



廈門大學

本科畢業論文  
(主修專業)

## 服务器 Linux 操作系统学习

Linux system server operation learning

姓 名：郭欣宸

学 号：19020162203201

学 院：数学科学学院

专 业：统计学

年 级：2016

指导教师：周达（副教授）

二〇二〇年 月

## 厦门大学本科学位论文诚信承诺书

本人呈交的学位论文是在导师指导下独立完成的研究结果。本人在论文写作中参考其他个人或集体已经发表的研究成果，均在文中以适当方式明确标明，并符合相关法律法规以及《厦门大学本科毕业论文（设计）规范》。

学生声明（签名）：

年 月 日



## 致 谢

值此论文完成之际，谨向所有关心和支持我的人们致以诚挚的谢意！

首先，我要衷心地感谢我的导师周达教授，从论文选题，内容和整体结构的确定，直至最后定稿，周达老师都以及其负责的态度给与细心指导，为我提出了许多宝贵的意见和建议，使我获益良多。他渊博的学识、严谨的治学态度以及朴实的学术作风时刻激励我不断努力完善自己，对我悉心关怀和教诲也将鼓舞我在今后的学习和工作上不断努力向上。在此，谨向周达老师致以最诚挚的感谢！

在此，还要感谢与我一起完成这个题目的所有同学，没有他们的帮助和共同努力，就没有本文的形成。在此，向他们表示衷心的感谢！



## 摘 要

随着计算机技术的发展,统计学要能更好地应用于实际离不开对计算机技能的掌握。传统的统计学科的学习往往重视理论推导和对统计思维的培养,相对忽视对计算机编程语言和算法结构的掌握。在运行一些具有复杂抽样的迭代算法时,普通计算机显得力不从心,现实要求我们接触具有更强大计算能力得到服务器来满足我们的学习需要。学院在去年引进了一台服务器用于机器学习的研究和探索,一方面是对统计发展的积极迎合,另一方面,掌握好使用服务器的方法将是对同学们解决实际问题、研究复杂问题能力的有力提升。本文的目的在于抛砖引玉,从入门者角度总结一些对服务器使用的建议。

文章主要分为三部分,第一部分了解服务器与其所搭载的 linux 系统,有利于快速建立对 linux 系统的初步印象;第二部分介绍如何使用服务器完成任务,具体分为如何连接服务器,如何上传文件以及提交不同软件编写的任务;第三部分是一个应用实例,简略介绍了 mcmc 算法的原理和两种基本抽样方法,提出一个问题并编写解决问题代码,最后提交到服务器运行并得出结果。第四部分附录是一些放在正文稍显累赘的基本命令介绍和程序脚本。

**关键词:** 服务器操作; linux 系统; mcmc 举例

## **Abstract**

With the development of computer technology, the mastery of computer skills is essential for the better application of statistics in practice. The study of traditional statistical discipline often attaches great importance to theoretical derivation and the cultivation of statistical thinking, and relatively ignores the mastery of computer programming language and algorithm structure. When running some iterative algorithms with complex sampling, ordinary computers seem to be overwhelmed, and the reality requires us to contact the server with more powerful computing power to meet our learning needs. Last year, the college introduced a server for the research and exploration of machine learning. On the one hand, it actively caters to the development of statistics, on the other hand, mastering the method of using the server will be a powerful improvement of students' ability to solve practical problems and study complex problems. The purpose of this article is to introduce some Suggestions for server usage from a beginner's point of view.

The article is mainly divided into three parts. The first part is to understand the server and its Linux system, which is conducive to the know the Linux system roughly. The second part describes how to use the server to complete tasks, specifically divided into how to connect to the server, how to upload files and submit different software written tasks; The third part is an application example. It briefly introduces the principle of MCMC algorithm and two basic sampling methods. It proposes a problem and writes the code to solve the problem. The fourth part of the appendix is a slightly cumbersome introduction to the text of the basic commands and scripts.

**Key words:** server operation; Linux system; MCMC





# 目 录

1 服务器与 Linux 系统 .....	1
1.1 服务器.....	1
1.2 文件系统.....	2
1.3 基本命令.....	3
1.3.1 Linux 命令.....	3
1.3.2 常用的基本命令.....	4
2 如何用服务器完成任务 .....	5
2.1 连接服务器.....	5
2.2 提交任务.....	7
2.2.1 提交 R 代码.....	7
2.2.2 提交 matlab 代码.....	10
2.3 查看任务情况.....	12
3 实例：用服务器计算 MCMC 算法（R 语言） .....	13
3.1 马氏链与 MCMC 原理.....	13
3.1.1 马氏链和 MCMC 方法.....	13
3.1.2 MH 方法的基本步骤.....	16
3.1.3 Gibbs 抽样的基本步骤 .....	17
3.2 运用服务器完成任务的一个 mcmc 实例.....	17
3.3 任务的提交与完成.....	19
4 结论 .....	23
参考文献.....	24
附录.....	25
附录 A.....	25
附录 B（提交 R 代码的 perl 脚本） .....	29

# Content

<b>1. Servers and Linux systems</b> . . . . .	错误!未定义书签。
<b>1.1 Servers</b> .....	错误!未定义书签。
<b>1.2 The file system</b> .....	错误!未定义书签。
<b>1.3 Basic commands</b> .....	错误!未定义书签。
1.3.1 Linux commands.....	错误!未定义书签。
1.3.2 Common basic commands .....	4
<b>2.How to get things done with the server</b> . . . . .	5
<b>2.1 Connect to server</b> .....	5
<b>2.2 Submit a task</b> .....	7
2.2.1 Submit the R code .....	7
2.2.2 Submit matlab code.....	10
<b>2.3 View task status</b> .....	12
<b>3. Example: computing MCMC algorithm with server (R)</b> . . . . .	13
<b>3.1 Markov chain and MCMC principle</b> .....	13
3.1.1 Markov chain and MCMC method .....	13
3.1.2 The basic steps of the MH method .....	16
3.1.3 The basic steps of Gibbs sampling.....	17
<b>3.2 An example of using a server to complete a task</b> .....	17
<b>3.3 Task submission and completion</b> .....	19
<b>4 Conclutions</b> .....	23
<b>References</b> .....	24
<b>The appendix</b> . . . . .	25
<b>A: Common basic commands</b> .....	25
<b>B: The perl script for submitting R code</b> .....	29

# 1 服务器与 Linux 系统

## 1.1 服务器

服务器指一个管理资源并为用户提供服务的计算机设备。从功能上来讲，服务器跟我们日常学习办公使用的台式机和笔记本没有本质区别，主要也是由最为关键的三大件：CPU、内存、硬盘组成，但是服务器的应用场景主要是提供给企业等角色用来支撑形形色色业务，不仅仅是用来安装浏览器访问网页、安装播放器看看电影，因此服务器会使用更强的配置，即更强劲的 CPU、更大的内存、更大的硬盘存储。从外形上来说，服务器存在多种外形，如机架式、刀片式、塔式等等，最主要的区别在于服务器一般不会配置显示器、键盘、鼠标等部件用于近端操作，一般通过 IP 远程连接的方式访问。

学院服务器的配置如下：

硬件：cpu:4\*Intel Xeon Platinum 8270

核：4（物理 cpu 个数）\*26（每颗物理 cpu 的核数）

线程：支持超线程技术，逻辑 cpu 个数 4（物理 cpu 个数）\*26（每颗物理 cpu 的核数）\*2（超线程数）

内存：256G, 不包括交换扇区

硬盘：非系统盘共 6.5T

GPU：3\*Nvidia 2080Ti

软件：

Centos7:linux 基于 Red Hat Enterprise Linux 源代码编译而成的操作系统

安装软件：R, Matlab, C, C++, python3, Perl

Gnome:图形化桌面，不常用，常用命令行桌面

## 1.2 文件系统

了解 linux 的文件系统可以帮助我们认清自己在服务器中的位置，更好地明确自己能进行的操作。

Linux 系统与常用的 windows 系统一个显著区别就在于其文件系统。

在 windows 系统中，点开桌面“计算机”，会看到不同的驱动器盘符：



图 1-1 windows 系统中驱动器盘符

每个驱动器盘符都有自己的根目录结构，这样就形成了多个树并列的结构：

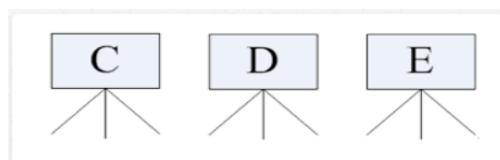


图 1-2 linux 树形文件系统示意

而在 linux 系统中，“一切都是文件”，看不见驱动器盘符，取而代之的是一个树形的文件系统。只有一个根目录“/”，所有文件都在这个根目录下。

在 linux 系统中，存在超级用户（root）和普通用户。对于普通用户来说，每个用户都有一个主目录，通常位于 /home 下。现代 Linux 系统上的 /root（“slash-root”）目录仅是 root 用户（或超级用户或系统管理员帐户）的主目录。在多用户系统上，/home 目录基础结构通常作为单独的文件系统安装在其自己的分区上，可以根据用户的部门或职能对其进行分组。然后，可以在 /home 目录下为每个组创建子目录。

被分配账号后，我们就是学院服务器的一名普通用户。在学院的服务器上，用户的主目录位于 /project1 下，在 /project1 底下又有 /project1/(姓名)，不同用户可以在这个以自己名字命名的文件夹下建立不同的文件夹管理自己的文件。

## 1.3 基本命令

使用 Linux 时，可以使用命令行界面（CLI）或图形用户界面（学院服务器上安装的 Gnome 可以实现这一点）。使用 CLI 工作时，必须记住用于执行任务的程序和命令，以及如何快速准确地获取有关其使用和选项的更多信息。使用 GUI 则快速而简单，它允许通过图形图标和屏幕与系统进行交互。在服务器中，我们使用的是命令行界面，因此必须熟悉一些常用的基本命令。

