

笔记

1、Python Matplotlib

1.1、柱状图绘制

```
#!/usr/bin/python3

import numpy as np
import matplotlib.pyplot as plt

a = np.random.randint(0,20,10000)
b = np.arange(1,10001)
y = a[0::3]
y1 = a[1::3]
y2 = a[2::3]

x = b[0::3]
x1 = b[1::3]
x2 = b[2::3]

rate = (len(y) - list(y).count(0))/len(y)
rate1 = (len(y1) - list(y1).count(0))/len(y1)
rate2 = (len(y2) - list(y2).count(0))/len(y2)

# plt.bar(x, y, color='', label='', width=1]) width为1时·数据之间没有空隙
plt.bar(x, y, color='yellow', label="x rate:{0:.2f}".format(rate),width=1)
plt.bar(x1, y1, color="red",label="x1 rate:{0:.2f}".format(rate1), width=1)
plt.bar(x2, y2, color="green", label='x2 rate:{0:.2f}'.format(rate2), width=1)

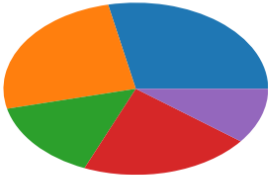

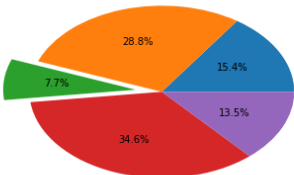
plt.legend() # 自动添加 bar 图例
plt.ylim(0,30) # 设置 y 轴 的 最大 高度

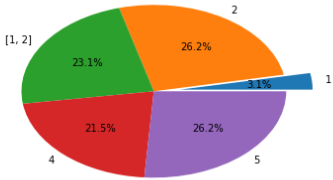
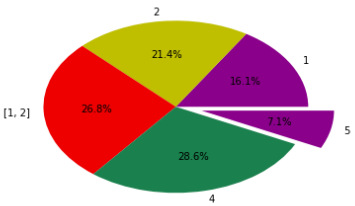
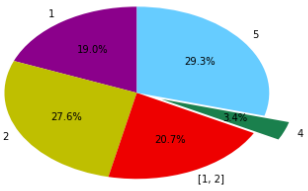
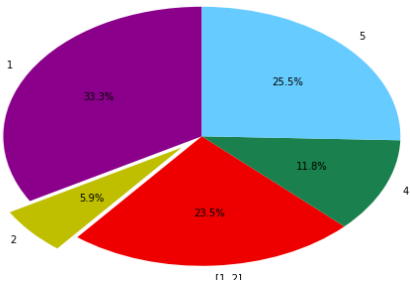
plt.ylabel("random")
plt.xlabel("x")
plt.title("random.randint")
plt.show()
```

1.2、饼图

```
#!/usr/bin/python3
```

```
def pie(  
    x, explode=None, labels=None, colors=None, autopct=None,  
    pctdistance=0.6, shadow=False, labeldistance=1.1,  
    startangle=None, radius=None, counterclock=True,  
    wedgeprops=None, textprops=None, center=(0, 0), frame=False,  
    rotatelabels=False, *, data=None):  
    return gca().pie(  
        x, explode=explode, labels=labels, colors=colors,  
        autopct=autopct, pctdistance=pctdistance, shadow=shadow,  
        labeldistance=labeldistance, startangle=startangle,  
        radius=radius, counterclock=counterclock,  
        wedgeprops=wedgeprops, textprops=textprops, center=center,  
        frame=frame, rotatelabels=rotatelabels, **({"data": data} if  
        data is not None else {}))
```

参数	意义	举例/结果
x	类似于一维数组	
explode=None	就是每个模块与中心的距离。如果指定，就要每个模块都指定	explode_[2] = 0.1 
autopct=None	显示百分比的格式	ausopct="%.1%%" 

