Problem Set 2
Statistics, Computation and Applications

Felipe del Canto

October, 2021

Problem 2.1

For part (a), I first load the data and compute the mean (μ) and standard deviation (σ) of $X := \log(1 + \texttt{interaction\_frequency})$, which are 1.13 and 0.40, respectively. Both numbers are rounded to the nearest hundredth. To compute these numbers, I use the fact that the sample variance ($\hat{\sigma}^2$) of data points $\{X_1, \ldots, X_N\}$ can be computed as

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N} X_i^2 - \mu^2$$

where

$$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i$$

and compute, separately, the number of samples N, the sum of the squares of X and the sum of X by loading the datasets one at a time. Then, compute μ using the formula above and $\sigma = \sqrt{\hat{\sigma}^2}$.

For part (b), in Figure 1 is presented the heatmap representing the interaction frequencies for chromosomes 19 and 20. As can be seen in the image, there are some contiguous darker zones that are indicators of intermingling portions of these chromosomes.

For part (c), we have that under the null hypothesis, H_0, the entries of the interaction matrix M between any pair of chromosomes are independent and identically distributed according to a Normal distribution with mean μ and standard deviation σ. Consequently, given a $k \times \ell$ sub matrix of M, with mean m, we have that m also follows a Normal distribution with mean μ but with standard deviation $\sigma/\sqrt{k\ell}$. Under the null, the probability of the m being greater than μ is equal to

$$1 - \Phi\left(\frac{m-\mu}{\sigma/\sqrt{k\ell}}\right) \tag{1}$$

This corresponds to the p-value of the test that rejects the null hypothesis of μ being the real mean, agains the alternative that the mean is lower. However, when looking for potential interaction zones (that is, submatrices with higher-than-average mean value), there are several hypothesis being tested at the same time. In fact, if M is a $N_r \times N_c$ matrix, then for a fixed size of $k \times \ell$, there is a total of

$$N_{\text{sumatrices}} := (N_r - k + 1)(N_c - \ell + 1)$$

Figure 1: Heatmap for the average interaction between chromosomes 19 and 20. The values correspond to $\log(1 + \text{interaction frequency})$, darker colors represent a higher interaction frequency.

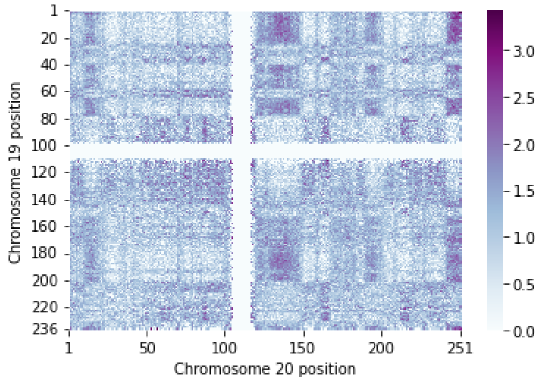
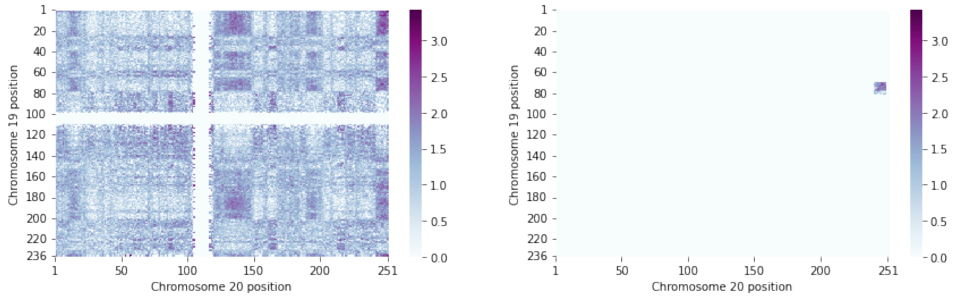