### 1.3.1 Linux 命令

格式一般为：命令[选项][参数]

其中[选项][参数]不一定每次都要填写

[选项]分为两种：

第一种：短选项

例如：-h、-l、-s。（PS：-后面接单个字母）

- 1) 短选项使用 ‘-’ 引导，当有多个短选项时，各选项之间使用空格隔开
- 2) 有些命令的短选项可以组合，例如：-l-h =>-lh
- 3) 有些命令的短选项可以不带 ‘-’，这种叫做 BSD 风格，例如：ps aux
- 4) 有些短选项需要带选项本身的参数，例如：-L 512M

第二种：长选项

例如：--help、--list 等。（PS：后面接单词）

- 1) 长选项都是完整单词
- 2) 通常不能组合
- 3) 如果需要加参数，长选项的参数通常需要 “=”，例如：--size=1G

[参数]

参数是指命令的作用对象，例如 ls 命令，不加参数显示当前目录，ls 后加上目录参数可以查看该目录。

### 1.3.2 常用的基本命令

参考本文附录。

## 2 如何用服务器完成任务

使用服务器的重要目的就是利用服务器强大的计算功能来为机器学习、深度学习服务。平时我们编写的 R 代码、matlab 代码、python 代码都可以在配置了相应环境的服务器中进行运算。

### 2.1 连接服务器

准备工作：在个人电脑上下载 MobaXterm, 校外登陆时连接 vpn, 服务器帐号密码  
点开 MobaXterm, 点击左上角“session”进入如下页面，点击“SSH”：

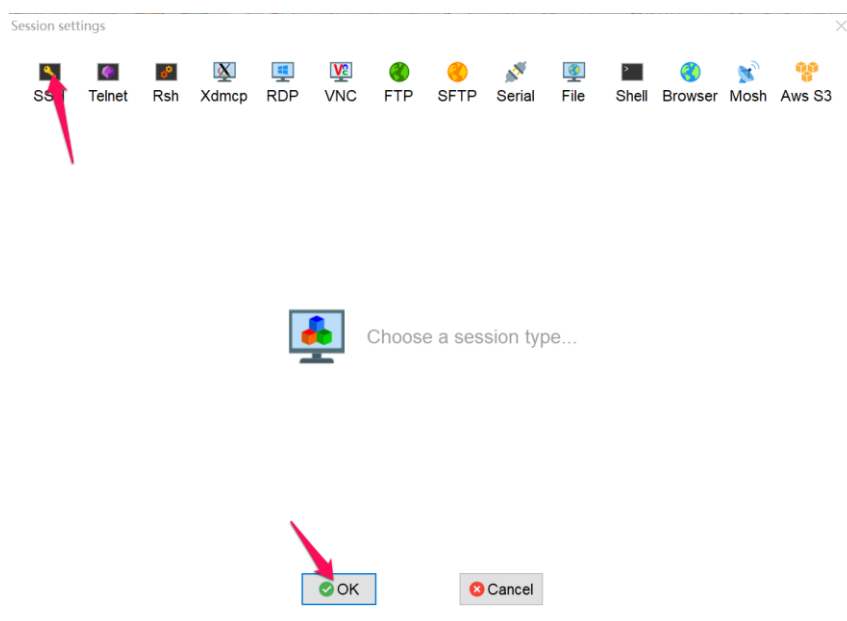


图 2-1 通过 MobaXterm 登陆服务器（1）

进入下列页面，输入服务器地址和自己的用户名：

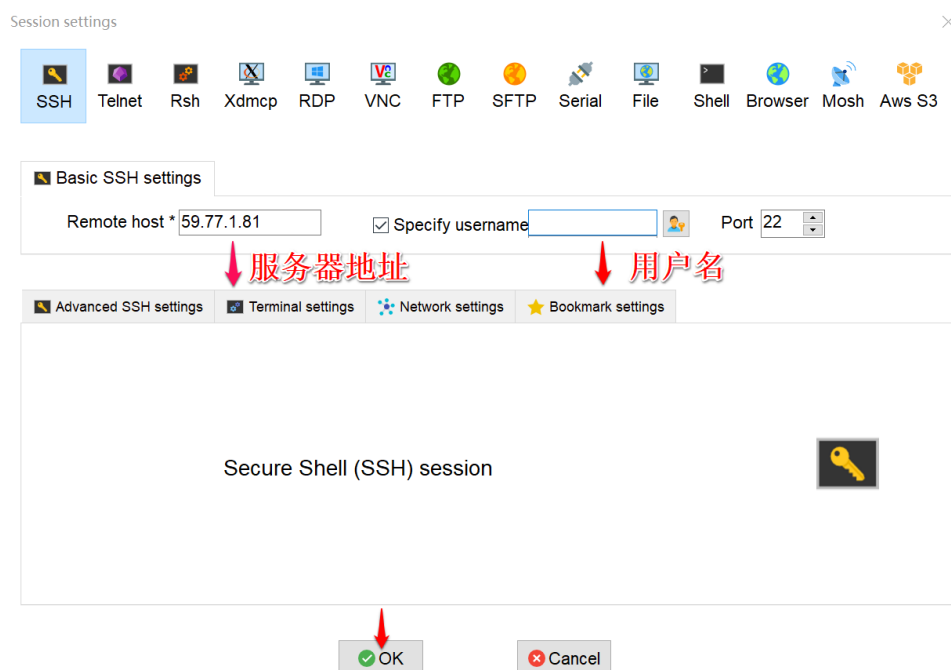


图 2-2 通过 MobaXterm 登陆服务器 (2)

进入服务器页面后，需要输入 password，就是用户的账号密码，输入密码不会显示出来，完成输入后按下回车即可登陆。登陆成功界面如下：

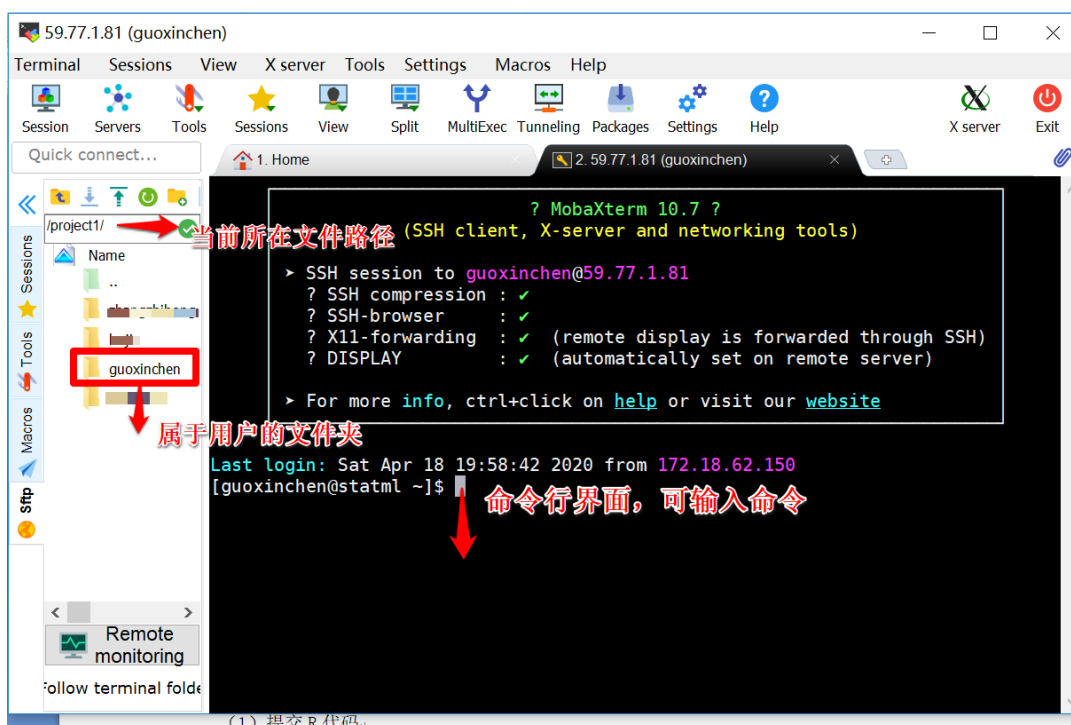


图 2-3 成功登陆服务器后界面展示



点属于自己的用户文件夹，可以直接拖拽本地文件到服务器：

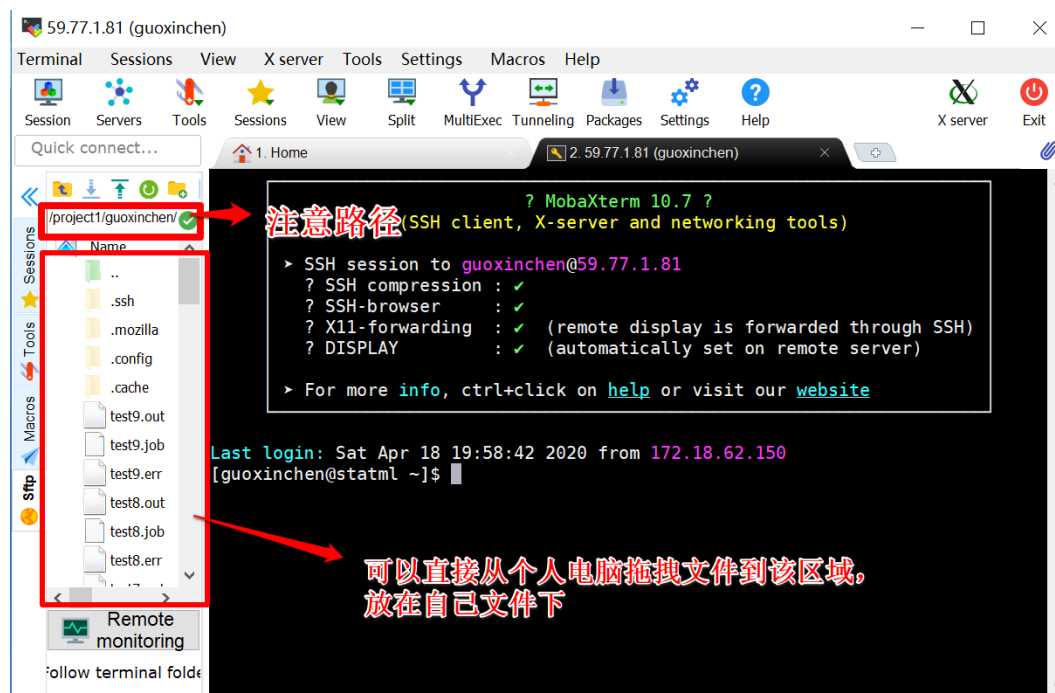


图 2-4 拖拽上传文件示意图

## 2.2 提交任务

### 2.2.1 提交 R 代码

步骤：

用 perl 编写一个 qsub 脚本，脚本文件为 qsub\_Wrapper\_0to9.pl（见附录），将其放到目录/project1/(姓名)下，同样地，直接把 R 文件拖拽到文件夹里：

