[1,] == 1]))
 locus6 = rbind(negative, positive)

armitage_test = function(m){
  ## M is 2x3 matrix, first row is control ,second is case
  X= matrix(c(1,1,1,0,1,2), ncol=2, byrow=F)
  p = m[2,]/colSums(m)

  alpha_0_hat = rowSums(m)[2]/sum(m)
  sigma = alpha_0_hat * (1- alpha_0_hat) / colSums(m)
  W_0 = diag(sqrt(1/sigma))

  X_0 = W_0 %*%X
  p_0 = W_0 %*%p

  beta = solve(t(X_0)%*%X_0)%*%t(X_0)%*%p_0
  var = solve(t(X_0)%*%X_0)
  list(beta=beta, var=var)
}
result = armitage_test(locus6)
test_T = result$beta[2]/sqrt(result$var[2,2])
test_T
## [1] -2.465552
2*pnorm(test_T)
## [1] 0.01368024
```

(b) To account for possible confounding population structure we cannot use the aggregated data, so return to the original data, where each entry x_{ji} is the number of major alleles at locus j observed in a sample of size 2 from the ith diploid individual. Compute the (global–across subpopulations, if they exist) wild-type allele frequency for each locus. Pre-process the data by subtracting the row means and standardizing so each entry x_{ij} has population variance approximately 1. (**Hint:** You may run into some computational difficulties working with such large matrices; be assured there are ways that do not make you wait overnight or more for calculations

to finish.) (**Note:** Actually, Price2006 neglects a factor of 2, the number of alleles sampled within each diploid individual, when standardizing the genotypes, but since it is a constant, it does not interfere with the analysis.) Point out two reasons this standardization is only an approximation. What could be the consequences of a failure to actually standardize?

(c) Perform a PC analysis on the standardized $\boldsymbol{X}$ matrix. Do you find evidence of genetic subpopulations in these data? How many do you think?

(d) Regress $\boldsymbol{y}$, the disease indicator, on the top 3 eigenvectors $\boldsymbol{G}^{(3)}$ *and* the sixth locus. Do your conclusions about the association of locus 6 with the disease change? Interpret.

(e) The above regression can be written as

$$\boldsymbol{y} = \left(\boldsymbol{G}^{(3)}, \boldsymbol{x}\right)\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $n \times 1$ vector $\boldsymbol{x}$ is the standardized data at the sixth locus. The least squares estimators (and MLEs when $\boldsymbol{\epsilon} \sim N(0, \sigma^2)$) of the coefficients are, from the usual matrix formula applied to this special case,

$$\hat{\boldsymbol{\beta}} = \begin{pmatrix} \boldsymbol{I} & (\boldsymbol{G}^{(3)})'\boldsymbol{x} \\ \boldsymbol{x}'\boldsymbol{G}^{(3)} & \boldsymbol{x}'\boldsymbol{x} \end{pmatrix}^{-1} \begin{pmatrix} (\boldsymbol{G}^{(3)})' \\ \boldsymbol{x}' \end{pmatrix} \boldsymbol{y}.$$

The 4×4 matrix $\boldsymbol{M} = \begin{pmatrix} \boldsymbol{I} & (\boldsymbol{G}^{(3)})'\boldsymbol{x} \\ \boldsymbol{x}'\boldsymbol{G}^{(3)} & \boldsymbol{x}'\boldsymbol{x} \end{pmatrix}$ is simple enough to invert analytically (you might find a formula in a book, or you can take the inverse as the product of elementary matrices that convert $\boldsymbol{M}$ to $\boldsymbol{I}$, *i.e.* if $\boldsymbol{E}_1, \boldsymbol{E}_2, \ldots, \boldsymbol{E}_L$ are such that $\boldsymbol{E}_L\boldsymbol{E}_{L-1}\cdots\boldsymbol{E}_2\boldsymbol{E}_1\boldsymbol{M} = \boldsymbol{I}$, then $\boldsymbol{M}^{-1} = \boldsymbol{E}_L\boldsymbol{E}_{L-1}\cdots\boldsymbol{E}_2\boldsymbol{E}_1$). Verify that

$$\boldsymbol{M}^{-1} = \begin{pmatrix} 1+\boldsymbol{g}'_{(1)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(1)} & \boldsymbol{g}'_{(1)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(2)} & \boldsymbol{g}'_{(1)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(3)} & -\boldsymbol{g}'_{(1)}\boldsymbol{x}/D \\ \boldsymbol{g}'_{(2)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(1)} & 1+\boldsymbol{g}'_{(2)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(2)} & \boldsymbol{g}'_{(2)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(3)} & -\boldsymbol{g}'_{(2)}\boldsymbol{x}/D \\ \boldsymbol{g}'_{(3)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(1)} & \boldsymbol{g}'_{(3)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(2)} & 1+\boldsymbol{g}'_{(3)}\boldsymbol{x}\boldsymbol{x}'\boldsymbol{g}_{(3)} & -\boldsymbol{g}'_{(2)}\boldsymbol{x}/D \\ -\boldsymbol{x}'\boldsymbol{g}_{(1)}/D & -\boldsymbol{x}'\boldsymbol{g}_{(2)}/D & -\boldsymbol{x}\boldsymbol{g}_{(3)}/D & 1/D \end{pmatrix},$$

where $D = \boldsymbol{x}'[\boldsymbol{I} - \boldsymbol{G}^{(3)}(\boldsymbol{G}^{(3)})']\boldsymbol{x}$. Therefore, the least squares estimate of the coefficient of $\boldsymbol{x}$ is

$$\hat{\beta}_4 = \frac{\boldsymbol{x}'[\boldsymbol{I} - \boldsymbol{G}^{(3)}(\boldsymbol{G}^{(3)})']\boldsymbol{y}}{\boldsymbol{x}'[\boldsymbol{I} - \boldsymbol{G}^{(3)}(\boldsymbol{G}^{(3)})']\boldsymbol{x}}.$$

Show that this is exactly the estimate of the coefficient you get if you regress the residuals $\boldsymbol{\epsilon}_{y3}$ on residuals $\boldsymbol{\epsilon}_{x3}$ after setting $\boldsymbol{\epsilon}_{x0} = \boldsymbol{x}$ and $\boldsymbol{\epsilon}_{y0} = \boldsymbol{y}$ and repeatedly performing simply linear regressions:

$$\boldsymbol{\epsilon}_{x,l-1} = \beta_{x,l}\boldsymbol{g}_{(l)} + \boldsymbol{\epsilon}_{xl}$$
$$\boldsymbol{\epsilon}_{y,l-1} = \beta_{y,l}\boldsymbol{g}_{(l)} + \boldsymbol{\epsilon}_{yl}$$

for $l = 1, 2, 3$ as recommended in Price2006 to remove the confounding factor of population structure.