autopipe模块使用手册

声明

autopipe是用来辅助进行生物信息学相关流程搭建的一个模块，模块对目前常见的流程构架，进行了归纳总结，自定义了一些模块化的类和函数，方便快速的进行分析流程的搭建。只需要提供相应的分析点和对应运行命令，即可自动化的生成可读性较强的目录结构、分析脚本及投递脚本。

反馈邮箱：pai@genedenovo.com

版本

目前版本为 `autopipe v1.0`

更新说明

暂无

使用需求

1. Perl 5.x
2. 设置环境变量 `AP_PATH` , 例如： `export AP_PATH=/Bio/User/aipeng/bin/autopipe`

使用示例

```
# 导入模块
use lib "$ENV{'AP_PATH'}/lib";
use PIPE;

# 初始化流程
my $conf_file = "";
my $pipe = PIPE->new(-name=>"reads_qc",-conf=>$conf_file);

# 添加分析 （一级）
my $filter = $pipe->step(-name=>"filter")

# 添加分析（二级）
my $fastp = $fliter->substep(-name=>"fastp",-cpu=>4,-mem=>4,-queue=>"all.
q")

# 给fastp分析添加命令
$fastp->cmd("fastp -i S1_1.fq.gz ...")
$fastp->cmd("fastp -i $2_1.fq.gz ...")
...

# 添加分析（二级）
my $rm_rRNA = $filter->substep(-name=>"rRNA",-cpu=>6,-mem=>4,-queue=>"av
x.q")
...

# 给filter添加命令
$filter->cmd("...")

# 生成脚本并运行
$pipe->run(-mode=>"track")

# 只生成脚本
```

```
$pipe->run(-mode=>"shell")

__END__
```

模块PIPE

主模块，用来初始化流程并对流程进行各种操作。同时会读取配置文件，并对配置文件的内容进行解读。

new()

使用方法

```
my $pipe = PIPE->new(-name=>'pipe_name',-conf=>$conf_file,-mode=>"shell",
-outdir=>"outdir"
```

功能

创建并初始化一个流程，初始化流程包括读取配置文件并进行解析。

返回值

返回一个**PIPE**对象

参数

- **-name:** 流程的名称，不可缺省。
- **-conf:** 流程的配置文件，可以是一个文件的路径（符合Config::General模块要求的格式），也可以是一个哈希的连接，不可缺省。
- **-mode:** 流程的运行模式，只有2个值，**shell**和**track**，缺省值为**shell**。
- **-outdir:** 流程的输出路径，缺省值为**-name**。

path

PIPE的属性名，调用方法 `$pipe->{path}` ，保存的是流程的根目录，即设置的 `-outdir` 。

shelldir

PIPE的属性名，调用方法 `$pipe->{shelldir}` ,保存的是流程脚本存放的根目录。

step()

使用方法

```
my $step1 = $pipe->step(-name=>"name1")
```

```
my $step2 = $pipe->step(-name=>"step_name",-cpu=>1,-mem=>1,-queue=>"all.q")
```

功能

用来给pipe添加一个一级分析点[1]。

返回值

返回一个STEP对象

参数

- **-name:** 一级分析点的名称，不可缺省，**不同的STEP之间不能使用相同的名称!**
- **-cpu:** 分析点所使用的cpu（用来设置qub-sge.pl投递该步骤脚本时-max_job参数的值），缺省为1.
- **-mem:** 分析点所使用的最大内存，缺省为1（1G）。
- **-queue:** 分析点投递的队列名称，缺省为"all.q"。

make_path_tree()

使用方法

```
$pipe->make_path_tree(-clear=>0)
```

功能

创建该流程所涉及到的所有目录结构。

返回值

无

参数

- **-clear:** 1/0, 是否覆盖之前的目录结构，当设置为1时，会将之前的目录结构清空（如果有），缺省值为0。

create_shell()

使用方法

```
$pipe->create_shell(-run=>0)
```

功能

创建流程的脚本并运行。

返回值

无

参数

- **-run:** 1/0，创建脚本的时候是否执行，缺省值为0（不执行）.

pipeinfo()

使用方法

```
$pipe->pipeinfo(-file=>"pipe.md");
```

功能

将流程的各个步骤的信息，包括开始和结束时间都写入一个markdown格式的文件中，便于后面查看和检查。

返回值

保存的文件路径

参数

- **-file:** 保存的文件名称，缺省值为$pipe->{'pipe_name'}，最终生成的文件名为： `$pipe->{path}/'{-file}.pipe.md'`

run()

`$pipe->run()` 创建流程的目录结构，并生成各级的任务脚本和投递脚本。等同于下面的代码：当 `-mode` 为 `shell` 时，-run=>0；当 `-mode` 为 `track` 时，-run=>1。

```
$pipe->make_path_tree();
$pipe->create_shell(-run=>0|1);
$pipe->$pipeinfo();
```

all_steps()

`$pipe->all_steps()` 用来返回该流程所有的一级分析点，一般情况下用不到。

all_analysis()

`$pipe->all_steps()` 用来返回该流程所有的二级分析点，一般情况下用不到。

samples()

使用方法

```
my @samples = $pipe->samples();
my @samples = $pipe->samples(-attr=>"samples_order",-sep=>"|");
```

功能

用来从配置文件中提取样本名称。

返回值

回样本名称的数组。

参数

- **-attr:** 配置文件中设置样本名的属性名，缺省值为“samples”;
- **-sep:** 配置文件中样本名称的分隔符，可以是正则表达式，缺省值为“[;,\s\t]”。

模块STEP

analysis()

使用方法

```
$step->analysis()
```

功能

给一级分析点添加一个二级分析点。

返回值

返回一个ANALYSIS的对象。

参数

substep()

analysis()的同名函数，功能跟analysis一致

cmd()

使用方法

```
$step->cmd('-nobr'=>1|0)
```

功能

用来给STEP对象添加命令。

返回值

STEP的所有命令。

参数

- **-nobr:** 1/0，1表示不在命令的末尾加上换行符，缺失值为0（即默认都会在命令的末尾加上换行符）。

all_analysis()

`$step->analysis()` 用来返回该STEP所有的二级分析点，一般情况下用不到。

path

STEP的属性名，调用方法 `$step->{path}` ，保存的是该一级分析点的结果目录。

shelldir

STEP的属性名，调用方法 `$step->{shelldir}` ,保存的是该分析点的二级分析所在的脚本目录，当该分析点没有二级分析点的时候该属性值为空。

模块ANALYSIS

cmd()

使用方法同STEP的cmd()函数，用来给ANALYSIS对象添加命令。

path

ANALYSIS的属性名，调用方法 `$analysis->{path}` ，保存的是该二级分析点的结果目录。

---

1. 目前该模块生成的流程架构最多只支持到2层，step表示第一层，analysis（substep）表示第二层。且第二层不是必须的。 ↩