

dy的生信学习笔记

一.Linux

some tips and knowledge

遇到过的一些问题及解决

二.python&R 常见数据处理

jupyter

单细胞分析

GWAS&QTL

dy的生信学习笔记

一.Linux

some tips and knowledge

常见中值得注意的

```
1 top （查看cpu使用情况）
2 ssh -L localhost:[number]:[remoteip]:[number] 服务器地址 账号 （可在本地端口运行服务器端口的程序）
3 rm -rf （强制删除文件夹）
4 conda search + conda install 很好用!!!
5 ls -hl （可以以K,M,G的单位显示当前文件夹大小）#没法显示子文件夹里面的内存
6 file=`ls path/` ; echo ${file##*/} # 只输出目录后面的文件名！（琢磨半小时弄出来的）
```

补：

```
1 #conda是个p, mamba才是王道
2 mamba search + mamba install!
3 #使用mamba后，最好把anaconda里面的清空，有时候会出现混用的情况
```

装R包的tips（我放到了python&R 常见数据处理那一部分）

sh脚本文件

```
1 脚本的参数传入：
2 $0:该脚本
3 $1:输入的的第一个参数
4 $2:输入的第二个参数
5 .. ${n}
6 example: vim test.sh
7 ##in vim
8 echo $1 （其实shell里命令之间用‘;’分割即可，为了美观这里还是使用换行）
9 echo $2
10 ESC :wq## out of vim
11 bash ./test.sh(. ./test.sh亦可) ustdc ayd
12 >ustdc
13 >ayd
```

处理文本的工具awk

```
1 这里用一个例子来说明
2  cat pruned_coatColor_maf_geno.vcf | awk 'BEGIN{FS="\t";OFS="\t";}/#/{next;}
    {{if($3==".")$3=$1:"$2;}print $3,$5;}' > alt_alleles
```

看起来很复杂，一步一步看

"|"代表左边的输出结果直接作为右边的输入，即cat读取文件后传入右边的awk，故awk的最后一个参数（输入文件）省去了

awk实际是逐行读取文件，awk -F 可以选择分隔的符号。awk -F [,]可以先空格分隔，再','分隔一次

awk 'BEGIN{ }' 可以填写在读取文件前的操作，如print。此处设置FS与OFS都等于"\t"（读取文件分隔符和输出所用分隔符）

/#/代表正则表达式含"#"的部分，若该行满足条件，则运行后面的next，跳过该次循环（即进入下一行），否则继续运行后面的部分

if (\$3 == "."),若第三列是".",则使第三列等于第一列加": "加第三列，然后输出第三列和第五列

最后的>,非常常用的重定向，写入文件

如果报错backslash not last character on line，记得删掉\$3这种变量或引号前面的反斜杠\（直接运行不要加，提交作业可能要加，巨sb）

-F也可以起到这种作用：awk -F '/' '{print \$1}'

一种选择多文件操作的方法

```
1  files="./set*.vcf"      #注意shell语言中'='的两边不要有空格！
2  file=`ls files | head -n $id | tail -n 1`
```

ls files会将满足正则的文件全部列出，随head -n 次数的累加与tail，每轮依次选择不同文件。id需是可以循环累加或类似于 \$SLURM_ARRAY_TASK_ID这种变量

文本工具sed

```
1  sed 's/.gz//g'
```

s为更改，即：将.gz更改为/g'(应该是空格)

使用软件的方法（软链接）

以存放在/home/path/路径的tool软件为例

```
1  /home/path/tool 即可运行
2  可使用软链接放入bin目录中
3  cd xxx/bin
4  ln -s /home/path/tool atool #软链接，后面是快捷用法
5  atool #即可
6
7  ll可查看目录下的软链接
8  which可查看命令存储的位置（类似这里软链接）
```

自定义命令的方法

例：想自定义一个叫“work”的命令，作用是切换到“/home/work”目录并检索目录，同时输出一段话

```
1 vim ~/.bashrc
2 #在.bashrc中写入
3 alias work="cd /home/work && ls && echo ""It is time for work!"" " #两个双引号
   是为了区别单个双引号（也可使用单引）
4 #退出.bashrc
5 source ~/.bashrc
```

