STAT243 Lecture 3.7 Examples and Challenges

1 Bash Shell Examples

1.1 示例一：统计不同州的气象站数量

目标：从压缩文件 `coop.txt.gz` 中分析每个州的气象站数。

核心命令：

Shell

```
cd fall-2025/data
gzip -cd coop.txt.gz | less        # 查看压缩文件内容
cut -b60-61 coop.txt | sort | uniq -c  # 提取第60-61列（州代码）并统计频数
```

说明：

- `gzip -cd`：解压并输出到标准输出。
- `cut -b60-61`：按字节位置提取州代码字段。
- `sort | uniq -c`：统计各州出现次数。
- 也可以直接在一行中完成：

  Shell

  ```
  gzip -cd coop.txt.gz | cut -b60-61 | sort | uniq -c
  ```

要点：这是快速查看文本数据结构和分布的命令行替代方案，无需用 Python/R 加载数据。

1.2 示例二：程序化计算 CSV 文件字段数

目标：自动计算 CSV 文件中字段的数量。

核心命令：

Shell

```
tail -n 1 cpds.csv | grep -o ',' | wc -l
nfields=$(tail -n 1 cpds.csv | grep -o ',' | wc -l)
nfields=$((${nfields}+1))
echo $nfields
```

说明：

- `tail -n 1`：取文件最后一行。
- `grep -o ','`：提取所有逗号。
- `wc -l`：统计逗号数。
- 由于字段数 = 逗号数 + 1，因此再加 1。
- 也可以使用 `bc` 计算：

  Shell

  ```
  nfields=$(echo "${nfields}+1" | bc)
  ```

延伸：可以写成函数，检查所有行字段数是否一致。

1.3 示例三：查找最近修改的 Quarto 文件中是否使用 `requests` 包

目标：判断 `requests` 是否出现在最近 5 个 `.qmd` 文件中。

核心命令：

Shell

```
cd ../units
ls -tr *.qmd | tail -n 5 | xargs grep -l 'import requests'
```

说明：

- `ls -tr`：按修改时间排序。
- `tail -n 5`：取最近的 5 个文件。
- `xargs grep -l`：在这些文件中搜索包含 `import requests` 的文件。
- `xargs` 将标准输入（stdin）转换为命令参数（arguments）。

替代写法（命令替换）：

Shell

```
grep -l 'import requests' $(ls -tr *.qmd | tail -n 5)
```

要点：展示了命令间信息传递的三种方式：

- `|` 管道（stdout → stdin）
- `$()` 命令替换
- `>` 文件重定向

1.4 示例四：移动最近下载的 n 个文件

目标：编写函数，把最近下载的 n 个文件移动到指定目录。

核心命令：

Shell

```
function mvlast() {
    for ((i=1; i<=${1}; i++)); do
        mv "/accounts/vis/paciorek/Downloads/$(ls -rt \
           /accounts/vis/paciorek/Downloads | tail -n 1)" ${2}
    done
}
```

说明：

- `${1}`：第一个参数，文件数。
- `${2}`：第二个参数，目标目录。
- `ls -rt`：按时间排序（最旧→最新）。
- `tail -n 1`：取最近一个文件。
- 引号确保文件名中有空格时不出错。

示例用法：

Shell

```
mvlast 3 ~/Desktop
```

1.5 示例五：自动提取所有 qmd 文件中的 Python 包并安装

目标：自动找出所有 `.qmd` 文件中导入的 Python 包并生成 `requirements.txt`。

核心命令：

Shell

```shell
grep --no-filename "^import " *.qmd | cut -d'#' -f1 | \
    sed "s/as .*//" | sed "s/import //" > tmp.txt
sed "s/,/\n/g" tmp.txt | sed "s/ //g" | sort | uniq | tee requirements.txt
pip install -r requirements.txt
```

说明：

- `grep "^import "`：提取所有以 `import` 开头的行。
- `cut -d'#' -f1`：删除注释。
- `sed`：去掉 `as` 别名并只保留包名。
- `tee`：同时输出到文件和终端。
- 最终生成 `requirements.txt` 并安装依赖。

要点：展示如何用 shell 快速自动化文本分析与环境管理任务。

1.6 示例六：批量终止后台 Python 任务

目标：若误启动多个 Python 进程，批量终止。

核心命令：

Shell

```shell
nJobs=30
for (( i=1; i<=${nJobs}; i++ )); do
   python job.py > job-${i}.out &
done

ps -o pid --sort=start_time -C python | tail -n ${nJobs} | xargs kill
```

说明：

- `&`：后台运行。
- `ps -o pid -C python`：列出 Python 进程 ID。
- `tail -n`：取最近的几个。
- `xargs kill`：逐个杀死进程。
- Mac 上略有不同，需用 `grep python` 获取 PID。

要点：展示如何通过管道与 `xargs` 结合批量操作进程。

2 Bash Shell Challenges

2.1 First Challenge — 统计单词出现次数

目标：统计文件中某个单词出现的次数，并打印为完整句子。

Shell

```shell
# 基本版（以 Belgium 为例）
```

```
count=$(grep -o "Belgium" cpds.csv | wc -l)
echo "There are ${count} occurrences of the word 'Belgium' in this file."
```

2.2 Second Challenge — 检查字段是否为数字

目标:

1. 找出第 4 列的唯一值。
2. 检查这些值中是否有非数字。

Shell

```
cut -d',' -f4 RTADataSub.csv | sort | uniq > uniq_field4.txt
grep -E '[^0-9]' uniq_field4.txt
```

