

Unixによるテキストファイル処理

作業場所

- 以降の作業は、以下のディレクトリで行います。

```
~/data/5_text/
```

cd コマンドを用いてディレクトリを移動し、

pwd コマンドを利用して、カレントディレクトリが上記になっていることを確認してください。

実習で使用するデータ

- 講習で使用するデータは以下のフォルダ内。
- ファイルがあることを確認してください。

```
~/data/5_text/
```

(確認するためのコマンド)

```
$ ls 5_text/
```

主なファイルの内容

batter.txt	2014年セリーグ打撃成績上位5名
ecoli.sam	マッピング結果ファイル
ecoli.gtf	アノテーションテーブルファイル
ecoli.htseq	アノテーションテーブルを使用して得られたリード数

コマンド復習

- `wc` [ファイル名]
 - ファイルの行数、単語数、文字数を出力する
- `head` [-行数] [ファイル名]
 - ファイルの先頭から指定した行数(指定しないと10行)を出力する
- `tail` [-行数] [ファイル名]
 - ファイルの最後から指定した行数(指定しないと10行)を出力する
- `less` ファイル名
 - ファイルの内容を閲覧する

本講で扱うテキスト処理コマンド

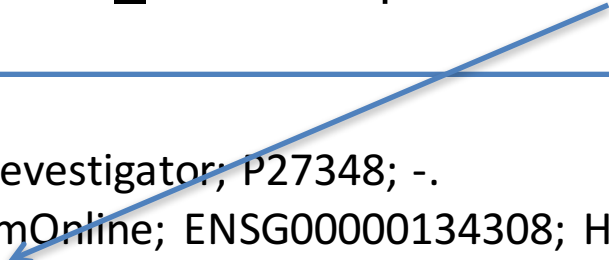
- `grep` 正規表現パターンの検索
- `sed` 文字列置換等によるファイルの変換
- `sort` ファイルのソート
- `awk` 様々なテキストファイル処理

正規表現による文字列検索 (grep)


- `grep 'パターン' [ファイル名 ...]`
 - ファイル中でパターンを含む行を出力する

例) `grep 'GO' 1433T_HUMAN.sprot`

- 1433B_HUMAN.sprot から GO を含む行を検索する。



```
...  
DR  Genevestigator; P27348; -.  
DR  GermOnline; ENSG00000134308; Homo sapiens.  
DR  GO GO:0005813; C:centrosome; IDA:HPA.  
DR  GO; GO:0005634; C:nucleus; IDA:HPA.  
DR  Bgee; P27348; -.  
....
```



```
DR  GO; GO:0005813; C:centrosome; IDA:HPA.  
DR  GO; GO:0005634; C:nucleus; IDA:HPA.
```

正規表現による文字列検索 (grep)

- `grep 'パターン' [ファイル名 ...]`
 - ファイル中でパターンを含む行を出力する
- 例) `grep 'GO' 1433T_HUMAN.sprot`
 - 1433B_HUMAN.sprot から GO を含む行を検索する。
- 例) `grep '^FT' 1433T_HUMAN.sprot`
 - 1433B_HUMAN.sprot から FT で始まる行を検索する。
- `grep -v` パターンを含まない行を出力する。
- `grep -i` 大文字小文字を区別しない。
- `grep -w` パターンを単語としてマッチ
- ファイル名は複数指定可能
- ファイル名を省略すると、標準入力から文字列を読み込んでパターンを検索する

正規表現

grepは「正規表現」によってパターンを指定し、照合したい文字列集合を規定する

- 通常の文字列はそのまま表現される
 - 例) `File1` (`File1`にマッチ)
- 特殊な意味を持つ文字(メタキャラクタ)によって規則を表現
 - 例) `[]`は文字集合を規定する
`File[1-3]` (`File1`, `File2`, `File3` のいずれにもマッチ)
- `\`によってメタキャラクタの特殊な意味を打ち消せる
 - 例) `\[abc\]` (`[abc]` という文字列にマッチ)

注意) 正規表現にはシェルのメタキャラクタが含まれるので、そのままコマンドラインで指定すると思わぬエラーになることが多い。そこで、パターンは `'` で囲むようにする。

正規表現(一部)

- **. (ドット) 任意の1文字**
 - 例) a.c
abc, adc など、aとcの間に任意の1文字を含む文字列にマッチ
- **[] (角形括弧) 文字の集合**
 - 例) [ad3@]
a, d, 3, @のいずれにもマッチ
 - 例) [a-d]
a, b, c, dのいずれにもマッチ
 - 例) [^abd]
a, b, d 以外のいずれにもマッチ
- **^ 行の先頭 \$ 行の終端**
 - 例) ^ID
行の先頭がIDである行とマッチ
- *** 0回以上の繰り返し**
 - 例) a.*m
aとmの間に任意の文字列を含む (am, arm, alarm, am am など)