一个自定义命令的截图 (*don't mind the details*)

```
(base) [dingyi@login01 ~]$ hhh
you hhh your horse?
```

遇到过的一些问题及解决

issue 1

焦虑等级：****

我想自定义一个脚本“me”，使得我输入me -xxx 可以实现不同的功能（如跳转到相应的目录）

```
#!/bin/sh

while getopts ":weadj" opt
do
    case $opt in
        w)
            cd /storage/yangjianLab/dingyi/work && echo "It is working time! You can make it!" && ls
            ;;
    esac
done
```

(部分截图)

使用bash命令运行后，可以看到，切换目录的需求并没有实现

```
(base) [dingyi@login01 ~]$ bash /storage/yangjianLab/dingyi/tools/me -w
It is working time! You can make it!
data paper_gastric qtl_exp STR_nat_gen SV_QTL test
(base) [dingyi@login01 ~]$
```

这是因为bash(以及 sh、./)运行脚本时，会新开一个shell运行，最后将结果返回到当前shell中（可以与作业系统类比），所以cd在新开的shell里面被实现后，无法对当前shell产生影响。

于是我使用了source (./等价)，该命令会在当前shell中运行脚本

```
(base) [dingyi@login01 ~]$ source /storage/yangjianLab/dingyi/tools/me -w
It is working time! You can make it!
data paper_gastric qtl_exp STR_nat_gen SV_QTL test
(base) [dingyi@login01 work]$
```

可以看到成功切换目录了

然后最操蛋的事情就来了..

```
(base) [dingyi@login01 ~]$ source /storage/yangjianLab/dingyi/tools/me -w
(base) [dingyi@login01 ~]$
```

.....

在这个shell中，再次运行就会像图里面那样，毫无反应

经过漫长的debug过程，我可以确定问题出在getopts上，应该和当前shell有关。终于，我找到了解答

通过 source 多次执行脚本对 OPTIND 的影响

通过 source 命令调用脚本是运行在当前 shell 下，由于 shell 不会自动重置 OPTIND 的值，如果要调用的脚本使用了 getopt 命令解析选项参数，在每次调用 getopt 之前，一定要手动重置 OPTIND 为 1，否则 OPTIND 的值不是从 1 开始递增，会获取到不预期的选项参数值。

假设有一个 test.sh 脚本，其内容如下：

```
#!/bin/bash

echo \${#}: ${#}, \${@}: ${@}, OPTIND: $OPTIND
getopts "abc" opt
echo $?, $opt
```

分别不使用 source 命令和使用 source 命令执行该脚本，结果如下：

```
$ ./test.sh -a
${#}: 1, ${@}: -a, OPTIND: 1
0, a
$ ./test.sh -b
${#}: 1, ${@}: -b, OPTIND: 1
0, b
$ source ./test.sh -a
${#}: 1, ${@}: -a, OPTIND: 1
0, a
$ source ./test.sh -b
${#}: 1, ${@}: -b, OPTIND: 2
1, ?
```

可以看到，执行 ./test.sh -a 命令和 ./test.sh -b 命令的输出结果都正常。

执行 source ./test.sh -a 命令的结果也是正常。

但是接着执行 source ./test.sh -b 命令，调用 getopt 之前，打印出 OPTIND 的值是 2，要获取第二个选项参数。

由于没有提供第二个选项参数，获取到的选项参数值是问号 ?，用 \$? 获取 getopt 命令的返回值是 1，执行报错。

即，如果一个脚本使用了 getopt 命令，而该脚本又要用 source 命令来执行时，脚本需要手动设置 OPTIND 变量的值为 1，否则会遇到上面的异常。

来源：<https://segmentfault.com/a/1190000021491911>

```
#!/bin/sh
OPTIND=1
while getopts ":weadj" opt
do
    case $opt in
        w)
            cd /storage/yangjianLab/dingyi/work && echo "It is working time! You can make it!" && ls
    ;;
    *)
    ;;
done
```

