Lab 8, NumPy and T-test

Due date: Friday, May 6, 11:59pm.

Overview

After some further deliberations, I decided to split your next assignment into 2-3 smaller labs. The next few labs will use the same Jokes dataset that you studied in Lab 7. Each of lab will build upon the code base you created in previous labs. Specifically, Lab 8 will require you to use some of the data acquisition and modeling code that you have been asked to build for Lab 7.

Lab 8 has two parts. First, you will implement a series of Python functions that jointly will allow you to perform a variety of versions of Student's t-test on data. Second, you will test your t-test implementation by analyzing some of the Jokes dataset data, testing a number of hypotheses. Finally, you will record your observations and submit them as part of your Jupyter notebook.

Assignment Preparation

This is an individual assignment.

For this assignment you will build one Jupyter notebook, `TTest.ipynb`.

To download the instructor's version of the notebook, open a Jupyter terminal, change to your lab assignments directory and type the following command:

```
$ nbgrader fetch --course=dekhtyar Lab8
```

Data

*This section is a copy of the **Data** section of Lab 7 document. It is kept here to give you easier access to reference material.*

You will be using *joke ratings data* from the **Jester** project, run by Professor Ken Goldberg at UC Berkeley. **Jester**[1] is an on-line joke recommender system available at

http://shadow.ieor.berkeley.edu/humor/

Disclaimer. Please read this note before proceeding! Jester has a database consisting of 100 jokes. The jokes are shown to the user, and the user's reaction to them is measured on a continuous scale. **Please be aware** that the jokes available through `Jester` may contain some examples that you personally will find objectionable. Please know, that *it is not my intent to offend anyone's sensibilities* (and neither is it the intent of the authors of the dataset and **Jester**).

The Dataset

The portion of the **Jester** dataset that you will be studying consists of two files uploaded to the `/home/dekhtyar/data` directory on the Jupyter server.

List of jokes. The first file is `Jokes.xml`. This file contains the list of jokes that **Jester** uses. The file has the following simple format:

```
<jokes>
  <joke>
    Joke 1 goes here...
  </joke>
   ...
  <joke>
     Joke 100 goes here...
  </joke>
</jokes>
```

There are 100 jokes in the file — these are all the jokes **Jester** uses. This file can be easily parsed with `BeautifulSoup` using the techniques you have aquired from the previous labs. The jokes are not numbered internally in the XML file, but their order corresponds to the joke ids in the ratings data file.

Ratings Data. The `jester-data-1.csv` file contains our joke ratings data. The full dataset contains three data files of roughly equal size. Of the three files, you will be using only the first one, `jester-data-1`.

The **Jester** dataset web page describes the format of the data as follows[2]:

"Format:

1. 3 Data files contain anonymous ratings data from 73,421 users.

2. Data files are in .zip format, when unzipped, they are in Excel (.xls) format
3. Ratings are real values ranging from -10.00 to +10.00 (the value "99" corresponds to "null" = "not rated").
4. One row per user
5. The first column gives the number of jokes rated by that user. The next 100 columns give the ratings for jokes 01 - 100.
6. The sub-matrix including only columns $\{5, 7, 8, 13, 15, 16, 17, 18, 19, 20\}$ is dense. Almost all users have rated those jokes (see discussion of "universal queries" in the above paper)."

There are 24,983 rows (ratings records) in the `jester-data-1.csv` file. Each row has 101 values as described above.

Lab Assignment

Before starting your work, make sure you download the instructor's version of the `Ttest.ipynb` notebook.

The notebook asks you to perform the following set of actions:

1. **Import Joke Ratings.** Reuse your Lab 7 code to import the array of joke ratings in the format/formats convenient for future use (see the remainder of instructions to determine what specific format the data should be imported in).

2. **Develop t-test routines.** Implement a number of Python functions for hypothesis testing using Student t-test in a variety of forms. Specifically, you will implement functions that compute the t-value for the following three versions of the t-test:

   (a) One-sample t-test

   (b) Two-sample unpaired t-test under the assumption of *unqual sample sizes* and *unequal variance*

   (c) Two-sample paired t-test

   In addition, to complete the t-test process, you will develop a function that translates the t-value computed in one of the functions above to `p-value`, i.e., the probability that the observed results are due to the *null hypothesis* being tested by one's analysis.