演習 (grep)

- `ecoli.sam` ファイルから、`grep` (`egrep`) コマンドを用いてヘッダ行(行頭に`@`を含む)を表示せよ。

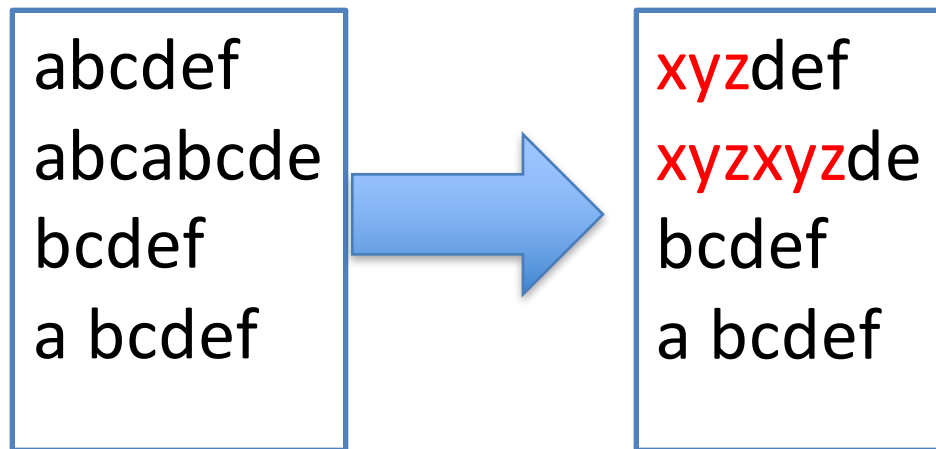
解答例

```
grep '^@' ecoli.sam
```

文字の置換など (sed)

```
sed 's/置換対象パターン/置換文字列/g' [ファイル名]
```

- ファイル中の、指定した正規表現パターンに合致するすべての文字列を、指定した置換文字列で置き換える。置換文字列が空の場合は文字列の削除になる。
 - 例) `sed 's/abc/xyz/g' file`
file中の文字列abcをすべてxyzに置き換える。



文字の置換など (sed)

```
sed 's/置換対象パターン/置換文字列/g' [ファイル名]
```

- ファイル中の、指定した正規表現パターンに合致するすべての文字列を、指定した置換文字列で置き換える。置換文字列が空の場合は文字列の削除になる。
 - 例) `sed 's/abc/xyz/g' file`
file中の文字列abcをすべてxyzに置き換える。
- 最後のgをつけない場合は、各行で最初にマッチしたパターンのみが置換される。
 - 例) `sed 's/:/ /' file`
各行で最初に出現した `:` をスペースに置き換える

sedコマンドにも一般にシェルのメタキャラクタが含まれるので、パターンの指定は常に'で囲むようにする。

文字の置換など (sed)

通常、wcを使ってファイルの行数を出すと、ファイル名まで出力されてしまう。行数だけ取り出してみよう。

```
$ wc -l ecoli.gtf | sed 's/ .*//g'
```

wcの出力結果から、スペース以降の文字を削除(=何も無いものに置換)している。


```
$ wc -l ecoli.gtf
```

演習) この2つのコマンドをそれぞれ実行し、違いを見よ。

行の並べかえ(sort)

- `sort [オプション] [ファイル名...]`
 - ファイルを行単位で並べかえる。
 - `-k FLD1,FLD2` ソートのキーを、スペース文字で区切られたフィールド単位で指定できる(FLD1開始フィールド、FLD2終了フィールド)。
 - `-k 2,2` —第2フィールドをキーとしてソート


Murton	T	.338	14	84
Kikuchi	C	.325	11	58
Yamada	S	.324	29	89
Ooshima	D	.318	2	28
Luna	D	.317	17	73



Kikuchi	<u>C</u>	.325	11	58
Luna	<u>D</u>	.317	17	73
Ooshima	<u>D</u>	.318	2	28
Yamada	<u>S</u>	.324	29	89
Murton	<u>T</u>	.338	14	84

- `-k 2,2 -k 3,3nr` —第2フィールドを1番目のキーとし、第3フィールドを2番目のキーとして、数値として(n)逆順(大きい順)で(r)ソート

Murton	T	.338	14	84
Kikuchi	C	.325	11	58
Yamada	S	.324	29	89
Ooshima	D	.318	2	28
Luna	D	.317	17	73



