实习生标注、修改和审核教程

注：1. 数据集分发时就需要在 redmine 上标清楚是分发给谁的，方便溯源。

2. 二审/三审负责人发现错误后需要在 redmine 上更新 feedback 来发布修改任务

3. report 中需要用修订模式指出错误，返回修改时一并附上。

4. 实习生数据标注指南、修改指南和审核人员审核指南见下文。

5. 同一时间下发的数据集争取在同一个周期内完成。如，每周下发 10 个数据集，那么每周尽量收上来 10 个数据集。

目录

如何给新实习生注册用户？

Redmine 用户注册：需要由管理员权限的人在 administration 中的 user 选项卡中添加用户。

Linux 用户注册：由李玥负责提供 端口号、密码和服务器 IP。

注册好的实习生该做些什么？

1. 安装软件登陆账户

2. 教程：

1) 网页版教程（包含流程步骤及注意事项）

http://118.190.148.166/biodb/dataset_curation/

2) 视频教程（有错，不建议细看，之后会有更新），以及邮件模板

https://pan.baidu.com/s/1NAJ6_BVTdB1IarZXh6gHaw 提取码: hbmm

3. 按照《实习生数据标注指南》完成用户帐号下的测试数据集，并把做好的 script.ipybn 中的所有表格展开，把这一网页生成 report.pdf，交给二审负责人：张萌栩。等待批注回复。

实习生数据集标注以及修改指南

数据集标注指南

注：实习生若是在一周内无法完成分配的数据集，需要及时向管理人员汇报。

进行数据标注，需要从 data/tutorial/template/code/ 文件夹下复制 script.ipynb 代码文件模板到自己的文件夹下参考运行. **不可以直接在模板中运行！！！脚本会随时更新，请随时关注新加入的字段，不要一直使用自己存在文件夹下面的 script，每次用都从 template 下复制最好。**

流程：审核人员下发任务到 redmine——拿到数据集——进行标注——生成 pdf 版本的 report——redmine 回复完成并附件添加 report——等待二审和修改

1. Report：

每新标注一个数据集，都要把自己写的代码，每一个表格尽可能的展开，把网页打印-选择生成 pdf 版，并把文件重新命名为 report_数据集编号_用户名和名字_日期.pdf，然后交给二审人员

如，report_GSE*****_user_72_zhangsan_20200320.pdf

2. unstructuredData 需要摘录的信息：

- ✧ **subDataset：由一审人员写在 description.txt 上，请复制过来**
- ✧ **description：由一审人员写在 description.txt 上，请复制过来**
- ✧ **correspondingFigure：由一审人员写在 description.txt 上，请复制过来**

- **title**：注意这里填写的是文章的 title，不是 GSE 界面的 title！**注意不要 title 不要加句号**

- **authors**：使用内置函数获取，script 中的引号内填 pubmedID

- **accessionNumber**：这里填写 GSE 开头的编码,可以在 pubmed 网站中文章下面的 Associated data 这一栏获取，一般 GSE 三个字母后面有 5 位数。

- **pubmedID**：填写 pubmed 编号，pubmed 网站上文章题目下面的 PMID 就是。

- **keywords：** 使用内置函数获取，script 中的引号内填 pubmedID

- **abstract**：把文中的 abstract 一段复制下来，注意不要多了或少了。不要填写 GSE 界面的 summary!

- **sourceID**：出现在 pubmed 网站中文章题目下方。DOI 的网络连接，如一篇文章的 DOI 为 10.1038/nature12172，那么它的网络链接为 https://doi.org/10.1038/nature12172

- **libraryPreparationMethod**：是指细胞测序所使用的技术，可在文章中查找；也可以在文章相应的 GEO 网址中（即在 GEO 网站首页输入 GSE 编码即可）查找。文章中的技术，使用 my_inspector.libmethod_keywords 调出关键词：10x chromium(注意大小写!), drop-seq, microwell-seq, C1 Fluidigm, inDrops, Smart-seq2, Smart-seq, CEL-seq, CEL-seq2, MARS-seq, msSCRB-seq, SCRB-seq 基本上都出自于上面几种测序技术。

- **sequencingPlatform：**这里填写测序平台。可以在文章相应的 GEO 网址中找到。如，Illumina HiSeq 2500，Illumina HiSeq 500，Illumina HiSeq 2000 等。

- **clusteringMethod**：这里填写聚类分析的所使用的方法，可在文章中查找。聚类分析常用方法：k-means, affinity propagation, mean-shift, spectral clustering, Ward hierarchical clustering, agglomerative clustering, DBSCAN, Gaussian mixtures, birch.

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

想要更详细的了解可以参考以下网址：https://blog.csdn.net/ztf312/article/details/97951928

- **biomarkerDerivationMethod**：是指 marker gene 的算法，一般在文章的 method 里面有，是找 cluster 下游的特异基因的方法。一般是 t-test 或者 wilcoxon 之类的，尽量在文章中找到对应的 marker genes 的算法。

- **fastqURL**：在 EBI 网址上查找文章名字，然后点击相应链接，查看 data 信息，链接就会跳转到一个有大写字母 PRJNA 和一串数字结尾的地址，如：https://www.ebi.ac.uk/ena/data/view/PRJNA542142 其实只要网页里面有文章的 fastq 相关信息即可

- **figureURL**：填写文章的摘要图网址。对于明确表示有 graphic abstract 的文章，我们需要把这张图放在展示页面上，如果没有 graphic abstract，那么放文章的第一张图。可以在文章页面访问原图，使用原图链接，或者访问杂志网站，使用杂志提供的图片链接。图片链接结尾一般是*.jpg .png .gif 之类的文件形式。链接需能在浏览器中打开，但不可使用只自动下载的图片。

