|  |
| --- |
|  |
| 生物信息技能训练 |
| 实验记录 |
| **院部 医学部**  **专业 生物信息学**  **学号 1730416017**  **姓名 张涵养**  **题目 Chip-seq数据分析之复现文章**  **组长 王颖娟**  **组员 王颖娟、罗晓琦、韩淑清、谢陈瑶、张涵养**  **2019/9/8** |
|  |

目录

[step-0、实验平台 3](#_Toc51254726)

[step-1、从 Genome下载 S288C 全基因组序列以及注释⽂档 3](#_Toc51254727)

[step-2、从 UniprotKB，下载物种 S288C 所在分类的所有已知蛋白质 3](#_Toc51254728)

[step-3、选择、安装、测试并运行基因预测软件 3](#_Toc51254729)

[step-4、从NCBI⽹站下载、安装并测试 blast 系列软件 4](#_Toc51254730)

[step-5、使用makeblastdb⼯具，把S288C的基因组序列建立成本地blast 数据库 4](#_Toc51254731)

[step-6、使用 blast 软件，将第2步下载的已知蛋白与基因组序列进⾏对比 4](#_Toc51254732)

[step-7、使用 blast92gff3.pl 程序，把 blast 比对结果转化成 GFF3 格式 4](#_Toc51254733)

[step-8、编写 python 脚本，根据该物种的注释⽂档训练基因的区间长度 5](#_Toc51254734)

[step-9、编写脚本，从基因组序列文件中提取序列 8](#_Toc51254735)

[step-10、寻找并测试可对短片段序列进⾏预测和建模的软件 8](#_Toc51254736)

[step-11、编程分析第 10 步的结果，评估基因结构的完整度 10](#_Toc51254737)

[step-12、对 9、10 和 11 步功能进⾏封装、迭代和预测 11](#_Toc51254738)

[1、封装模块 11](#_Toc51254739)

[2、进行迭代预测 14](#_Toc51254740)

[3、迭代预测的后续处理 15](#_Toc51254741)

[4、迭代预测后续处理之基因内部原件回贴 16](#_Toc51254742)

[附录 18](#_Toc51254743)

[1、FilterGFF3 18](#_Toc51254744)

[2、IterativePrediction 20](#_Toc51254745)

[3、dedup 21](#_Toc51254746)

[4、reattachment 23](#_Toc51254747)

## step-0、实验平台

PC：

System：window 10 专业版

Processor：Intel（R） core（TM）i5-7300HQ CPU @2.50GHz 2.5GHz

Memory: 4+4GB

Disk: 512GB

机房PC：

代号：HP31

System：Ubuntu 16.04 LTS

Processor：Intel® Pentium(R) CPU G4400T @2.90GHz\*2

Memory: 7.7GB

Disk: 483.8GB

语言：

R、python、perl

## step-1、从 Genome下载 S288C 全基因组序列以及注释⽂档

由『谢陈瑶』同学完成

## step-2、从 UniprotKB，下载物种 S288C 所在分类的所有已知蛋白质

排除该物种自身的已知蛋白质

由『谢陈瑶』同学完成

## step-3、选择、安装、测试并运行基因预测软件

比如AUGUSTUS、GeneMark\_ES和 geneid并使用这些预测软件，对物种 S288C 的全基因组序列进⾏基因预测和结构建模，保存结果⽂档，并将其统⼀转换为 GFF3 格式

由『AUGUSTUS：王颖娟、GeneMark\_ES：韩淑清、geneid：韩淑清』同学完成

## step-4、从NCBI⽹站下载、安装并测试 blast 系列软件

由『谢陈瑶』同学完成

2020/9/17

在由『王颖娟』同学进行gffcompare比对后，如下图所示，我们发现同源搜索指导后的从头预测的内含子预测水平为0，比同源搜索和从头预测都要低，这很不符合常理

检阅整理后的迭代预测文件，如下图所示，我们发现基因内部元件的起始位点与终止位点全部相同

这是因为Augustus预测基因结构时，按照输入序列从1计算位置，所以我需要按照blast结果该序列原始位置、已经延伸的长度以及Augustus预测出的起始位点计算该基因在基因组中的位置。但是，我在编写筛选完整基因的代码时，直接将一个基因内部所有原件的起始位点与终止位点统一使用正则替换，造成了无法区分内部元件的错误

编写python脚本augustus\_reattchment.py，遍历所有augustus迭代预测出的文件，重新分别针对每个基因元件计算其起始位点和终止位点，代码见附录4、reattachment，并去冗余

去冗余后得到3291条结果

## step-5、使用makeblastdb⼯具，把S288C的基因组序列建立成本地blast 数据库

由『谢陈瑶』同学完成

## step-6、使用 blast 软件，将第2步下载的已知蛋白与基因组序列进⾏对比

由『谢陈瑶』同学完成

## step-7、使用 blast92gff3.pl 程序，把 blast 比对结果转化成 GFF3 格式

将其中 LOWSCORE\_SKIP 阈值修改为 0

由『罗晓琦』同学完成

2020/9/14

经张高川老师提醒后，我们发现blast比对结果存在大量冗余信息，编写python脚本，将同染色体、同条链上重叠的基因取score值最大者保留，代码见附录3、dedup，先将原gff3文件按照正负链、染色体以及起始位点进行排序，然后执行py文件

