1. The first file “**Author's Sample Visualisation**” is about the the code that has been provided by the author with a sample input in the code itself to check the functioning of the tool - “JDINAC (joint density-based non-parametric differential interaction network analysis and classification using high-dimensional sparse omics data) “

    a. This contains no graph as the data which has been provided is a matrix and the visualization pictures entered in the paper are from a specific dataset and was particularly for an experiment carried out for BRCA patients.
    b. Just copy paste this entire code in the R environment and it will run. Also the desired output is given as comments in the code.

2. The second file “**Author**” is about the existing tool - JDINAC, how it functions and is basically the normal working of the JDINAC tool. First this code needs to be run before running the first file. Both these tools have to be run in an R environment.

    a. Working :

        i. Step 1: Data Splitting - We take our bunch of observations and randomly split them into two groups, called D1 and D2.

        ii. Step 2 : Kernel Density Estimation - We look at group D1 and figure out how genes work together by making fancy curves for each pair of genes (like drawing squiggly lines). This helps us understand how genes interact in our dataset.

        iii. Step 3 : L1 Penalized Logistic Regression - Now, with group D2, we use math to make a special formula that guesses if something is a particular type of biology thing based on how genes act. We tweak this formula to make it work better, kind of like adjusting a recipe until it tastes just right.

        iv. Step 4 : Repetition and Prediction - We do steps 1 to 3 over and over again a bunch of times (we decide how many times before we start). Each time, we make guesses about biology and write down how sure we are about each guess.

        v. Step 5 : Aggregation and Differential Dependency Weight Calculation - After all the guessing, we put together all our guesses and average them to make a final guess.

3. The third file “**Visualization**” contains the code in the python environment where the dataset provided by Dr. Anna has been used to do the correlation of genes provided in the dataset. Now in order to make the visualization look more appealing, the names of the genes were given short initials so as to understand the visualization better. The code is divided into different parts as follows :

a. Data loading and exploration

    i. First, we load the gene expression data from a TSV file and explore its structure

b. Data Filtering and exploration

    i. To focus on the most informative genes, we filter out those with low variance.

c. Renaming the columns

d. Correlation Matrix and Network graph

    i. We calculate the correlation matrix and create a network graph to visualize the relationships between genes. Only strong correlations are included to reduce clutter

    ii. Exploration of the network graphs :
    The network graphs show the correlations between gene expressions across different samples for various conditions. Each node represents a gene, and edges are drawn between nodes with a correlation above a specified threshold. This helps identify clusters or groups of genes with similar expression patterns.

e. Binary Classification and Model Evaluation

    i. Next, we create binary labels randomly, split the data into training and testing sets, and train a logistic regression classifier. We then evaluate the model using ROC and PRC curves.

    ii. ROC & PRC curves illustrate performance of logistic regression model

        1. **ROC Curve**: The ROC (Receiver Operating Characteristic) curve shows the trade-off between sensitivity (true positive rate) and specificity (false positive rate). The area under the ROC curve (AUC) indicates the overall performance of the model. A model with an AUC close to 1 is considered to have a good performance

        2. **PRC Curve**: The PRC (Precision-Recall Curve) focuses on the trade-off between precision and recall. This curve is particularly useful for evaluating models on imbalanced datasets. The average precision score is shown, which provides a single value summarizing the precision-recall trade-off