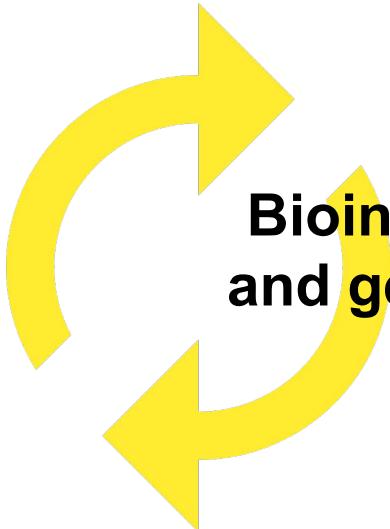




Modules de formation 2019





Bioinformatics platform dedicated to the genetics
and genomics of tropical and Mediterranean plants
and their pathogens

genome assembly SNP detection
phylogeny structural variation
comparative genomics transcriptome assembly differential expression
GWAS pangenomics
population genetics metagenomics
polyploidy



Rice



Banana



Palm



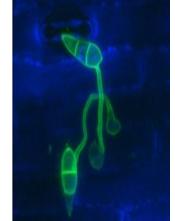
Sorghum



Coffee



Cassava



Magnaporthe

South Green

bioinformatics platform



Larmande Pierre
Sabot François
Tando Ndomassi
**Tranchant-Dubreuil
Christine**



Comte Aurore
Dereeper Alexis



Orjuela-Bouniol Julie



Bocs Stephanie
De Lamotte Fredéric
Droc Gaetan
Dufayard Jean-François
Hamelin Chantal
Martin Guillaume
Pitollat Bertrand
Ruiz Manuel
Sarah Gautier
Summo Marilyne



Rouard Mathieu
Guignon Valentin
Catherine Breton



Mahé Frédéric
Ravel Sébastien



Sempere Guilhem





Workflow manager

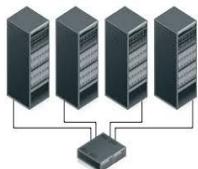
TOGGLE

Toolbox for generic NGS analyses

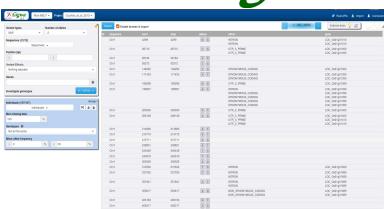
 SNAKEMAKE

Galaxy

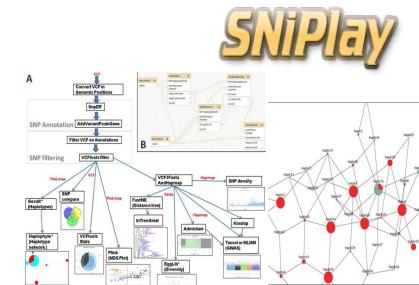
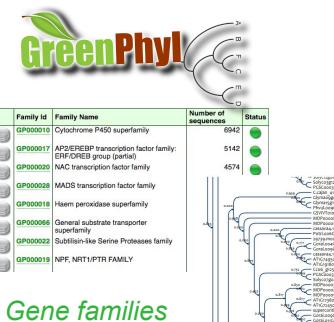
HPC and trainings....



Genome Hubs & Information System



SNPs and Indels



<https://github.com/SouthGreenPlatform>



@green_bioinfo

The South Green portal: a comprehensive resource for tropical and Mediterranean crop genomics, Current Plant Biology, 2016

18-19/03	Guide de survie à Linux - IRD
21/03	Initiation à l'utilisation du cluster CIRAD – CIRAD
22/03	Initiation à l'utilisation du cluster itrop - IRD
15-16/04	Initiation au gestionnaires de workflow SG & Gigwa – IRD
18-19/04	Guide du Jedi en Linux & bash - CIRAD
13-16/05	Python - IRD
17/05	Initiation aux analyses de données transcriptomiques – IRD
21/05	Utilisation avancée du cluster IRD – IRD
23-24/05	Initiation aux analyses de données métagénomiques – IRD
6/06	Manipulation de données et figures sous R – CIRAD
26-28/06	Assemblage et annotation de transcriptomes - IRD