- **isFigurePublic**：填 True or False，是否对所有网络均公开可见，如一般 abstract 中的 figure 为公开的，就填写 True。

- **taxonomyID**：可查看文章相应的 GEO 网址中的 sample 里的信息，一般人填 9606，鼠填 10090（只用填写了这个才能生成 geneannotation）

- **genomeBuild**：可查看文章相应的 GEO 网址中是否有相应字段。人为 hg/GRCh，小鼠为 mm/GRCm 这类格式，其他物种可以填 notAvailable。

- **annotation:** 是指基因的注释信息是什么

- **journal**： 使用内置函数获取，引号内填 pubmedID，如无法获取，使用命令 my_inspector.journal_keywords 调用 journal 的列表查看

- **citation:** 指引用次数

- **tissue**：文中选填,填为 list 格式。需要在 https://www.ebi.ac.uk/ols/ontologies/bto 这个网址查询是否有填写的 tissue。

- **tissueOntology:** 不填写，向下运行代码自动生成

- **clusterAvailability：**填 True or False，意思是能否找到对应的 cluster 信息

- **以下字段为文章的研究主题**

- **disease：**填 True or False

- **methodology：**填 True or False （True 仅包括本身研究测序方法的文章）

- **cancer：**填 True or False

- **neuroscience：**填 True or False

- **developmentalBiology**：填 True or False （发育生物学，如细胞的分化）

- **immunology：**填 True or False

✧ **cellAtlas：** 填 True or False （遗传图谱）

✧ **tSNEAvailability**：填 True or False，意思是能否找到 tSNE 的坐标信息

✧ **（新加入）isCultured：**填 True or False，意思是 scRNA-seq 所用的细胞是作者自己传代培养（True）的还是原代细胞（False）。

✧ **（新加入）isTPMNotAvailable**：填 True or False。Some articles provide norm matrix only and cannot generate TPM (we can only treat the norm as if it is TPM)，此时填上 True。但在矩阵的 normalizationMethod 这一列里要标注清楚这个不是 TPM 矩阵！

✧ **（新加入）diseaseOntology：**在 https://disease-ontology.org/中寻找 disease_name。

```
metadata['isCultured'] = ''#填Ture or False, 意思是scRNA-seq所用的细胞是作者自己传代培养 (True) 的还是原代细胞 (False)
metadata['isTPMNotAvailable'] = ''#填True or False. Some articles provide norm matrix only and cannot generate TPM (we can only t
```

```
# 若disease为True需要填diseaseOntologyName, 根据文章到https://disease-ontology.org/网站查找
# 若disease为False直接留空,并运行即可
disease_name = ['']
metadata['diseaseOntologyName'] = disease_name
metadata['diseaseOntologyID'] = my_builder.get_disease_ontology_id(disease_name)
```

✧ （当 cancer 为 True 时需要填写）cancerDescription：按照 script 脚本里面的要求填写，文章不是 cancer 相关可以不填

```
# 若cancer为True需要填写以下字段
cancerDescription = {}
cancerDescription['TIL'] = '' # 填 True or False.Tumor Infiltrating Lymphocyte肿瘤浸润淋巴细胞的缩写
cancerDescription['TME'] = '' # 填 True or False.Tumor Micro-environment肿瘤微环境的缩写
cancerDescription['NCIBodyLocation'] = '' # 从给的keywords选填, 如: Head and Neck
cancerDescription['NCIBodyLocationSubtype'] = '' # 根据NCIBodyLocatio对应的keywords选填, 如: Hypopharyngea
cancerDescription['TCGAStudyAbbreviation'] = '' # 从给的keywords选填, 如: LAML
cancerDescription['TCGAStudyName'] = '' # TCGAStudyAbbreviation对应的全称如: LAML对应的全称为: Acute Myelo


# 运行 my_inspector.nci_cancer_typing 查看 NCIBodyLocatio的keywords
# 运行 my_inspector.tcga_cancer_typing 查看 TCGAStudy的keywords
```

```
# 若cancer为False cancerDescription 填{}即可
cancerDescription = {}
```

```
metadata['cancerDescription'] = cancerDescription
```

```
my_builder.get_metadata('')[3]
```

对于各字段的说明：

- ✓ disease：是指文章研究内容是否与疾病有关，例如研究某种疾病、从疾病患者采样等，但对于一般性的研究某一通路、细胞等作用，最后认为可能与某种疾病有关时需要多加判断，准则是此篇文章出发的目的和主题内容是否是与疾病相关，此外，cancer 必为 disease；

- ✓ methodology：此为高出错字段，应更加注意。方法学是指文章开发了某一新的算法、分析方法、技术手段或者比较现有算法并进行评价的。很多文章习惯性采用“我们使用了新的策略”这样的表述，这种并不是方法学文章的特征，也并不是用了某一算法、框架的就是方法学，准则是“新”的方法以及文章主旨；

- ✓ cancer：很容易理解，研究主旨是关于癌症的；

- ✓ neuroscience：研究主旨、研究样本关于神经生物学或神经组织/细胞；

- ✓ immunology：研究主旨、研究样本关于免疫学或免疫器官/细胞；

- ✓ developmentalBiology：研究的是某一个体/器官/细胞类型/组织的时间跨度下的发展变化，关键字如发育、衰老、胚胎、分化等，此外，使用胚胎的文章大多可归为发育；

