テキスト処理

筑波大学医学医療系精神医学

根本　清貴

本日の内容

- 第1部：UNIX系OSのお作法を知る（兼予習）
- 第2部：画像解析ソフトのコマンドに慣れる
- 第3部：テキストを処理してみる
- 第4部：明日から使えるシェルスクリプトを知る
  1. 繰り返し
  2. 条件分岐

勉強会のルール

- ターミナルでタイプするものは、青色（0000cc）で表示
  - 例：`$ fslhd V_ID001.nii.gz`
  - $はタイプする必要はない
- スクリプトに記載する内容は緑色（007e00）で表示
- コマンドやスクリプトではあるが、タイプしなくていいものは、紫色（9933ff）で表示
- #以降は、解説でありタイプする必要はない
- 「フォルダ」＝「ディレクトリ」
  - Linux/UNIXは、「ディレクトリ」を好む

テキスト処理に便利な記号とコマンド

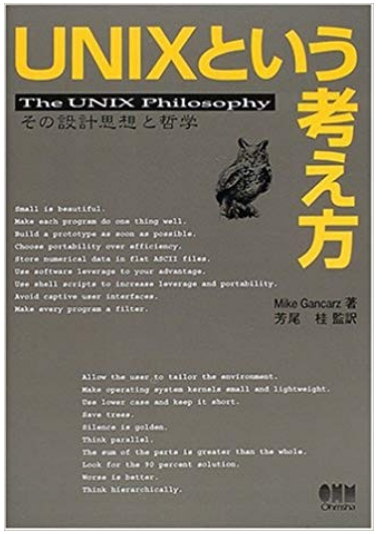
- 記号
  - リダイレクション `>`, `>>`

  コマンド
  - **`grep`, `sed`, `awk`**
  - **`paste`, `cut`, `join`, `sort`, `uniq`**

  これらの記号とコマンドを使えるようになると仕事が一気に進む
- UNIX系OSは、テキストファイルの処理はとても得意

UNIXの思想

- 小さいものは美しい
- 各プログラムが一つのことをうまくやるようにせよ
- できる限り早く原型（プロトタイプ）を作れ
- 効率よりも移植しやすさを選べ
- 単純なテキストファイルにデータを格納せよ
- ソフトウェアを梃子(てこ)として利用せよ
- 効率と移植性を高めるためにシェルスクリプトを利用せよ
- 拘束的なユーザーインターフェースは作るな
- 全てのプログラムはフィルタとして振る舞うようにせよ



UNIXという考え方―その設計思想と哲学

リダイレクション

- コンピュータは基本、出力を画面に出す
- UNIX系OSでは、出力を画面に出さずにファイルに出力することができる
- 出力する向き（direction）を変更することから、リダイレクション（redirection）と呼ばれる
- 記号は2つ
  - > 上書き（以前にあったものはなくなる）
  - >> 追記（以前のファイルに追記される）

データの準備

```
$ cd ~/shell_practice
$ ls
UnixIntro mri pair scripts textprocessing zip
$ cd textprocessing
$ ls
fsleyes_fh.txt  id.txt     roi1.txt  subj_hippo.txt  tissue_volumes.csv
help.txt        names.txt  roi2.txt  subj_info.txt
```

- `cd` したら `ls` する習慣を!

ヘルプ集の作成

- help.txt に dcm2niix と bet のヘルプを出力する
- 最初に自分自身で"Help of dcm2niix" と表示したい
- こういう時は、echo コマンドを用いる

```
$ echo "Help of dcm2niix" > help.txt
$ cat help.txt
Help of dcm2niix
$ dcm2niix -h > help.txt
$ less help.txt
```