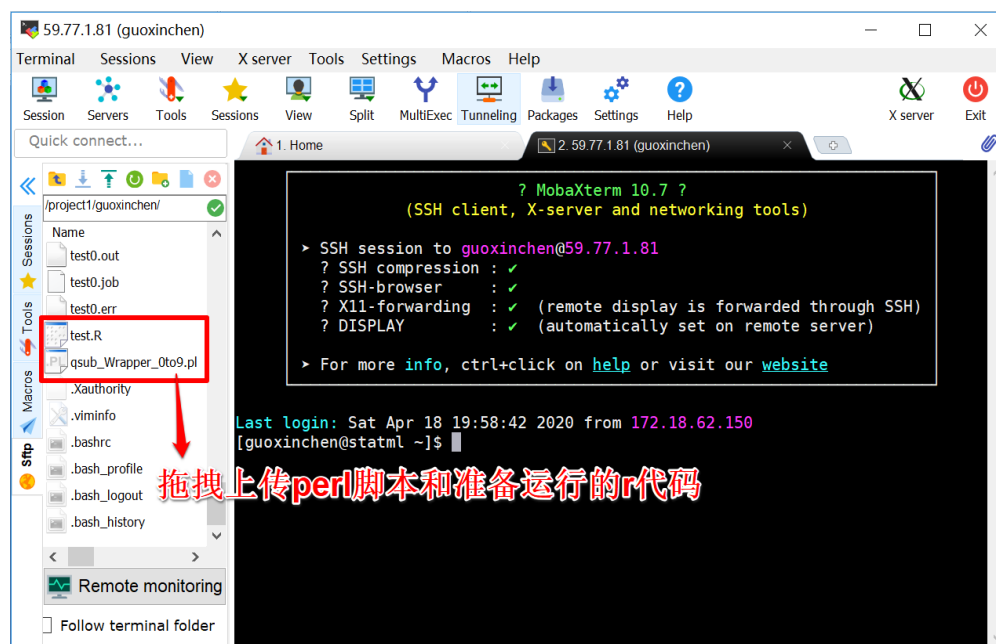


图 2-5 拖拽上传 R 文件示意

注意：此时应该对 R 文件进行一定的修改，增加头文件：

```
args=commandArgs(T);jobB=args[1];jobE=args[2];DirPre[3]
```

循环改为 jobB:jobE

接下来提交 R 代码：

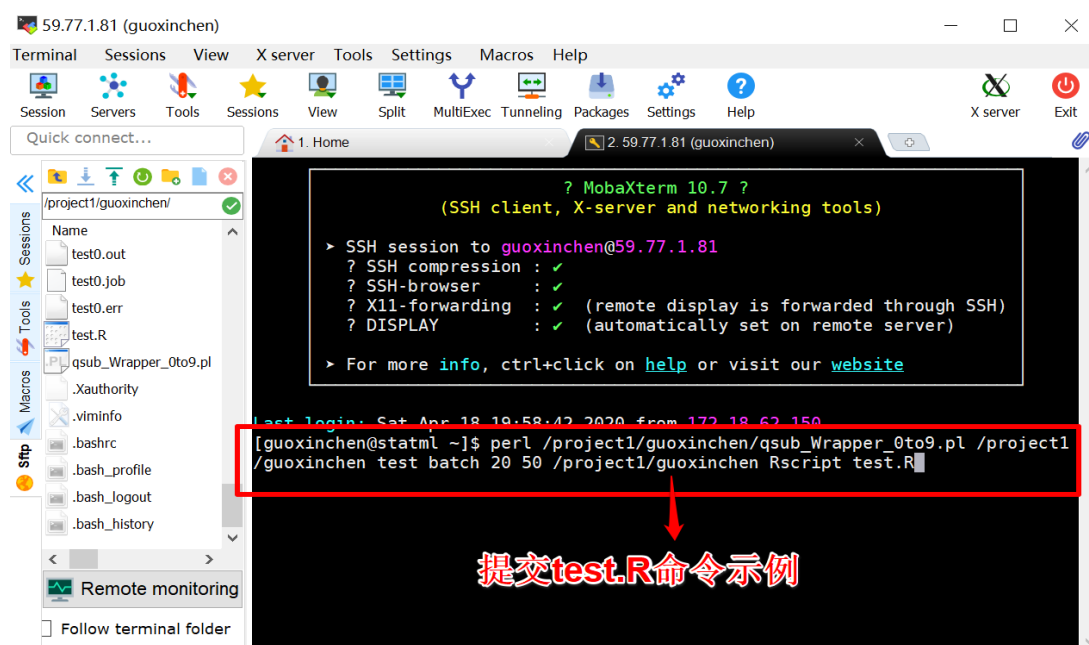


图 2-6 提交 test.R 命令示意

命令解读：从左至右按顺序进行

Perl/project1/guoxinchen/qsub\_Wrapper\_0to9.pl:

perl+qsub\_Wrapper\_0to9.pl 目录

/project1/guoxinchen(任务所在目录) test(任务名称) batch(队列名称)

20(核数) 50(单个核运算次数)

/project1/guoxinchen : 作业运行的工作目录，存放任务运行的中间数据。

Rscript (提交 R 文件固定) test.R(待运行的 R 文件)

解释：

batch 是任务队列名称，可以询问服务器管理员，比如在学院的服务器中就只有 batch 这一个任务队列。如何分配服务器核数和单个核运算数的分配基于你代码中的循环个数。比如循环数为 1000，那么可以分配给 20 个核，每个核运算 50 次。循环次数为 9999 次，可以分配给 11 个核进行，每个核运算 909 次。这个命令的问题在于只能规定循环的总次数，如  $20 \times 50 = 1000$  次，循环默认起点和终点是 1: 1000。如果想要改变循环的起点和终点，不妨在 r 代码中在循环的部分添加一个常数。在执行上述命令的时候尤其需要注意文件路径是否填写正确，可以灵活地使用相对路径，但是在命令执行出错的时候选择绝对路径是一个稳妥的选择。可以自行设定一个目录来存放运行得到的数据。

任务提交成功以后会返回生成的任务编号。

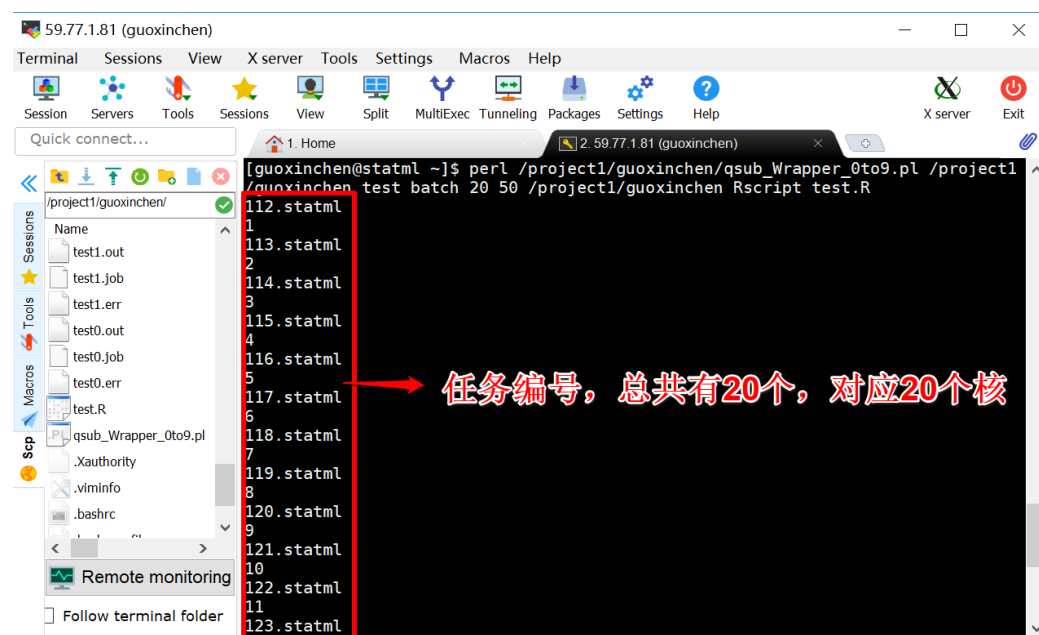


图 2-7 任务提交成功后生成编号

## 2.2.2 提交 matlab 代码

（一）执行多核运算任务—提交嵌套 pbs 脚本的 perl 脚本

要提交 matlab 代码做多核运算，如果仅提交 pbs 脚本，无法有效地控制运算的核数和单个核运算的次数。因此需要把 pbs 脚本嵌套到 perl 脚本中。前文用于提交 r 代码的 perl 脚本就是这个思路，因此，提交对象换成 matlab 文件时，只需要对前文的 perl 脚本进行一点修改。首先对 perl 脚本进行了解。

```
01. #!/usr/bin/perl -w
02. #
03. # qsub_wrapper.pl: this script creates a series job files and submits them to the queue
04. #
05. # perl qsub_wrapper_v1.pl $WorkingDir $JobNamePrefix $QueueName $NumberOfJobs $Range $DataDir $ExeCmd $ExeFile
06. # perl qsub_wrapper_v1.pl $HOME/example jobname es4 10 100 /scratch/prj00004/XXX Rscript XXX.r
07.
08.
09. if(@ARGV!=8)
10. {
11.     print STDERR "\nSyntax:\n ./$0 WorkingDir JobNamePrefix QueueName NumberOfJobs Range DataDir ExeCmd ExeFile\n";
12.     exit 0;
13. }
14.
15. my ($WorkingDir, $JobNamePrefix, $QueueName, $NumberOfJobs, $Range, $DataDir, $ExeCmd, $ExeFile) = @ARGV;
16.
17. my $fpout;
18.
19. my $count;
20. my $JobName;
21. my $JobFileName;
```