搞定

二.python&R 常见数据处理

当读取很大的datafram文件时

```

1 | scipy.io.mmread
2 | scipy.io.mmwrite("xxx.mtx") (写为稀疏矩阵)
3 |
4 | df.to_hdf('xxx.h5', key='df')
5 | pd.read_hdf('xxx.h5', key='df')

```

```

1 | data.table::fwrite() 速度较快
2 |
3 | as(data.matrix(df), 'dgCMatrix')
4 |
5 | library(Matrix)
6 | writeMM(obj, file="xxx.mtx")
7 | readMM()

```

R值得注意的

```

1 | keeplist <- c()
2 | for (i in 1:20){
3 |     if (xxx){
4 |         keeplist <- c(keeplist,i)
5 |     }
6 | }
7 | #python的append在R中类似的用法
8 |
9 | list <- 1:5
10 | paste0("x_", list)
11 | #输出
12 | > x_1 x_2 x_3 x_4 x_5
13 |
14 | 想输出字符串的长度，请用nchar!! 不要用length
15 | length(c('aaa', 'bbcd'))#输出字符串的个数
16 | > 2
17 | nchar(c('aaa', 'bbcd'))#字符串长度
18 | > 3 4
19 |
20 | i=2;j=8
21 | c((i-1):(j+1)) #加括号!
22 |
23 | str_split_fixed("a.b.c", "\\.", Inf)
24 | > a b c
25 | #需要\\.的格式表示你真的希望他识别为一个点，而不是任意字符（R中正则语言的标准是双反斜杠）






```

R装包tips (保你能装上)

补：如果是网络问题，下载工具打不开链接那就没办法了

以R包 plink2R 为例 (阴间包)

- 1.按惯例试试 `install.packages("plink2R")`，不行就用 `BiocManager::install("plink2R")`
- 2.上面两个要是都不行，就只能用 `conda/mamba search +conda/mamba install`
- 3.conda也找不到的话，那基本就是小众到离谱的阴间R包了。去你看到这个包的网站(rcran、github等等),把R包文件夹手动下载下来，然后用 `devtools::install_local("path/xxx")` (`devtools`、`remote`都行)
- 4 在这里，我从github上下载了 `master.zip`,解压后发现其中的 `Plink2R`子文件夹是所需的R包，于是 `devtools::install_local("xxx/master/plink2R")`,成功安上

 R	Added imputation of missing genotypes by sampling proportional to the
 man	initial commit
 src	Removed old dependency on mmap
 DESCRIPTION	Removed old dependency on mmap
 NAMESPACE	initial commit

这是子文件夹 `plink2R` 里面的模样，R包一般需具有R代码文件、description（必需）、其他文件。

运行截图:

```
devtools::install_local("/storage/yangjianLab/dingyi/tools//plink2R-master/plink2R/")
```

Skipping 2 packages not available: RcppEigen, Rcpp

```
— R CMD build —
* checking for file ‘/tmp/RtmpTGDBNP/file375441688a1c2c/plink2R/DESCRIPTION’ ... OK
* preparing ‘plink2R’:
* checking DESCRIPTION meta-information ... OK
* cleaning src
* checking for LF line-endings in source and make files and shell scripts
* checking for empty or unneeded directories
* building ‘plink2R_1.1.tar.gz’
Warning in sprintf(gettext(fmt, domain = domain), ...) :
  one argument not used by format 'invalid uid value replaced by that for user 'nobody'
Warning: invalid uid value replaced by that for user 'nobody'
```