- help.txt に "Help of dcm2niix" がない。なぜ？

よくあるリダイレクションのミス

- リダイレクションは最初のみ > で後は >> と覚えておく
- 先の例では、2回 > を使っているため、上書きされている
- 再度ヘルプ集を作成

```
$ echo "Help of dcm2niix" > help.txt
$ echo "====" >> help.txt
$ dcm2niix -h >> help.txt
$ echo "====" >> help.txt
$ echo "Help of bet" >> help.txt
$ echo "====" >> help.txt
$ bet -h >> help.txt
$ (Lin4Neuro) xdg-open .
$ (MacOS) open .
```

xdg-open / open

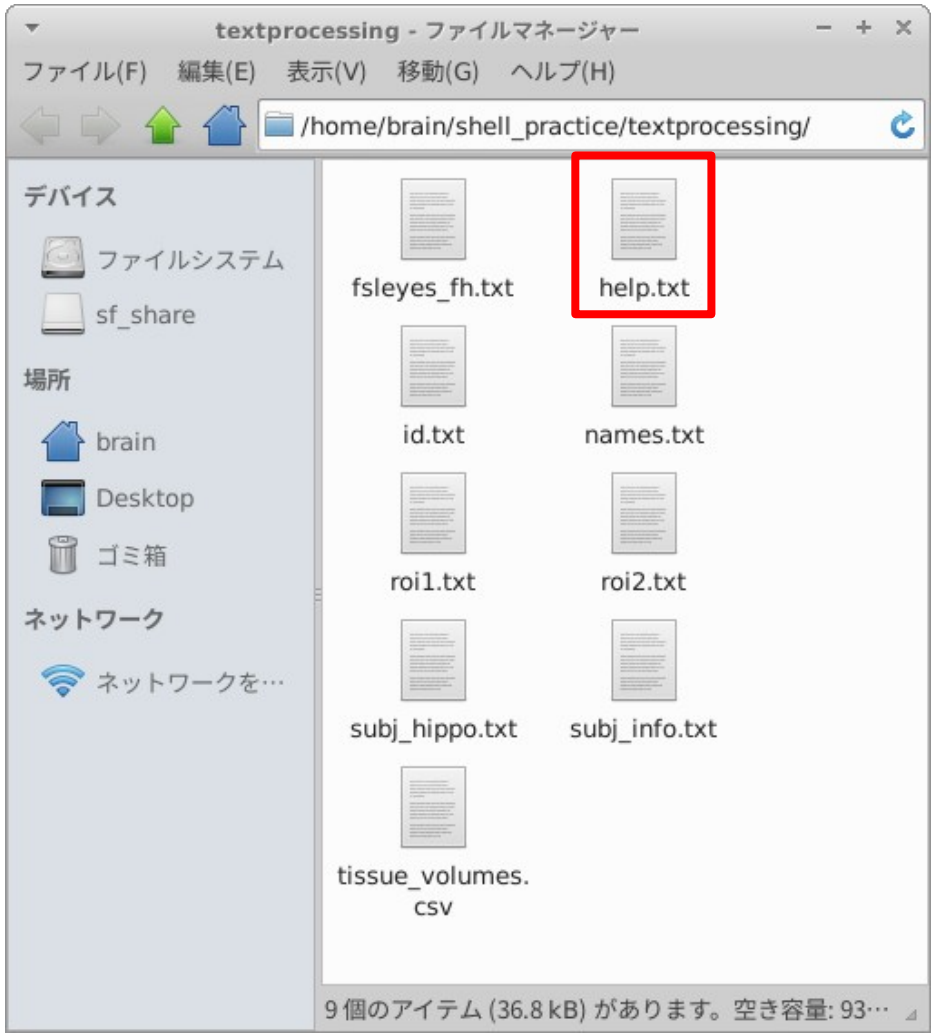
- Linuxでは **xdg-open**、MacOS では **open** はターミナルからGUIを操るのに有用
- コマンドの後にカレントディレクトリを意味する . をつけると、ファイルマネージャー / Finder を起動することができる