参数	意义	举例/结果
labels=None	显示标签的内容，强行当作一维List_like	
colors=None	列表传参:支持16进制，单字母，英文代码，(0~1,0~1,0~1), 周末有时间写随机颜色脚本	colors=["DarkMagenta",'y','#ee0000', (0.1,0.5,0.3)] 
startangle=None	逆时针旋转角度，以第一个块的右边半径为坐标轴，旋转	startangle=90 
radius=None	radius默认为1	radius=1.5 
wedegrops=None	略	
textprops=None	略	
data=None		

1.3、折线图

```
#!/usr/bin/python3
import matplotlib.pyplot as plt
```

2、格式化输出

format

数字	格式	输出	描述
3.1415926	{:.2f}	3.14	保留小数点后两位
3.1415926	{:+.2f}	+3.14	带符号保留小数点后两位
-1	{:+.2f}	-1.00	带符号保留小数点后两位
2.71828	{:.0f}	3	不带小数
5	{:0>2d}	05	数字补零 (填充左边, 宽度为2)
5	{:x<4d}	5xxx	数字补x (填充右边, 宽度为4)
10	{:x<4d}	10xx	数字补x (填充右边, 宽度为4)
1000000	{:,}	1,000,000	以逗号分隔的数字格式
0.25	{:.2%}	25.00%	百分比格式
1000000000	{:.2e}	1.00e+09	指数记法
13	{:10d}		右对齐 (默认, 宽度为10)
13	{:<10d}		左对齐 (宽度为10)
13	{:^10d}		中间对齐 (宽度为10)

3、数据类型转化

函数	说明
int(x [,base])	将x转换为一个整数
long(x [,base])	将x转换为一个长整数
float(x)	将x转换到一个浮点数

函数	说明
complex(real [,imag])	创建一个复数
str(x)	将对象 x 转换为字符串
repr(x)	将对象 x 转换为表达式字符串
eval(str)	用来计算在字符串中的有效表达式,并返回一个对象
tuple(s)	将序列 s 转换为一个元组
list(s)	将序列 s 转换为一个列表
chr(x)	将一个整数转换为一个字符
unichr(x)	将一个整数转换为Unicode字符
ord(x)	将一个字符转换为它的整数值
hex(x)	将一个整数转换为一个十六进制字符串
oct(x)	将一个整数转换为一个八进制字符串

4、linux——awk命令

```
#!/usr/bin/bash
```

花括号两边必须是单引号

基操

```
awk '{pattern + action}' {filenames}
```

分隔符可以是列表

```
awk -F"分隔符" '{pattern + action}' {filenames}
```

```
awk -F"[\t',':']" '{pattern + action}' {filenames}
```

选择第20-30行

if的条件必须加括号

```
awk '{if(NR>=20 && NR<=30) print $1}' {filenames}
```

BEGIN

```
awk '{count++;print $0;} END{print "user count is ",count}' {filename}
```

END

```
awk 'BEGIN {count=0;print "[start] user count is ",count} {count=count+1;print $0} END{print "[e
```

awk '布尔表达式{action}' file 仅当对前面的布尔表达式求值为真时， awk 才执行代码块。

```
awk -F: '$1=="root"{print $0}' {filename}
```

```
awk -F: '($1=="root")&&($5=="root") {print $0}' {filename}
```

5、pandas

5.1.1 读取数据

```
#!/usr/bin/python3
```

```
import pandas as pd
```

header默认第一行

```
df = pd.read_csv("file_name" [,sep=" ", header=True])
```

5.1.2 保存

```
#!/usr/bin/python3
```

```
df.to_csv(output_file, sep=" ", header=True, index=True)
```

```
df.to_excel('*.xlsx', sheet_name='', index=False)
```

```
writer = pd.ExcelWriter("*.xlsx", mode='a')
```

```
df.to_excel(writer, sheet_name='test_2')
```