Figure 2: Greedy search algorithm finds an interaction region between chromosomes 19 and 20. To the left, the original interaction matrix. To the right, the submatrix found by the greedy search algorithm.



possible submatrices of that size to test for interaction zones. Consequently, the p-value in (1) must be adjusted for multiple testing hypothesis. In particular, if we follow the Bonferroni correction, the hypothesis is rejected (that is, we find interaction zones) at significance level α if

$$1 - \Phi\left(\frac{m - \mu}{\sigma / \sqrt{k\ell}}\right) \leqslant \frac{\alpha}{N_{submatrices}},$$

which is what we wished to prove. In the case of the chromosomes 19-20 interaction matrix, we have that M is of size 236×251 and if we wished to test for a 10×10 interaction region, we would have:

$$N_{\text{submatrices}} = 54\,934$$

For part (d), the greedy search algorithm is resumed in Algorithm 1. This procedure works because, as the name suggests, greedily searches for submatrices whose mean is highly likely to be greater than μ. This, because for any given initialization, the algorithm tries to find local minima in the adjusted p-value of each submatrix. To achieve this goal, when expanding the initial submatrix M chooses the row or column that reduces the most the adjusted p-value. An example of how this algorithm finds an interacting region is presented in Figure 2. As can be seen from the picture, the greedy search algorithm is capable of finding an interacting region. However, in order to find all possible interacting regions a more general purpose algorithm must be used. This new procedure is resumed in Algorithm 2 and the result of running it for chromosomes 19 and 20 is presented in Figure 3.

For part (e), I first run Algorithm 2 on all non equal pair of chromosomes to compute the number of intermingling regions. Since the goal is to embed these data into a 3D space, there is no need to compute these regions for identical chromosomes, as the distance between a chromosome and itself should be 0. The results are presented in Figure 4a. As can be seen from the image, some chromosomes have almost none interaction, while other share many regions. In particular, the most intermingled chromosomes are those at the lower right corner of the interaction counting matrix. These are the pairs: 16-22, 18-19, 19-20, 19-22, 20-21, 20-22, and 21-22.

Figure 3: Greedy search algorithm finds an interaction region between chromosomes 19 and 20. To the left, the original interaction matrix. To the right, the submatrix found by the greedy search algorithm.

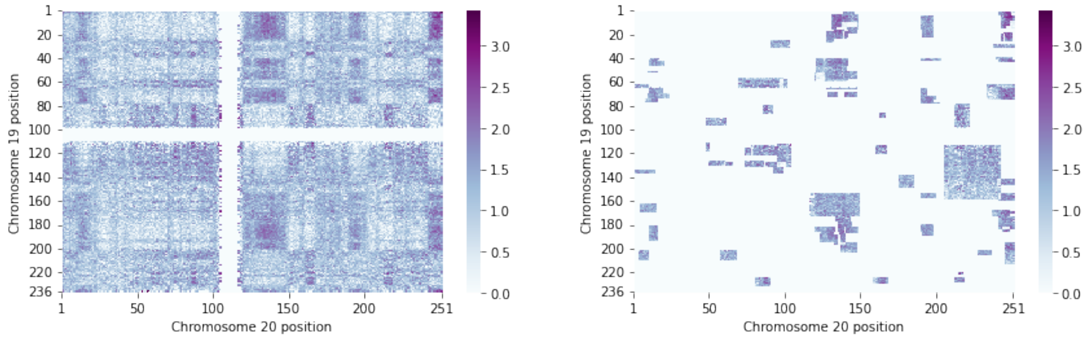
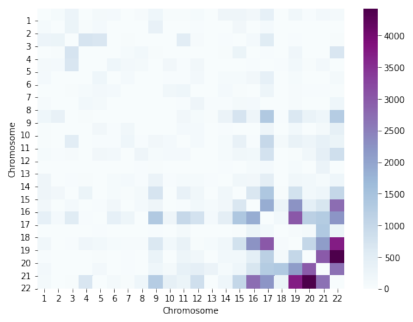
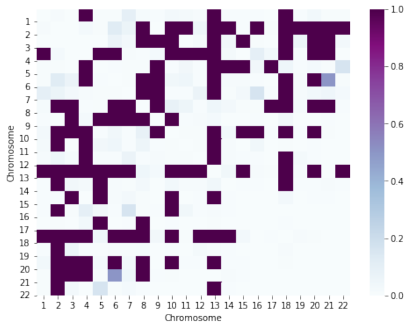