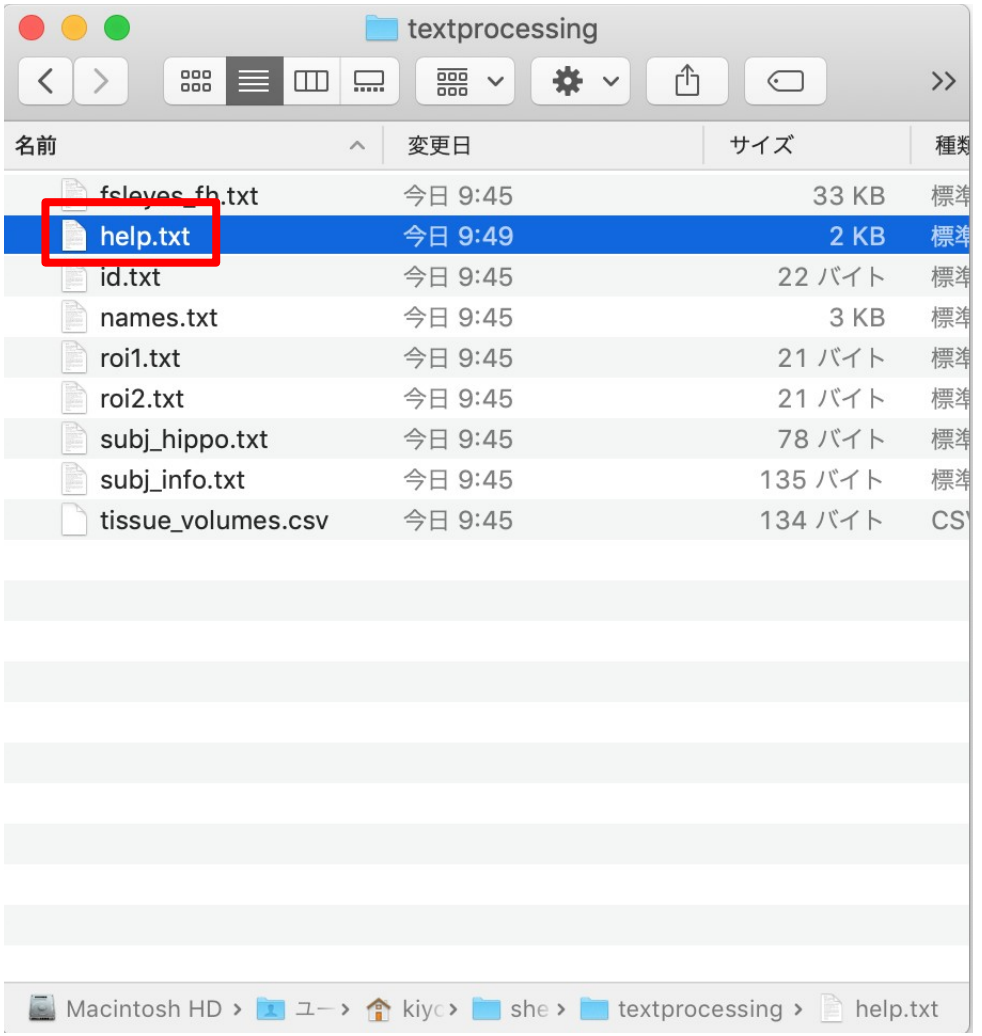
```
$ xdg-open .
```

```
$ open .
```

- その他、登録されている拡張子ならば、（Excelファイルや Wordファイルなど）**"xdg-open/open ファイル名"**でそのまま起動できる

xdg-open . / open .





リダイレクションの応用

- 画像に関する情報など、文字が出力されるものは、リダイレクションを用いることでテキストファイルとして保存し、利用できる
  - FSLであれば、**`fslinfo, fslhd, fslstats`** など
    - **`fslinfo`**: 画像のヘッダー情報（簡略版）
    - **`fslhd`**: 画像のヘッダー情報（フル版）
    - **`fslstats`**: 画像の信号値の統計（平均、標準偏差など）

テキスト処理に役立つコマンド

コマンド	由来	機能
grep	**g**lobal **r**egular **e**xpression **p**rint	検索
sed	**s**tream **ed**itor	行ごとの置換、削除など
awk	作者3人の姓の頭文字	表形式テキストの操作
paste		横方向に貼り付ける
join		2つのファイルをキーを元に結合する
cut		ファイルの一部を切り出す
sort		並べ替える
uniq	**uniq**ue	重複を取り除く

grep

- 「ファイルの中のパターンを検索しなさい」

  ```
  $ grep パターン ファイル
  ```

- **fsleyes_fh.txt** から "**linevector**" というキーワードを調べたい場合

  ```
  $ grep linevector fsleyes_fh.txt
  Available overlay types: label, linevector, mask, mesh, mip, rgbvector,
  sh,
    These options are applied to 'linevector' overlays.
  ```

  - 検索キーワードがある行だけ表示される

- 該当した箇所の行番号も表示したい場合

  ```
  $ grep -n linevector fsleyes_fh.txt
  139: Available overlay types: label, linevector, mask, mesh, mip,
  450:  These options are applied to 'linevector' overlays.
  ```