Kikuchi	<u>C</u>	<u>.325</u>	11	58
Ooshima	<u>D</u>	<u>.318</u>	2	28
Luna	<u>D</u>	<u>.317</u>	17	73
Yamada	<u>S</u>	<u>.324</u>	29	89
Murton	<u>T</u>	<u>.338</u>	14	84

演習 (sort)

- `ecoli.htseq` を使い、リード数をカウントされた回数が多いものから20個を表示せよ。
また、少ないものから20個を表示するにはどうすればよいか？

解答例

```
sort -k 2,2nr ecoli.htseq | head -20
```

少ないものから20個の場合

```
sort -k 2,2n ecoli.htseq | head -20
```

テキストファイルの処理 (awk)

awk 'コマンド' ファイル

- テキストファイル进行处理する多機能なコマンド
- コマンドの一般形式は
パターン {アクション}
パターンに指定した条件に合致した行について、アクションで指定した操作を行う。パターンを省略するとすべての行が対象になる。
- タブ区切りテキストなどテーブル形式のファイルでは、\$1, \$2, ... によって各フィールド(カラム)の値を参照できる。

テーブルデータの処理 (awk)

- テーブルカラムの抽出

```
awk ' {print $3,$4,$5} ' datafile
```

- 3,4,5カラム目を出力
- パターンが指定されていないのですべての行が出力される。

ecoli.gtf

```
...  
chr  eschColi_K12_refSeq  stop_codon  253  255 ...  
chr  eschColi_K12_refSeq  exon        190  255 ...  
chr  eschColi_K12_refSeq  start_codon 337  339 ...  
chr  eschColi_K12_refSeq  CDS         337  2796 ...
```



```
# awk ' {print $3,$4,$5} ' ecoli.gtf  
...  
stop_codon  253  255  
exon        190  255  
start_codon 337  339  
CDS         337  2796
```

テーブルデータの処理 (awk)

- 条件を指定したフィルタリング

```
awk ' $4<200 {print}' datafile
```

- 4カラム目が200未満の行を出力
- 出力フィールドが指定されていないので行全体を出力

ecoli.gtf

```
...  
chr  eschColi_K12_refSeq  stop_codon  253  255 ...  
chr  eschColi_K12_refSeq  exon        190  255 ...  
chr  eschColi_K12_refSeq  start_codon 337  339 ...  
chr  eschColi_K12_refSeq  CDS         337  2796 ...
```



```
# awk ' $4<200 {print}' ecoli.gtf  
chr  eschColi_K12_refSeq  start_codon 190 192 ...  
chr  eschColi_K12_refSeq  CDS         190 252 ...  
chr  eschColi_K12_refSeq  exon        190 255 ...
```

テーブルデータの処理(awk)

- テーブルカラムの抽出

```
awk ' {print $1,$2,$5} ' datafile
```

- 1,2,5カラム目を出力
- パターンが指定されていないのですべての行が出力される。

- 条件を指定したフィルタリング

```
awk '$3<200 {print}' datafile
```

- 3カラム目が200未満の行を出力
- 出力フィールドが指定されていないので行全体を出力

- 複数の条件の指定

```
awk '$2~/target/ && $3<200{print}' datafile
```

- 2カラム目にtargetを含み、3カラム目が200以下の行を出力
- 変数~/パターン/ は正規表現の照合

awk コマンドにも一般にシェルのメタキャラクタが含まれるので、常に''で囲むようにすると良い

算術計算(awk)

- 合計値の出力

```
awk '{sum=sum+$2} END{print sum}' ecoli.htseq
```

b0001	11
b0002	117
b0003	33
b0004	44
b0005	3
b0006	14
...	