去除冗余后，原24672个结果剩余3833个结果

我们发现这个数据与原基因组注释文档中显示的基因数目（6399）存在很大差异，猜测是原基因组中基因本身存在重叠现象，编写python脚本，计数存在重叠的基因（同样先排序），代码如下：

class countoverlap():

"""docstring for countoverlap"""

def \_\_init\_\_(self, gff):

self.num = 0

self.gff = gff

def readwrite(self):

old\_chr = ""

overlap\_set = []

old\_interval = range(1)

with open(self.gff, "r") as f:

for line in f.readlines():

gene = line.split("\t")

new\_chr = gene[0]

if new\_chr == old\_chr:

new\_interval = range(int(gene[3]), int(gene[4]))

if len(set(old\_interval).intersection(set(new\_interval))) > 0:

self.num += 1

overlap\_set.append(gene)

elif len(overlap\_set) > 0:

old\_interval = new\_interval

else:

old\_chr = new\_chr

d = countoverlap("gff3\_onlygene\_sorted\_chain.txt")

d.readwrite()

d.num # 0

经测试后，发现原基因组不存在基因重叠现象。

疑问：为什么blast同源预测的基因结果与原基因组存在如此数量的差异？

## step-8、编写 python 脚本，根据该物种的注释⽂档训练基因的区间长度

2020/9/8

编写python脚本，从该物种的gff注释文档统计每个基因的长度

import re

chr = ''

genes = []

with open('S288C.gff', 'r') as f:

for line in f.readlines():

if "region\t" in line:

chr = re.findall(r"chromosome=(.+?);", line)[0]

if "gene\t" in line:

new\_gene = [chr,line.split("\t")[3],line.split("\t")[4]]

new\_gene.append(str(int(new\_gene[2]) - int(new\_gene[1]) + 1))

genes.append(new\_gene)

with open('gene\_length.txt', 'w') as f:

for gene in genes:

f.write("\t".join(gene) + "\n")

编写R语言脚本，分析基因长度，并尝试由此决定迭代预测时每次扩增的长度。

# setwd("D:/Script/final/lab\_1/")

genes <- read.table("gene\_length.txt", header = T)

len <- genes$len

shapiro.test(len[1:5000])

# W = 0.8256, p-value < 2.2e-16

min\_l <- min(len, na.rm = T) # 51

max\_l <- max(len, na.rm = T) # 14733

ave\_l <- mean(len, na.rm = T) # 1399

sd\_l <- sd(len, na.rm=TRUE) # 1160

ds\_l <- density(len, na.rm=TRUE) #density

med\_l <- median(len, na.rm = T) # 1134

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

mod\_l <- getmode(len) # 72

hist(len, freq = FALSE, breaks = 500, xlim = c(0,10000),

main="Density Histogram of Length", xlab="length", col = "gray") #绘制频率分布直方图

lines(ds\_l, col="black", lty=3, lwd=3)#从样本估计概率密度

abline(v = mod\_l, col = "red", lty = 3, lwd = 2)

abline(v = med\_l, col = "blue", lty = 3, lwd = 2)

text(x = mod\_l, y = 0, labels = as.character(mod\_l), col = "red")

text(x = med\_l, y = 0, labels = as.character(med\_l), col = "blue")

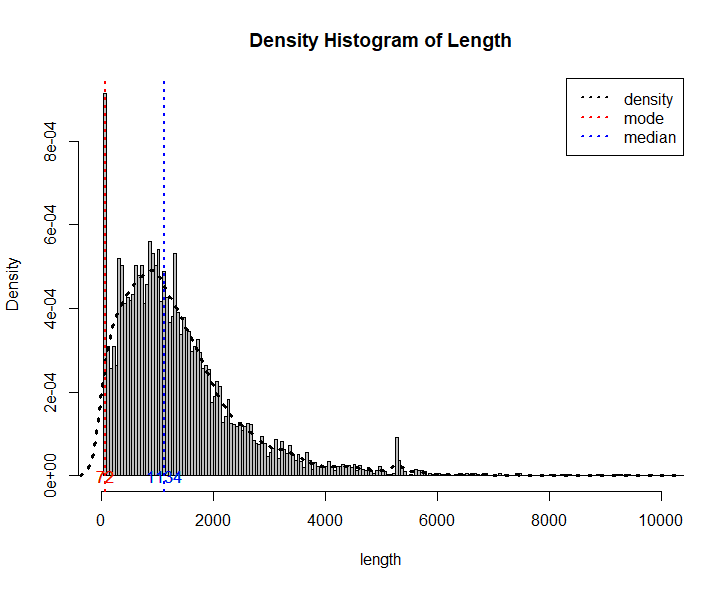
legend("topright", c("density","mode","median"),

lty=c(3,3,3), lwd=c(2,2,2),

col=c("black", "red", "blue"))

boxplot(data = genes, len~chr)