正規表現

- grep は検索ワードに正規表現を利用できる
- 知っておくと便利な正規表現4つ

  ^ 行頭

  $ 行末

  . 任意の一文字

  * 直前の文字の0回以上の繰り返し

  実際は、.* の組み合わせで「任意の文字列」

正規表現の具体例

```
dim1
dim2
dim3
pixdim1
pixdim2
pixdim3
```

grep dim1

- dim1 と pixdim1 が合致

grep ^dim1

- dim1 のみ合致

grep ^dim.

- dim1, dim2, dim3 が合致

grep ...dim.$

- pixdim1, 2, 3が合致

パイプと grep

- 他のプログラムで得られた結果から、特定の文字を検索したい場合、パイプでつなげて **grep** を使う
- 実際にはこの使い方が多い
- 具体例
  - nifti ファイルのヘッダは **fslinfo** で表示できる
  - ヘッダのうち、dim は 画像の dimension（各方向に何ボクセルあるか）
  - 以下で画像のdimensionが求められる

```
$ fslinfo 画像ファイル | grep ^dim.
```

【演習】パイプと grep

1. ~/shell_practice/UnixIntro に移動してください
2. fslinfo と grep を用いて、highres.nii.gz の dim3 のみ表示してください
3. 同様に fslinfo と grep を用いて、highres.nii.gz の pixdim を表示することで、この画像のボクセルサイズを求めてください

【演習の回答】パイプと grep

1. ~/shell_practice/UnixIntro に移動してください

   ```
   $ cd ../UnixIntro
   ```

   相対パスの威力はこういう時に発揮される

2. fslinfo と grep を用いて、highres.nii.gz の dim3 のみ表示してください

   ```
   $ fslinfo highres.nii.gz | grep ^dim3
   dim3 128
   ```

   # ^dim3 としないと、pixdim3 も表示されます

【演習の回答】パイプと grep

3. 同様に **fslinfo** と **grep** を用いて、**highres.nii.gz** の **pixdim** を表示することで、この画像のボクセルサイズを求めてください

```
$ fslinfo highres.nii.gz | grep pixdim
```

```
pixdim1    1.000000
pixdim2    1.000000
pixdim3    1.250000
pixdim4    1.000000
```

この画像のボクセルサイズは **1x1x1.25mm** で3次元ファイルであることがわかる

pixdim4は4次元ファイルであるかどうか。拡散**MRI**画像では**b0+**軸数、機能**MRI**ではボリューム数が表示される

sed

- 「ファイルに対して、1行ずつ、ルールを実行しなさい」

  `sed ルール ファイル`
- 多機能だが、最も使われるのは「置換 substitute」
- 置換のルールは `'s/検索文字列/置換文字列/'`
- 区切り文字 / は @ や : など他の文字でもよい
- オプション -e を使うことでルールはいくつも並べられる
  （ルールがひとつならば -e は省略可）
- 基本は、行の中で最初にヒットした文字を置換
- `'s/検索文字列/置換文字列/g'` ですべて置換

sed の例

- fileA の中にある

  subj1 D_subj1.nii F_subj1.nii V_subj1.nii

  subj1 subj1_info.txt

  という文字列の subj1 を subj2 に置換したい場合

```
sed 's/subj1/subj2/' fileA
  subj2 D_subj1.nii F_subj1.nii V_subj1.nii
  subj2 subj1_info.txt      #最初にマッチしたものだけ置換される
sed 's/subj1/subj2/g' fileA
  subj2 D_subj2.nii F_subj2.nii V_subj2.nii
  subj2 subj2_info.txt      #すべて置換される
```

sed の実例(1)

```
$ cd ../textprocessing
$ ls
$ cat tissue_volumes.csv
File,Volume1,Volume2,Volume3
V_subj0154_seg8.mat,0.70,0.52,0.36
V_subj0155_seg8.mat,0.80,0.54,0.43
V_subj0156_seg8.mat,0.74,0.49,0.41
(Lin4Neuro) $ xdg-open tissue_volumes.csv
(MacOS) $ open tissue_volumes.csv
```