图 2-8 提交 r 代码的 perl 脚本（1）

```
22.
23. for($count=1; $count<=$NumberOfJobs; $count=$count+1) {
24.     $Posfix=$count-1;
25.     $JobName=join('', $JobNamePrefix, $Posfix);
26.     $JobFileName=join('', $JobName, ".job");
27.     open($fpout, '>', $JobFileName) || die("Can't open output file $JobFileName: $!");
28.     print $fpout "#!/bin/sh.\n";
29.     print $fpout "#PBS -N $JobName\n";
30.     print $fpout "#PBS -q $QueueName\n";
31.     print $fpout "#PBS -o $JobName.out\n";
32.     print $fpout "#PBS -e $JobName.err\n";
33.     print $fpout "\n";
34.     print $fpout "cd $WorkingDir\n";
35.     print $fpout "\n";
36.     print $fpout join(" ", $ExeCmd, $ExeFile, $Range*($count-1)+1, $Range*$count, $DataDir)."\n";
37.
38.     close($fpout);
39.     #sleep(1);
40.     system("qsub $JobFileName");
41.     print $count."\n";
42. }
```

图 2-9 提交 r 代码的 perl 脚本（2）

```
43.
44. $count=$count-1;
45. print "\nDone for $count lines.\n";
46.
47.
48.
49. #####end of main function#####
```

图 2-10 提交 r 代码的 perl 脚本（3）

9 行-13 行：

规定了提交 perl 脚本时，命令行需要输入 8 个参数。结合图 2-6 分析 8 个参数对应命令行：

\$WorkingDir: /project1/guxinchen，工作目录。

\$JobNamePrefix: test，任务名称前缀

`$QueueName: batch`, 任务队列

`$NumberOfJobs: 20`, 核数

`$Range: 50`, 单核计算次数

`$DataDir: /project1/guoxinchen`, 存放任务运行中产生数据的路径

`$ExeCmd: Rscript`, 执行任务的应用程序

`$ExeFile: test.R`, 待执行的 R 文件

15 行 -21 行:

定义一系列变量。Perl 语言中, 用 `$` 定义简单变量, 用 `@` 定义数组。

23 行-42 行:

通过 for 循环创建并提交 pbs 脚本达到控制多核运算的目的。(需要几个核, 每个核运行几次。

在提交 R 代码时, 观察 perl 脚本中创建 pbs 脚本的命令。

图 2-9 的第 36 行为:

```
print $fpOut join(" ", $ExeCmd, $ExeFile, $Range*($count-1)+1, $Range*($count),  
$DataDir)."\n";
```

`$ExeCmd:Rscrip`, 调用执行命令的应用程序

`$ExeFile: test.R`, 待执行的 R 文件

`$Range*($count-1)+1`: 第 `$count` 个核上循环的起点

`$Range*($count)`: 第 `$count` 个核上循环的终点

`$DataDir` 含义见前。

前面提到, 用 perl 脚本提交 r 文件时需要在 r 文件中加入头文件:

```
Args=commandArgs(T)
```

```
jobB=args[1]
```

```
jobE=args[2]
```

```
PreDir=args[3]
```

并且将循环改为 `jobB:jobE`。

在 R 中，`commandArgs` 函数是用于从命令行读取参数，示例如下：

```

1  ##test.R
2  args=commandArgs(T)
3  print (args[1])##第一个外部参数
4  print (args[2])##第二个外部参数
5  ##运行脚本:Rscript test.R first second
6  结果:
[wangjq@mgmt bismark]$ Rscript test.R first second
[1] "first"
[1] "second"

```

图 2-11 `commandArgs` 函数示范（图源网络）

因此不难理解，在 R 文件中加入头文件以后，文件执行时就会从命令行自动获取参数  $\$Range * (\$count - 1) + 1$ （第一个位置参数 `args[1]` => jobB）和  $\$Range * (\$count)$ （第二个位置参数 `Args[2]` => jobE），这样就达到了把一个循环切割到多个核上运行的目的。

同时，这也提示我们修改 perl 脚本的方向：

（1）参考命令行使用 matlab 的方法

（2）将 perl 脚本的参数  $\$Range * (\$count - 1) + 1$  和  $\$Range * (\$count)$  传递到 matlab 文件中的循环位置。

于是修改该行命令为：

```
print $fpout "/usr/local/bin/matlab -nodesktop -nosplash -r 'rand6($Range*(\$count-1)+1,$Range*(\$count))'\n";
```

图 2-12 修改图 2-9 第 36 行代码

其中，`/usr/local/bin/matlab` 是服务器上 matlab 的安装路径，`-nodesktop` 表示不开 matlab 的操作界面，`-nosplash` 表示不显示启动 matlab 时的 logo 画面，`-r` 表示运行 matlab cmd 命令，后接要运行的 matlab 文件，所运行的文件必须在当前目录。（那么根据脚本，所运行的文件就应当在路径 `$WorkingDir` 下。）

`rand6` 是运行的 matlab 文件的名称（不带后缀.m）。`rand6()` 括号内是 `rand6.m` 内以循环起始点为自变量的函数的自变量赋值。`rand6.m` 文件内容如下：

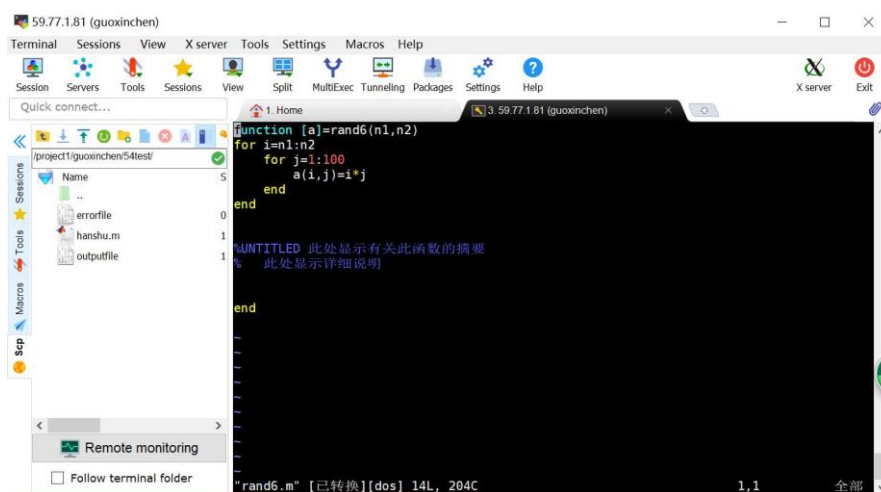


图 2-13 rand6.m

对 rand6.m 的一点说明：这是一个生成 100\*100 矩阵的函数，外层 i 循环代表矩阵的行数，可以拆分在多核计算。

脚本修改完成后，如果所要提交的 matlab 文件中的函数是以循环起点和循环终点为自变量，那么提交前只需要修改“rand6()”中“rand6”的位置为待提交 matlab 文件的名称即可。