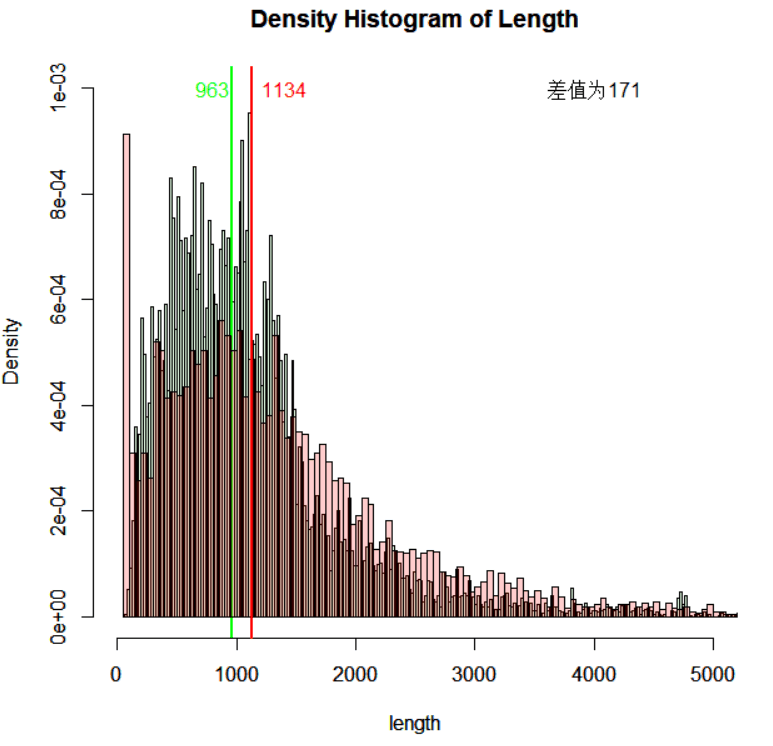
由于p < 0.05，有理由拒绝该物种的基因长度分布并非呈现正态分布，所以放弃使用正态分布中的95%置信区间限制迭代中的扩增长度。



另一方面，我们发现基因长度分布落在几十~1W5Kbp，据此数据，我们计划在迭代过程中，每次上下各扩增100bp，并设置最大迭代次数为80次（>75，防止意外）

2020/9/12

经张高川老师指导后，我们将原基因组注释文档的基因长度分布频率直方图与blast比对结果的叠加在一起，以二者分布的差异来决定迭代过程中的延伸长度，如下图所示：



该图由『王颖娟』同学完成

由上图可知二者的中位数差异为171bp，所以我们设定迭代过程中，起始延伸长度为上下游各100bp，以及后续延伸长度为上下游各100bp。

## step-9、编写脚本，从基因组序列文件中提取序列

由『罗晓琦』同学完成

## step-10、寻找并测试可对短片段序列进⾏预测和建模的软件

由『王颖娟』和『韩淑清』同学完成

经过两位同学测试，Augustus和GeneID可对短片段序列进行基因预测和建模

经张高川老师指导后，我们发现GeneID的预测结果存在很大问题，全基因组预测的基因数量过少，所以舍弃此工具。

2020/9/9

尝试通过python使用Genscan

Python 执行

from selenium import webdriver

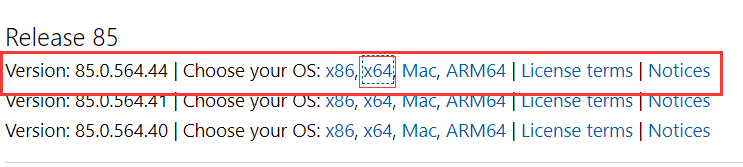
from selenium.webdriver.common.keys import Keys

dirver = webdriver.Edge()

报错：

selenium.common.exceptions.WebDriverException: Message: 'MicrosoftWebDriver.exe' executable needs to be in PATH. Please download from <http://go.microsoft.com/fwlink/?LinkId=619687>

去官网下载webdiver驱动，将该驱动程序移动至代码目录下（或将驱动添加到python路径），与自己浏览器版本一致



再次测试，可成功调用浏览器进入GenScan网站，并进行预测，但是传输参数有误，无法产生正常结果

import json

import requests

seq = 'GGATAGGCATCGCCGTATTTACTACTTTGTAAACCAGTGGATTTTTGCTCAACATATAAA'

url = "http://argonaute.mit.edu/cgi-bin/genscanw\_py.cgi"

dat = {

"-o": "Vertebrate",

"-e": "0.5",

"-n": "test",

"-p": "Predicted CDS and peptides",

"-u": seq

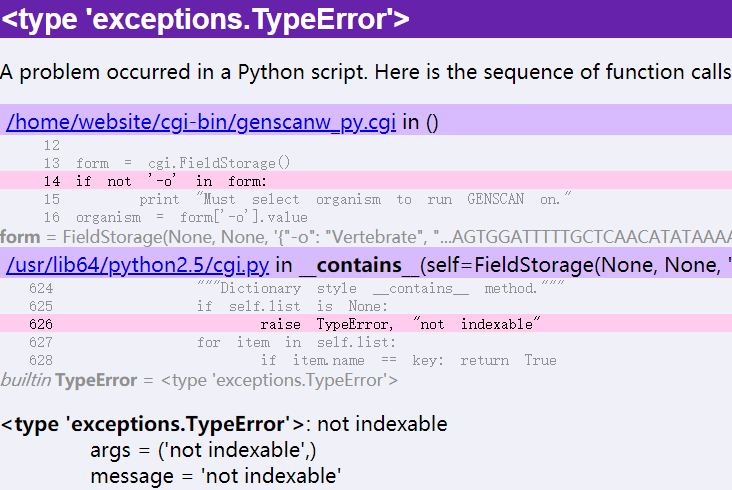
}

r = requests.post(url, json = dat)

re = r.content.decode("utf-8")

with open("test.html","w") as f:

f.write(re)



