Data sets: fields and field separator

Generally, a Dataset is organized in fields.

A **field is a unit of information** and can contain either numeric or non-numeric data arranged in rows or columns.

A field can be a group of columns or rows. Usually, fields are arranged in columns. In this course, we refer to fields as a group of columns.

In the following example, the data file has 5 fields separated by a space character. In this example, the fields are:

1. First name
2. Last name
3. Birth year
4. Career
5. Area of focus

```
Jane Bolden 1932 author economics
John Talbot 1945 poet english
```

In this other example, the field separator is a comma, and there are 5 fields

```
Jane,Bolden,1932,author,economics
John,Talbot,1945,poet,english
```

Introduction - Text processing

wc: get info on a data file

grep: extract lines matching a pattern

cut: extract columns or fields

sort: sort lines of text files

cat and **paste**: used to concatenate files together

sed: transform text

Before we start:

- Download the zip file **data-temp.zip from Canvas -> Files -> zip files**
- Put the data-temp.zip into your home directory
- Unzip the file data-temp.zip and a directory called data-temp will be created
- Open a terminal and cd to data-temp directory
- List the content of data directory. You will find data files temp.dat, temp-clean.dat and temp-clean1.dat

You can also unzip by typing at the terminal

```
unzip data-temp.zip
```

wc command

```
wc (options) filename
```

Try:

```
wc temp.dat
```

Use man page to find out which info the wc command provides

```
man wc
```

Which option will display specifically the number of lines?

Explore the file: display the first 10 lines, and the last 10 lines of the file

grep command

grep searches and prints lines that match one or more patterns

```
grep (options) pattern(s) filename
```

```
grep Data temp.dat
grep Climate temp.dat
grep AZ temp.dat
```

Some useful options of the grep command: -c, -e, -i, and –v

-c option: count the number of lines matching patterns

```
grep –c pattern filename
```

multiple -e options: search multiple patterns

```
grep –e pattern1 –e pattern2 filename
```

```
….pattern1…
```

```
….pattern2…
```

-v option: search lines excluding a pattern

```
grep –v pattern filename
```

Ignore case

```
grep –i pattern filename
```

sort command to sort lines

`sort (options) filename`

Useful options of the sort command:

-u	to sort and remove duplicates
-k*number*	to sort lines based on a certain field *number*
-n	to sort numerically (if you do not specify this, it will sort alphabetically)
-t*sep*	to specify *field separator* (only if it is different than whitespaces)

If the field separator is one or more than one whitespaces (spaces, tabs) you do not need to use option –t.

Try These

```
sort temp-clean.dat     #sort alphabetically based on 1st  field
sort –n temp-clean.dat #sort numerically based based on 1st  field

sort –k2 temp-clean.dat #sort alphabetically based on 2nd  field
sort -nk3 temp-clean.dat #sort numerically based on 3rd field

sort –t: -k3 temp-clean1.dat #specify that field separator is : and
sort alphabetically based on 3rd field
```

cut command to extract columns or fields

While grep extracts lines matching a pattern, cut extracts columns or fields

`cut option(s) filename`

- option **`–c`** ***`n`*** to cut specific columns by specifying column number n

  Try these

```
cut -c 1 temp-clean.dat

cut -c 1-3 temp-clean.dat

cut -c 1-3,7-9 temp-clean.dat
```

- option **`–d"`*`sep`*`" –f`** ***`n`*** to cut specific field(s) by specifying field separator *sep* and field number *n.* The field separator is also called a delimiter.