循环的次数计算方法、整体平移循环区间的方法同提交 R 代码中的说法一样。

这里给出提交 perl 脚本执行 rand6.m 的命令示例：

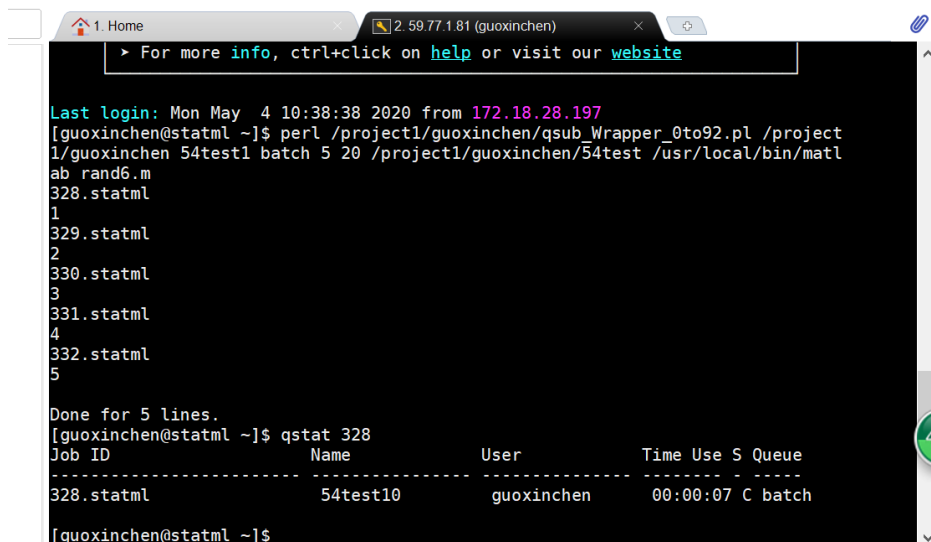


图 2-14 提交命令及生成任务编号

由于 rand6.m 待拆分的外层循环共 100 次，故拆分为 5 核\*20 次/核。

执行结果：

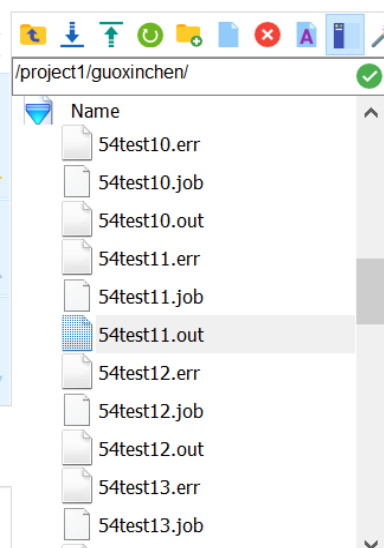


图 2-15 任务完成后输出的文件

（二）不执行多核运算任务—可以仅提交 pbs 脚本

在执行不需要准确掌握单核运行次数时，可以仅用 pbs 脚本提交任务。

此时有两种提交方式。

### 1. 脚本方式提交

用户可以将需要执行的程序或者命令写入脚本中，再加上一些必要或者可选的语句，保存后提交任务脚本，从而达到提交任务的目的。这种方式利于保存，方便更改，是比较推荐使用的方式。脚本编辑可以用 vim 完成。接下来演示如何利用 vim 编辑 pbs 脚本。

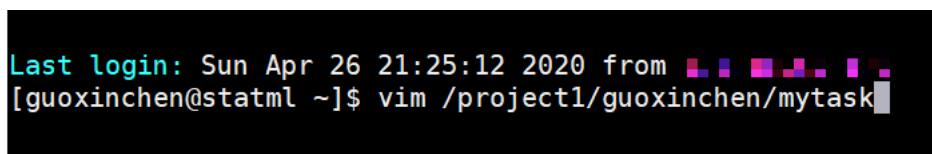


图 2-8 利用 vim 新建一个 pbs 脚本

通常情况下，vim+路径+文件名是打开一个文件，但是在打开的文件还不存在时，就在路径下新建该文件。图 2-8 中的命令就是在路径/project1/guoxinchen/下新建一个名为 mytask 的文件。进入文件界面后，按键盘上的 i 键即可进入编辑模式，此时在屏幕下方会显示出当前的状态。用上下左右键可以移动光标对文件进行编辑。





图 2-9 用 vim 编辑 pbs 脚本

编辑完成以后按 Esc 键退出编辑模式，再输入“:wq”退出并保存编辑的内容。

```
[guoxinchen@statml ~]$ qsub mytask
218.statml
```

图 2-10 提交任务脚本的命令及生成任务编号

生成任务编号即说明任务已经成功提交。

## （二）命令行提交

命令行提交就是键入 qsub 然后回车，把脚本里的内容再输一遍，输完后通过 ctrl+D 提交。

```
Last login: Thu Apr 30 11:07:24 2020 from [redacted]
[guoxinchen@statml ~]$ qsub
#!/bin/sh/
^C
[guoxinchen@statml ~]$ qsub
#!/bin/sh
#PBS -N mytask
#PBS -l nodes=1:ppn=20
#PBS -l walltime=00:10:00
#PBS -o outputfile
#PBS -e errorfile
cd $PBS_O_WORKDIR
/usr/local/bin/matlab<rand3.m
```

键入ctrl+D即可提交

图 2-11 命令行提交任务的方式

注意：提交脚本时，如果用户没有激活服务器安装版本的 matlab，很可能报错。错误为：“matlab is selecting software opened rendering”，此时选择在线激活。用学校邮箱上 mathworks 官网就可以完成注册。软件激活码可参考网址：

<http://genuine.xmu.edu.cn/matlab.html>

## （三）pbs 脚本注释

```
01.  #!/bin/sh
02.  #PBS -N mytask
03.  #PBS -l nodes=1:ppn=20
04.  #PBS -l walltime=00:10:00
05.  #PBS -o outputfile
06.  #PBS -e errorfile
07.  cd $PBS_O_WORKDIR
08.  /usr/local/bin/matlab<rand3.m
```

图 2-12 pbs 脚本内容

第一行：脚本开头

第二行：首先键入#PBS 设置任务属性，-N mytask 表示设定作业名称为 mytask

第三行：-l 作用是设置作业所需要的资源

-l nodes=1(节点数为 1)：ppn=20（20 个核）

第四行: `-l walltime=00:10:00` 表示作业最长运行时间为 10 分钟

第五行: `-o outputfile` 设定作业的标准输出文件路径为  
`/project1/guoxinchen/outfile`

在任务结束后文件 `outputfile` 将会出现在 `/project1/guoxinchen` 下。

要查看结果, 打开 `outputfile` 即可。

当然，这个文件路径是可以更改的，取决于用户如何设定。

第六行：`-e errorfile` 设定作业的标准错误路径文件。设定规则和标准输出文件路径是一样的。在这个文件内可以查看任务的报错信息。

第七行：进入执行命令的工作目录。

第八行：`/usr/local/bin/matlab` 是执行命令的 `matlab` 程序所在的路径。  
`rand3.m` 为待执行的 `matlab` 代码。

## 2.3 管理任务

管理任务的基本命令见附录。

### 3. 实例：用服务器计算 MCMC 算法（R 语言）

#### 3.1 马氏链与 MCMC 原理

##### 3.1.1 马氏链和 MCMC 方法

设  $\{X_t : t=0,1,\dots\}$  为随机变量序列，称为一个随机过程。设  $X_t$  为“系统在时刻  $t$  的状态”。为简单讨论起见，设所有  $X_t$  均取值于有限集合  $S = \{1,2,\dots,m\}$ ，称  $S$  为状态空间。

马氏链：

如果  $\{X_t\}$  满足：

$$P(X_{t+1}=j | X_0=k_0,\dots,X_{t-1}=k_{t-1},X_t=i) = P(X_{t+1}=j | X_t=i) = p_{ij}, t=0,1,\dots,k_0,\dots,k_{t-1},i,j \in S \quad (1)$$

则称  $\{X_t\}$  为马氏链。 $p_{ij}$  为转移概率，矩阵  $P=(p_{ij})_{m \times m}$  为转移概率矩阵。其中

$$\sum_{j=1}^m p_{ij} = 1, i=1,2,\dots,m$$

对马氏链， $P(X_{t+k}=j | X_t=i) = p_{ij}^{(k)}$  也不依赖于  $t$ ，称为步  $k$  步转移概率。如果对任意  $i,j \in S, i \neq j$  都存在  $k \geq 1$  使得  $p_{ij}^{(k)} > 0$  则称为不可约马氏链。不可约马氏链的所有状态是互相连通的，即总能经过若干步后互相转移。对马氏链  $\{X_t\}$  的某个状态  $i$ ，如果存在  $k \geq 0$  使得  $p_{ii}^{(k)} > 0$  并且  $p_{ii}^{(k+1)} > 0$ ，则称是非周期的。如果一个马氏链所有状态都是非周期的，则该马氏链称为非周期的。不可约马氏链只要有一个状态是非周期的则所有状态是非周期的。对只有有限个状态的非周期不可约马氏链有  $\lim_{n \rightarrow \infty} P(X_n=j | X_0=i) = \pi_j, i,j=1,2,\dots,m$ ，其中  $\{\pi_j, j=1,2,\dots,m\}$  为正常数，满足  $\sum_{j=1}^m \pi_j = 1$ ，称为  $\{X_t\}$  的极限分布。 $\{\pi_j\}$  满足方程组：

$$\begin{aligned}\sum_{i=1}^m \pi_i p_{ij} &= \pi_j, j=1, 2, \dots, m \\ \sum_{j=1}^m \pi_j &= 1\end{aligned}\quad (2)$$

称满足 (2) 的分布  $\{\pi_j\}$  为平稳分布或不变分布。对只有有限个状态的非周期不可约马氏链，极限分布和平稳分布存在且为同一分布。

如果允许状态空间  $S$  为可列个元素，比如  $S = \{0, 1, 2, \dots\}$ ，极限分布的条件需要更多的讨论。对状态  $i$ ，如从状态出发总能再返回状态  $i$ ，则称状态是常返的 (recurrent)。状态  $i$  常返可等价地定义为  $P(X_n = i, i.o. | X_0 = i) = 1$ ，其中  $i.o.$  表示事件发生无穷多次。

对常返状态  $i$ ，如果从  $i$  出发首次返回  $i$  的时间的期望有限，称  $i$  是正常返的。对于不可约马氏链，只要存在正常返状态则所有状态都是正常返，这时存在唯一的平稳分布  $\pi$ 。非周期、正常返的不可约马氏链存在极限分布，极限分布就是平稳分布：

$$\lim_{n \rightarrow \infty} p_{ij}^{(n)} = \pi_j, \forall i, j \in S$$

设正常返的不可约马氏链的平稳分布为  $\pi$ ，设  $h(\cdot)$  是状态空间  $S$  上的有界函数，则

$$P\left(\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^n h(X_k) = \sum_{x \in S} \pi_x h(x)\right) = 1 \quad (3)$$

这类似于独立同分布随机变量平均值的强大数律。当  $Y$  服从平稳分布  $\pi$  时，上式中的极限就等于  $Eh(Y)$ 。如果能设计转移概率矩阵满足正常返、不可约性质的马氏链且其平稳分布为  $\pi(\cdot)$ ，则从某个初始分布出发按照转移概率模拟一系列马氏链，由 (3) 可以估计关于  $Y \sim \pi(\cdot)$  的随机变量的函数的期望  $Eh(Y)$ 。