- ✓ cellAtlas：对某个个体、器官、组织、系统。进行全面普遍的研究，即关注的问题不是某一特定的细胞类型或组成的功能与变化，而是清楚表达其内部组织与层次关系，可以类比为“参考基因组”这样的概念，不是为了研究某个基因或某个人，而是为了提供可参考的人类基因组。也会有文章进行全面测序和分析后，只抓住某个点阐述，此时也仍属于 cellAtlas，判断标准是测序的样本和数据是否包含了其采样的全部，比如有的取了肝组织，但只测肝内免疫细胞，则不符合要求，应该包括基质细胞、上皮细胞等等。

3. cellAnnotation

这里需要查看一些矩阵里的信息来完善 cellAnnotation 表格，需要用到代码

注意：没有的数据，不要留下 NaN，若找不到信息就填成 notAvailable。

- **填写 CellID:** 一般在 Matrix_rawCounts 或者 Matrix_normalized 有对细胞的编号，只需要取出后安进 cellAnnotation 表格中的 cellID 这一列就可以了。

- **填写 meta_字段：**可以在读取了 sample 信息后查找是否有值得填写的字段，一般可以查看文章对应的 GEO 网址里面 characteristic 一栏中的信息来填写这一部分，注意需要与细胞一一对应。

- **sample 表格中的信息如何与 CellID 找到关系来填写 cellAnnotation 表格里的内容？**

  一般是通过 sample 表格中 title 这一列来寻找关系的：

1） title 中的**信息**与 cellID **信息相同**，**只是顺序不同**：可以借助循环代码历遍表格，利用当 title 中信息与 cellID 中信息相同时，才能填入 cellID 这一行的其他相对应的信息为条件，填入表格信息。

2） 如果 title 中的**信息**和 cellID **不同**，需要在 sample 这个表格中另外找能够跟 cellID 对应起来的信息。再进行 1）中操作。

- **填写 clusterName 和 clusterID（tSNE and UMAP）：**

1） 首先要在文章中的 supplemental information 栏下查找作者是否给了是否有相应的聚类信息（包括 clusterName/clusterID，tSNE1/2, UMAP1/2 等）。

2） 还可以在读取作者提供的矩阵（文章对应的 GEO 网址下载的）中查找。

3） 如果都没有，需要发邮件给作者。**（如果没有得到回复，审核人员可以再发一遍试试）**

4） 最后也没找到 clusterName/clusterID 的可以先空中，之后运行我们自己的计算脚本，自动生成为数字编号的 clusterName/ID、tSNE1/2、UMAP1/2。这部分代码出现在 template/script.ipynb 中的 **6.使用脚本自动生成其他项**中的 6.4，同时 cellAnnotation 的表格中会多出来两列：clusteringMethod 和 clusterName_scibet。**如下图：**

6.4 Clustering

原文及作者没有提供clusterName时，需运行下面的函数得到聚类结果

```
[ ]: my_builder.calculate_cluster()
```

5）若 tsne 和 umap 作者也没有提供，跟 cluster 一样，实在不行可以运行自己的脚本自动计算。6.3 中的代码生成 2D（第一行代码） 和 3D 的图（第二行代码）

6.3 生成tSNE,UMAP

如果出错首先查下TPM格式是否正确，是否存在负值

```
[ ]: # generate dim_red
     # 其中tSNE坐标需先从文章中寻找或者发邮件给作者询问，若无可使用下面函数
     my_builder.calculate_dim_red(tSNE = True, UMAP = True)
```

```
[ ]: my_builder.calculate_3d_dim_red(tSNE=True, UMAP=True)
```

- **填写 cellOntologyName 和 cellOntologyID：这两项与 clusterName 是相对应的**

1）如果文中提供 clusterName 等相关内容，cellOntologyName/ID 可以在 https://www.ebi.ac.uk/ols/index 网址搜索细胞信息，并输入与 clusterName 最相近的 cellOntologyName 和 ID **(id 现在已经不需要填写，运行 6.5 代码根据 cellOntologyName 自动生成)**

6.5生成cellontologyID

```
In [ ]: my_builder.generate_ontology_id()
```

2）如果文章没有找到 clusterName，那么就只能在文章中寻找进行聚类分析的是什么细胞，

有时候只能找到这篇文章是研究**细胞的，如骨髓细胞，那就只查找骨髓细胞相应的cellOntology信息并全部填上即可。目的就是填上就好，能填就填，轻易不要填notAvailable，更不可以空着。

4. Matrix

注：当只提供了 rawcounts 文件的时候需要把矩阵存成 mtx 格式，一旦存了 mtx 格式，所有矩阵都需要存成 mtx 格式。

从 Matrix_rawCounts 或 Matrix_normalized 生成 TPM

注意：Matrix_normalized 优先于 Matrix_rawCounts 使用生成 TPM。

Cell number，即有效细胞数的判断：

（有效细胞数即数据整合后或经过 normalization 后被使用的细胞数。）

1. 文章中给出，可以是几种细胞数的和。

2. 通过 Matrix_normalized 的 row number 数判断：cell number = row numbers **(有时需要根据情况随机应变)**

建议用方法 2 复查方法 1 中作者提供的有效细胞数。

为什么要用 TPM：

在大数据中查询基因表达时，不同数据集在同一个 scale 上便于比较，因而要有统一的标准。

为什么尽可能不用 rawCounts 生成 TPM：

Matrix_normalized 中数据往往是对 Matrix_rawCounts 数据做出某些处理后所得，尤其是

经过 gene or cell filtering。因此，通常 Matrix_rawCounts 可能有更多的 cell，即存在无效细胞。因而 Matrix_rawCounts 列表就可能和后面的 cellAnnotation 不对应，故后续无法用其向 cellAnnotaion 表格中插入数据。

判断 downloaded data 是 Matrix_rawCounts 还是 Matrix_normalized：

根据测序原理，检测结果 Matrix_rawCounts 中一定全为整数，有负数则不是 Matrix_rawCounts。当算法为 pseudo alignment（kallisto 等，这种算法越来越受欢迎）时，会出现小数。

判断是否直接使用 Matrix_rawCounts 生成 TPM：

当且仅当文章未提供 Matrix_normalized 数据，可以直接用 matrix_rawCounts 生成 TPM。

判断下载的 Matrix_normalized 是否可以直接用于生成 TPM：

作者在文章 method、analysis 部分可能提及使用 multi-step normalization strategy，诸如使用 previously described method, reduse batch effects, use ComBat method, etc.

