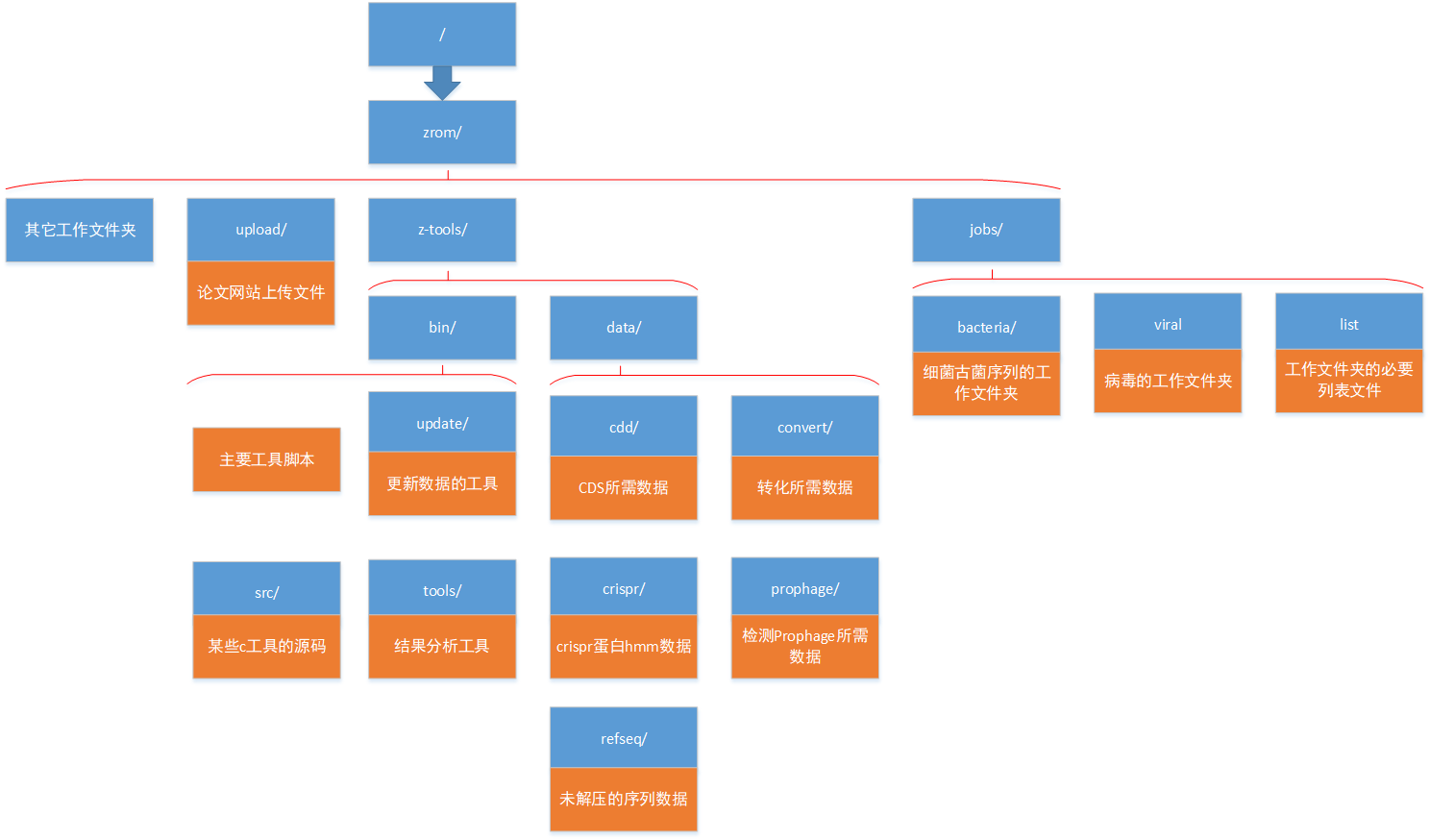
生信工具集Z-TOOLS使用文档

（周海彬工作部分）

作者：周海彬

时间：2017/8/31



一: 细菌、古菌 与病毒的 Interaction

工作思路:

1. spacer文件与phage的基因组fasta文件做blastn,参数evalue 0.01

通过公式missmatch < 筛选(长度小于22的默认在筛选列表中)

1. 得到匹配的spacer序列,(通过bacterial\_id,phage\_id,spacer\_id来标识一条匹配序列),通过blastn的结果中start,end标识,去拿到phage中与之匹配的hit序列,并加上前后5bp得到protospacer
2. 处理phage的genbank文件,得到标记为CDS的该phage表达的蛋白质,检查匹配的hitsequence是否在某段蛋白质表达基因里面,如果有,标注该蛋白质的id,position,product,protein\_sequence.