5.2 筛选

```
#!/usr/bin/python3

# 选取f中第0列包含list_like中的数据行
df[df[0].isin(list_like)]

# 选取f中第0列不包含list_like中数据的行
# ~ 表示反选
result = df[~df[0].isin(list_like)]

# 模糊筛选 （包含什么）

## 单条件
df.loc[df[0].str.contains("GendId")]
## 多条件 不能用 ' & ' 只能用 |
df.loc[df[0].str.contains("GenId|GenLenth")]

# 组合筛选时，每个条件都要添加括号
df2 = df[(df[3]>95) & (df[3]<96)]

# query 函数
df.query('列名'>4 & '列名' < 5)
```

行筛选

```
df[(df > 6).any(1)]
```

5.3 访问数据

```
#!/usr/bin/python3

# 因为人为规定的索引值可能会有重复，而索引位置不会重复

df.loc['索引']# 就是index，按照index的值来访问。它的类型根据index的类型改变
df.loc[行索引名称或条件，列索引名称]

df.iloc[int]# iloc只接受int，它表示第几行，从0开始
df.iloc[行索引位置，列索引位置]
```

5.4 更改数值

```
#!/usr/bin/python3

df.loc[0] = [..., ..., ..., ...]# 在原来的基础上改变值

df.loc[行值·列值] # 精准定位到某个数据进行更改
df.iloc['index_seq','column']# 同上

## 重命名索引值、列名

df.rename(index,columns,inplace=False) # 默认不替换原数据

df.rename(index={1:'first'},columns={'item':'object'},inplace=True)

## 寻找替换
## 然后需要再赋值给原来的表的列名才行
df["列名"].str.replace("", "") # 替换数据中的字符
df['列名'].replace("", "") # 更改数据

df.drop("", axis=0) # axis=0删除行·1·删除列

# 将df中大于10的值改为10
df[df>10] = 10

# 设置某一列为索引·必须是列名称
df.set_index('id_new')

# 将索引变为第一列
df.reset_index()
```

5.5 添加数据


```
#!/usr/bin/python3
```

```
df.insert(第几列, "新列名", [数据]) # 在第几列添加新的列·列名是什么·数据  
# 按列添加
```

```
col_name = df.columns.tolist()# 读取df列名为 list  
col_name.insert(1, 'D') # 就是对列表进行操作
```

```
df.reindex(columns=col_name)# 然后根据列名·重新排序df
```

```
# 利用 list.index的方法 ,  
col_name.insert(col_name.index('B'), 'D')# 在 B 列前面插入  
df.reindex(columns=col_name)
```

```
col_name.insert(col_name.index('B')+1, 'D') # 在 B 列后面插入  
df.reindex(columns=col_name)
```

```
# 按行添加数据  
temp=pd.DataFrame(like_list)  
df.append(temp, ignore_index=True)
```

```
# 在某一行添加数据暂时不考虑·因为如果在不同位置添加多个数据·名称和索引位置会改变,比较麻烦  
# 大多数思想都是拆表添加再合并
```

5.6 重复值处理

```
#!/usr/bin/python3
```

```
# 会对每个数据计数·重复返回True·否则返回False  
df.drop_duplicates(subset,keep)
```

```
# 根据subset中的列名进行重复值去除  
# 缺省subset—根据所有的列进行去重  
# 缺省keep—保留第一个值 False—不保留所有重复值, last—保留最后一个, first—保留第一个  
df.drop_duplicates(subset=[列名称1, 列名称2 ],keep)
```

5.7 排序

```
#!/usr/bin/python3
```

```
df.sort_values(by=[], axis=0, ascending=[], inplace=False, na_position='last')
```

```
# 根据低 1 · 2 · column_name3 排序 ·
```

```
# 分别是升序 · 降序 · 升序,
```

```
# 并且直接作用于原来的数据表
```

```
df.sort_values(by=[1,2,column_name3,...], ascending=[True,False,True], inplace=True)
```

```
df.sort_index(axis=1) # 对索引进行排序
```

```
df.sort_index(axis=0) # 对列名进行排序
```

5.8 计数

```
# 统计
```

```
df = df['colname'].value_counts()
```

```
df = df['colname'].value_counts(normalize=False) # 是否归一化 · 也就是百分比形式, 默认False
```

```
# 分组统计
```

```
df = df.groupby('user_id')['song_id'].value_counts().to_frame('count').reset_index()
```