- このファイルから、'V_' と '_seg8.mat' を取り除きたい
- 削除は「何もないと置換する」という発想

sed の実例(2)

```
$ sed -e 's/^V_//' -e 's/_seg8.mat//' tissue_volumes.csv
File,Volume1,Volume2,Volume3
subj0154,0.70,0.52,0.36
subj0155,0.80,0.54,0.43
subj0156,0.74,0.49,0.41
```

- sed の結果は画面に表示される
- ファイルに書き出すにはリダイレクションを用いる

```
$ sed -e 's/^V_//' -e 's/_seg8.mat//' tissue_volumes.csv > tisvol.csv
```

- 結果を GUI で確認してみる

```
(Lin4Neuro) $ xdg-open tisvol.csv
(MacOS) $ open tisvol.csv
```

パイプと sed

- grep と同様、sed も他の出力をパイプとして入力できる

```
$ cd ../UnixIntro
$ ls                    #一覧表示
```

- 画像ファイルの拡張子を取り除いた一覧を作成したい
- `$ ls *.nii.gz` #.nii.gz がついているファイルを一覧表示

```
example_func.nii.gz highres.nii.gz hr_zstat1.nii.gz
im1.nii.gz              #以降省略
$ ls *.nii.gz | sed 's/.nii.gz//'
example_func highres hr_zstat1 im1 #実際は縦に表示
```

置換の考え方

- 置換は、以下の考えにより、様々な応用ができる
- **subj1.nii.gz**
  - **subj1** を **subj2** に置換（普通の置換）

    `s/subj1/subj2/`　#この場合、s/1/2/でもよい
  - **.nii.gz** を削除　→　**.nii.gz** を**「何もない」**に置換

    `s/.nii.gz//`
  - ファイル名の最初に **V_** を付加 → 行頭を意味する **^** を **V_** に置換

    `s/^/V_/`
  - ファイル名の最後に **.bak** を付加 → 行末を意味する **$** を **.bak** に置換

    `s/$/.bak/`

【演習】パイプと sed

- **UnixIntro** ディレクトリの中にある **im*.nii.gz** のファイル名から拡張子を取り除き、最初に **V_** をつけたリストを作成したい（例: **im1.nii.gz** → **V_im1**）

1. **ls** と **sed** を用いて **imlist.txt** というファイルを保存してください

【演習の回答】パイプと sed

- **UnixIntro** ディレクトリの中にある **im*.nii.gz** のファイル名から拡張子を取り除き、最初に **V_** をつけたリストを作成したい（例: **im1.nii.gz → V_im1**）

1. **ls** と **sed** を用いて **imlist.txt** というファイルを保存してください

```
$ ls im*.nii.gz                          # im1.nii.gz, im2.nii.gz, …
$ ls im*.nii.gz | sed 's/.nii.gz//'      # im1, im2, …
$ ls im*.nii.gz | sed -e 's/.nii.gz//' -e 's/^/V_/'
    # V_im1, V_im2, ...
$ ls im* | sed -e 's/.nii.gz//' -e 's/^/V_/'
  > imlist.txt
$ cat imlist.txt                         #ターミナル内で確認
(Lin4Neuro) $ xdg-open imlist.txt        #Lin4Neuroの gedit で確認
(MacOS) $ open imlist.txt                #Macのテキストエディットで確認
```

- 実際、このように段階を踏んで確認しながら作業をしている


Break

awk

- awk は表形式のテキスト処理が得意なプログラム
- ファイルに対して1行ずつ、パターンに合致する場合、アクションを実行していく
- `awk '/パターン/{アクション}' ファイル`
- 多機能だが、基本は以下
  - パターンは省略も可能→省略した場合、全行が対象
  - アクションで多く使うのは、`{ print }`
  - 区切り文字（デフォルトは半角スペース）で区切られている第1列は `$1`, 第2列は `$2`, … として表示
    - `'{ print $1 }'`
  - 行全体を表示するには `$0`
  - 区切り文字は `-F` の後に指定
    - コンマ区切りの場合、`-F,` となる

awk の実例(1)

```
$ pwd                          #~/shell_practice/UnixIntro
$ cd ../textprocessing         #~/shell_practice/textprocessing
$ ls                           #ファイル一覧を表示
$ cat tisvol.csv
File,Volume1,Volume2,Volume3
subj0154,0.70,0.52,0.36
subj0155,0.80,0.54,0.43
subj0156,0.74,0.49,0.41
```