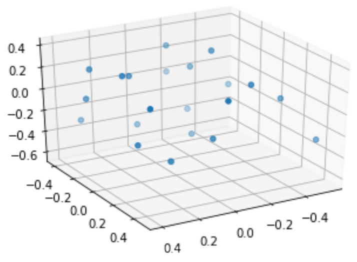
Figure 4: 3D embedding process for the chromosomes. In the left panel, the heatmap of the number of intermingling regions for each pair of chromosomes. In the center panel, the associated dissimilarity matrix. In the right panel, the MDS embedding is presented.



(a) Heatmap of the number of intermingling regions for each pair of chromosomes.



(b) Heatmap representing the dissimilarity matrix for each pair of chromosomes.



(c) MDS embedding

Algorithm 1 Greedy search algorithm to the largest interaction region in interaction matrix Z

```
function GREEDY_SEARCH( )
    $M \leftarrow$ random $Z_{ij}$
    $p \leftarrow$ adjusted $p$-value of $M$
    current_$p \leftarrow p$
    return_$M \leftarrow M$
    while $p \geqslant$ current_$p$ do
        for side in [left, right, top, bottom] do
            $M_2 \leftarrow M$ + side row/column
            $p_2 \leftarrow$ adjusted $p$-value of $M_2$
            if $p_2 <$ current_$p$ then
                current_$p \leftarrow p_2$
                return_$M \leftarrow M_2$
            end if
        end for
    end while
    return return_$M$
end function
```

For part (f), in order to transform the count of intermingling region into a dissimilarity matrix that is accepted for the MDS algorithm, we need to transform the previous heatmap. This, in order to transform smaller (reps. large) counts into large (resp. smaller) distances. Let D be the dissimilarity matrix that is wished to obtain. I first define that the diagonal of D will contain only zeros, that is, each chromosome is identical to itself. Then, if the intermingling region count between chromosomes i and j is 0, then $D(i,j) = 1$.

Algorithm 2 General procedure to find all interaction regions in an interaction matrix Z.

```
function OVERALL_GREEDY_SEARCH(max_rep)
    min_$p \leftarrow 0.01$
    while True do
        for $i$ in 1,...,max_rep do
            $M \leftarrow$ GREEDY_SEARCH( )
            $p_2 \leftarrow$ adjusted $p$-value of $M$
            if $p_2 \leqslant$ min_$p$ then
                chosen_$M \leftarrow M$
                min_$p \leftarrow p_2$
            end if
        end for
        if min_$p \geqslant 0.01$ then
            break while
        end if
        $Z_{ij} \leftarrow Z_{ij}$ - mean chosen_$M$
    end while
    return return_$M$
end function
```

Finally, if the count is not zero, then $D(i,j)$ is equal to the inverse of the intermingling region count. This way, $D(i,j) \in [0,1]$ and lower values correspond to closer chromosomes. The final dissimilarity matrix is presented in Figure 4b. As expected from Figure 4a and the definition of D, the heatmap shows a smaller distance for pairs of chromosomes whose interaction count was high and a distance close to 1 for those that had less (or none) interactions. Finally, using D I perform the MDS embedding. The result is presented in Figure 4c.