经『韩淑清』和『王颖娟』同学测试和比较后，我们选择Augustus软件作为第12步迭代测试使用的工具

2020/9/10

## step-11、编程分析第 10 步的结果，评估基因结构的完整度

编写python脚本FilterGFF3.py，筛选augustus预测结果中不完整的基因。读入gff3结果文档，已gene为单位进行甄别操作：

* 将预测正负链错误的gene删去；
* 对未被预测出的序列删去，并在predicate\_error.txt中进行标记；
* 对被预测出多个匹配基因的序列，选择其与原序列overlap最大的一个保留，且若最大overlap < 0.6，同样删去，并标记；
* 记录每一次迭代过程中未被完整预测的基因数目。

每次使用augustus软件进行基因预测的结果命名为augustus\_out\_\*-0.gff3，将所有未被完整预测的基因信息另存为augustus\_out\_\*-2.gff3文件，预测完整的数据另存为augustus\_out\_\*-1.gff3文件（\*表示迭代的次数，如此命名，方便进行批量迭代操作），并将每次预测成功的结果追加写入augustus\_out\_sum.gff3。

代码见附录1、FilterGFF3

## step-12、对 9、10 和 11 步功能进⾏封装、迭代和预测

### 1、封装模块

**1.1、接入“序列提取”模块**

感谢『罗晓琦』同学提供的“提取序列”R语言脚本，现使用python调用

尝试使用python通过cmd调用R，编写bat文件测试

echo hello

R

source("test.R")

t(1,1)

q()

pause

echo hello world

pause

在cmd中键入

cmd /c test.bat

# 回显：

E:\Stu\semester20-2\LAB-1\step-11>echo hello

hello

E:\Stu\semester20-2\LAB-1\step-11>R

R version 3.6.3 (2020-02-29) -- "Holding the Windsock"

Copyright (C) 2020 The R Foundation for Statistical Computing

Platform: x86\_64-w64-mingw32/x64 (64-bit)

R

'license()''licence()'

R.

'contributors()'

'citation()'RR

'demo()''help()'

'help.start()'HTML

'q()'R.

>

> q()

是否保存工作空间映像? [y/n/c]: n

E:\Stu\semester20-2\LAB-1\step-11>source("test.R")

'source' 不是内部或外部命令，也不是可运行的程序

或批处理文件。

E:\Stu\semester20-2\LAB-1\step-11>t(1,1)

't' 不是内部或外部命令，也不是可运行的程序

或批处理文件。

E:\Stu\semester20-2\LAB-1\step-11>q()

'q' 不是内部或外部命令，也不是可运行的程序

或批处理文件。

E:\Stu\semester20-2\LAB-1\step-11>pause

请按任意键继续. . .

E:\Stu\semester20-2\LAB-1\step-11>echo hello world

hello world

E:\Stu\semester20-2\LAB-1\step-11>pause

请按任意键继续. . .

E:\Stu\semester20-2\LAB-1\step-11>

cmd批处理无法进一步使用R窗口。

查阅资料，cmd可以执行R文件，编写test.R文件

# test in cmd

a <- data.frame(1:4,2,2)

write.table(a,"1.txt")

cmd执行

r CMD BATCH test.R

有1.txt文件输出，结果正常

编写r脚本，尝试通过cmd给r函数传入参数

args <- commandArgs(TRUE)

if(length(args)==0)

{

print("No arguments supplied.")

##supply default values

a = 1

b = c(1,1,1)

}else

{

for(i in 1:length(args))

{

print(args[i])

print(parse(text=args[[i]]))

print(eval(parse(text=args[[i]])))

}

}

cmd运行

R CMD BATCH --save test.R test.out --args 1 c(1,2,3)

R CMD BATCH --save <test.R >test.out '--args a=1 b=c(1,2,3)'

R CMD BATCH '--args a=1 b=c(1,2,3)' test.R test.out &

R CMD BATCH --no-save --no-restore '--args a=1 b=c(2,5,6)' test.R test.out &

均不能将参数成功传入R函数

尝试使用rpy2包调用R脚本

pip install rpy2

报错：

ERROR: Exception:

Traceback (most recent call last):

File "d:\python\lib\site-packages\pip\\_vendor\urllib3\response.py", line 437, in \_error\_catcher

yield

File "d:\python\lib\site-packages\pip\\_vendor\urllib3\response.py", line 519, in read

data = self.\_fp.read(amt) if not fp\_closed else b""

File "d:\python\lib\site-packages\pip\\_vendor\cachecontrol\filewrapper.py", line 62, in read

data = self.\_\_fp.read(amt)

File "d:\python\lib\http\client.py", line 461, in read

n = self.readinto(b)

File "d:\python\lib\http\client.py", line 505, in readinto

n = self.fp.readinto(b)