- このファイルに対して Volume1, Volume2, Volume3の合計を求めたい

awk の実例(2)

```
$ awk -F, '/subj/{ print $1, $2+$3+$4 }' tisvol.csv
subj0154 1.58
subj0155 1.77
subj0156 1.64
```

- これによって subj という文字列がある行に対してVolume1～3 の合計を求めることができた
  - `/subj/` がパターン、`{ print $1, $2+$3+$4 }` がアクション
  - ここで **$1** は ファイル名、**$2+$3+$4** はVolume1～3の合計を意味
- **awk** を上手に使えば、区切られている文字列を取り出すことができる

（参考）awk の応用

- デフォルトでは、出力の区切り文字はスペース
  - -v 'OFS=,' というオプションでコンマ区切りにできる
- BEGINフィールドを使うことで最初の行の表示を設定できる

```
$ awk -F, -v 'OFS=,' 'BEGIN{ print "File", "ICV" }
  /subj/{ print $1, $2+$3+$4 }' tisvol.csv
  > icv.csv
$ cat icv.csv
File,ICV
subj0154,1.58
subj0155,1.77
subj0156,1.64
```

パイプと awk

- パイプを使うと、表形式で出てくる結果を awk で上手に取り出すことができる
- **fslinfo** の結果などは取り出しやすい

```
data_type   INT16
dim1        256
dim2        256
```

（以降省略）

```
$ cd ../UnixIntro
$ ls        #ファイル一覧を表示
$ fslinfo highres.nii.gz | awk '{ print $1 }'
data_type
dim1
dim2
```

（以降省略）

【演習】パイプと awk

1. **fslinfo** で **highres.nii.gz** のヘッダー情報を表示してください
2. **fslinfo** と **awk** を使って **highres.nii.gz** のヘッダーの第2列のみ表示してください
3. **fslinfo** と **awk** を使って、**highres.nii.gz** のヘッダーの **dim3** の行のみ表示してください
   - ヒント: **awk** のパターンは **/^dim3/** となる
4. 3.の結果を応用して、**dim3** の行の第2列のみ取り出ししてください

【演習の回答】パイプと awk

1. **fslinfo** で **highres.nii.gz** のヘッダー情報を表示してください

```
$ fslinfo highres.nii.gz
```

2. **fslinfo** と **awk** を使って **highres.nii.gz** のヘッダーの第2列のみ表示してください

```
$ fslinfo highres.nii.gz | awk '{ print $2 }'
```

3. **fslinfo** と **awk** を使って、**highres.nii.gz** のヘッダーの **dim3** の行のみ表示してください

```
$ fslinfo highres.nii.gz | awk '/^dim3/{ print }'
```

4. 3.の結果を応用して、**dim3** の行の第2列のみ取り出ししてください

```
$ fslinfo highres.nii.gz | awk '/^dim3/{ print $2 }'
```

paste

- 「fileA, fileB, fileC を横方向に連結しなさい」
- `paste fileA fileB fileC`
- `paste` は複数のファイルを横方向に連結する
- `id.txt, roi1.txt, roi2.txt` を連結する
- `cat` は縦方向への連結、`paste` は横方向への連結
- まずは `cat` から確認

```
$ cd ../textprocessing/
$ ls     #ファイル一覧を表示
$ cat id.txt roi1.txt roi2.txt
id
subj1
subj2
subj3

roi1
3.45
4.56
5.67

roi2
3.21
4.32
5.43
```

paste

```
$ paste id.txt roi1.txt roi2.txt
id      roi1    roi2
subj1   3.45    3.21
subj2   4.56    4.32
subj3   5.67    5.43
```

- 3つのファイルが横方向に連結された
- 行の長さが同じものであれば非常に便利

paste

行数が違うものを連結すると結果がぐちゃぐちゃになる

subj_info.txt と **subj_hippo.txt** を **paste** で横方向に連結する

```
$ cat subj_info.txt        #subj_info.txtの内容を表示
$ cat subj_hippo.txt       #subj_hippo.txtの内容を表示
$ paste subj_info.txt subj_hippo.txt
```

```
fsid gender age  mmse eTIV fsid LtHippo  RtHippo
0013 Female 80   30   1230 0013 3115.9   2889.3
0022 Female 61   30   1313 0048 4106.2   4376.6
0048 Male   66   19   1695 0067 2650.3   3141.6
0050 Male   71   20   1461
0067 Male   67   30   1440
```