如若不确定直接下载的 Matrix_normalized 数据是否经过 logarithm transciption，则使用下载的 Matrix_normalized 数据生成 TPM 矩阵，并检查 TPM 的正确性。TPM 正确即表明 Matrix_normalized 中数据可以直接使用。若 TPM 不正确，则需间接使用 Matrix_rawCounts 生成 TPM。

如何填写 normalization method?

下载的 normalized 矩阵和生成的 TPM 矩阵都需要填写一列 normalizationMethod。

1) **Normalized 矩阵：**一般是下载得来，需要在文献中找出 normalized Matrix 是怎样被标准化的，有的是 FPKM,有的是 RPKM,还有 log2(TPM+1)等等形式，载入 normalizedMatrix 的时候需要在第一列标明 normalizedMethod。格式： 如，FPKM。（直接填写原文作者

把 rawcounts 标准化的方法，不要添加其他字符。）

2) **TPM 矩阵：**

- 如果由 rawcounts 矩阵转化而来，填写：**TPM from raw counts**
- 如果由 normalized 矩阵转化而来，填写： **TPM from FPKM** （根据 normalized 矩阵的方法而改变，如：TPM from log2（TPM+1） ）

3) **如果遇到无法转换成 TPM 的**，TPM 的 normalizationMethod 一定**要强调这个不是 TPM**！并且一定要很**详细的注明这个矩阵的标准化方法是什么**，如：**矩阵不是 TPM! 是 TMM 矩阵**。

检查 TPM 矩阵正确与否方法：

TPM 中横行代表基因，纵列代表细胞。对于任意单个细胞，当其对应横行中的基因值相加和为 1000 000 (10^6)，则 TPM 正确。**如果文章提到使用 UMI，那么 CPM（也叫 RPM） = TPM,** 行和都是一百万，M 代表 million。

间接使用 Matrix_rawCounts 生成 TPM：

注：当需要用 Matrix_rawCounts 自动生成 TPM 时需要先填写 unstructuredData 中的 libraryPreparationMethod。

在文章作者给出 Matrix_normalized 情况下，判断是否可以直接用 Matrix_normalized 生成 TPM，或者仍需使用 Matrix_rawCounts 部分数据生成 TPM 的方法：

1. 作者于文章 data analysing/processing 中明确指出 median normalize (to zero)，或者 Z-Score、TMM，以及一些依赖函数库的结果不能使用 Matrix_normalized 生成 TPM，要根据 matrix_normarlized 中的的 cell ID 挑选出 Matrix_rawCounts 中的的有效 cell ID，使用有效 cell ID 生成仅包含有效细胞的新 Matrix_rawCounts，用其生成 TPM。

2. 作者文章 data analysis/process 部分未发现不可用 normalized data 生成 TPM 的条

件，则可做一个 Matrix_normalized。如果该 Matrix 里有负数，则作者 normalization 方法可能为 median normalize (to zero)，或者 z-score 等等。此时同样需从下载的 Matrix_rawCounts 中根据 cell ID 挑选出 Matrix_normalized 中使用的有效细胞，再使用筛选细胞后的新 Matrix_rawCounts 生成 TPM。

从 logarithmic transformed Matrix_normalized 到 TPM：

若作者明确提供的 Matrix_normalized 已经经过 log 变换，如 log2（TPM+1），则需逆推原始 Matrix_normalized，再使用原始 Matrix_normalized 生成 TPM。

logaN, a = 2, 10, e..., TPM = a^N /100 000，normalizationMethod: loga(TPM+1)

特例：如果文章写明使用 logarithmic transformed data，但并未指出是何种 log 变换，尝试用 log2、log10、ln 等等变换后 Matrix 每行基因和均不为 1000 000，且每行基因的和相近，则可以直接在 Matrix_normalized 基础上进行归一化，即每个基因除以该行基因的和，再乘上 1000 000。

从 FPKM, RPKM 到 TPM 的转换：

接将 Matrix_normalized 这个 矩阵除以 行的和 乘上 1000 000，这样得出来的就是 TPM 矩阵了。

示例代码：（需要根据不同的情况变换，仅作示例）

```
df_downloaded = pd.read_csv
('../downloaded_data/GSE_supplementary_data/GSE75413_genes.fpkm_table.txt.gz', sep =
"\t",index_col = 0)                              #读取矩阵

df_downloaded.head()    #显示矩阵前前 5 行出来看看



df_norm = df_downloaded.T   #行和列交叉互换，即行变成列，列变成行
```

```
df_norm.head()
```

```
fpkm_array = df_downloaded.T

b = np.sum(fpkm_array,axis=1)

df_tpm = fpkm_array / b.values.reshape(-1,1) * 1e6                #1e6 代表 1 million
```

5. 生成的其他文件：

1） TPM 需要从 rawcounts 生成时，运行 6.1（注意：只有当 libraryPreparationMethod 填写完了之后才能运行成功）

2） geneAnnotation 文件：这个是运行代码自动生成的，在 script.ipynb 中的 6.2 可见。（注意：只有 taxonomy 填写了物种的编码之后才能成功生成）