Modules de formation 2019

- Toutes nos formations :
<https://southgreenplatform.github.io/trainings/>
- Topo & TP : [Linux For Jedi](#)
- Environnement de travail : [Logiciels à installer](#)



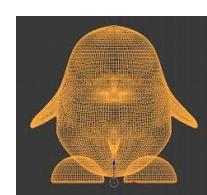


Linux Avancé

www.southgreen.fr

<https://southgreenplatform.github.io/trainings>





Objectifs du module

The objectif!

Optimiser vos analyses bioinformatiques sur
un cluster en utilisant la puissance de Linux



Applications

- Travailler avec de larges volumes de données (eg.: fastq, bam, gff, vcf).
- Filtrer rapidement des fichiers volumineux pour par ex substituer un motif, filtrer sur la taille de séquence, sur un chromosome
- Modifier le contenu d'un fichier avec des outils puissants : ***sed*, *awk***
- Réaliser rapidement la même action sur plusieurs fichiers
- Ecrire de simple scripts bash



Rappel Commandes de Base





Previously

- pwd** Affiche le chemin (où je suis)
- ls -alrt** Liste le contenu d'un répertoire
- cd** Change de répertoire



Previously

pwd Affiche le chemin (où je suis)
ls -alrt Liste le contenu d'un répertoire
cd Change de répertoire

mkdir Crée un répertoire
rmdir Supprime un répertoire vide
rm Supprime un fichier
rm -r Supprime répertoire & fichiers
cp source cible Copie/renomme
mv Déplace un fichier/répertoire



Previously

pwd	Affiche le chemin (où je suis)
ls -alrt	Liste le contenu d'un répertoire
cd	Change de répertoire
mkdir	Crée un répertoire
rmdir	Supprime un répertoire vide
rm	Supprime un fichier
rm -r	Supprime répertoire & fichiers
cp source cible	Copie/renomme
mv	Déplace un fichier/répertoire

cat	Affiche fichier (court)
less	Affiche fichier (long)
head/tail	Affiche début/fin fichier
wc -l	Compte nombre de lignes



Previously

pwd	Affiche le chemin (où je suis)
ls -alrt	Liste le contenu d'un répertoire
cd	Change de répertoire
mkdir	Crée un répertoire
rmdir	Supprime un répertoire vide
rm	Supprime un fichier
rm -r	Supprime répertoire & fichiers
cp source cible	Copie/renomme
mv	Déplace un fichier/répertoire

cat	Affiche fichier (court)
less	Affiche fichier (long)
head/tail	Affiche début/fin fichier
wc -l	Compte nombre de lignes
grep -icv	rechercher un motif
cut -d -f	Extrait colonnes d'un fichier
sort -t -kngr	Trie une colonne d'un fichier



Previously

pwd Affiche le chemin (où je suis)
ls -alrt Liste le contenu d'un répertoire
cd Change de répertoire

mkdir Crée un répertoire
rmdir Supprime un répertoire vide
rm Supprime un fichier
rm -r Supprime répertoire & fichiers
cp source cible Copie/renomme
mv Déplace un fichier/répertoire

chmod Change les droits
chown Change le propriétaire
chgrp Change le groupe

cat Affiche fichier (court)
less Affiche fichier (long)
head/tail Affiche début/fin fichier
wc -l Compte nombre de lignes

grep -icv recherche un motif
cut -d -f Extrait colonnes d'un fichier
sort -t -kngr Trie une colonne d'un fichier



