LAB4_Tips

Install requests, scipy, pandas and scikit-learn libraries and download dataset

Install the following libraries: request, scipy, pandas, scikit-learn.

Gene expression dataset comes from an RNA-seq experiment onto two breast cancer subtypes: Luminal A and Luminal B. Please, download the dataset (***dataset.csv***) from the Teaching Portal.

T test statistics

In many cases, we analyze microarrays with "one gene at a time" approach. **That is, for each gene we would like to know if this gene is "different in the two classes"**. In our example the two classes are two breast cancer subtypes (Luminal A and Luminal B). A classic analytical method to answer to this question is to perform Independent Student's t-test (so called Welch test).

1. **Define hypotheses and level of significance α:**

Null Hypothesis: Means of the two populations are equal, so the two groups are from the same populations. In our example it means that the gene we are investigating is not differentially expressed between Luminal A and Luminal B breast cancer subtypes and we cannot exploit its gene expression to distinguish between the two-cancer subtypes.

$$H_0 : \mu_1 = \mu_2$$

Alternative Hypothesis: The mean of the two populations are un-equal and the two groups are from different populations. In our example it means that the gene we are investigating is differentially expressed between Luminal A and Luminal B breast cancer subtypes we can exploit its gene expression to distinguish between the two-cancer subtypes.

$$H_1 : \mu_1 != \mu_2$$

Level of significance **α** is defined a priori and it is the risk we are prepared to take in rejecting H_0 when it is in fact true. For example, **α** = 0.05 means a possibility of 5% in rejecting H_0 when in reality H_0 is true.

2. **For each gene perform Welch T-test**

Welch T-test is used to investigate the significance of the difference between the means of two populations. Use scipy library to perform the test. E.g.

```
t_value, p_value = stats.ttest_ind(np.array(Lum_A), np.array(Lum_B), equal_var=False)
```

3. **Reject null hypothesis if the p-value is lower or equal to α**

If the p value is lower than α, we can reject the null hypothesis and say that the gene is differentially expressed between Luminal A and Luminal B cancer subtypes.

Bonferroni adjustment

However, when we perform multiple statistical tests (in our case more than 1000, one for each gene), the overall probability in rejecting the null hypothesis when actually it is true is given by $\alpha \times G$, where G is the number of genes. This overall error is called Family-Wise Error Rate (FWER).

To achieve the global FWER lower than α, we can use Bonferroni adjustment that requires to select only genes for which p value ≤ α/G.

From the ENSEMBL gene notation (ENSG00000268889.1) to the common gene name

Use **ENSEMBL API** (https://rest.ensembl.org/documentation/info/overlap_id) to perform the translation from ENSG to common name. Please note that the following script accepts ENSG gene name without the version (ENSG00000268889.1 is ENSG00000268889)

```
import requests, sys
 server = "https://rest.ensembl.org"
ext = "/overlap/id/ENSG00000157764?feature=gene"
 r     =     requests.get(server+ext,     headers={     "Content-Type"     :
"application/json"})
if not r.ok:
       r.raise_for_status()
       sys.exit()
 decoded = r.json()
print(repr(decoded))
```

Common gene name is placed inside decoded list, at the dictionary key 'external_name'

Scikit-learn library

Scikit-learn is a useful library to perform machine learning with Python. In order to work with, you have to import the library in your program:

```
import sklearn
```

Check for useful functions at this link:

https://scikit-learn.org/stable/

Some useful scikit-learn classes

`sklearn.preprocessing.StandardScaler`

`sklearn.decomposition.PCA`

`sklearn.model_selection.train_test_split`

`sklearn.metrics.accuracy_score`

`sklearn.metrics.precision_recall_fscore_support`

```
sklearn.neighbors.KNeighborsClassifier

sklearn.svm.SVC

sklearn.ensemble.RandomForestClassifier

sklearn.naive_bayes.GaussianNB
```

PCA + KNN example

```
# PCA and number of components

pca = PCA(n_components=80) # creates pca object

# fit pca object onto train set features and transform train set

X_train_pca = pca.fit_transform(X_train)

# transform test set using the features fitted onto train_set

X_test_pca = pca.transform(X_test)


# KNN on dataset with PCA

neigh = KNeighborsClassifier(n_neighbors=3) # creates knn object

neigh.fit(X_train_pca, y_train) # fit knn object onto training set

# use knn to predict the outcome onto training set

y_test_predicted = neigh.predict(X_test_pca)
```