或只使用一行代码:

Shell

```
cut -d',' -f4 RTADataSub.csv | sort | uniq | grep -E '[^0-9]'
```

2.3 Third Challenge — 各国最低失业率

目标:

1. 找出 Belgium 的最小失业率（第 6 列）。
2. 自动计算所有国家的最小值并输出。

Shell

```
# Belgium
grep "Belgium" cpds.csv | cut -d',' -f6 | sort -n | head -1
# 输出示例：6.2
```

> ⚠ **Remark**
>
> `sort` 后必须加上 `-n`，表示按照数值大小排序，否则默认会按照字符串顺序排序

完整自动化版本

Shell

```
countries=$(cut -d',' -f1 cpds.csv | tr -d '"' | sort | uniq)
for c in $countries; do
    minval=$(grep "$c" cpds.csv | cut -d',' -f6 | sort -n | head -1)
    echo "$c $minval"
done
```

2.4 Fourth Challenge — 删除含缺失值的行

目标：创建一个不含缺失符号 “x” 的新文件。

Shell

```
```

```
grep -v "x" RTADataSub.csv > RTADataSub_clean.csv
```

扩展：写成函数，可统计删除行数并可指定缺失符号

Shell

```
function clean_missing() {
    local missing=$1
    local infile=$2
    local removed=$(grep "${missing}" "${infile}" | wc -l)
    echo "${removed} rows removed."
    grep -v "${missing}" "${infile}"
}

# 用法示例（输出可用于管道）：
clean_missing x RTADataSub.csv > clean.csv
```

2.5 Fifth Challenge — 自动提取州字段位置

目标：利用 `grep` 找到州字段在 coop.txt 中的起始字节位置。

Shell

```
grep -b "CA US" coop.txt | head -1
```

输出示例：

```
59:CA US
```

→ 州字段从第 60 字节开始。

自动提取位置并用于 cut

Shell

```
offset=$(grep -b "CA US" coop.txt | head -1 | cut -d':' -f1)
start=$((offset + 1))
end=$((start + 1))
cut -b${start}-${end} coop.txt | sort | uniq -c
```

2.6 Sixth Challenge — 替换分隔符以避免嵌套逗号问题

目标：将含双引号内逗号的 CSV 改成用其他分隔符（如 `|`）的文件。

示例输入：

```
1,"America, United States of",45,96.1,"continental, coastal"
2,"France",33,807.1,"continental, coastal"
```

解决方案（简单替换）

Shell

```
sed 's/","/"|"/g' input.csv | sed 's/,/|/g' | sed 's/|"/","/g' > output.csv
```

输出示例：

```
1,"America - United States of",45,96.1,"continental - coastal"
2,"France",33,807.1,"continental - coastal"
```

3 Assignment 1 题目

3.1 Question 1 — 获取 Python 路径

Shell

```shell
mypython=$(which python)
```

3.2 Question 2 — 用户名 + 主机名

Shell

```shell
username_machinename=$USER@$(hostname)
```

3.3 Question 3 — 一行创建复杂目录结构

Shell

```shell
mkdir -p temp/proj{1,2,3}/{code,data}
```

3.4 Question 4 — 统计文件行数

Shell

```shell
wc -l < data.txt          # 常规情况
grep -c "" data.txt       # 若最后一行缺少换行符
```

3.5 Question 5 — 打印前 3 行与第 3 行

Shell

```shell
head -3 FILENAME
head -3 FILENAME | tail -1
```

3.6 Question 6 — 将第 3 行写入新文件

Shell

```shell
head -3 FILENAME | tail -1 > NEWFILENAME
```

3.7 Question 7 — 追加第 5 行到同一文件

```shell
head -5 FILENAME | tail -1 >> NEWFILENAME
```

3.8 Question 8 — 提取 Australia 数据

```shell
grep "Australia" cpds.csv > cpds_australia.csv
# 或更严格匹配：
grep -E '^[^,]*,"Australia",' cpds.csv > cpds_australia.csv
```

3.9 Question 9 — 查找不含逗号的行

```shell
grep -v ',' FILENAME
# 或：
grep -E '^[^,]*$' FILENAME
```

3.10 Question 10 — 批量创建文件

```shell
for i in {1..N}; do
    echo "blah" > "file${i}.txt"
done
```

4 Assignment 2 题目

4.1 Question 1 — 匹配任意大小写形式的 “dog”

```shell
grep -E "[Dd][Oo][Gg]" FILENAME     # 正则匹配
grep -i "dog" FILENAME              # 忽略大小写
```

4.2 Question 2 — 匹配 “cat”、"caat"、"caaat" 等

```shell
grep -E "ca+t" FILENAME
grep -E "caa*t" FILENAME
grep -E "ca{1,}t" FILENAME
```

4.3 Question 3 — 匹配 “cat”、"at"、"t"

```shell
```

```shell
grep -E "c?a?t" FILENAME     # 允许可选的 c 和 a
grep -E "cat|at|t" FILENAME  # 明确枚举匹配
```

4.4 Question 4 — 匹配由任意空白分隔的两个单词

Shell

```shell
grep -E "[[:alnum:]]+[[:space:]]+[[:alnum:]]+" FILENAME
grep -E "[A-Za-z]+[[:space:]]+[A-Za-z]+" FILENAME   # 仅限英文字母
```