工作依赖:

1. 依赖于世家的工作中找到spacer文件并存入/zrom/jobs/bacterial 相应的文件夹中
2. 依赖于一个需要匹配的细菌或者古菌列表,放入list

该list格式为:

NZ\_AEMG01000024 1 5 n

NZ\_AMPO01000010 1 20 n

NZ\_AMPO01000012 1 3 n

\t分隔,只需要第一个ID信息

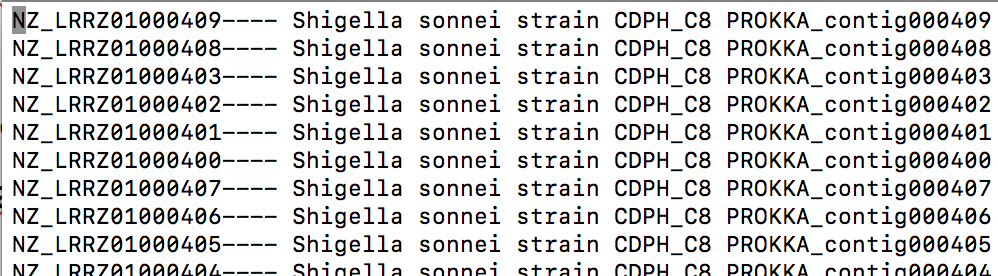
即新增LIST的时候可以在文本这样写入

id\t\n

id\t\n

1. 依赖一个phagedb(用makeblastdb命令将phage 的基因组fasta 文件建phagedb库)加速blastn速度.
2. 依赖于一个bacteria与archaea的 id-name的对应关系list, 本工具集 中存放的文件为:/zrom/jobs/list/bacterialId 文件

bacterialId 格式:



id--- name的形式为一行

(不用\t是怕有些名字中带空格,\t分割的时候错分)

调用脚本:

1. interblast.py脚本:

功能:将spacer文件与phage的基因组序列做blastn

此脚本生成的相应文件为如NC\_000853.asn这样的asn文件代表了该ID的spacer与phage 有匹配的情况

(2)interactionPaser.py脚本:

功能: 将(1)生成的asn文件解析,并筛选得到匹配的spacer序列,(通过bacterial\_id,phage\_id,spacer\_id来标识一条匹配序列),通过blastn的结果中start,end标识,去拿到phage中与之匹配的hit序列,并加上前后5bp得到protospacer,处理phage的genbank文件,得到标记为CDS的该phage表达的蛋白质,检查匹配的hitsequence是否在某段蛋白质表达基因里面,如果有,标注该蛋白质的id,position,product,protein\_sequence.

此脚本生成三个文件:

生成文件有三个:  
hitproteins\*.csv interation\*.csv sequences\*.csv \*代表了脚本运行的时间  
  
interaction表含义: (`bacteria\_id`, `bacteria\_name`, `specie`, \  
`genus`, `phage\_id`, `phage\_name`, `spacer\_id`, `missmatch`, `phage\_position`, `flag` )  
phage\_position 是匹配部分在phage基因组中的位置  
flag 为1/0 1代表其匹配的基因段在phage中参与编码区域,0 代表不参与编码区域  
  
hitproteins表信息:`bac\_id`, `phage\_id`, `spacer\_id`, `protein\_id`, `protein\_position`, `protein\_product`, `protein\_sequence  
其中bac-id,phage-id,spacer-id 三者共同代表了一个Hit 序列,这个hit序列在phage中参与编码,其蛋白质信息protein\_id`, `protein\_position`, `protein\_product`, `protein\_sequence

Sequences表信息:`bac\_id`, `phage\_id`, `spacer\_id`, `spacer\_sequence`, `hit\_sequence`, `long\_hit\_sequence`, `coverage` ,其中hit\_sequence为phage中与spacer\_sequence匹配的部分,long\_hit\_sequence为’protospacer’,即hit\_sequence加上上下游5bp形成序列

(3)converTosql.py 脚本:

按理说,形成excel表格已经够了,但是为了方便数据能加载到我们的web应用中,应该转换为sql 语句的形式,此脚本为功能为该脚本是将得到的三个表格变成sql插入语句的形式,方便更新数据库