Problem 2.2

For part (a), note that duplicating the variable and running an *unregularized* version of the logistic regression changes the output coefficient to $\frac{1}{2}\hat{\beta}$. Indeed, let $X' = 2X$ be the duplicated variable and let h be the loss function for the logistic regression (which depends on the product βY). Note that

$$\arg\min_{\beta} h(\beta Y) = \frac{1}{2} \arg\min_{\beta} h\left(\frac{1}{2}\beta Y\right). \tag{2}$$

Indeed, if β^* solves the left-hand side problem, then $2\beta^*$ solves the right-hand one, and (2) is obtained. Hence,

$$\arg\min_{\beta} h(\beta Y) = \frac{1}{2} \arg\min_{\beta} h(\beta X)$$

and the previous result is proved. Denoting β^* the output coefficient obtained in the unregularized problem with X doubled, we have that

$$|\beta^*| \leqslant |\hat{\beta}|, \tag{3}$$

with equality if and only if $\hat{\beta} = 0$. Finally, it should be expected that

$$|\hat{\beta}_{\ell_1}| \leqslant |\beta^*|, \tag{4}$$

since the ℓ_1-regularized problem induces a lower 1-norm for the output coefficient. To see why this happens, consider the regularized problem

$$\arg\min_{\beta} h(\beta Y) + \lambda|\beta|. \tag{5}$$

for which β^* is feasible and achieves the lowest h. Let $\hat{\beta}_{\ell_1}$ be the solution of (5). Then, $h\left(\hat{\beta}_{\ell_1} Y\right) \geqslant h\left(\beta^* Y\right)$. But since $\hat{\beta}_{\ell_1}$ is optimal, it has to occur that $|\hat{\beta}_{\ell_1}| \leqslant |\beta^*|$, otherwise β^* would be the solution of (5).

In conclusion, we have that (3) and (4) imply

$$|\hat{\beta}_{\ell_1}| \leqslant |\hat{\beta}|$$

with equality if and only if $\hat{\beta} = 0$, in which case $\hat{\beta}_{\ell_1} = 0$ as well.

For part (b), we have that (3) still holds. Moreover, the same argument use with optimization problem (5) holds with the ℓ_2-regularized problem

$$\arg\min_{\beta} h(\beta Y) + \lambda\beta^2. \tag{6}$$

and thus we have that

$$\hat{\beta}_{\ell_2}^2 \leqslant (\beta^*)^2, \tag{7}$$

but since $\hat{\beta}_{\ell_2}$ and β^* are scalars, then (7) is equivalent to (4). Showing that

$$|\hat{\beta}_{\ell_2}| \leqslant |\hat{\beta}|$$

with equality if and only if $\hat{\beta} = 0$, in which case $\hat{\beta}_{\ell_2} = 0$ as well.

Finally, for part (c), one of the advantages of the elastic net is that, as opposed to ℓ_1 regularization, there is always a unique solution. In situations with many features it might be the case that ℓ_1 achieves two distinct and optimal results that are difficult to compare. If in that case the ℓ_2 was used, many (if not all) features will have a nonzero coefficient, making the regularization less useful. Instead, by using the elastic net, it is possible to reduce de dimensionality of the coefficient vector while still ensuring a unique and tractable solution.

An example where the use of an elastic net could be beneficial would be a clinical application, where from the many risk factors only a few must be selected in order to provide physicians with a manageable tool to evaluate their patients. Another case would be selecting engineered features that can better explain a corpus when performing text classification. In order for results to be explainable and improved between training-testing iterations, a stable and interpretable solution must be obtained at each time.

Problem 2.3

For part (a), note that if $Y \mid X \sim \text{Bernoulli}\,(p(x))$, then

$$P(Y = y) = (1 - p(x))^{1-y}\, p(x)^y$$

Consequently, the likelihood of the n iid observations is

$$L\left(\{x_i, y_i\}; \beta, \beta_0\right) = \prod_{i=1}^{n} \left(1 - p(x_i; \beta, \beta_0)\right)^{1-y_i} p(x_i; \beta, \beta_0)^{y_i},$$

where

$$p(x_i;\beta,\beta_0) = \frac{1}{1+\exp\left(\beta_0+\beta^T x_i\right)}$$

And finally, the log-likelihood is

$$\ell\left(\{x_i,y_i\};\beta,\beta_0\right) = \sum_{i=1}^{n}(1-y_i)\log\left(1-p(x_i;\beta,\beta_0)\right) + y_i \log p(x_i;\beta,\beta_0)$$

which after rearranging and canceling terms is

$$\ell\left(\{x_i\};\beta,\beta_0\right) = \sum_{i=1}^{n}(1-y_i)\left(\beta_0+\beta^T x_i\right) - \log\left(1+\exp\left(\beta_0+\beta^T x_i\right)\right)$$