5.9 长—宽列表转换

```
pd.melt(data,)
```

```
# 如果不指定values, 会有多层级所列名
```

```
df.pivot(columns='features', index='user_contig', values='feature_count') # 长 —> 宽
```

5.10 分割列

```
expand=True # 返回一DataFrame
```

```
expand=False # 返回列表
```

```
df['colname'].str.split('pattern', expand=True [,n=..]) # 将表格中某一列以pattern分割n次 · 默认有多次
```

```
df['colname'].str.rsplit(...) # 反向分割 · 参数同上
```

5.11 删除满足条件的列/行

```
df.ix[:, ~((df==1).all() | (df==0).all())]
```

```
df.ix[:, (df != df.ix[0]).any()]
```

5.12 数据透视表

```
a = pd.read_clipboard(header=0, sep="\t")
pd.pivot_table(a, index=['family'], columns=['HC>RA', 'p.signif'], values=['.y.'], aggfunc=[len], f
```

5.13 pandas显示问题

```
import pandas as pd
pd.set_option('expand_frame_repr', False) # 解决长度过长时，自动换行的问题
pd.set_option("max_colwidth", int) # 设置显示的值的长度
pd.set_option("max_column", int) # 设置显示的列数
pd.set_option("max_rows", int) # 设置显示的行数
```

5.14、pandas挑选列名相同的列

因为pandas如果读取文件时，设置header=0，出现相同的列名时，会自动在后面加\d
所以，为了挑选同名的列，可以设置header=None，然后对第0行进行筛选即可

```
import pandas as pd
df = pd.read_csv("./test_data/pandas_5.14.txt", header=0, sep="\t")
print(df)
```

5.15、填充空值

```
import numpy as np
import pandas as pd
df = pd.DataFrame({"name":["apple", "pear", "pig", "dog", "cat"], "number_1":[1,np.nan,3,np.nan,
```

5.16、整张表处理

```
# 每个值站每一列和的百分比
df = df.div(df.sum(axis=0), axis=1)
```

5.17、pandas 天坑

计算每列的和 **axis=0**

```
import pandas as pd
df = pd.read_csv("./test_data/pandas_5.17.csv", header=0, sep="\t")
print(df.sum(axis=0))
```

	variable	GA	GC	GA-GC	ABS(GA-GC)
0	Bacteroidia	48.608199	63.083423	-14.475224	14.475224
1	Actinobacteria	6.452157	0.806468	5.645689	5.645689
2	Gammaproteobacteria	6.249311	2.652160	3.597150	3.597150
3	Bacilli	1.833909	0.143504	1.690405	1.690405
4	Negativicutes	2.897499	1.373483	1.524016	1.524016
5	Clostridia	27.745406	29.187009	-1.441604	1.441604

删除行 **axis=0**

```
df.drop(0, axis=0)
```

	variable	GA	GC	GA-GC	ABS(GA-GC)
1	Actinobacteria	6.452157	0.806468	5.645689	5.645689
2	Gammaproteobacteria	6.249311	2.652160	3.597150	3.597150
3	Bacilli	1.833909	0.143504	1.690405	1.690405
4	Negativicutes	2.897499	1.373483	1.524016	1.524016
5	Clostridia	27.745406	29.187009	-1.441604	1.441604

6、匿名函数

```
#!/usr/bin/python3
```

```
# lambda 表达式
```

```
正确:      lambda x,y:x+y
```

```
错误表达:  lambda x,y:a=x+y
```

```
# 对可迭代对象中的每一个元素进行function操作
```

```
map( function, iterable )
```

```
# reduce 连续映射
```

```
from functools import reduce
```

```
reduce(lambda x,x+y, list_x [,初始值/list_like])
```

```
# 过滤函数,保留条件为True的元素
```

```
filter ( lambda x: True if x==1 else False, list_like )
```

```
# 结合后的例子
```

```
map( lambda x: 满足条件返回值 if 条件 else 不满足条件返回值 ,iterable )
```

```
# 可同时迭代两个对象
```

```
map( lambda x,y: True_value if condition else False_value, iterable_1,iterable_2 )
```