用法与参数:

-i 输入的csv表格名字  
 -s 保存的文件  
 -t 表格的类型,可选sequences,hitProtein,interaction

更新数据时需要的操作与代码改动:

1. 更新bacterial 或者archea

依赖改动

(1)(arche类似)更新bacterial 依赖于以更新的bacterial 的spacer文件已经存放于/zrom/jobs/bacterial 相应的文件夹中

(2)依赖于bacteria与archaea的 id-name的对应关系list 已经更新(追加到/zrom/jobs/list/bacterialId后面).或者新做一份文件,并将interactionParser.py中main函数的idToNameDic变量改为文件的路径与名字

(3) 依赖于bacterial 与aechaea 的 id-specis-genus对应文件/zrom/jobs/list/idToSpecisGenus.txt, 或者新做一份文件,并将interactionParser.py中main函数的idToGenus\_specisDic = get\_genus\_specis('/zrom/jobs/list/idToSpecisGenus.txt'),传入参数改为新做的文化的路径加文件名.

代码改动:

1. 1) 修改interactions.py代码中69行 getqfastalist方法如:qfastalist=getqfastalist('/home/admin01/spc/spclist') 括号内修改为需要更改的bacterial 或者archea 的ID的一个list,格式参照/home/admin01/spc/spclist文件

(2) interactionParser.py中 asnDir = '/zrom/tmp/archaeadbinteraction/' 改为(1)中asn保存文件夹. save\_path = '/zrom/tmp/sql/'改为你想要的csv保存路径

(3)在保存的路径里面

hitproteins\*.csv interation\*.csv sequences\*.csv \*代表了脚本运行的时间.

即为更新的内容

注意,以上改动代码的部分均在\_\_main\_\_的函数中

1. 更新phage

将新加的phage与原来phage的基因组fasta文件用makeblastdb做一个blast的对比库,假设输出对比库路径为:/home/username/phagedb/

改动其代码第28行中NcbiblastnCommandline方法的db 参数为/home/username/phagedb

如原代码为:

blast\_cline=NcbiblastnCommandline(query=BASE\_DIR+**'jobs/bacteria/%s/%s/'** % (gethash(refseqaccession),refseqaccession)+refseqaccession+**".spc"**,  
 db=**'/zrom/jobs/phagefna/phagefna'**,  
 word\_size=20,out=savepath+refseqaccession+**".asn"**,  
 outfmt=**"6"**,evalue=1e-2,  
 )  
更新后:

blast\_cline=NcbiblastnCommandline(query=BASE\_DIR+**'jobs/bacteria/%s/%s/'** % (gethash(refseqaccession),refseqaccession)+refseqaccession+**".spc"**,  
 db=**’**/home/username/phagedb**’**,  
 word\_size=20,out=savepath+refseqaccession+**".asn"**,  
 outfmt=**"6"**,evalue=1e-2,  
 )

然后改变保存路径,运行Interaction.py脚本,

然后 将修改的保存路径修改到interactionParser.py脚本中的asnDir 变量 ,

运行 interactionParser.py脚本即可得到新的数据,数据更新完毕

Example:

更新archaea的过程为例:

1. 准备工作:

archaea的spacer文件已经准备在/zrom/jobs/bacterial目录下

有一个需要更新的archaea id 的文件/zrom/jobs/list/cripsrarchaea

将Interaction.py中的 qfastalist=getqfastalist('/home/admin01/spc/spclist')代码改为:

qfastalist=getqfastalist('/zrom/jobs/list/cripsrarchaea

')

运行的命令为:

Interaction.py && interactionParser.py

等代码运行完时候,在/zrom/tmp/sql/目录下 即可看到三个表格文件:

hitproteins\*.csv interation\*.csv sequences\*.csv \*代表了脚本运行的时间

二.上传一段未知spacer文件的interaction

(前面的是基于整个库的Interaction,这个是单个文件)

调用脚本:

getInteractions.py -s \*.spc

\*.spc即上传的未知ID的spacer文件

结果文件:

/zrom/tmp/tmpasn/\*.res文件 ,\*代表了网站上传spacer文件的时候保留的一个标识符

文件解释:

若该spacer与phage dna存在intertactions,则文件中会有以下的N行(几条匹配就几行)

subject,spacerId,identity, coverage,length,hit\_position, hit-sequence

example:  
 比如我有一个spacer文件叫 NC\_000853.spc

则我调用的代码是:

getInteractions.py -s NC\_000853.spc

更新:

此工具为单个spacer与phagedb的interaction,所以只能更新phage的时候改变.