3） 当真的没有 tsne 和 umap 坐标没办法从作者那里得到回复时，可以运行我们自己的计算脚本，计算 tSNE 和 UMAP 的二维和三维， 运算脚本在 script.ipynb 中的 6.3 可见。

4） clustering 计算，在 6.4 中，仅在找不到作者又未回复时使用，记得运行第二行代码来判断 isbadtSNE，来判断 tSNE 图上的细胞簇点是否能分得开。

5） cellOntologyID，在 6.5 中运行后根据 cellOntologyName 自动填写

6.6 有rawCounts需要计算qc

```
In [ ]: my_builder.calc_qc_value()
```

6.7 自动生成其他文件

```
In [ ]: my_builder.auto_calculation()
```

6.8 物种为人需要运行以下代码

```
In [ ]: my_builder.calc_cell_cycle_score()
```

6）

7） 填写 README:

a） 第一个 block 里面运行的代码用来赋值和读取 readme.json

b） 第二个 block 里面填写信息

Readme['author'] = '' #填写自己的名字拼音，如 Xiaoming

Readme['date'] = '' #填写完成日期

Readme['modificationDate'] = '' #填写修改日期

Readme['unfinishedParts'] = [''] #未完成的部分，包括矩阵及 cellAnno 中的 cluster 等

Readme['authorComments'] = '' #如果数据分成了几个 part，写清楚该 part 代表什么类型的数据

Readme['otherComments'] = ''

c） 最后记得保存

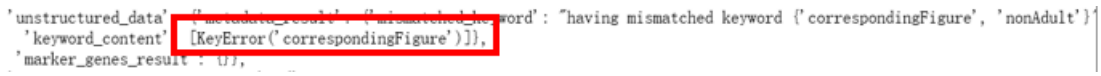
6. 运行检查器（之后可以让陈淳继续添加）

运行代码 my_inspector.inspect()

在运行检查器后经常会出现报错，这时就需要根据检查器报错来排查错误信息

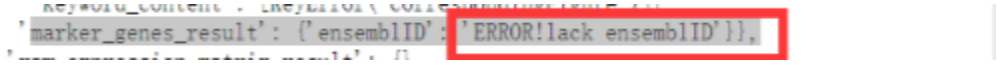
常见报错：

1. unstructuredData 中缺少内容：



只需要补充上即可。

2. markergenes



markergenes 中缺少 ensemblID,这个 ID 是有内置函数换算的，是跟 gene 一一对应的。

这里报错是因为 cluster 里的 ensemblID 都为 notAvailable，这可能是因为函数没有运行正确，也有可能是基因库没有对应的。可以重新运行试试，如果还是没有，可以在群里询问。

3. 如果没填写 libraryPreparationMehtod 或者 taxonomy 就想生成 toTPM 或者 geneAnnotation，也会报错提示

7. 遇到的问题与解答

1） rawcounts 文件太大，生成 TPM 总是崩溃怎么办？

答：可以试试存成 sparse matrix，这样会小一点，然后再转成 TPM。细胞超

过 5 万以上的就不要存成 tsv 格式了。现在如果只提供 rawcounts，都需要生成 mtx 格式的矩阵。

2） libraryPreparationMethod 里面的 msSCRB-seq 是什么？

答：是 molecular crowding SCRB-seq。https://omictools.com/mcscrb-seq-tool

3） jypyter_no_port 错误解决方法？

答：百度网盘里有文档解释

https://pan.baidu.com/s/1NAJ6_BVTdB1IarZXh6gHaw 提取码: hbmm

9) **TPM 矩阵如何从 sparse matrix 转换过来？**

答：rawcounts 存储为 sparse matrix 时运行 my_builder.toTPM()这个函数生成 TPM #运行这个函数前，需要保证 rawcounts.tsv 这个文件是空的才行

10) **在读取文或运行件的时候，跳出 received signal 15，stopping 然后 shutdown 是什么原因呢？有什么处理办法吗？**

答：这种情况一般是因为运行的文件占用内存太大，被程序自动 kill 了，可以尝试在人少的时候再试试，是在不行就联系发任务的人。

11) **在运行 auto_calculation 的时候，显示 clusterName Error！怎么办？**

如果出错首先查下TPM格式是否正确，是否存在负值

```
In [134]: # generate dim_red
          my_builder.calculate_dim_red(tSNE = True, UMAP = True)

          TPM loaded
          feature selection and PCA compression finished
          UMAP finished
          tSNE finished

In [136]: my_builder.auto_calculation()

Out[136]: 'clusterName ERROR!'
```

答：没有 cluster 的信息时要先填上 notAvailable，再用我们自己的脚本计算 cluster，不能留下 NaN。

12) **在运行 calculate_cluster（RUN = True）的时候报错，是什么原因?**

(1)

```
    408             return self._fit_truncated(X, n_components, self._fit_svd_solver)

/usr/share/intel/intelpython3/lib/python3.6/site-packages/daal4py/sklearn/monkeypatch/pca.py in _fit_full(self, X, n
_components)
    126
    127     def _fit_full(self, X, n_components):
--> 128         return _fit_full_copy(self, X, n_components)

/usr/share/intel/intelpython3/lib/python3.6/site-packages/daal4py/sklearn/monkeypatch/pca.py in _fit_full(self, X, n
_components)
     55                         "min(n_samples, n_features)=%r with "
     56                         "svd_solver='full'"
---> 57                         % (n_components, min(n_samples, n_features)))
     58     elif n_components >= 1:
     59         if not isinstance(n_components, (numbers.Integral, np.integer)):

ValueError: n_components=50 must be between 0 and min(n_samples, n_features)=47 with svd_solver='full'
```