Previously

pwd

Affiche le chemin (où je suis)

ls -alrt

Liste le contenu d'un répertoire

cd

Change de répertoire

mkdir

Crée un répertoire

rmdir

Supprime un répertoire vide

rm

Supprime un fichier

rm -r

Supprime répertoire & fichiers

cp source cible

Copie/renomme

mv

Déplace un fichier/répertoire

chmod

Change les droits

chown

Change le propriétaire

chgrp

Change le groupe

find

rechercher un fichier

cat

Affiche fichier (court)

less

Affiche fichier (long)

head/tail

Affiche début/fin fichier

wc -l

Compte nombre de lignes

grep -icv

rechercher un motif

cut -d -f

Extrait colonnes d'un fichier

sort -t -kngr Trie une colonne d'un fichier

history

zcat, zgrep

tar / gzip

Compresser

tar/gunzip

Décompresser

df -h

du -sh

wget

In -s



Previously

Caractères joker

- * N'importe quel caractère
- [sb] Caractère de l'ensemble



Previously

Caractères joker

*

N'importe quel caractère

[sb]

Caractère de l'ensemble

Redirection Entrées/sorties

> >>

vers un fichier

|

vers une commande



Previously

Caractères joker

- * N'importe quel caractère
- [sb] Caractère de l'ensemble

Redirection Entrées/sorties

- > >> vers un fichier
- | vers une commande

Interagir avec les processus

<Ctrl> + C Arrêter le processus en cours sous le terminal



Previously

Caractères joker

- * N'importe quel caractère
- [sb] Caractère de l'ensemble

Redirection Entrées/sorties

- > >> vers un fichier
- | vers une commande

Interagir avec les processus

<Ctrl> + C Arrêter le processus en cours sous le terminal

Tab completion

<Tab> Complète automatiquement le nom d'un fichier/ répertoire qui est en cours de saisie (choix unique)

<Tab><Tab> Affiche la liste des différentes possibilités si le choix n'est pas unique



Previously

Interagir avec l'historique de commandes

Flèche bas/haut

Afficher la commande précédente/suivante

Presser plusieurs fois pour naviguer dans l'historique

<Ctrl> + R

Afficher la dernière commande qui contient les caractères saisis.

Presser les touches et commencer à taper la commande recherchée



Previously

Interagir avec l'historique de commandes

Flèche bas/haut

Afficher la commande précédente/suivante

Presser plusieurs fois pour naviguer dans l'historique

<Ctrl> + R

Afficher la dernière commande qui contient les caractères saisis.

Presser les touches et commencer à taper la commande recherchée

Nomenclature fichiers

- Linux = sensible à la casse
- PAS d'espaces, accents et caractères spéciaux & ~ # " ' { ([| ` ^ @)] } \$ * % ! / ; , ?
- Suffixe des noms de fichiers (.txt, .fasta, .fa, .fq etc.) optionnel



Environnement de travail

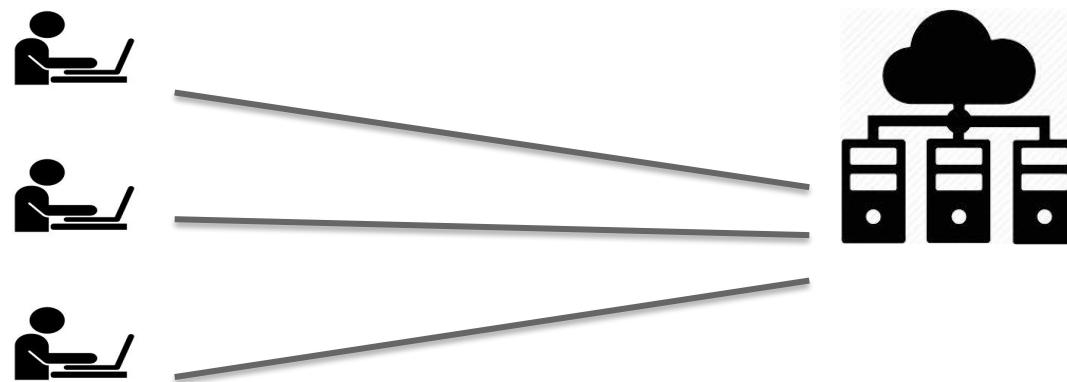
Comment travailler sur le serveur ?



Comment travailler sur le serveur ?