得到有平稳分布的转移概率矩阵的一个充分条件是转移概率满足细致平衡条件。

如果存在  $\{\pi_j, j \in S\}, \pi_j \geq 0, \sum_{j \in S} \pi_j = 1$ ，使得

$$\pi_i p_{ij} = \pi_j p_{ji}, \forall i \neq j,$$

称这样的马氏链为细致平衡的 (detailed balanced)，这时  $\{\pi_j\}$  是  $\{X_t\}$  的平稳分布。事实上，若  $P(X_t = i) = \pi_i, i \in S$ ，则

$$P(X_{t+1} = j) = \sum_i \pi_i p_{ij} = \sum_i \pi_j p_{ji} = \pi_j \sum_i p_{ji} = \pi_j, \forall j \in S$$

只要再验证转移概率矩阵满足不可约性和正常返性就可以利用 (3) 估计服从平稳分布的随机变量  $Y$  的函数的期望  $Eh(Y)$  了。

马氏链的概念可以推广到  $X_t$  的取值集合  $\mathcal{X}$  为  $\mathbf{R}^d$  的区域的情形。如果各  $\{X_t\}$  的有限维分布是连续型的，则 (1) 可以改用条件密度表示，这时的  $\{X_t\}$  按照随机过程论中的习惯应该称作马氏过程，但这里还是叫做马氏链。

如果非周期、正常返的不可约马氏链  $\{X_t\}$  的平稳分布为  $\pi(x)$ ， $x \in \mathcal{X}$ ，则从任意初值出发模拟产生序列  $\{X_t\}$ ，当  $t$  很大时， $\{X_t\}$  的分布就近似服从  $\pi$ ，抛弃开始的一段后的序列可以作为分布的相关的样本，抛弃的一段序列叫做老化期。如果只是为了估计  $Y \sim \pi(\cdot)$  的函数的期望，可以仅要求马氏链为正常返不可约马氏链。设  $Y \sim \pi(\cdot)$ ， $h(y), y \in \mathcal{X}$  是有界函数，为估计  $\theta = Eh(Y)$ ，用

$X_t, t=k+1, \dots, n$  作为  $\pi$  的样本，用估计量  $\hat{\theta} = \frac{1}{n-k} \sum_{t=k+1}^n h(X_t)$  来估计  $\theta$ ，则  $\hat{\theta}$  是  $\theta$  的强相合估计。老化期长度  $k$  可以从  $\hat{\theta}$  的变化图形经验地选取。

以上就是 MCMC 方法（马氏链蒙特卡洛）。MCMC 方法的关键在于如何从第  $t$  时刻转移到第  $t+1$  时刻。好的转移算法应该使得马氏链比较快地收敛到平稳分布，并且不会长时间地停留在取值空间  $\mathcal{X}$  的局部区域内（在目标分布是多峰分布且峰高度差异较大时容易出现这种问题）。

Metropolis-Hasting 方法 (MH 方法) 是一个基本的 MCMC 算法，此算法在每一步试探地进行转移 (如随机游动)，如果转移后能提高状态  $x_t$  在目标分布  $\pi$  中的密度值则接受转移结果，否则以一定的概率决定是转移还是停留不动。

Gibbs 抽样是另外一种常用的 MCMC 方法，此方法轮流延各个坐标轴方向转移，且转移概率由当前状态下用其它坐标预测转移方向坐标的条件分布给出。因为利用了目标分布的条件分布，所以 Gibbs 抽样方法的效率比 MH 方法效率更高。

### 3.1.2 MH 方法的基本步骤

- 1) 从一个使  $f(x^{(0)}) > 0$  的初始位置  $x^{(0)}$  开始
- 2) 对  $X^{(t)} = x^{(t)}$ ，从建议分布  $g(\cdot | x^{(t)})$  中抽取一个建议值  $x^*$  计算 Metropolis-Hasting 比率

$$R(x^{(t)}, x^*) = \frac{f(x^*) / g(x^* | x^{(t)})}{f(x^{(t)}) / g(x^{(t)} | x^*)} = \frac{f(x^*) g(x^{(t)} | x^*)}{f(x^{(t)}) g(x^* | x^{(t)})}$$

- 4) 从  $U(0, 1)$  中抽取  $u$ ，并计算  $X^{(t+1)}$

当  $u \leq R(x^{(t)}, x^*)$  时， $X^{(t+1)} = x^*$

当  $u > R(x^{(t)}, x^*)$  时， $X^{(t+1)} = x^{(t)}$



MH 算法生成的过程具有马氏性。

### 3.1.3 Gibbs 抽样的基本步骤

1) 在  $t=0$  时刻选择初始值  $x^{(0)}$

2) 依次产生

$$X_1^{(t+1)} | \cdot \sim f(x_1 | x_2^{(t)}, \dots, x_p^{(t)})$$

$$X_2^{(t+1)} | \cdot \sim f(x_2 | x_1^{(t+1)}, x_3^{(t)}, \dots, x_p^{(t)})$$

...

$$X_{p-1}^{(t+1)} | \cdot \sim f(x_{p-1} | x_1^{(t+1)}, x_2^{(t+1)}, \dots, x_{p-2}^{(t+1)}, x_p^{(t)})$$

$$X_p^{(t+1)} | \cdot \sim f(x_p | x_1^{(t+1)}, x_2^{(t+1)}, \dots, x_{p-1}^{(t+1)})$$

其中  $\cdot$  表示对其它所有变量的最近更新值取条件

3) 增加  $t$ ，并转到 2)

对  $X$  的所有元素完成第 2) 步称为一个 Gibbs 抽样循环

## 3.2 运用服务器完成任务的一个 mcmc 实例

接下来将提出一个问题应用算法，并介绍怎样将解决问题的代码作为任务提交到服务器做并行计算。

问题：（估计一个混合参数）

一个混合分布为  $\delta N(3, 0.5^2) + (1-\delta)N(8, 0.7^2)$ ， $\delta$  的先验分布为  $Unif(0, 1)$ 。

可以利用 MCMC 技术构造一个平稳分布等于  $\delta$  后验分布的链。样本从  $\delta = 0.6$  的分布中抽取。样本数量为 100。

在本例中，我们使用独立链，用先验分布  $Unif(0, 1) = Beta(1, 1)$  来作为提案密度。此时的 Metropolis-Hastings 比率等于似然比。

写出似然函数：
$$L(\delta | y) = \prod_{i=1}^{100} f(y_i | \delta) = \prod_{i=1}^{100} (\delta \varphi(y_i | 3, 0.5^2) + (1-\delta) \varphi(y_i | 8, 0.7^2))$$

编写 r 程序，完成下述 3 个任务：

1) 输出  $\delta$  的后验均值估计

2) 输出构造的马氏链

3) 设定 1000 个不同种子，即抽取 1000 组不同的样本进行估计

示例代码如下：

```

01. #####估计混合分布参数#####
02. #增加在服务器提交任务所需要的头文件
03. args=commandArgs(T)
04. jobB=args[1]
05. jobE=args[2]
06. DirPre=args[3]
07. #设定工作空间、设定输出文件名filenames
08. setwd('/project1/guoxinchen/test/')
09. filenames<-paste('seed',1:1000,sep='')
10. ##外层for循环改变设定的种子
11. ##内层for循环是一个函数中的迭代步骤，用于生成xt
12. for(j in jobB:jobE){
13.
14.     set.seed(j)
15.     #在给种子下，抽取混合分布样本值
16.     y=rep(0,100)
17.     for(i in 1:100){
18.         u=runif(1)
19.         y[i]=ifelse(u<0.6,rnorm(1,3,0.5),rnorm(1,8,0.7))
20.     }
21.

```

图 3-1 示例代码 part1

```

22. ##独立链，变量为样本值y，链长n.its，预烧值n.burnin
23. mcf1<-function(y, n.its, n.burnin){
24.     x.val = NULL
25.     f = function(x) prod( x*dnorm(y,3,0.5) +
26.         (1-x)*dnorm(y,8,0.7))
27.     g = function(x) dbeta(x,1,1)
28.     R = function(xt,x) f(x)*g(xt)/(f(xt)*g(x))
29.     x.val[1]=rbeta(1,1,1)
30.     for(i in 1:n.its){
31.         xt = x.val[i]
32.         x = rbeta(1,1,1)
33.         p = min(R(xt,x),1)
34.         d = rbinom(1,1,p)
35.         x.val[i+1] = x*d + xt*(1-d)
36.     }
37.     return(x.val[(n.burnin + 1):n.its])
38. }

```

图 3-2 示例代码 part2

```

39. ##设定输出文件
40. #设定后验均值估计输出为名为seedj的txt文件
41. write.table(mean(mcf1(y,10000,500)),paste(filenames[j],'.txt',sep=''))
42. #设定输出马氏链图像为pdf文件
43. pdf(paste(filenames[j],'.pdf',sep=''))
44. plot(mcf1(y,10000,1000),type='c')
45. dev.off()
46. }

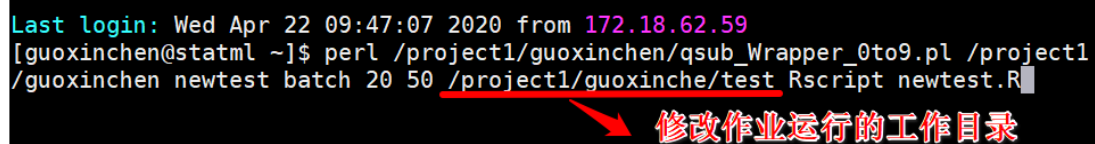
```