- 横方向に連結されたが、意味がない
- こういう時に **join** が役立つ

join

- 「2つのファイルをキーとなる列を元に横方向に結合しなさい」

```
$ join -1 m -2 n fileA fileB
```

- **join** は最初のファイルの第m列と2番めのファイルの第n列を使ってファイルを横方向に連結する
  - どちらも同じ第m列の場合は、`join -j m fileA fileB` でよい
- **subj_info.txt** と **subj_hippo.txt** を連結してみる

```
$ cat subj_info.txt
fsid    gender      age     mmse    eTIV
0013    Female      80      30      1230
0022    Female      61      30      1313
0048    Male        66      19      1695
0050    Male        71      20      1461
0067    Male        67      30      1440
```

```
$ cat subj_hippo.txt
fsid    LtHippo     RtHippo
0013    3115.9      2889.3
0048    4106.2      4376.6
0067    2650.3      3141.6
```

join

```
$ join -j 1 subj_info.txt subj_hippo.txt
fsid gender  age  mmse eTIV LtHippo   RtHippo
0013 Female  80   30   1230 3115.9    2889.3
0048 Male    66   19   1695 4106.2    4376.6
0067 Male    67   30   1440 2650.3    3141.6
```

- **join** は2つのファイルに共通するものだけ抜き出す
- **-j 1** はファイルの1列目が共通する場合のオプション
- 注意: **join** を使う際は、元のデータはソートされていないといけない。空白行もエラーになる

sort

- 「文字列をルールに沿って並べ替えなさい」

  `$ sort ファイル`

- 文字通り、ファイルのソートを行うコマンド
- Excelの並べ替えを使っている人は同じイメージでいける

  `$ sort -r` で降順

- **sort** もパイプで活躍する
- 次で紹介する **uniq** や **join** で **sort** は必須

sort のオプション

- sort は数字の並べ替えの際、オプションで挙動が変わる
- seq 20 というコマンドを作って、数を1から20まで1つおきに出力し、それをパイプで sort し、オプションの違いを見てみる

```
$ seq 20 | sort
$ seq 20 | sort -r
$ seq 20 | sort -n
$ seq 20 | sort -nr
```

cut

- 「ファイルからオプションで設定された範囲を切り出しなさい」

  `$ cut オプション ファイル`

- オプションで便利なのは　`-c` 文字数
  - `-c 3-6` で3文字目～6文字目
- `tissue_volumes.csv` から、IDのみを抜き出したい

```
$ cat tissue_volumes.csv
File,Volume1,Volume2,Volume3
V_subj0154_seg8.mat,0.70,0.52,0.36
V_subj0155_seg8.mat,0.80,0.54,0.43
V_subj0156_seg8.mat,0.74,0.49,0.41
```

- 3文字目～10文字目がID

cut

```
$ cut -c 3-10 tissue_volumes.csv
le,Volum  #←邪魔
subj0154
subj0155
subj0156
```

- ファイル全体から3〜10文字目を切り出すため、最初の列も3〜10文字目が切り出されている
- これをなくす方法は？→ grep を使ったらいい

```
$ cut -c 3-10 tissue_volumes.csv | grep subj
```

uniq

- 「ファイル内の連続する同じ行を取り除いて表示しなさい」

  `$ uniq ファイル`

- 画像解析において、ファイルの欠損などを調べる時にとても便利
- uniq はオプションがとても大事

  `$ man uniq`

  `-c, --count`

  行の前に出現回数を出力する

  `-d, --repeated`

  重複した行のみ出力する。出力はグループ毎に 1 回行われる

  `-u, --unique`

  重複していない行のみ出力する

uniq

- **names.txt** にはある地域に済んでいる人のfirst nameが記載されている
- どの名前が多いのかを知りたい

```
$ cat names.txt
```