And since the first term is nonzero only if $y_i = 0$, we can write

$$\ell\left(\{x_i,y_i\};\beta,\beta_0\right) = \sum_{\substack{i=1\\ y_i=0}}^{n}\left(\beta_0+\beta^T x_i\right) - \sum_{i=1}^{n}\log\left(1+\exp\left(\beta_0+\beta^T x_i\right)\right)$$

For part (b), let ϕ be the standard normal pdf. For $c \in \{0,1\}$, let

$$z_{i,c} := -\frac{1}{2}(x_i-\mu_c)^T\Sigma^{-1}(x_i-\mu_c)$$

Then, following the same generalization for part (a), the likelihood for this n iid observations is

$$L(\{x_i,y_i\};\{\mu_c\},\Sigma) = \prod_{i=1}^{n}\phi(z_{i,0})^{1-y_i}\phi(z_{i,1})^{y_i}$$

And thus, the *log*-likelihood is

$$\ell(\{x_i,y_i\};\{\mu_c\},\Sigma) = \sum_{\substack{i=1\\ y_i=0}}^{n}\log\phi(z_{i,0}) + \sum_{\substack{i=1\\ y_i=1}}^{n}\log\phi(z_{i,1})$$

Problem 2.4

For part (a), in order to choose the appropriate number of clusters (k), first we visualize the dataset to obtain a clue of how many groups of cells we could encounter. In Figure 5 are presented the scatterplots that result of fitting a t-SNE and a PCA with two components each. In the left image, t-SNE hints at the possibility of three clusters, as the scatterplot shows a triangular shape. On the other hand, the image on the right suggests that PCA is not a good tool to search for clusters in this setting. Overall, I consider the possibility of up to 8 clusters. To test how many clusters is optimal in this case, in Figure 6 is presented the Within Cluster Sum of Squares (WCSS) for different number of clusters, when running k-means for $k \in \{1,\ldots,k\}$ on both the original data points and the t-SNE transformed ones. In this figure, is possible to appreciate that applying k-means on the transformed points achieves a much better WCSS than when applied to the original data points. This is expected as t-SNE makes closer points appear closer, while increasing the distance between points that are separate beforehand. Looking at the image, it is possible to conclude that a number of clusters between 3 and 5 appears to be enough, as the elbow point in both images is between this range.

In order to visualize how the clustering is being performed, in Figure 7 are presented the scatterplots showing each of the clusters when applying k-means in the original and transformed data points. As can be seen in the pictures, the clusters formed when using k-means in the transformed data points are more structured than its counterparts to the left. For the rest of this problem, 4 clusters on the t-SNE transformed points were considered.

Figure 5: t-SNE (left) and PCA (right) plots for the sample of cells.

