MMCT Python Package Documentation

Version 1.1.2

Chris Walther Andersen

April 9, 2022

1 Introduction

The purpose of this software is to perform a statistical test on a dataset, in order to determine whether the data has been generated from a multinomial distribution. The name of the package, `mmct`, is an abbreviation of "Multinomial Monte Carlo Test". As the name suggests, the statistical test is performed using a Monte Carlo simulation. This document describes the mathematical foundations for this task.

The package is inspired by two other software packages, both of which do the same job: The Python-package met[1] is also a Python-package for performing multinomial tests, however `met` performs an *exact* test. While this strategy is certainly preferred to the Monte Carlo version (which will always be an approximate test), an exact test quickly becomes infeasible as the problem size grows. The other package to have inspired this project is the XNomial[2] package for the R-programming language. `mmct` is basically an XNomial-clone for Python.

2 Mathematical background

Assume we have done an experiment in which N observations of a system that can end up in one of k different states has been made (e.g. rolling a D6 20 times: $N = 20$, $k = 6$). We have tracked the number of times each state occurred and denote them $m_1, m_2, \ldots, m_k$, such that $\sum_{i=1}^{k} m_i = N$.

We want to test the hypothesis that this result is compatible with a multinomial distribution with parameters N and $p_1, p_2, \ldots, p_k$, where p_i is the probability of the state i occurring and $\sum_{i=1}^{k} p_i = 1$. The result of this test should be a p-value for the null hypothesis that $m_1, \ldots, m_k$ are drawn from this multinomial distribution.

2.1 Monte Carlo sampling

The test is performed in two steps: Monte Carlo sampling a bunch of multinomially distributed sets of data, and afterwards determining the likelihood of the original dataset occuring based on the sampled data.

First we write the original data under test as

$$X = \{m_1, m_2, \ldots, m_k\}\,, \tag{1}$$

[1] https://pypi.org/project/met/

[2] https://cran.r-project.org/web/packages/XNomial/vignettes/XNomial.html

where m_i is the number of observations in bin i. We note that the probability observing X given the hypothesised parameters of the distribution is

$$\begin{aligned} P &= \Pr\left(m_1, \ldots, m_k | N, p_1, \ldots, p_k\right) \\ &= \frac{N!}{m_1! \cdots m_k!} p_1^{m_1} \cdots p_k^{m_k}. \end{aligned} \tag{2}$$

We now draw N_S random samples from the hypothesised distribution, i.e.

$$X^j = \left\{m_1^j, m_2^j, \ldots, m_k^j\right\}, \quad 1 \leq j \leq N_S. \tag{3}$$

Each sample has been drawn with the probabilities of the hypothesised distribution $p_i, 1 \leq i \leq k$ and for each sample $\sum_i m_i^j = N$. If N_S is large enough, we can now estimate the likelihood that X (the data under test) is drawn from the same distribution by comparing it to the randomly generated samples.

2.2 Sample grading

To do this we need a way of grading our samples to determine what is likely and what is unlikely. To do this we grade each sample by its probability of occurring according to the probability distribution. This probability is calculated as

$$\begin{aligned} P^j &= \Pr\left(m_1^j, \ldots, m_k^j | N, p_1, \ldots, p_k\right) \\ &= \frac{N!}{m_1^j! \cdots m_k^j!} p_1^{m_1^j} \cdots p_k^{m_k^j}. \end{aligned} \tag{4}$$

Calculating P for all samples and our data under test in this way, we can now rank all the samples from the most likely (largest P^j) to most unlikely (smallest P^j) and look where P, the probability of the data under test, fits in. The p-value of our test is simply the number of samples with a probability P^j smaller than P divided by the total number of samples N_S:

$$\text{p-value} = \frac{N_{\text{smaller}}}{N_S} \tag{5}$$