3. **Data Analysis.** Using the t-test functions you have implemented, you are asked to test a variety of *null hypotheses* and to construct some comprehensive analytical outputs describing the Jokes dataset.

4. **Explanations.** The final part of the notebook is your explanations and thought on what you have observed.

Step 1. Import of Ratings.

For the first part of the notebook, import and if necessary extend your Lab 7 code for loading the Joke ratings data into main memory. For this lab, we will not be using the texts of the jokes, so there is no need to import your inverted index construction.

Determine in what format you want to store the ratings, and what auxillary data structures you want to have based on the tasks you are asked to do in Step 3 of the process.

All ratings data must be stored in `NumPy arrays`.

Step 2. T-test functions.

This part is at the heart of the lab. You will implement and test the following Python functions (their stubs are available in the instructor's notebook):

- `tValueOneSample(sample, mu)`: this function, given a one-dimensional `NumPy array` containing an observed sample, and a value μ (`mu`), which typically represents the expected *population mean* performs a one-sample t-test to determine whether the null hypothesis that *the population from which the sample is drawn has a mean* μ shall be *rejected*.

  Arguments:

  `sample`: one-dimensional `NumPy array` representing sample being tested.
  `mu`: the population mean value against the sample mean is being tested.

  The function shall compute and return two values:

  `tvalue`: the t-value statistic the the given sample size, mean and variance.
  `df`: number of degrees of freedom

- `tValueTwoSample(sample1, sample2)`: this function, given a two one-dimensional `NumPy arrays` containing two observed samples, performs an independent two-sample t-test to determine whether the null hypothesis that *the means of the two observed samples coincide* shall be *rejected*. The computation of the t-value (the t-test statistic) is performed under the assumptions that the sample sizes may be different and that *the variance of the two samples may also be different*.

  Arguments:

  `sample1`: one-dimensional `NumPy array` representing the first sample being tested.
  `sample2`: one-dimensional `NumPy array` representing the second sample being tested.

  The function shall compute and return two values:

  `tvalue`: the t-value statistic
  `df`: number of degrees of freedom

- `tValuePaired(sample1, sample2)`: this function, given a two one-dimensional `NumPy arrays` containing two observed samples, performs a paired two-sample t-test to determine whether the null hypothesis that *the means of the two observed samples coincide* shall be *rejected*.

Arguments:

`sample1`:	one-dimensional `NumPy array` representing the first sample being tested.
`sample2`:	one-dimensional `NumPy array` representing the second sample being tested.

The function shall compute and return two values:

`tvalue`:	the t-value statistic
`df`:	number of degrees of freedom

- `t2p(t, df, tails)`: this function given the number of degrees of freedom in the t-test performed, and the type of the test desired (one-tailed or two-tailed), converts the t-value obtained during the t-test computation to the p-value, representing the probability that the observations studied by the t-test procedure are consistent with the null hypothesis.

  Arguments:

`t`:	the t-value to be converted into the p-value
`df`:	number of degrees of freedom in the experiment that generated the data
`tails`	either 1 or 2, indicating whether a one-tailed or a two-tailed test is performed

  The function shall return the p-value.

Computing t- and p-values

Student t-test is a standard statistical procedure used in *statistical hypothesis testing*.

There are versions of the t-test based on the experiment design and the question studied (i.e., the nature of the null and alternative hypothesis), but all versions of the t-test follow the same logic.

What t-test is measuring. t-test is designed to test the hypotheses about the *means* of samples when compared to either means of other samples or population means. The tests are based on the comparison of the observed means and variances.

Null hypothesis. A *null hypothesis* in statistics is a hypothesis which your statistical significance test is trying to reject. In plain speak, the *null hypothesis* largely advocates the *status quo*. For t-tests, the null hypothesis usually takes the form of a statement that the means of the sample are the same (or that the mean of the sample is equal to the provided population mean).

Alternative hypothesis. An alternative hypothesis is the statement that we are trying to find support for in the observed data. Alternative hypothesis is a counter to the null hypothesis. Depending on the formulation of the alternative hypothesis, the t-test you are conducting may be *one-tailed* ot *two-tailed*.

A *two-tailed alternative hypothesis* has the form of a *"the means are not the same"* statement.