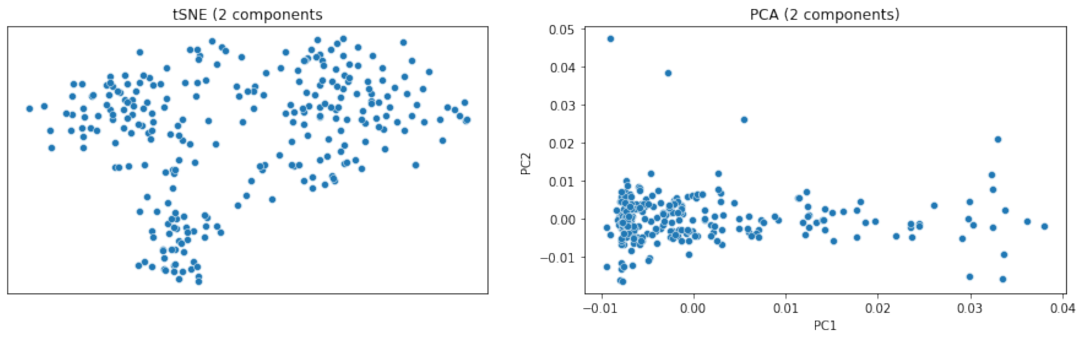
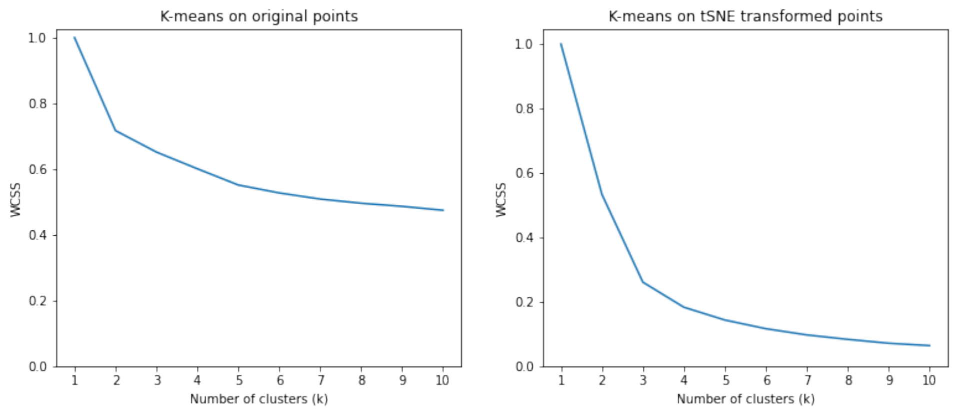


Figure 6: Within Cluster Sum of Squares (WCSS) for k-means algorithm ran with different number of clusters. The left (resp. right) image, shows WCSS for k-means applied on the original (resp. t-SNE transformed) data points.