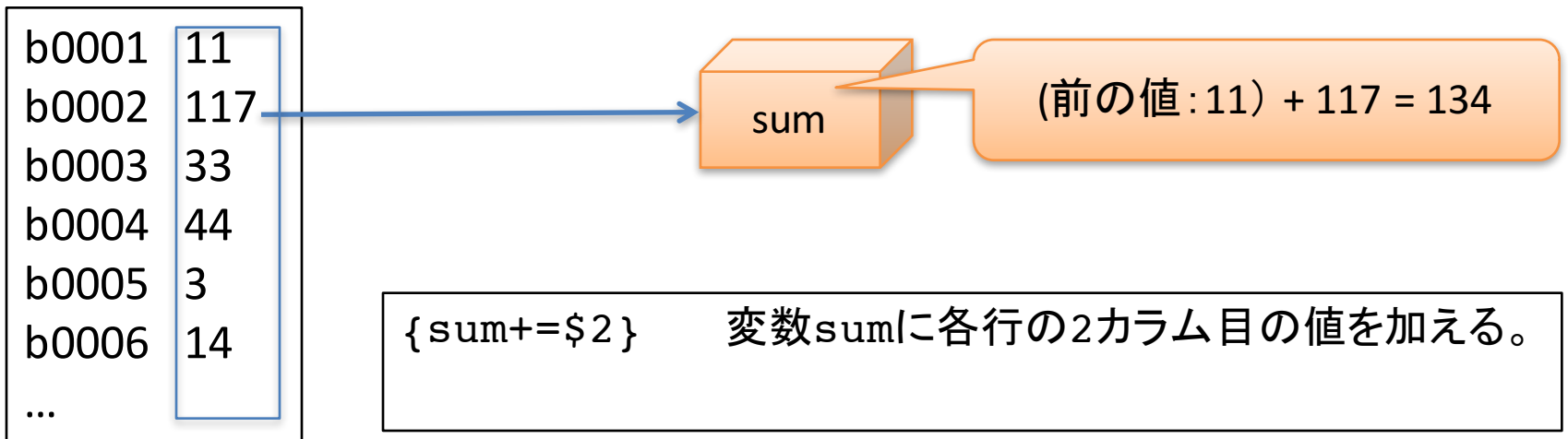


`{sum+= $2}` 変数sumに各行の2カラム目の値を加える。

算術計算 (awk)

- 合計値の出力

```
awk ' {sum=sum+$2} END{print sum}' ecoli.htseq
```

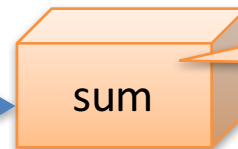


算術計算 (awk)

- 合計値の出力

```
awk ' {sum=sum+$2} END{print sum}' ecoli.htseq
```

b0001	11
b0002	117
b0003	33
b0004	44
b0005	3
b0006	14
...	



(前の値: 134) + 33 = 167

`{sum+= $2}` 変数sumに各行の2カラム目の値を加える。

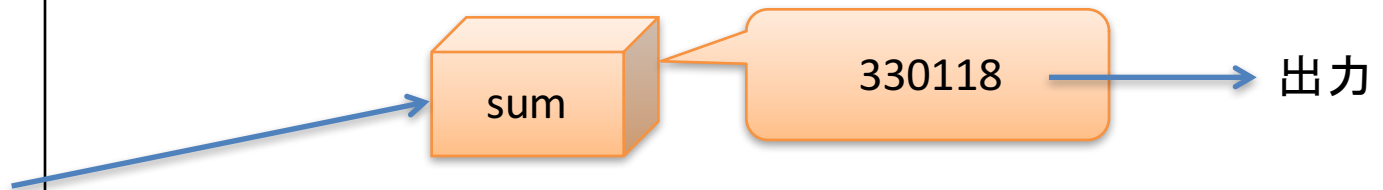
算術計算 (awk)

- 合計値の出力

```
awk ' {sum=sum+$2} END{print sum} ' ecoli.htseq
```

```
b0001 11  
b0002 117  
b0003 33  
b0004 44  
b0005 3  
b0006 14  
...  
[EOF]
```

END{print sum}
最終行で sum の値を出力する。



算術計算(awk)

- 合計値の出力

```
awk ' {sum=sum+$2} END{print sum}' ecoli.htseq
```

- 2カラム目の合計値を出力

- プログラムは2つのブロックからなる

`{sum=sum+$2}` パターン部がないのですべての行が対象となる。
変数sumに各行の2カラム目の値を加える。

`sum+=x` は `sum=sum+x`と同じ。

`END{print sum}` パターンENDは最終行のみにマッチ。
最終行で `sum` の値を出力する。

参考) パターン `BEGIN`は先頭行のみにマッチする。これを用いて変数の初期化などができる。

例) `BEGIN{sum=0} {sum+= $2} END{print sum}`

最初に変数 `sum` を0に初期化する。これはデフォルトの動作として省略できるため、上記のプログラムと同じ結果になる。

演習 (awk)

- awkコマンドを用いて、ecoli.htseq の2カラム目(カウント数)の平均値を出力せよ。
 - ヒント: 行数を数える必要がある。変数lnを使って行数を数えるにはどうすればよいか？
 - カラムの和を変数sumを用いて表せば、最後にsumを行数で割り算することで平均が出せる。割り算は a/b で計算できる。

演習 (awk)

- awkコマンドを用いて、ecoli.htseq の2カラム目(カウント数)の平均値を出力せよ。

解答例1)

```
awk '{sum=sum+$2; ln=ln+1} END{print sum/ln}' ecoli.htseq
```

解答例2)

```
awk '{sum+= $2; ln++} END{print sum/ln}' ecoli.htseq
```