补: *BiocManager*的使用:

```
1 R<3.5.0
2 source("https://bioconductor.org/biocLite.R")
3 BiocInstaller::biocLite(c("packages1", "packages2"))
4 R>=3.5.0
5 if (!require("BiocManager", quietly = TRUE))
6   install.packages("BiocManager")
7 BiocManager::install()#install the BiocManager
8 BiocManager::install(c("packages1", "packages2"))
```

jupyter

在新的账号上安装jupyter (base环境)

```

1 conda/pip install jupyter
2 conda/pip install jupyterlab
3 jupyter notebook --generate-config #生成~/.jupyter/jupyter_notebook_config.py
  文件
4 jupyter notebook password # 然后输入想设置的密码并重复密码，之后即可打开文件
  ~/.jupyter/jupyter_notebook_config.json，复制文件中保存的密钥xxxxxxx（是一串字符）
5 #打开~/.jupyter/jupyter_notebook_config.py文件，加入
6 c.NotebookApp.allow_remote_access = True#允许远程访问
7 c.NotebookApp.ip='*' #似乎'*'或者'0.0.0.0'效果一样
8 c.NotebookApp.password = u'xxxxxxx'#这里是刚才的哈希密码（那一串字符）
9 c.NotebookApp.open_browser = False#不打开浏览器
10 c.NotebookApp.port =8888#这个端口是只运行命令jupyter notebook时的默认端口，可以通过
   命令jupyter notebook --port xxx 来设置端口

```

运行命令改为jupyter lab即可使用lab，运行notebook后，在本地浏览器链接后加上lab同样可以 如 localhost:2000/lab

创建多个kernel

R

```

1 install.packages("IRkernel") #不行就用conda
2 IRkernel::installspec(name = 'ir33', displayname = 'R 3.3')

```

python

```

1 pip install ipykernel #conda 也行
2 python -m ipykernel install --user --name py_38

```

little tricky

ctrl + 鼠标左键 可以自由产生多个输入光标

alt + 鼠标左键上拉/下拉 产生一个包括多行的大光标

单细胞分析

一些细胞种类的marker

B cells:CD19,MS4A1

plasma cells(浆细胞):IGHG1,CD79A

CD4+T cells:CD3D,CD4

CD8+T cells:CD3D,CD8A

natural killer cells(NK): NVR1, FGFBP2

myeloid cells(髓细胞):CD14,CD68

mast cells(肥大细胞):TPSAB1,TPSB2

endothelial cells(内皮细胞):RAMP2,PECAM1

fibroblasts(成纤维细胞):DCN,LUM

mural cells(壁细胞):PDGFRB,ACTA2

glial cells(胶质细胞):PLP1,SOX10

epithelial cells(上皮细胞):PGC,PGA3

GWAS&QTL

文件格式

0代表缺失

*.ped										*.map			
FID	IID	PID	MID	Sex	P	rs1	rs2	rs3		Chr	SNP	GD	BPP
1	1	0	0	2	1	CT	AG	AA		1	rs1	0	870000
2	2	0	0	1	0	CC	AA	AC		1	rs2	0	880000
3	3	0	0	1	1	CC	AA	AC		1	rs3	0	890000

*.fam						*.bed		*.bim					
FID	IID	PID	MID	Sex	P	Contains binary version of the SNP info of the *.ped file. (not in a format readable for humans)		Chr	SNP	GD	BPP	Allele 1	Allele 2
1	1	0	0	2	1			1	rs1	0	870000	C	T
2	2	0	0	1	0			1	rs2	0	880000	A	G
3	3	0	0	1	1			1	rs3	0	890000	A	C

Covariate file						Legend			
FID	IID	C1	C2	C3		FID	Family ID	rs(x)	Alleles per subject per SNP
1	1	0.00812835	0.00606235	-0.000871105		IID	Individual ID	Chr	Chromosome
2	2	-0.0600943	0.0318994	-0.0827743		PID	Paternal ID	SNP	SNP name
3	3	-0.0431903	0.00133068	-0.000276131		MID	Maternal ID	GD	Genetic distance (morgans)
						Sex	Sex of subject	BPP	Base-pair position (bp units)
						P	Phenotype	C(x)	Covariates (e.g. Multidimensional Scaling (MDS) components)

Plink

--geno 筛选variants

--mind 筛选samples

osca

--eqtl:

- --bfile: reads individual-level SNP genotype data (in PLINK binary format), i.e. .bed, .bim, and .fam files.
- --bfile: to input molecular phenotypes (e.g. DNA methylation or gene expression measures in BOD format).
- --qcovar : reads quantitative covariates from a plain text file
- something else