(2)

```
    403         df_test_selected_genes = df_test.reindex(columns=self.reference_genes, fill_value=0)
    404         df_test_logged = np.log1p(df_test_selected_genes)

/usr/share/intel/intelpython3/lib/python3.6/site-packages/pandas/core/frame.py in __init__(self, data, index, columns, dtype, copy)
    422             else:
    423                 mgr = init_ndarray(data, index, columns, dtype=dtype,
--> 424                                    copy=copy)
    425
    426         # For data is list-like, or Iterable (will consume into list)

/usr/share/intel/intelpython3/lib/python3.6/site-packages/pandas/core/internals/construction.py in init_ndarray(values, index, columns, d
type, copy)
    165         values = maybe_infer_to_datetimelike(values)
    166
--> 167     return create_block_manager_from_blocks([values], [columns, index])
    168
    169

/usr/share/intel/intelpython3/lib/python3.6/site-packages/pandas/core/internals/managers.py in create_block_manager_from_blocks(blocks, a
xes)
   1658         blocks = [getattr(b, 'values', b) for b in blocks]
   1659         tot_items = sum(b.shape[0] for b in blocks)
-> 1660         construction_error(tot_items, blocks[0].shape[1:], axes, e)
   1661
   1662

/usr/share/intel/intelpython3/lib/python3.6/site-packages/pandas/core/internals/managers.py in construction_error(tot_items, block_shape,
axes, e)
   1689         raise ValueError("Empty data passed with indices specified.")
   1690     raise ValueError("Shape of passed values is {0}, indices imply {1}".format(
-> 1691         passed, implied))
   1692
   1693

ValueError: Shape of passed values is (2700, 33538), indices imply (2700, 2)
```

答：（1）报错中显示细胞数只有 47 个，细胞数太少的时候不需要进行细胞聚类。

（2）遇到报错可以先运行检查器，查看其他部分是否填写完整无误。在这里如果 rawcounts 存成 mtx 格式在生成 tpm 就不需要运行 geneAnnotation 函数了，要不然会覆盖掉。（现在已经优化了代码，不让你运行 geneAnnotation 了）

（3）出现 ValueError: cannot reindex from a duplicate axis 报错,是因为矩阵中基因名或者细胞名可能有重复。基因名有重复可以使用 ensemblID 代替基因名，若不行可以在重复的那一列做个标记区别开来。系报名重复时，同一个细胞的基因表达量应该是一样的，因此需要去除重复的细胞。

13) **metadata 中的信息有时使用 my_builder.get_metadata('')#引号中填写 pubmedID 获取不到相应的信息，这是为什么?**

答：可能是因为文章中本来就没有，可以在文献里确认一下，文献中确实没有的就算了。

9) **metadata 中的 tSNEAvailability 需要发邮件问作者吗?**

答：新标注的时候需要发，修改数据集的时候可以不用发。

10) **citation 的次数在哪里找?**

答：pubmed 中会有，也可以直接 google

11) **从 rawcounts 生成 TPM 有时候会有基因数不一样的情况**

检查后发现 rawcounts 矩阵中多出的基因为 ERCC 开头，这种情况时不用管就行了

12） 矩阵存成 mtx 格式之后运行 auto_calutation 还是会超时断开，就只能收回来在 infinity 上跑了。或者先清一下服务器内存看看。清服务器内存可以看 数据节省内存文件夹

实习生数据集修改指南

标注数据集之后的修改注意事项用黑色写出，不用新建文档，在自己的代码中修改即可

回溯型数据集的修改注意事项用绿色写出（当有回溯型的数据集需要修改的时候，再放出给实习生）

实习生收到修改意见后，需要另外新建一个 phthon3 的文件，命名为 revision.ipynb，在这个文件中修改标注的数据。

1 以下代码是运行配置项，需要在修改数据集的时候运行

```
import importlib.util

import sys

import os

import pandas as pd

import numpy as np

from scipy.io import mmread, mmwrite

import scanpy as sc

import anndata as an

import louvain

sys.path.append('/home/biodb/data/abio_database_pipeline/') #注意如果是在阿里云服务器上，该地址改为：'/home/ztr/abio_database_pipeline/'

from pipeline.datasets_curation import datasetBuilder
```

```
from pipeline.datasets_curation import inspector

from pipeline.datasets_curation import downsample

starting_dir = '/home/biodb/data/user_33/No_1/part_1' #根据自己更改的数据集更换路径

my_inspector = inspector.Inspector(starting_dir)

my_builder = datasetBuilder.DatasetBuilder(starting_dir)

my_downsample = downsample.Downsample(starting_dir)
```

2. 以下三个部分的数据如果进行了修改，需要注意运行下面的自动化函数：

一、矩阵

1.1 raw counts, normalized 变动，需要考虑重新生成 TPM 矩阵；（注意 TPM 如果重新生成了，参考 1.2 运行自动生成函数）

1.2 TPM 矩阵如果变化了，所有自动生成函数都需重新运行，包括：

```
my_builder.generate_geneAnno()

my_builder.calculate_dim_red(tSNE = True, UMAP = True)

my_builder.calculate_cluster(RUN = True) #注意参数 RUN=True 会将原 cluster 信息删除并更新，所以如果原文提供 cluster 信息的时候不要运行该行!

#以下两个函数最后运行

my_builder.auto_calculation()
```

my_downsample.downsample() #如果需要运行该函数，必须先将原来的 downsample_data 文件夹删除!