A *one-tailed alternative hyptothesis* has the form of a *"one mean is higher than the other"* statement.

t-value. The t-test proceeds by computing a statistic called the t-value. A t-value, or a t-statistic is usually defined as a *ratio of the departure of the estimated parameter from its notional value*[1]. The general form of a t-statistic for an estimator $\hat{\beta}$ of some parameter β is

$$t_{\hat{\beta}} = \frac{\hat{\beta} - \beta_0}{SE(\hat{\beta})},$$

where β_0 is the "notional" value being compared to the estimator and $SE()$ is the standard error of the estimator.

t-distribution. For smaller sample sizes, given a population, the means of samples will be distributed according to a t-distribution with a single parameter called number of degrees of freedom, which in most (but not all) cases is the sample size minus one. The probability density function of t-distribution can be described using the following formula:

$$f_d(t) = \frac{\Gamma\left(\frac{d+1}{2}\right)}{\sqrt{d\pi}\,\Gamma\left(\frac{d}{2}\right)}\left(1 + \frac{t^2}{d}\right)^{-\frac{d+1}{2}}$$

Here, d is the number of degrees of freedom, $\Gamma()$ is the Gamma-function.

The good news is that Python's NumPy package contains the ability to draw samples from the t-distribution!

One sample t-test. For one-sample t-test, the mean of a given sample $X = (x_1, \ldots, x_n)$ of values is compared to a known quantity μ - typically a known or a notional populaiton mean.

The *null hypothesis* for a one-sample t-test is *"the sample mean is equal to μ*. The alternative hypothesis for a one-tailed test is *"the sample mean is greater/smaller than μ*. The alternative hyptothesis for a two-tailed test is *"the sample mean is not equal to μ*.

The t-value for this test can be computed using the following formula:

$$t = \frac{\bar{X} - \mu}{\frac{s}{\sqrt{n}}},$$

where $\bar{X}$ is the sample mean and s is the sample standard deviation.

The number of degrees of freedom d for this test is set to be $d = n - 1$.

[1] https://en.wikipedia.org/wiki/T-statistic

Independent two-sample t-test under the unequal sample size and unequal variance assumption. Independent two-sample t-test checks for the equality of the means of two samples where the samples were drawn independently. The test described below assumes that (a) sample sizes may be different and (b) sample variances are different.

The test works with two samples $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_m)$. The null hypothesis is *"the means of samples X and Y are the same"*. The two-tailed alternative hypothesis is *"the mean of sample X is different from the mean of sample Y"*. The one-tailed alternative hypothesis is *"the mean of X is greater/smaller Y"*.

The `t-value` for this test can be computed as follows:

$$t = \frac{\bar{X} - \bar{Y}}{s_{XY} \cdot \sqrt{\frac{1}{n} + \frac{1}{m}}},$$

where $\bar{X}$ and $\bar{Y}$ are sample means and s_{XY} is computed s follows:

$$s_{XY} = \sqrt{\frac{(n-1)s_X^2 + (m-1)s_X^2}{m+n-2}}$$

The number of degrees of freedom for this test is

$$df = \frac{(s_X^2/n + s_Y^2/m)^2}{(\frac{s_X/n)^2}{n-1} + \frac{(s_Y/m)^2}{m-1}}$$

where s_X^2 and s_Y^2 are the respective variances of X and Y.

Paired t-test. In some cases, the two samples whose means are to be compared are not independent. For example, one sample may provide information about a response variable for a group of patients before treatment and the other sample has information about the same response variable after treatment. In general, a *paired two-sample test* is any test of two samples where for each value in one sample a "corresponding" value in the other sample can be found.

The paired t-test for samples $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ can be performed as follows:

1. Compute the sample $Z = X - Y = (z_1, \ldots, z_n)$, where $z_i = x_i - y_i$.
2. Convert your null and alternative hypotheses about X and Y to null and alternative hypotheses for Z.
3. Conduct one-sample t-test on Z.
4. Reject or fail to reject the null hypothesis for X,Y based on the ability to reject/fail to reject the null hypothesis for Z.

The number of degrees of freedom for a paired t-test is $d = n - 1$.

Rejecting/Failing to reject null hypotheses. Before proceeding with hypothesis testing, the researcher must establish the weight of the evidence that would lead them to reject the null hypthesis. The weight of the evidence is usually presented in form of a statement

> *The probability that the observed sample mean is the same as the base value[2] is p.*

The value p at which you are ready to reject the null hypothesis is called the p-value.