File "d:\python\lib\socket.py", line 589, in readinto

return self.\_sock.recv\_into(b)

File "d:\python\lib\ssl.py", line 1071, in recv\_into

return self.read(nbytes, buffer)

File "d:\python\lib\ssl.py", line 929, in read

return self.\_sslobj.read(len, buffer)

socket.timeout: The read operation timed out

……

再次尝试：

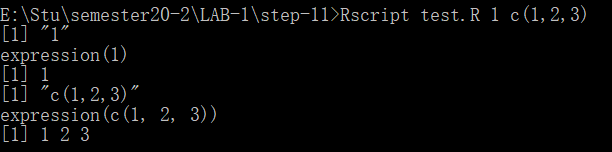
easy\_install rpy2

无结果

尝试使用Rscript命令调用R脚本

Rscript test.R 1 c(1,2,3)

回显



成功传入参数

2020/9/11

经过在机房电脑的测试，该语法可成功在Linux下运行，输出结果正常

**1.2、接入“判断预测结果完整性”模块**

由于FilterGFF3.py已被封装完毕，只需导入模块并实例化对象，即可使用

# 关键代码

from FilterGff3 import filter

# ……

fi = filter()

# ……

fi.filter\_au(gff, times, p)

2020/9/12

### 2、进行迭代预测

选择Augustus作为预测基因的工具

代码见附录2、IterativePrediction

step-12中针对Augustus软件，对第9、10 和 11 步的功能使用python脚本封装完毕，将源文件（blast\_out.gff3; FilterGff3.py; IterativePrediction.py; SeqFetch.R; S288C.fna）copy至机房电脑（s31）（将blast结果转的gff3文件重命名为augustus\_out\_0-0.gff3，便于迭代中的识别与命名），进行测试。

在修改部分语法错误后，为方便比对augustus预测出的基因位置与blast比对结果的基因位置，将提取序列的格式改为如下所示：

>NC\_001142.9;gene;627586;628152;+

TCAACCATACCACTCCGAACCACCATCCATCCCTCTACTTACTACCACTCACCCACCGTTACCCTCCAATTACCCATATCCAACCCACTGCCACTTACCCTACCATTACCCT

9/14/2020

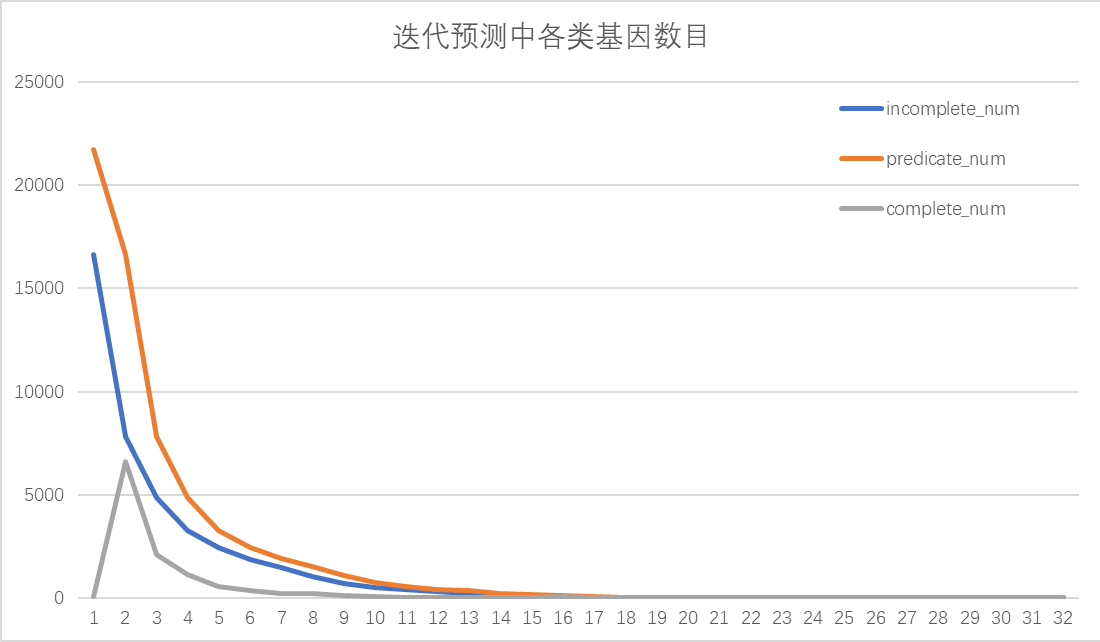
再次修正部分代码错误后，在当前文件夹路径的shell下，输入

python IterativePrediction.py

等待约2h后

共得到11529条结果

整理每次迭代时，Augustus预测的基因数目、筛选出来的完整的基因数目和不完整的基因数目，并将其可视化，得下图：



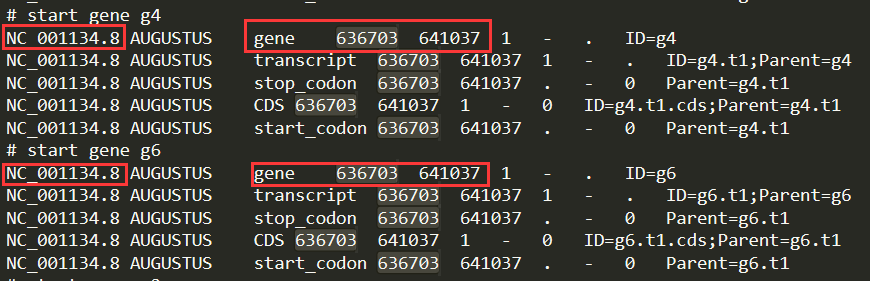
该图由『韩淑清』同学完成

结合基因频数分布直方图可知，在基于blast结果未延伸时，Augustus预测结果中完整基因数目很低，而在第一次延伸（上下游各100bp）后，预测结果的完整基因数目陡增，说明blast比对结果与原本基因的位置存在200bp左右的错位，这与基因频数分布直方图中二者中位数差值171bp相符