--sqt (THESTLE 工具): 与 eqtl类似, 看说明文档就好

也可直接输入qtl summary文件

--meta 用来合并来自同一cohort的多个qtl结果文件

--Mecs 合并不同cohort的qtl文件

sQTL

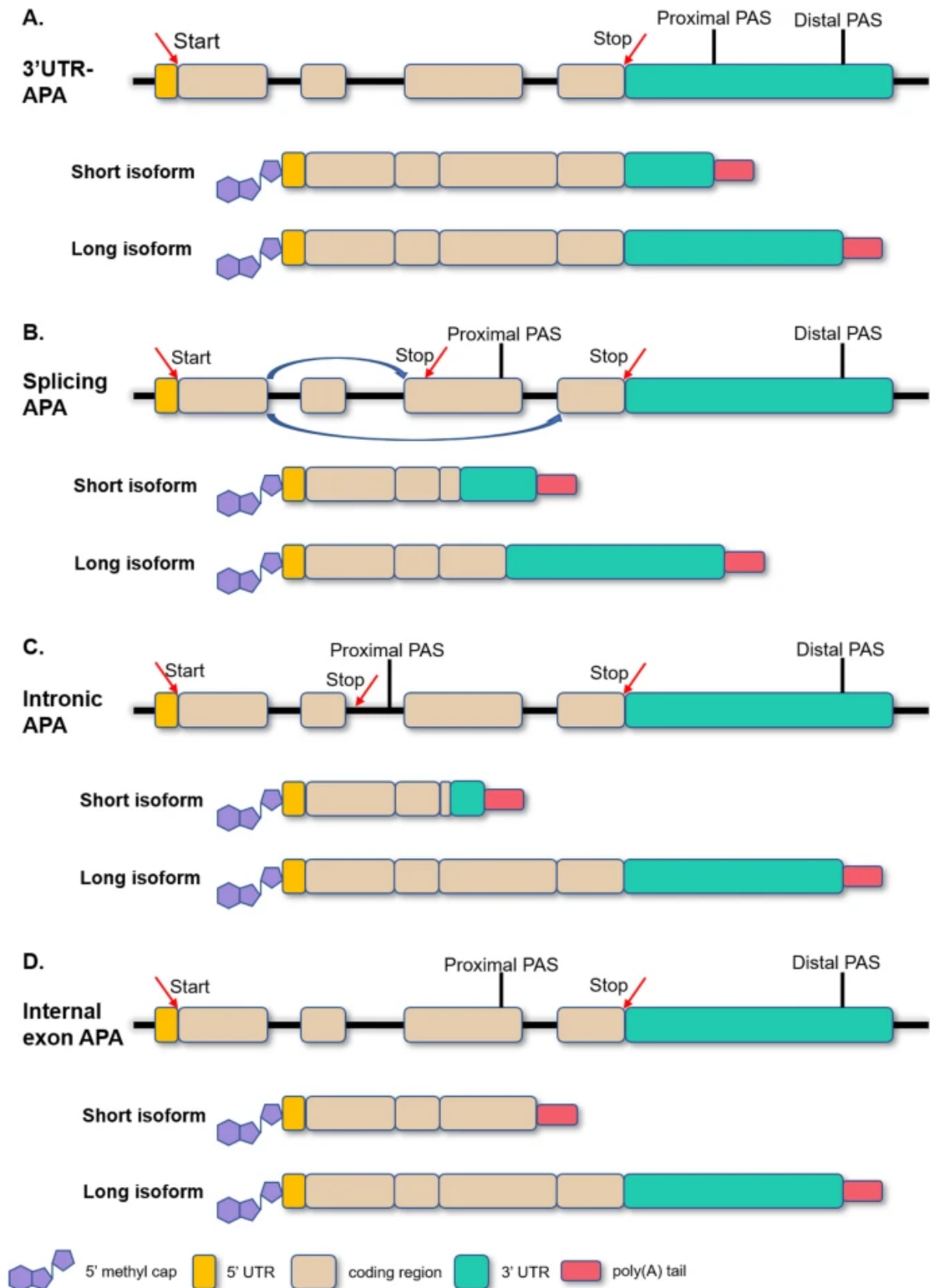
transcript-level: 需要reference, 通过map提取出不同的isoform。