The **key observation that allows us to find the p-values lies in the nature of the t-statistics computed.** A t-value obtained in a t-test is drawn from the t-distribution with a given number of degrees of freedom. The p-value associated with a specific test depends on whether the test is on-tailed or two-tailed.

For *one-tailed* tests the p-value is the probability to observe a value *higher* than the computed t-value for the test.

For *two-tailed* tests the p-value is the probability to obsever a value *with higher absolute value* than the computed t-value for the test.

If the p-value computed this way is smaller than the pre-selected probability, then the **null hypothesis** is rejected.

Computing p-values. Given a t-value t and the number of degrees of freedom df you can approximate the p-value using Python and NumPy as follows.

- Create a large sample of values drawn from the t-distribution (`numpy.random.standard_t()` function).

  My sample size was around $100,000$ for relatively reasonable results.
- Given a t-value t, compute the number of sample values whose abosolute value is larger than t.
- If you are working on a one-tailed t-test, divide that number by two.
- Convert this value into a fraction of 1. **This is the p-value**.

Step 3. Hypothesis testing for Jester

Your second most important task for this assignment is to test your t-test implementations by testing a number of hypotheses related to the Jester dataset.

Here is a list of the hypotheses you will test. For each null and alternative hypothesis determine if a one-tailed, or two-tailed t-test is needed, what

[2] Either the other sample mean or the pre-specified value

type of t-test (one-sample, independent two-sample, paired two sample) is needed, and perform the appropriate data manipulations and analysis.

Your acceptance/rejection level for all one-tailed tests shall be at 5% (p-value of 0.05 or lower), and at 10% (p-value of 0.1 or lower) for all two-tailed tests.

Null Hypothesis 1. The first null hypothesis to be tested is

> *All jokes have received, on average, a similar reaction from those who rated them.*

The alternative hypothesis for this case is

> *There are jokes that received distinctly different (on average) reaction from those who rated them.*

To test this hypothesis, you must conduct a pairwise comparison of the means and variances of existing ratings *for each pair of jokes.*

The result of your hypothesis testing shall be:

- a 100x100 t-value matrix for all pairwise mean comparisons.
- a 100x100 p-value matrix for all pairwise mean comparisons.
- Count of the number of pairs of jokes on which the null hypothesis can be rejected.
- Count of the number of pairs of jokes on which the null hypothesis cannot be rejected.

Note: please make sure you do not test the joke against itself - you can simply put a value of 1 or 0 on the main diagonals of the matrices you are constructing.

Null Hypothesis 2. For this null hypothesis you will compare two jokes with the lowest Ids for which you have a complete set of responses. We will call these jokes *Joke 1* and *Joke 2*. The null hypothesis is

> All users of the Jester system rated *Joke 1* and *Joke 2* similarly.

The alternative hypothesis is

> Users tended to rate *Joke 1 higher* than *Joke 2*.

To test this hypothesis, find the *appropriate* t-value and p-value, report them and determine if the hypothesis is rejected or accepted.

Study 3. Consider joke number 50 (this is the highest rated joke). You have a lot of ratings for this joke. Let us consider the average score this joke received from everyone who rated it a population mean for this exercise.

Consider the following setup. You are sampling (without replacement) the population of people who rated *joke 50*. For each sample you create you can find the sample mean. You want the following:

- Your sample size to be as small as possible.
- The sample mean to be as close to the population mean as possible.

Conduct a study to find the sample size at which the probability that we reject the null hypothesis

> The population mean is the same than the sample mean (i.e., we can postulte that the sample was drawn from the population of people who rated joke 50).

is less than 5%. That is, of 100 samples of this size that you draw, you shall observe no more than 5 on which you reject the null hypothesis.

Put your simulation code into the notebook, run it, report the sample size that allows you to do this. (You may want to run more than 100 samples of the same size for each sample size you test just to be on the safe side).

Step 4. Explanations

After completing your studies, include a Markdown cell that contains your observations, thoughts and explanations of the observed results.

Deliverables and submission instructions

References

[1] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2), 133-151. July 2001.

[2] Ken Goldberg, Anonymous Ratings Data from the Jester Online Joke Recommender System, http://www.ieor.berkeley.edu/~goldberg/jester-data/.