### 3、迭代预测的后续处理

9/15/2020

查阅结果gff3文件，发现其中存在大量冗余数据（如下图所示），猜测是由于blast比对结果的冗余造成的



编写python脚本，去除此结果中的冗余信息（将染色体编号、基因起始位置与终止位置全部相同的数据视作冗余数据），代码如下：

gene = []

gene\_id = ""

gene\_dat = ""

gene\_dict = {}

with open("augustus\_out\_sum.gff3", "r") as f:

for line in f.readlines():

if "# " in line and len(gene\_dat) != 0:

gene = gene\_dat.split("\n")[0].split("\t")

gene\_id = ";".join([gene[0], gene[3], gene[4]])

if gene\_id not in gene\_dict:

gene\_dict[gene\_id] = gene\_dat

gene\_dat = ""

elif "# " not in line:

gene\_dat += line

with open("augustus\_out\_sum\_dedup.gff3", "w") as f:

for key in gene\_dict.keys():

f.write(gene\_dict[key])

# 3291

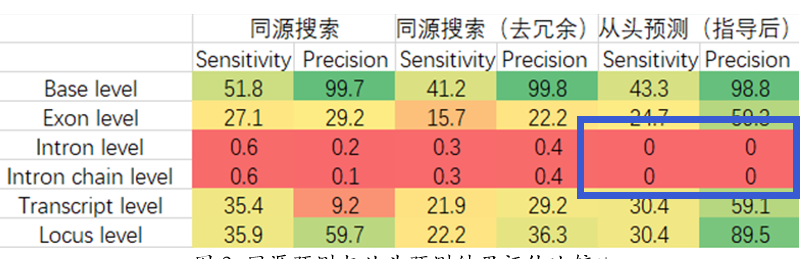
去除冗余后，还剩下3291个结果

这个数据与去除冗余后的blast比对结果相近

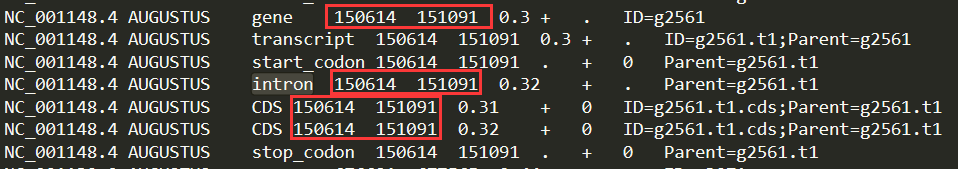
### 4、迭代预测后续处理之基因内部原件回贴

2020/9/17

在由『王颖娟』同学进行gffcompare比对后，如下图所示，我们发现同源搜索指导后的从头预测的内含子预测水平为0，比同源搜索和从头预测都要低，这很不符合常理



检阅整理后的迭代预测文件，如下图所示，我们发现基因内部元件的起始位点与终止位点全部相同



这是因为Augustus预测基因结构时，按照输入序列从1计算位置，所以我需要按照blast结果该序列原始位置、已经延伸的长度以及Augustus预测出的起始位点计算该基因在基因组中的位置。但是，我在编写筛选完整基因的代码时，直接将一个基因内部所有原件的起始位点与终止位点统一使用正则替换，造成了无法区分内部元件的错误

编写python脚本augustus\_reattchment.py，遍历所有augustus迭代预测出的文件，重新分别针对每个基因元件计算其起始位点和终止位点，代码见附录4、reattachment，并去冗余

去冗余后得到3291条结果

## 附录

### 1、FilterGFF3

import re

# for Augustus

# filename = augustus\_out\_1-0.gff3

class filter():

"""docstring for filter

"""

def \_\_init\_\_(self):

self.flag = "F"

self.out\_gff = ""

self.pass\_gff = ""

self.gene\_dict = {}

self.incomplete\_num = 0

def whether\_complete(self, gene, extend\_len):

items = gene.split("\n")

initial\_start = int(items[1].split("\t")[0].split(";")[2])