  Try these

```
#field separator of temp-clean.dat is a blank character
#field separator of temp-clean1.dat is a :

cut -d":" -f 1 temp-clean1.dat

cut -d" " -f 1  temp-clean.dat

cut -d" " -f 1,3 temp-clean.dat
```

Slide 6 video Lecture4-cat and paste-2min

cat and paste

`cat` and **paste** commands can be used to concatenate files together

Try this:
Make a file called file1 and write in it `Hello`
Make another file called file2 and in it write `Unix`

Now try these and look at the output

```
cat file1 file2           #vertically concatenate

paste file2 file1         #horizontally concatenate
```

The sed command

sed (*stream editor*) is a Unix utility that parses and transforms text

```
sed 's/word1/word2/'  filename

sed 's/word1/word2/g'  filename
```

Make a file called sed-example and write in it:

```
I love bla. I said I love bla
```

Then run sed:

```
sed 's/bla/tea/' sed-example
sed 's/b/Co/' sed-example

sed 's/bla/tea/g' sed-example
sed 's/b/Co/g' sed-example
```

If you add g at the end, it will replace all occurrences

Slides 8-9 video Lecture4-printf-type7min

printf – format and print text

Example

```
var1=mass
var2=18.547
echo $var1 $var2 Kg
```

```
printf "%-8s %.2f Kg\n" $var1 $var2
```

```
mass     18.55 Kg
```

printf to format text: define the type

printf formats and prints argument(s) under control of the format(s)

```
printf "format" argument
```

```
printf "%[width].[precision]type" argument
```

width and precision are optional modifiers

```
%type
%s string
%i or %d integer
%f float or real number
%e scientific notation or exponential
\n new line
```

Try these

```
printf "%f\n" 1.6547
printf "%e\n" 1250000
printf "%d\n" 2
printf "%s\n" Two
```

printf to format text: specify precision

```
printf "%[width].[precision]type" argument
```

```
the precision modifier is a positive integer number which follows
the dot %.precisiontype

%.precisions    for string type specifies the number of string
characters to output
%.precisionf    for float type specifies the number of decimal
places to output
%.precisione for scientific notation type specifies the number of
decimal places to output
```

Try these

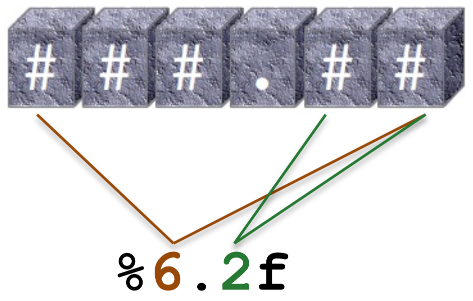
```
printf "%.2f\n" 1.6547  #format float with 2 decimal places
printf  "%.3f\n" 1.6547
printf "%.0f\n" 1.6547
#sometimes 1 is included for width: printf "%1.2f\n" 1.6547

printf "%.3e\n" 1250000 #format in scientific notation with 3
decimal places
printf "%.1s\n" Two   #output 1 character
printf "%.2d\n" 2
```

printf to format text: specify width

```
The width modifier specifies the number of characters to print.

%width.precisiontype
The width is an integer number, which precedes the dot.
If the width is larger than the number of characters of the
output, it will add whitespace characters to the left.

Negative integer - whitespace characters are added to the right
%-width.precisiontype
```

Try these

```
printf "%.2f\n" 1.6547
printf "%5.2f\n" 1.6547
printf "%6.2f\n" 1.6547

printf "%3s\n" Two
printf "%4s\n" Two
printf "%5s\n" Two

printf "%-5s\n" Two
printf "%-6.2f Kg\n" 1.6547
```



printf – define type and precision of multiple arguments and include text

```
printf "format1 format2" argument1 argument2

printf "%w.ptype1 %w.ptype2" argument1 argument2
```

```
printf "%-10.4s %.2f\n" Temperature 1.6547
```

The text you include within the " " will be printed to the screen, including the spaces characters

```
printf "Mass %.2f  .. in %s\n" 65.4747 Kg
```

```
Mass 65.47  .. in Kg
```

Play with the numbers

```
printf "%s %.2f Kg\n" mass 18.547
printf "%6s %7.2f Kg\n" mass 18.547
printf "%-6.2s %7.2f Kg\n" mass 18.547
```