7、biopython

7.1.1、访问NCBI Entrez数据路

7.1.2、Entrez获取数据库信息

7.1.3、ESearch：搜索Entrez数据库

7.1.4、Epost：上传identifiers列表

7.1.5、ESummary：通过主要的IDs获取摘要

7.1.6、EFetch：从Entrez下载更多记录

7.1.7、FLink：在NCBI Entrez中搜索相关的条目

7.1.8、EGQuery：全局搜索-统计搜索的条目

7.1.9、ESpell：获得拼写建议

7.1.10、解析大的Entrez XML文件

7.1.11、错误处理

7.1.12、专用解析器

7.1.13.1、解析Medline记录

7.1.13.2、解析GEO记录

7.1.13.3、解析UniGENe记录

7.1.14、使用代理

7.1.15、实例

8、生物信息中常见的数据记录格式

8.1、.fastq

每条reads只占用4行

第四行是碱基质量

sequence identity

```
@<instrument>:<run number>:<flowcell id>:<lane>:<title>:<x-pos>:<y-pos>:<read>:<is filtered>:<control number>:<ir
```

格式如下：

[illegible]

8.2、.fasta

每条序列占两行

下一行 是碱基序列

格式：

省略

8.3 gff

跑基因注释

```
##gff-version 3
ctg123 . exon 1300 1500 . + . id=exon00001
ctg123 . exon 1050 1500 . + . id=exon00002
ctg123 . exon 3000 3902 . + . id=exon00003
ctg123 . exon 5000 5500 . + . ID=exon00004
ctg123 . exon 7000 9000 . + . ID=exon00005
```

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

feature type

start coordinate
end coordinate
score
strand
frame
attributes

名称	说明
reference sequence	参照序列
annotation source	注释来源
feature type	特征类型
start coordinate	起点
end coordinate	终点

|得分| 针对一些量化的属性来表示程度得分

strand |链|+表示正链，-表示负链，
frame |.表示不指定步长|通常是编码蛋白质制定下一密码子开始位置。
attributes |属性|一个包含众多属性的列表。格式为(tag=value),不同属性之间以分号相隔

9. Shell

```
# 递归复制多个目录的相同文件
cp -r /share/data7/zhangy2/projects/2019-07-19/ 3 .anno/**/*.sort.depth.log ./
nohup ./program > program.log 2>&1 & # nohup重定向
```

10. conda

更改 .condarc 可以设置conda的下载源
windows会在用户加目录下，有.condarc文件，更改为一下内容即可

```
channels:
  - defaults
show_channel_urls: true
default_channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/main
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/r
custom_channels:
  conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  msys2: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  bioconda: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  menpo: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
```

命令行更改

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/main
conda config --set show_channel_urls yes
# 恢复默认下载源
conda config --remove-key channels
```

删除环境： `conda remove -n name --all` # 删除环境中所有的包

重命名： `conda create -n tf --clone rcnn`

相当于clone，删除原环境即可

11.Git


```
# 在自己的服务器上，创建一个空仓库
git init --bare sample.git
# 克隆非22端口服务器仓库
git clone ssh://jiangminghao@git.data_analyse.cn:22220/home/jiangminghao/sourcecode/.git
# 查看历史提交记录，短格式
git reflog
# 推送
git push -u origin master
# 拉取远程仓库
git pull origin master

# 版本控制
git revert <commit-hash> # 往后加一个版本，状态时所选版本

git diff <hash1> <hash2> # 比较两个版本有什么差异

# 撤销暂存区中，未提交的件
git checkout -- <file-name>

# 添加新的远程地址
git remote add origin ssh://serves@172.16.0.30/~yafeng/.git
```

中文显示问题

```
git status # 不能显示中文时，先执行
git config --global core.quotePath false

# 如果git reflog不能显示中文，
git config --global core.pager more
```

进入你的项目根目录

1.设置git gui的界面编码

```
git config --global gui.encoding utf-8
```

2.设置 commit log 提交时使用 utf-8 编码，可避免服务器上乱码，同时与linux上的提交保持一致！