[now\_start, now\_end] = [int(items[1].split("\t")[3]), int(items[1].split("\t")[4])]

new\_start = initial\_start - extend\_len + now\_start - 1

new\_end = initial\_start - extend\_len + now\_end - 1

gene = re.sub(r"\t[0-9]+\t[0-9]+\t", "\t"+str(new\_start)+"\t"+str(new\_end)+"\t", gene)

if "start\_codon" in gene and "stop\_codon" in gene:

with open(self.pass\_gff, "a") as p:

p.write(re.sub(r";gene.+;[\+\-]","",gene))

else:

self.flag = "T"

self.incomplete\_num += 1

with open(self.out\_gff, "a") as o:

o.write(re.findall(r"g[0-9]+\n(.+\n?)NC\_", gene)[0])

def write\_error(self, gff, times):

pre\_dict = {}

if times == 1:

with open("extend\_0bp.fasta", "r") as f:

for line in f.readlines():

if ">" in line:

id = line.strip()[1:]

if id not in pre\_dict:

pre\_dict[id] = 1

else:

pre\_num = int(re.findall(r"out\_(.+)\-", gff)[0])

pre\_gff = re.sub(r"[0-9]+\-0", str(pre\_num - 1)+"-2", gff)

with open(pre\_gff, "r") as p:

for line in p.readlines():

if "NC\_" in line:

id = line.split("\t")[0]

if id not in pre\_dict:

pre\_dict[id] = 1

for key in pre\_dict.keys():

if key not in self.gene\_dict:

with open("predicate\_error.txt", "a") as f:

f.write(key.replace(";", "\t") + "\tpredicate\_null\n")

def max\_overlap(self, genes, pre\_dat, strand, extend\_len):

max\_index = -1

max\_overlap = 0

native\_interval = range(extend\_len + 1, int(pre\_dat[3]) - int(pre\_dat[2]) + 2 + extend\_len)

for i in range(len(genes)):

if pre\_dat[4] == strand:

new\_interval = range(int(genes[i].split("\t")[3]), int(genes[i].split("\t")[4])+1)

new\_overlap = len(list(set(native\_interval).intersection(new\_interval)))

if new\_overlap > max\_overlap:

max\_overlap = new\_overlap

max\_index = i

if max\_overlap/len(native\_interval) < 0.6:

max\_index = -1

with open("predicate\_error.txt", "a") as f:

f.write("\t".join(pre\_dat)+"\tdislocation\n")

return(max\_index)

def filter\_au(self, gff, times, p):

self.flag = "F"

id = ""

new\_gene = ""

old\_gene = []

self.pass\_gff = gff.replace("-0.gff3","-1.gff3")

self.out\_gff = gff.replace("-0.gff3","-2.gff3")

with open(gff, "r") as f:

for line in f.readlines():

if "# start gene" in line:

new\_gene = ""

if "# protein sequence" in line:

gene\_head = re.findall(r"g[0-9]+\n(.+\n?)NC\_", new\_gene)[0].split("\t")

strand = gene\_head[6]

pre\_dat = gene\_head[0].split(";")

if gene\_head[0] == id:

old\_gene.append(new\_gene)

else:

if len(old\_gene) > 0:

ind = self.max\_overlap(old\_gene, pre\_dat, strand, (times-1)\*p)

if ind >= 0:

self.whether\_complete(old\_gene[ind], (times-1)\*p)

self.gene\_dict[id] = 1

id = gene\_head[0]

old\_gene = [new\_gene]

new\_gene = ""

elif "# command line" in line:

ind = self.max\_overlap(old\_gene, pre\_dat, strand, (times-1)\*p)

if ind >= 0:

self.whether\_complete(old\_gene[ind], (times-1)\*p)

gene\_head = re.findall(r"g[0-9]+\n(.+\n?)NC\_", old\_gene[ind])[0].split("\t")

id = gene\_head[0]

self.gene\_dict[id] = 1

else:

new\_gene += line

self.write\_error(gff, times)

n = self.incomplete\_num

self.incomplete\_num = 0

return([self.flag, self.out\_gff, n])

### 2、IterativePrediction

import os

from FilterGff3 import filter

p = 100

times = 1

flag = "F"

fi = filter()

gff = "augustus\_out\_0-0.gff3"

# 从blast结果gff中提取序列并预测、筛选，并判断预测结果是否完整

os.system("Rscript SeqFetch\_0.R 0 0 " + gff)

gff = "augustus\_out\_" + str(times) + "-0.gff3"

# augustus --gff3=on --outfile=Sc\_augustus\_out.gff3 --species=saccharomyces\_cerevisiae\_S288C S288C.fna

os.system("augustus --gff3=on --species=saccharomyces\_cerevisiae\_S288C --outfile=" + gff + " extend\_0bp.fasta")

[flag, gff, incomplete\_num] = fi.filter\_au(gff, times, 0)

with open("incomplete\_num.txt", "a") as n:

n.write("extend\tincomplete\_num\n0bp\t" + str(incomplete\_num) + "\n")

while flag == "T" and times <= 80:

# call R script to extract sequences

os.system("Rscript SeqFetch.R " + str(p) + " " + str(times) + " " + gff)

# run to predict

times += 1

gff = "augustus\_out\_" + str(times) + "-0.gff3"

fasta = " extend\_" + str((times-1)\*p) + "bp.fasta"

os.system("augustus --gff3=on --species=saccharomyces\_cerevisiae\_S288C --outfile=" + gff + fasta)

# filter incomplete

[flag, gff, incomplete\_num] = fi.filter\_au(gff, times, p)

with open("incomplete\_num.txt", "a") as n:

n.write(str(times\*p) + "bp\t" + str(incomplete\_num) + "\n")

### 3、dedup

class dedup():