二、cellAnnotation（以下情形为矩阵未改动的情形，如矩阵变动参考 1.1）

2.1 clusterName 改动，需要重新运行：

my_builder.auto_calculation()

my_downsample.downsample(tpm_downsampled = True)

2.2 clusterName 没变，其他部分变动

不需要运行其他代码

my_downsample.downsample(tpm_downsampled = True)

2.3 其他

cluster 如果是用函数生成的话会多出两列：clusteringMethod 和 clusterName_scibet。

所以如果是后面找到原文提供的 cluster 信息填入之后，需要将原来这两列删除。

三、metadata

直接更改 unstructuredData.json 文件即可

有几个字段需要格外注意下:

1.关于人的数据一般都能找到 tissue，不可填为 notAvailable;

2.research topic 那几个字段，即 cancer，disease 等已经统一检查过了，一般情况下不需要

更改，不确定的请咨询；

3.genomeBuild 也是统一核查过的，一般没有错，注意不要改错了。

4.metadata 里面多出来的字段不要删掉！少的需要加上！

5.增加字段：metadata['correspondingFigure'] = ''

这里填写该数据对应的文章 tSNE 聚类图，格式为形如"1-a"，意思为 figure1 的图 a，没有对应的图填为 notAvailable

四、其他注意事项

4.1 每改完一个问题需要进行汇报；更改数据的代码需新建一个代码脚本进行更改，不要直接在别人的代码上改，相应代码的脚本需放在 code 文件夹中，命名为 revision，同时需要有详细的注释解释更改内容。

4.2 不清楚的函数或者代码内容可以参考代码目录里面的脚本

4.3 更改完成之后需通过检查器检查之后再进行报备

审核人员审核指南

审核人员拿到实习生标注过的数据集之后，需要在每个数据集 code 文件夹下建立新的 python3 文件，命名为 inspection.ipynb **不可以直接在别人的代码上改动**！负责人不代表工作内容只有一个人干，**如果一下子需要审核太多数据集，请及时求助其他组员，保证项目进行的的流畅性。**

一审

负责人：门语实

派发步骤:

1. 一审人员**（只可以在正式员工里面选择哦！）**在确定每周要发包的数据集有哪些之后，需要给每个数据集分 part，并保证每个数据集下面只有相应的 part 需要用的 GEO 数据信息，并写出一部分 unstructuredData 的内容在数据集文件夹下的 descriptio.txt 中。

2. 由实习生标注的时候把相应内容黏贴到数据集里，防止直接写到文简里被随意更改掉。

3. 记录到本地检查表格中，并把表格交给**二审负责人，**并把分好 part 的数据集，放入二审负责人用户下文件夹里

Part 分类方法(在把数据集下发给实习生之前就已分好):

分 part 只可以正式的审核人员来管理。

实际操作：和原作者一致即可，避免 part 划分人员的主观判断。

总原则：不同测序方法要分开，不同物种分开。例：part_1: human, part_2: mouse.

细化：不同细胞大类可以作为一个 part。例：part_1: T cell, part_2: B cell.

1） 最主要的还是根据文章中的聚类分析来进行分 part，不管分了几次聚类分析，只要 cluster 和 tSNE/UMAP 等的相关信息齐全，就可以算是一个 part。

2） 如果信息不那么齐全

若是文中进行了 2 次独立的聚类分析，那么应该将数据拆分成 2 个 part。

- 但是，如果 part2 的聚类分析是对 part1 聚类中的某一簇细胞进行的再聚类，只算一个 part。

- 如果多次聚类，都是对同一组细胞的，那么只保留一次。

例：如果一篇文章涉及 T cell 和 B cell，作者仅做了一次整体聚类，那么我们不需将 T cell 和 B cell 再聚类。如果作者后续单独对 T cell 的集合做了新的聚类，那么我们就新增细化的 T cell part。

3） 如果实在不知道该不该分 part，那就分就可以了。

unstructuredData 填写三项：

划分 part 后，在数据集文件夹(../GSE*****/)下新建一个 description.txt 文档,写入每个 part 对应的 description，subDataset，和 correspondingFigure。

保证 part_1, part_2, part_3 文件中仅含该 part 所需 GSE 数据信息，如果需要 数据标注人员从同一个 GSE 中挑选该 part 所用细胞时，一审人员需要在 description 中添加描述。

1) **subDataset：**如果有不只一个 part，则根据 part_n 填为 SubDataset-n。只有一个 part 就空着即可，不可以填。

2) **description：**填写对 part，即 subDataset 的描述。只有一个 part 就空着即可，不可以填。格式：description: The original article contains ? subdatasets, based on e.g. species, cell type(s). This subdataset is based on e.g. Mouse, cell A, B and C. (should match the initial description).

3) **correspondingFigure：**这里填写该数据对应的文章 tSNE 聚类图, 格式为形如"1-a", 意思为 figure1 的图 a，如果是附录里面的图，可以填写成“s1-a”的形式，没有对应的图填 notAvailable

二审：

负责人：张萌栩

二审 修改回收步骤：

负责人分配任务，

1） 二审（可以找优秀的实习生帮忙了），首先把数据集标注的任务在 redmine 上 asign 给空闲的实习生，把数据集 scp 到修改人的目录中(针对在 infinity 服务器上审核的人)或者 move 到修改人的目录下面（针对在阿里云服务器上修改的人）

2） 之后等实习生做完标注并上传 report 在 redmine 上，就需要在 report 和数据集的基础上进行审核，如果有错误，记录在 redmine 上并在 report 中使用修订模式指出，然后返回实习生修改。除 metadata 的错误可由审查人员直接在原文件中自行改动之外，其余错误均需按照以下步骤！

