原文

来源 2020-04-10（可能会更新）

1. 定义Javascript函数。 d = condition_specific_correlation(g1, g2, info)。输入g1是g1在各个样本中的表达值，JSON类型。 输入g2是g2在各个样本中的表达值，JSON类型。 输入info是每个样本的信息。 其中返回值d是个字典。Key 是生物条件，比如 'leaf'。 Value 是 {'r':0.93, 'n':62, 'p-value':0.005}， 即在生物条件leaf下所有62个样本所得的g1,g2相关性系数是 r=0.93， p值是0.005。
2. 计算相关性系数的empirical p-value。 具体的做法是使用Permutation Test，把g1的所有表达值顺序打乱，把g2的所有表达值顺序打乱，然后求g1与g2的相关性系数r'。重复上面的步骤n次，数出超过r的r'的个数n'。 n'除以n就是empirical p-value。 n至少要在十的三次方级别。
3. 计算相关性系数theoretical p-value。需要与R的cor.test返回的值一致。
4. 对代码更加详细的注释（为了帮助后期维护）。
5. 对代码更加全面的测试（我们已经产生了三类测试数据，正确性测试数据，速度测试数据，出错测试数据）。测试数据由 generate_test_data.py 产生。
6. 更好的ReadTheDocs文档（对输入数据的说明，对输出的解释）。文档中不应该有错别字，不应该有语句不通顺的地方，不应该有含糊其辞的地方。。
7. 更加简洁的用户界面。能省则省。

总结

改善1：增加注释

对代码文件增加注释。

帮助大家理解我这个项目。

改善2： 调整UI

大后期的混子行为。

功能1：字典函数

`condition_specific_correlation(g1, g2, info)`

其实已经实现了，可以稍微整理整出来。

功能2：empirical p-value

就是在原来的基础上复杂几步，不算太难实现。

功能3：导出csv

据说加几句小代码就可以了。

文档1：测试文档

要写出一个符合他测试工具的txt文件，来表示当前版本的测试情况。

文档2：用户文档+开发文档

分别是DevDoc.md UserDoc.md，放在doc路径下。

这个协作路线比较长，可以慢慢扯。