"""docstring for dedup"""

def \_\_init\_\_(self, gff):

self.gff = gff

def maxscore(self, overlap\_set):

max\_s = 0

max\_re = []

for gene in overlap\_set:

if float(gene[5]) > max\_s:

max\_re = gene

return(max\_re)

def readwrite(self):

old\_chr = ""

gene\_dat = ""

overlap\_set = []

old\_interval = range(1)

with open(self.gff, "r") as f:

for line in f.readlines():

if "gene" in line and len(gene\_dat) != 0:

gene = gene\_dat.split("\t")

new\_chr = gene[0]

if new\_chr == old\_chr:

new\_interval = range(int(gene[3]), int(gene[4]))

# percent = len(set(old\_interval).intersection(set(new\_interval)))/len(set(new\_interval))

# if percent > 0.6:

if len(set(old\_interval).intersection(set(new\_interval))) > 0:

overlap\_set.append(gene)

elif len(overlap\_set) > 0:

with open("blast\_output\_gff3\_dedup.gff3", "a") as b:

b.write("\t".join(self.maxscore(overlap\_set)))

overlap\_set = [gene]

old\_interval = new\_interval

else:

old\_chr = new\_chr

with open("blast\_output\_gff3\_dedup.gff3", "a") as b:

b.write("\t".join(self.maxscore(overlap\_set)))

overlap\_set = [gene]

gene\_dat = line

else:

gene\_dat += line

gene = gene\_dat.split("\t")

new\_interval = range(int(gene[3]), int(gene[4]))

if len(set(old\_interval).intersection(set(new\_interval))) > 0:

overlap\_set.append(gene)

with open("blast\_output\_gff3\_dedup.gff3", "a") as b:

b.write("\t".join(self.maxscore(overlap\_set)))

else:

with open("blast\_output\_gff3\_dedup.gff3", "a") as b:

b.write("\t".join(self.maxscore(overlap\_set)))

b.write("\t".join(self.maxscore([gene])))

d = dedup("blast\_output\_gff3\_sorted\_chain.txt")

### 4、reattachment

import re

class reattachment:

def \_\_init\_\_(self):

self.pass\_gff = "augustus\_reattachment.gff3"

def whether\_complete(self, gene, extend\_len):

initial\_gene = ""

items = gene.split("\n")

initial\_gene += items[0] + "\n"

initial\_start = int(items[1].split("\t")[0].split(";")[2])

for i in range(1, len(items)-1):

[now\_start, now\_end] = [int(items[i].split("\t")[3]), int(items[i].split("\t")[4])]

new\_start = initial\_start - extend\_len + now\_start - 1

new\_end = initial\_start - extend\_len + now\_end - 1

initial\_gene += re.sub(r"\t[0-9]+\t[0-9]+\t", "\t"+str(new\_start)+"\t"+str(new\_end)+"\t", items[i]) + "\n"

if "start\_codon" in initial\_gene and "stop\_codon" in initial\_gene:

with open(self.pass\_gff, "a") as p:

p.write(re.sub(r";gene.+;[\+\-]","",initial\_gene))

def max\_overlap(self, genes, pre\_dat, strand, extend\_len):

max\_index = -1

max\_overlap = 0

native\_interval = range(extend\_len + 1, int(pre\_dat[3]) - int(pre\_dat[2]) + 2 + extend\_len)

for i in range(len(genes)):

if pre\_dat[4] == strand:

new\_interval = range(int(genes[i].split("\t")[3]), int(genes[i].split("\t")[4])+1)

new\_overlap = len(list(set(native\_interval).intersection(new\_interval)))

if new\_overlap > max\_overlap:

max\_overlap = new\_overlap

max\_index = i

if max\_overlap/len(native\_interval) < 0.6:

max\_index = -1

return(max\_index)

def filter\_au(self, gff, times, p):

gene\_id = ""

new\_gene = ""

old\_gene = []

with open(gff, "r") as f:

for line in f.readlines():

if "# start gene" in line:

new\_gene = ""

if "# protein sequence" in line:

gene\_head = re.findall(r"g[0-9]+\n(.+\n?)NC\_", new\_gene)[0].split("\t")

strand = gene\_head[6]

pre\_dat = gene\_head[0].split(";")

if gene\_head[0] == gene\_id:

old\_gene.append(new\_gene)

else:

if len(old\_gene) > 0:

ind = self.max\_overlap(old\_gene, pre\_dat, strand, (times-1)\*p)

if ind >= 0:

self.whether\_complete(old\_gene[ind], (times-1)\*p)

# self.gene\_dict[gene\_id] = 1

gene\_id = gene\_head[0]

old\_gene = [new\_gene]

new\_gene = ""

elif "# command line" in line:

ind = self.max\_overlap(old\_gene, pre\_dat, strand, (times-1)\*p)

if ind >= 0:

self.whether\_complete(old\_gene[ind], (times-1)\*p)

gene\_head = re.findall(r"g[0-9]+\n(.+\n?)NC\_", old\_gene[ind])[0].split("\t")

gene\_id = gene\_head[0]

else:

new\_gene += line

r = reattachment()

for i in range(1,33):

gff = "augustus\_out\_" + str(i) + "-0.gff3"

r.filter\_au(gff, i, 100)