3） 当修改人在 redmine 上反馈修改完成之后，审查人去该目录下进行二次检查。除了原有问题之外，需重点看以下几点：

- 如果是矩阵需要改动，需检查其相应的自动生成函数是否都运行了（可根据文件更新时间进行判断），包括：tsne, cluster, auto_calculation, downsample 等；

- metadata 里面的 research topic，clusterAvailability 以及 tSNEAvailability 需进行二次核查

4） 如果没有错误，把 report 一起保存在数据集中，并回收数据集到：

针对在 infinity 服务器上修改的人：将数据集打包压缩(如果数据集较小可不打包压缩，但还是建议先打包压缩)，并 scp 回来，拷贝到路径：/home/biodb/data/dataset_collection/datasets/2_inspection_stage/revised/；针对在阿里云服务器上修改的人：将数据集 move 到路径：/home/biodb/data/revised/。也就是说除了 metadata 的错误，cellAnnotation 的错误和 矩阵的错误，在数据改完之后均需要放在以上这个路径中。

5） 确认数据集已拷贝至指定路径后，需要删除原来的修改人目录中的数据（不要留有冗余数据）。

6） **在本地表格记录审核情况，每周五交给数据集流程管理人员进行汇总。**错误记录只需简略提及要点即可，标注人、审核人（在二审表格中）一定要写清楚。

7） 最后由三审人员在指定文件夹下做最后审查。小错误自己改改就好的，可以在直接改完后交到指定文件夹下。**做好本地表格记录即可！**

8） 如果有优秀的实习生，并且其本人有意愿，可以让他们接触审核工作，帮忙分担工作压力，但是质量控制要把握好。

补充教程：

1. Unstructured Data

在 inspection 中参照 **实习生数据标注指南** 核验 unstructured data 与 cell annotation。如有错误，记录在 report 中，并 update 在 redmine 上面。

2. Cell Annotation 和 tSNE 检查

cellAnnotation

重点检查 meta 部分

注意检查 cluster 以及 cellOntology 部分，set（）出来查看一下

还要注意检查细胞数量是否与矩阵和文章中一致

使用代码，调用计算脚本画出 tSNE 的图

```
import seaborn as sns

import matplotlib.pyplot as plt  #下一行开始需要在两个不同的 block 里运行

clusterName = df_cell['clusterName'].tolist()

tsne1 = df_cell['tSNE1'].tolist()

tsne2 = df_cell['tSNE2'].tolist()

plt.figure(figsize=(10,10))
```

sns.scatterplot(x = tsne1, y = tsne2, hue = clusterName)

检查 tSNE 图质量，不同颜色的点不可过多重合，过多重合即为 isBadtSNE：False

使用代码画图查看某个基因在细胞中表达的情况

3. 矩阵检查

1） TPM 检查

读取 expressionMatrix_TPM.tsv 这个文件，检验各行基因的和相加是否为 100 000。

2） rawCounts 矩阵检查

读取 expressionMatrix_rawCounts.tsv 文件，看是否有把 normalized 矩阵和 rawCounts 矩阵弄混。还需要去源代码 script.ipynb 中检查是否有矩阵拼接上的错误。cellAnnotation 中的细胞数量（即，有多少行）应该和矩阵里的细胞数量都是一样的。

3） Normalized 矩阵检查

读取 expressionMatrix_normalized.tsv 文件， 看是否有把 normalized 矩阵和 rawCounts 矩阵弄混。

cellAnnotation 中的细胞数量（即有多少行）应该和矩阵里的细胞数量都是一样的。

4） 如何检查矩阵的正确性？

分为两个层面：

第一看作者提供的原始数据，（downloaded data）看看实习生在拼接过程中是否有出错。

第二，找一个文章里的 marker gene 在测试网站上画一下图，如果跟文章的图差不多，就问题不大了。

三审

负责人：陈淳、天然（暂定）

三审 修改回收步骤：

1）再进行一次二审的操作，不过可以大体浏览。发现错误后在 renmine 上记录，同时返回实习生修改。(请注意**记录二审的漏检率**，用作汇报时证明三道审核制度下的优势)

2）完全没有错误了之后，需要把 README.json 中的 notPassed 改为 passed，并保存。

3）在 redmine 网站上 close 相应的任务。

4）**在本地表格记录自己的审核情况（包括漏检率的记录），每周五交给数据集流程管理人员进行汇总。**标注人、审核人（在三审表格中）一定要写清楚。

5）最后，把更改好的数据集转移到 infinity 上。

审核的汇总表格流程

周期：每周更新一次数据集回收情况

流程

1. 每月生成一张新的 dataset_inspection.xlsx 表格记录整个月的数据集标注情况，

2.每周下发数据集时，由**一审负责人**更新 excel 表格中**数据集编号和 title** 信息。

3. 收包时由**二审负责人**更新完成的信息到 excel 表格中，包括三道审查中改出的错误用“×”记录，还要填写标注人员的信息，**每周五收集各个二审和三审人员的本地记录表格**，完成汇总后每周并上传至 Github，共享给组员。

4. 除此之外，管理员需每周回收数据集的时候查看各个实习生完成的数据集数量，若是超过两个周的时间中完成不超过两个数据集，需要沟通了解情况。

下发数据及之后长时间不做怎么办？

在每周回收数据集时，会发现是否有本周下发的数据集没能 close，这时需要审核没做完的数据集是因为什么原因。

1）数据集过于复杂（可以适当延长回收日期）

2）实习生没有做，并且没有及时通知（需询问是何原因，是否还有意愿继续做数据集？）如果没办法完成数据集，及时换人。