- この結果を **sort** でソートし、**uniq** につなげる

uniq

```
$ cat names.txt                          #names.txtの内容を表示
$ cat names.txt | sort                   #names.txtの内容をソート
$ cat names.txt | sort | uniq            #ソートした結果から重複を削除
$ cat names.txt | sort | uniq | wc -l #重複を削除した結果の行数
  20
$ cat names.txt | sort | uniq -d   #重複している名前
$ cat names.txt | sort | uniq -d | wc -l #重複している名前の行数
  18    #2人は名前が1回しか登場しないということ
$ cat names.txt | sort | uniq -u   #重複がない名前
  David Elijah
$ cat names.txt | sort | uniq -c   #重複回数を表示
$ cat names.txt | sort | uniq -c | sort -nr #降順で表示
```

【演習】パイプと uniq

```
$ cd ../pair
```

- ここには、研究対象者100人に対して縦断研究で得られたファイルがあり、ベースラインは **subjid_1** 、フォローアップは **subjid_2** となっている。しかし、いくつか欠損ファイルがあり、これを見つけたい

1. **ls** と **wc** を用いてベースラインとフォローアップのファイルがそれぞれいくつかあるか確認してください
2. **ls, cut, uniq** を用いてファイルが2つそろっていない **subjid** を同定してください
3. **grep** を用いて 欠損したファイルを同定してください

【演習の回答】パイプと uniq

1. ls と wc を用いてベースラインとフォローアップのファイルがそれぞれいくつかあるか確認してください

```
$ ls subj*_1                    #ベースラインのファイル一覧
$ ls subj*_1 | wc -l            #98
$ ls subj*_2                    #フォローアップのファイル一覧
$ ls subj*_2 | wc -l            # 99
```

2. ls, cut, uniq を用いてファイルが2つそろっていない subjid を同定してください

```
$ ls | cut -c 1-7 | uniq -u
```

#解説は次スライド

【演習の回答】パイプと uniq

2. **ls, cut, uniq** を用いてファイルが2つそろっていない **subjid** を同定してください

```
$ ls | cut -c 1-7 | uniq -u
```

ファイル名はsubj000_1のようにsubjidは最初の7桁なので `cut -c 1-7` で7桁を切り出し

`uniq -u` で重複がないものだけ表示

その結果: **subj047, subj067, subj094**

3. **grep** を用いて 欠損したファイルを同定してください

```
$ ls | grep subj047 #subj047_2 → subj047_1がない
$ ls | grep subj067 #subj067_2 → subj067_1がない
$ ls | grep subj094 #subj094_1 → subj094_2がない
```

【ケーススタディ】臨床での応用

- 筑波大学附属病院で、2018年1月1日～2018年12月31日に、「電気けいれん療法と下肢静脈エコーのどちらもした人をリストアップしてもらいたい」という依頼あり
- 電気けいれん療法の一覧、下肢静脈エコーの一覧はcsvファイルでともにあり
- ともに重複例あり
- **awk, sort, join** で5分で抽出完了！

csvファイルの内容

- ect.csv
  - 状態,予約日,予約時間,患者ID,患者名
  - 外来,2018/01/05, 09:00-09:30,0001234567,佐藤AA
  - 外来,2018/01/05, 09:30-10:00,0002345678,鈴木BB
- echo.csv
  - 状態,予約日,予約時間,患者ID,患者名
  - 外来,2018/01/04, 13:30-14:00,0003456789,高橋CC
  - 外来,2018/01/04, 13:30-14:00,0004567890,田中DD
- ふたつとも書式は同一
- 第4列の患者IDが一致しているものを抜き出せばよい

awk, sort, uniq, join の合わせ技

- ect.csv から 患者ID（4列目）と 患者名（5列目）を抜き出し、ソートしたうえで重複を削除し、ect.tmp として出力

```
$ awk -F, '{ print $4,$5 }' ect.csv | sort | uniq > ect.tmp
```

- echo.csv から 患者ID（4列目）を抜き出し、ソートしたうえで重複を削除し、echo.tmp として出力

```
$ awk -F, '{ print $4 }' echo.csv | sort | uniq > echo.tmp
```

- ect.tmp と echo.tmp を第1列を使って結合

```
$ join -j 1 ect.tmp echo.tmp
000138394X 豊田XX
000565120X 本田XX
000565563X 松田XX
000571903X 鈴木XX
患者ID 患者名
```

質疑応答