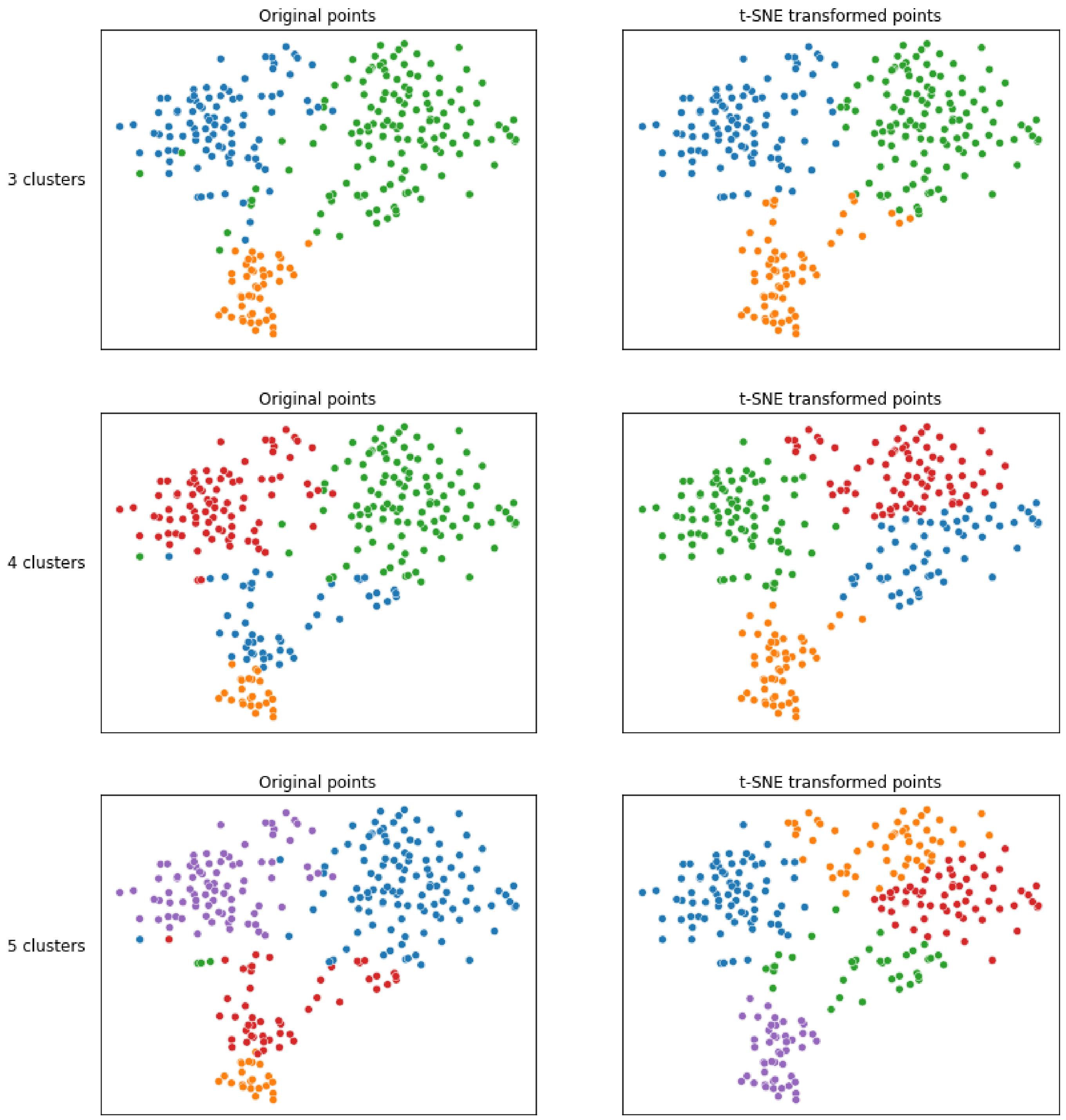


Finally, in order to find which genomic features are good markers for the 4 clusters selected, I train a classifier on the data. Specifically, a Logit model is trained to predict the labels of the 4 clusters using all features (RNA sequences) and regularizing using an elastic net. Since the data is extremely high-dimensional, the regularizing term allows for better interpretability of the results. Moreover, the elastic net also founds a unique solution no matter the initial conditions of the algorithm. The results are presented in Table 1. From what can be seen in the table, the sequence 13 895 plays an important role in differentiating these cells. For example, cluster 1 is characterized by an under-regulation of sequence 13 895, while cluster 3 is characterized by an over-regulation of the same sequence. Cluster 4, on the other hand is a type of cell that also has an under-regulation of sequence 13 895, albeit in a smaller scale. However, it is also different from cluster 1 since this cell has an over-regulation of sequence 94 839. Finally, cluster 4 has no relevant genomic features. In a not presented experiment, using 3 clusters leads to finding the same three relevant genomic features, suggesting cluster 4 is probably not a different cell type.

Table 1: Relevant genomic features which predict the selected number of clusters. A Logit classifier was trained on the 4 labels (4 clusters) and the relevant genomic features and their values were extracted.

Cluster	Relevant features		
	Number	Position	Value
1	1	13 895	- 0.32
2	2	13 895	- 0.04
		94 839	0.06
3	1	13 895	0.83
4	0	–	–

Figure 7: *t*-SNE showing clustering of data points. The rows represent different number of clusters. The left (resp. right) column, shows the clustering when the *k*-means was applied on the original (resp. *t*-SNE transformed) data points.


Original points
t-SNE transformed points
3 clusters
Original points
t-SNE transformed points
4 clusters
Original points
t-SNE transformed points
5 clusters