图 3-3 示例代码 part3

代码中的一些注意事项：

1) `Setwd('/project1/guoxinchen/test/')` 命令设定了 R 代码输出的 txt 和 pdf 文件存放的目录。相应地，在提交任务命令中也要修改作业运行的工作目录。

示例：



```
Last login: Wed Apr 22 09:47:07 2020 from 172.18.62.59
[guoxinchen@statml ~]$ perl /project1/guoxinchen/qsub_Wrapper_0to9.pl /project1
/guoxinchen newtest batch 20 50 /project1/guoxinchen/test Rscript newtest.R
```

修改作业运行的工作目录

图 3-4 修改工作目录示例

2) `filenames<-paste('seed', 1:1000, sep='')` 命令设定了一个由文件名构成的向量，运行该命令的结果是 (seed1, seed2, ..., seed1000)，配合循环给输出的 txt 和 pdf 文件命名。

3) 前面说到要修改 R 文件，即增加一个头文件，并且修改 for 循环起点和终点为 jobB: jobE。

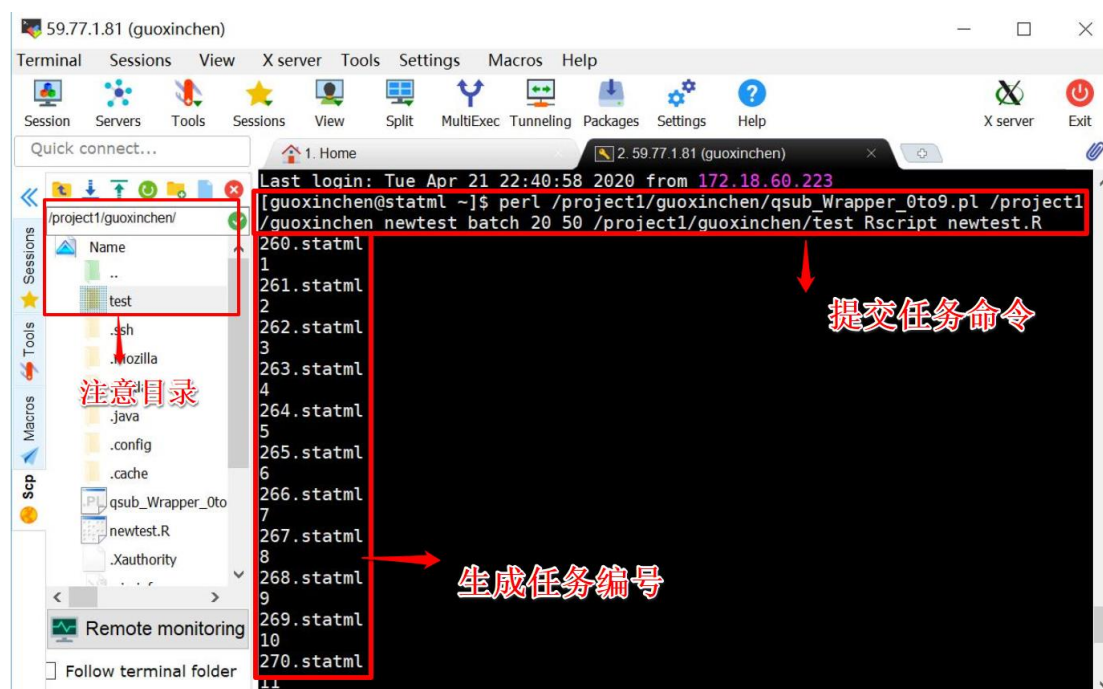
但是在上面的代码中可以看到最外层的 for 循环用来改变种子设定，不同种子下的模拟是可以并行的，即把总共 1000 次的任务分给 20 个核，每个核运行 50 次，提升了计算速度但是不影响计算结果。我们修改的是可以并行的 for 循环的起点和终点。

在 mcf1 函数中还有一个 for 循环是用来迭代计算的，迭代计算不能分开否则迭代会失败，因此 mcf1 中的 for 循环不需要修改起点和终点为 jobB: jobE。

### 3.3 任务的提交与完成

应用前文提到的命令提交任务：

图 3-5 提交任务命令示例



任务完成后，我们可以进入作业运行的工作目录 `/project1/guoxinchen/text` 下查看输出的结果。可以看到该目录下有名为 `seed1~seed1000` 的 `txt` 文件和 `pdf` 文件。在左侧文件栏里面可以直接双击打开文件。例如 `seed1.txt` 中包含了 `set.seed(1)` 下抽取样本得到的  $\delta$  的后验均值估计，`seed1.pdf` 中是该种子下生成的马氏链图像，可以观察该链收敛。

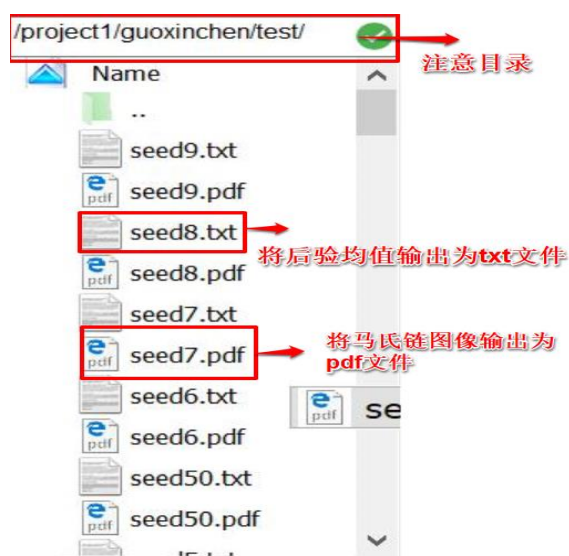


图 3-6 输出的 `txt` 和 `pdf` 文件示例

这里需要注意的是，如果 `r` 文件中设置输出命令的话，后验均值估计和图像就会出现在 `seed1.out` 文件中。如果程序报错，可以在 `seed1.err` 中查看。



## 4 结论

本文从入门的角度介绍了一些服务器的基本知识和操作，目的在于使读者能快速上手服务器。

首先简单浅显地介绍了服务器的基本知识和 Linux 操作系统，力求使读者能快速构建对陌生事物的新概念；接下来从日常应用角度介绍了如何在服务器上提交 R 脚本和 matlab 脚本；最后例举一个 MCMC 算例进行详细说明。

当然本文存在诸多不足，只能介绍一部分比较基础的使用，对于实际应用来说，不同情景下存在不同的操作，因此更多的操作有待读者探索。权当抛砖引玉，希望为读者使用服务器提供一点方便。

熟练地运用服务器要求我们对计算机知识有一定的了解，更重要的是通过实践不断地发现并解决问题，积累经验。要用服务器达到不同的目的，完成不同的任务，势必会遇上不同的问题与障碍，且想要完成的问题越困难、越复杂，操作层面要求也会越高。面对这些障碍时，应当戒骄戒躁，放平心态，多咨询同学老师，多到技术论坛积极寻找解决问题的方法。

## 参考文献

- [1] Michael K. Johnson ,Paul L. Rogers , Eric S. Raymond. Introduction to Linux[M].  
Iuniverse Inc, 2000.4
- [2] Geof\_H\_Givens. 计算统计 第二版[M]. 西安: 西安交通大学出版社, 2017. 12
- [3] Daniel J.Barrett. Linux Pocket Guide[M].O'Reilly Media,2016. 6. 25
- [4] 李东风. 统计计算[M]. 北京: 高等教育出版社, 2017. 3
- [5] CSDN 社区: <https://www.csdn.net/>



## 附录

## 附录 A

(选项不全使用 man 或者 info 命令查找)

别 类	命令	功能	格式
文件操作	ls	ls 命令用于显示文件目录列表, 和 Windows 系统下 DOS 命令 dir 类似。由于 linux 系统“一切皆文件”, 因此该命令十分重要	ls [选项] [目录或文件名]
	选项	-a: 显示所有的文件, 包括隐藏文件 -c: 将文件以文件状态最后改变时间排序 -clt: 显示列表并且以文件状态最后改变时间排序。 --cl:显示文件状态最后改变时间并且以文件名排序。 -d: --directory 的缩写, 仅列出目录本身, 而不列出目录里的内容列表。 -l: 列出长数据串, 显示出文件的属性与权限等数据信息	
	mkdir	在指定位置创建以指定的文件名命名的文件夹或目录。	Mkdir[选项] [指定目录名称]
	选项	-m: 对新建目录设置存取权限 -p: 后加路径名称。此时若路径中的某些目录尚不存在,加上此选项后, 系统将自动建立好那些尚不存在的目录,即一次可以建立多个目录;	
	cd	切换工作目录	cd[相对路径 or 绝对路径 or 特殊符号]
	选项	Cd -: 返回进入此目录之前所在目录 Cd ~:跳转到当前用户的家目录 root 用户, cd ~ 相当于 cd /root 普通用户, cd ~ 相当于 cd /home/当前用户名  相对路径: 指以当前文件所处目录而言文件的位置, 使用灵活 绝对路径: 指对站点的根目录而言某文件的位置, 不易出错 特殊符号: 包括~、-、..等。 ~表示用户主目录, 即 HOME 变量指定的目录, 如 root 用户的主目录为 /root。 -表示前一个工作目录。 ..表示上级目录。 .表示当前目录。	
	pwd	显示当前所在工作目录的全路径	Pwd[选项]
	选项	-L: 显示当前的路径, 有连接文件时, 直接显示连接文件的路径。(不加参数时默认此方式)	