THESTLE

event-level: 不需要reference,通过观察intron的去除来构建剪接过程,可以发现新的alternative splicing events

Leafcutter

APA (alternative polyadenylation)



3'UTR APA

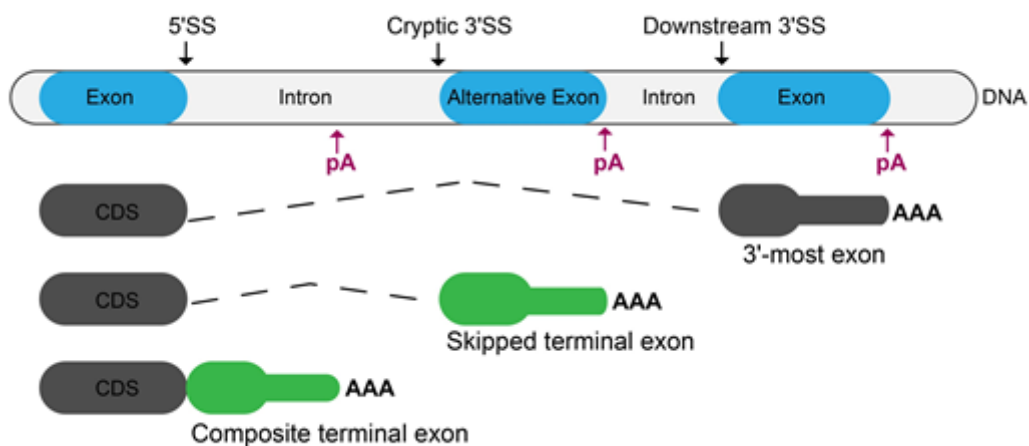
分析软件: DaPars

PDUI值 (Percentage of Distal polyA site Usage Index) : 变大表示UTR长度增加, 反之减少。

$$PDUI = \frac{w_L^{i*}}{w_L^{i*} + w_S^{i*}} \quad (\text{Eq.2})$$

where w_L^{i*} and w_S^{i*} are the estimated expression levels of transcripts with distal and proximal polyA sites for sample i . The greater the PDUI is, the more distal polyA site of a transcript is used and vice versa. Finally, the regression model is extended towards the internal exons, so that splicing coupled APA events can also be detected.

Intronic APA



分析软件:

APAlzyer (安这东西差点要了我半条命, R4.2怎么都不行, 最后R4.1用conda安装成功了.. 另外如果 BiocManager网页连不上, 试试终端里的R, 可能可以) (垃圾包, 别用😞)

IPAFinder:原理和DaPars有点像, 都是只依赖基因组注释信息(包含exon、intron等信息), 来发现novel的intronic APA。(说白了就是不需要poly(A)位点的注释文件) 遍历intron部分的每一个point, 分别计算point上游和下游read数的均方误差 MES_u 、 MES_d 以及整个intron区域read数均方误差 MES_e (应该是对每个碱基标准化后的coverage数求MES)

$Ratio_{MSE} = \frac{MSE_u + MSE_d}{MSE_e}$ 最小的点被预测为intronic poly(A)位点

IPAFinder先检测composite terminalexon IpA event, 然后检测 skipped terminal exon IpA event,之后通过junction-spanning reads来排除alternative splice位点

IPUI值

$$\text{IPUI} = \frac{E_{IPA}^j}{E_{IPA}^j + E_{FL}^j} = \frac{E_{IPA}^j}{E_{CPE}^j}, \quad (1)$$

where E_{IPA}^j , E_{FL}^j , and E_{CPE}^j are the estimated expression levels of IpA isoform, full-length isoform, and constitutive preceding exon located upstream of the IPA site for a given sample (j), respectively. In principle, E_{CPE}^j is equal to the sum of E_{IPA}^j and E_{FL}^j (Supplemental Fig. S22).