将新加的phage与原来phage的基因组fasta文件用makeblastdb做一个blast的对比库,假设输出对比库路径为:/home/username/phagedb/

改变getInteractions.py 代码13行中的 self.db='/zrom/jobs/phagefna/phagefna'为:

self.db='/home/username/phagedb/phagedb'

三:一个蛋白质fasta文件是否与已知anti-crispr蛋白质匹配:

思路: 做blastp(参数qcov\_hsp\_perc=80,evalue=0.01),然后筛选条件:

evalue<0.01 and identy>0.3 and qcovs>0.8

脚本: getanti.py脚本

输出: /zrom/tmp/tmpanti/\*.res文件

若有匹配,则会有如下的一行:

proteinID hitanti-crisprid evalue identy coverage proteinsequence

若无匹配,则文件中会有None字样

example:

getanti.py –f NC\_000853.faa

(假设NC\_000853.faa为储存了NC\_000853基因组的蛋白质信息的fasta文件)

则在/zrom/tmp/tmpanti/NC\_000853.res 即为结果文件

更新:

拓展已知的anti-crispr 与原来的anticrispr 序列信息,做成一个fasta文件:

例如下面的fasta格式:

>ID+名字

序列

>ID+名字

序列

保存该fasta文件,并用makeblastdb命令,将该fasta文件做成blast对比库,假设路径及命名为/home/username/anticrisprdb/

更改: getanti.py代码中的第19行中的self.dbpath='/zrom/jobs/anticrisprprotein/anticrisprprotein'

更改为

self.dbpath='/home/username/anticrisprdb/ anticrisprdb'

四:已知anticrispr蛋白质的同源蛋白质查找

思路:

(1)将已知的anticrispr蛋白质的序列与nr数据库做blastp,参数evalue 0.01

得到blastp的结果(xml格式),解析筛选条件(evalue<0.01,identy>0.3, coverage>80%),得到相应的蛋白质信息:

1. 将(1)中得到的蛋白质信息,回溯,找其所在的基因组的蛋白质序列信息,进行cd-search找蛋白质结构域,
2. 将与anticrispr蛋白质的同源蛋白质的上下游5个蛋白质的结构域信息显示

调用代码:

1. blastp 命令, 将anticrispr的蛋白质序列与nr 做blastp 输出格式xml,得到anticrispr.xml文件
2. 调用anticrispr.py脚本,解析anticrispr.xml并筛选蛋白质信息,得到antiHomo.txt文件
3. 调用rpsblast.py脚本,将(2)中的基因组的蛋白质做cd-search,找到同源蛋白质前后5个蛋白质的结构域信息
4. 调用inserthomo.py脚本,以antiHomo.txt文件为输入,输出相应已经筛选的protein的sql脚本
5. 调用Insertproteins.py脚本,将(3)中rps-blast(对蛋白质做cd-search看结构域)输出的xml结果解析,得到两个sql表

proteins(与anti-crispr蛋白质同源的蛋白质)表插入语句如下,表示其列分别为:

蛋白质ID,蛋白质所在基因组ID,蛋白质的product,蛋白质所在的位置,以及蛋白质序列

"INSERT INTO proteins(proteinId, geneid, product, geneposition, query)VALUES('%s', '%s', '%s', '%s', '%s');"

proteindomain表的插入语句为:

其列分别表示:

蛋白质ID,基因组ID,蛋白质结构域,CDD号,cdd描述,evalue,匹配率,

INSERT INTO proteindomain(proteinId, geneid , proteindomain, cdd, cdddescription, evalue, identy)VALUES('%s','%s' ,'%s', '%s', '%s', '%s', '%s');"

Proteindomain表中的蛋白质Id 一般连着,然后其中一个id在proteins表中出现,表示该id的蛋白质与anticrispr蛋白质同源, 其ID末尾+-5 就为上下游5个蛋白质的信息

更新工作:

1. 更新anticrispr蛋白质:

将更新的anticrispr蛋白质序列重新做一个fasta文件,与nr库做blastp得到新的anticrispr.xml文件,重新调用步骤(2)-(5)即可

1. 更新nr库

将更新后的nr库与anticrispr蛋白质序列做blastp得到新的anticrispr.xml文件,重新调用步骤(2)-(5)即可