	-p: 显示当前的路径，有连接文件时，不使用连接路径，直接显示连接文件所指向的文件，当包含多层连接文件时，显示连接文件最终指向的文件。	
mv	移动文件或者将文件改名； 当参数 2 是目标文件时，实现对源文件或者目录重命名的功能； 当参数 2 是目录时，把源文件或者目录移动到该目录下。	mv[选项][参数 1: 源文件或者目录][参数 2: 目标文件或者目录]
选项	-b:当覆盖文件之前先行备份。 -f(force 强制):当覆盖文件时，不询问直接覆盖 -i:当覆盖文件之前，会询问是否覆盖 -u :只有当源文件是最新更新时，才可以覆盖目标文件	
touch	创建新的空文件；把已存在的文件的时间标签更新为系统当前的时间（默认方式），它们的数据将原封不动地保留下来。	touch[选项][参数]
选项	-a: 只更改存取时间； -c: 不建立任何文件； -d: 使用指定的日期时间，而非现在的时间；	
cat	查看文件内容，创建文件，文件合并，追加文件内容。	cat [-AbeEnstTuv] [--help] [--version] fileName
选项	-n 或 --number 由 1 开始对所有输出的行数编号 -b 或 --number-nonblank 和 -n 相似，只不过对于空白行不编号 -s 或 --squeeze-blank 当遇到有连续两行以上的空白行，就代换为一行的空白行	
示例	cat -n textfile1 > textfile2 : 把 textfile1 的档案内容加上行号后输入 textfile2 这个档案里； cat -b textfile1 textfile2 >> textfile3 : 把 textfile1 和 textfile2 的档案内容加上行号（空白行不加）之后将内容附加到 textfile3 里 cat file_name: 显示文件所有内容 cat > file_name //当文件不存在时创建文件，存在时覆盖文件；ctrl+c 结束	
file	辨识该文件的类型	file [-bcLvz][-f <名称文件>][-m <魔法数字文件>...][文件或目录...]
选项	-b: 列出辨识结果时，不显示文件名称； -c: 详细显示指令执行过程，便于排错或分析程序执行的情形； -f<名称文件>: 指定名称文件，其内容有一个或多个文件名称时，让 file 依序辨识这些文件，格式为每列一个文件名称； -L: 直接显示符号连接所指向的文件类别； -m<魔法数字文件>: 指定魔法数字文件； -v: 显示版本信息； -z: 尝试去解读压缩文件的内容。	
find	在指定目录下查找文件。任何位	find path -option [-print] [-exec

	<p>于参数之前的字符串都将被视为欲查找的目录名。如果使用该命令时，不设置任何参数，则 <code>find</code> 命令将在当前目录下查找子目录与文件。并且将查找到的子目录和文件全部进行显示。</p>	<p>  <code>-ok command {} \</code>  <code>path</code>:要查找的目录路径  <code>~</code>表示<code>\$home</code>（用户）目录  <code>.</code>表示当前目录  <code>/</code>表示根目录  <code>-option</code> 表示查找方式，见选项  <code>[-print]</code>将结果输出到标准输出  <code>[-exec -ok  xargs  grep ]</code>:  <code>-exec</code>: 对匹配的文件执行该参数所给出的 <code>shell</code> 命令（就是后面的 <code>command {} \</code>；注意<code>{}</code>和<code>\</code>之间有空格）  <code>-ok</code>:作用同<code>-exec</code>，但在执行操作前会询问用户  <code> xargs</code>:作用同<code>-exec</code>，但后面主要跟删除命令，<code>-exec</code> 还可以跟复制、移动等命令</p>
选项	<p><code>-name filename</code> 查找名为 <code>filename</code> 的文件  <code>-perm</code> 按执行权限来查找  <code>-user username</code> 按文件属主来查找  <code>-ctime -n +n</code> 按文件创建时间来查找文件，<code>-n</code> 指 <code>n</code> 天以内，<code>+n</code> 指 <code>n</code> 天以前</p>	
示例	<p><code>find /etc -name 'host*' -print</code> 在 <code>/etc</code> 及其子目录中，查找 <code>host</code> 开头的文件  <code>find . -perm 755 -print</code> 在当前目录及子目录中，查找属主具有读写执行，其他具有读执行权限的文件  <code>find . -name 'del.txt' -ok rm {} \</code> 查找 <code>del.txt</code> 并删除，删除前提示确认</p>	
diff	行对行比较两个文件	<code>diff [参数] [文件 1 或目录 1][文件 2 或目录 2]</code>
选项	<p>- 指定要显示多少行的文本。此参数必须与<code>-c</code>或<code>-u</code>参数一并使用。  <code>-a</code>或<code>--text</code> <code>diff</code> 预设只会逐行比较文本文件。  <code>-b</code>或<code>--ignore-space-change</code> 不检查空格字符的不同。  <code>-B</code>或<code>--ignore-blank-lines</code> 不检查空白行。  <code>-c</code> 显示全部内文，并标出不同之处。  <code>-C</code>或<code>--context</code> 与执行<code>"-c"</code>指令相同。</p>	
du	检查硬盘使用情况，统计文件或目录及子目录使用硬盘的空间大小	<code>du [选项][文件]</code>
选项	<p><code>-a</code> 显示所有目录或文件的大小  <code>-b</code> 以 <code>byte</code> 为单位，显示目录或文件的大小  <code>-c</code> 显示目录或文件的总和  <code>-k</code> 以 <code>KB</code> 为单位输出  <code>-m</code> 以 <code>MB</code> 为单位输出</p>	

	chmod	变更文件与目录的权限	
	cmp	比较两个文件是否有差异	
PBS 作业管理	qstat	查询作业信息状态	qstat[参数][ID]
	选项	-f jobid 列出指定作业的信息 -R 列出磁盘预留信息 -q 列出队列状态，并以 alternative 形式显示 -au userid 列出指定用户的所有作业 -r 列出所有正在运行的作业 -Qf queue 列出指定队列的信息	
	qsub	向系统提交任务 (提交 r 作业没有用到该命令)	
	qdel	用于删除已经提交的作业	qdel [-W 间隔时间] 作业号
	示例	qdel -W 15 211: 15 秒后删除作业号为 211 的作业	
	qmove	把作业从现节点池删除，移到其他节点池	
命令帮助	man	查看系统中自带的各种参考手册	
	info	获取命令的帮助	
系统信息	w	显示目前登入系统的用户信息	
	whoami	显示用户自身名称	
	which	查看可执行文件的位置	
	free	查询剩余可用内存	
	echo	在屏幕上显示文字	
	date	显示和设置系统日期和时间	
	cal	显示日历	
	top	实时显示系统中各个进程的资源占用状况	
读取、筛选	more	基于 vi 编辑器文本过滤器，它以全屏幕的方式按页显示文本文件的内容，支持 vi 中的关键字定位操作	
	less	对文件或其它输出进行分页显示	
	head	显示开头或结尾某个数量的文字区块	
	tail	依照要求将指定的文件的最后部分输出到标准设备，通常是终端，通俗讲来，就是把某个档案文件的最后几行显示到终端上，假设该档案有更新，tail 会自己主动刷新，确保你看到最新的档案内容	
	grep	使用正则表达式搜索文本，并把匹配的行打印出来	
	egrep	在文件内查找指定的字符串	
网络	ssh	远程登陆上 linux 主机	
	scp	在 Linux 下进行远程拷贝文件	
	ping	测试主机之间网络的连通性	

	telnet	登录远程主机，对远程主机进行管理	
	nslookup	查询 DNS 信息	
	wget	从指定的 URL 下载文件	
Shells	clear	清除屏幕	
	history	显示历史指令记录内容，下达历史纪录中的指令	
	set	显示系统中已经存在的 shell 变量，以及设置 shell 变量的新变量值	
	alias	设置指令的别名	
	watch	监测一个命令的运行结果	
文件编辑	nano	一个适合 linux 初学者的字符终端的文本编辑器	
	vim	文本编辑器，较 nano 稍复杂	
	vi		
程序管理	ps		
	kill		
	killall		
	fg		
	bg		

## 附录 B（提交 R 代码的 perl 脚本）

```
#!/usr/bin/perl -w
#
# qsub_Wrapper.pl: this script creates a series job files and submits them
# to the queue
#
# perl qsub_Wrapper_v1.pl $WorkingDir $JobNamePrefix $QueueName
# $NumberOfJobs $Range $DataDir $ExeCmd $ExeFile
# perl qsub_Wrapper_v1.pl $HOME/example jobname es4 10
# 100 /scratch/prj00004/XXX Rscript XXX.r

if(@ARGV!=8)
{
    print STDERR "\nSyntax:\n ./$0 WorkingDir JobNamePrefix QueueName
NumberOfJobs Range DataDir ExeCmd Exefile\n";
    exit 0;
}
my ($WorkingDir, $JobNamePrefix, $QueueName, $NumberOfJobs, $Range,
    $DataDir, $ExeCmd, $ExeFile) = @ARGV;
```

---

```
my $fpOut;

my $count;
my $JobName;
my $JobFileName;
for($count=1; $count<=$NumberOfJobs; $count=$count+1) {
    $Posfix=$count-1;
    $JobName=join(' ', $JobNamePrefix, $Posfix);
    $JobFileName=join(' ', $JobName, ". job");
    open($fpOut, '>', $JobFileName) || die("Can't open output file
$JobFileName: $!");
    print $fpOut '#!/bin/sh'. "\n";
    print $fpOut "#PBS -N $JobName\n";
    print $fpOut "#PBS -q $QueueName\n";
    print $fpOut "#PBS -o $JobName.out\n";
    print $fpOut "#PBS -e $JobName.err\n";
    print $fpOut "\n";
    print $fpOut "cd $WorkingDir\n";
```

---

```
    print $fpOut "\n";
    print $fpOut join(" ", $ExeCmd, $ExeFile, $Range*($count-1)+1,
$Range*($count), $DataDir). "\n";

    close($fpOut);
    #sleep(1);
    system("qsub $JobFileName");
    print $count. "\n";
}
$count=$count-1;
print "\nDone for $count lines.\n";
```

```
#####end of main function#####
```