```
git config --global i18n.commitencoding utf-8

git config --global i18n.logoutputencoding utf-8
```

注：

windows系统默认编码为gbk，可改成gbk

如果系统设置了：

```
export LANG=zh_CN.UTF-8
```

则日志输出编码设置为utf-8

```
git config --global i18n.logoutputencoding utf-8
```

3.在 /etc/profile 中添加：

```
export LESSCHARSET=utf-8
```

12. numpy

```
import numpy as np
a = np.array([1,2,3,4,5,6])
# 分割一维列表
print(np.array_split(a, 3)) # 将data分割成几分
```

13、perl

```
# cpan 换源 /.cpan/CPAN/MyConfig.pm
修改为： 'urllist' => [q[http://mirrors.aliyuncs.com/CPAN/]],
```

14、scp

配置后传输还是出问题，是因为安全问题，将安全级别降到最低即可

参考链接[点击这里](#)

```
# 下载
scp -P <prot> -o stricthostkeychecking=no root@192.168.1.1:/path/ ./
```

15、VIM

小技巧

```
:set nu
:set ff "显示文件编码的方式dos/unix
:set ff=unix "change the code to unix
:r!command "插入命令行的输出内容
:read <filename> " 在下面插入filename的内容
:set cursorline " 高亮当前行
```

16、qsub提交节点任务

```
qsub -cwd -l vf=20g,p=8 -q all.q -V # 提交任务
qhost -j # 查看节点状态
qhold <job id> # 挂起任务
qrls <jobid> # 释放任务
```

17、awk

awk打印单引号，需要在双引号中使用一对单引号，转义的单引号

awk使用system () 可以直接执行shell命令

```
find ./*/storage/bin_stats.analyze.tsv | awk -F "/" '{cmd="sed '\''s/^/"$2"\t/"'\'' " $0;system(cr
```

18、Windows技巧收集

文件名输入空格 Alt+0160

19、python基础包技巧

1、反转字典键值对

```
a = {1:'1', 2:'2'}
print(dict(zip(a.values(), a.keys())))
```

2、创建自己的python包

导入的包名为zy2，下面是具体的方法, 在setup.py中添加一下代码

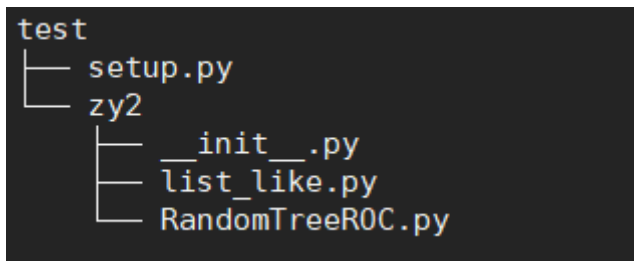
```
from distutils.core import setup

setup(name='zy2',
      version='1.0',
      author='YeXiaNingYue',
      py_modules=['zy2.list_like', 'zy2.RandomTreeROC']# 包下面的名称)
```

然后在setup.py所在目录中运行

```
python setup.py build
python setup.py sdist
```

在dist目录中，会有一个打包的压缩包，pip install <package> 即可安装



20、vscode技巧

1、运行代码块 code-chunk

使用插件Markdown Preview Enhanced 可以运行python、shell、R...代码

举例：

```
```python {cmd=true id='1'}
a = {1:'1'}
```
```

python {cmd=true continue="1" id="2"} # 使用id为1的代码块中的变量

python {cmd=true matplotlib=true} # 设置显示matplotlib代码块

参考：

[1]:<https://www.jianshu.com/p/7313d9840edc>

[2]:官方文档：<https://shd101wyy.github.io/markdown-preview-enhanced/>