En se connectant sur un serveur linux distant de son ordinateur via le **protocole ssh**



HPC South Green

- itrop (IRD)
- HPC AGAP (CIRAD)

bioinfo-master.ird.fr

cc2-login.cirad.fr



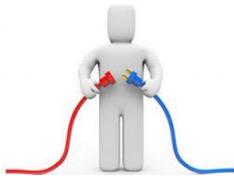
Environnement de travail

Comment transférer un fichier de son PC sur le serveur ?

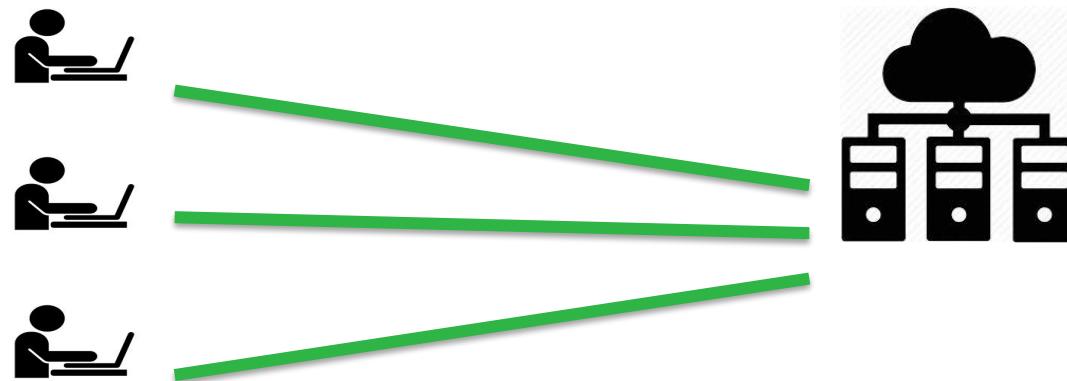
Comment éditer un fichier à distance ?



Copier un fichier de son PC sur le serveur ?



- En se connectant sur un serveur linux distant de son ordinateur via le **protocole sftp**



HPC South Green

- itrop (IRD) bioinfo-nas.ird.fr
- HPC AGAP (CIRAD) cc2.login.cirad.fr



Practice

mobaXterm
terminal, ssh

qrsh, cd, mkdir

1

Go to [Practice 1](#) & [Practice 2](#) on our github



Process monitoring

commande w, ps, kill, top

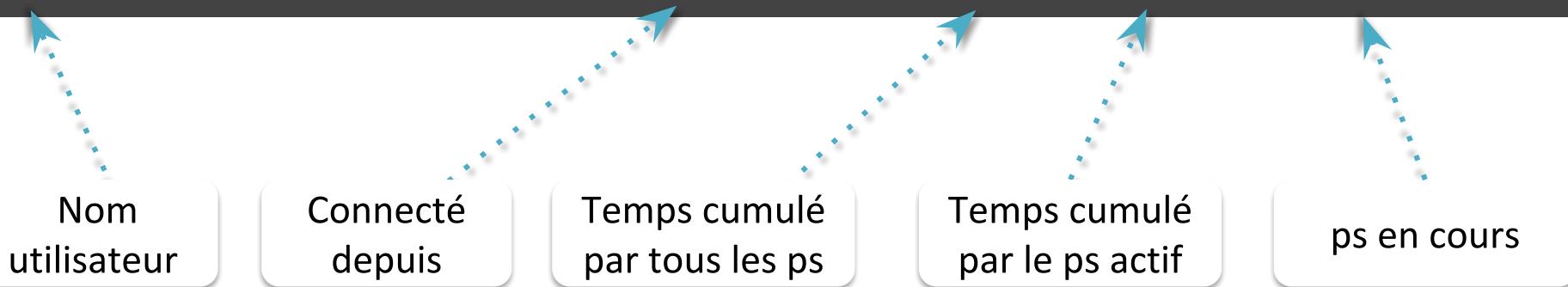


Comment suivre l'activité sur un serveur ?

w

affiche les utilisateurs et les processus associés

```
[tranchant@master0 ~]$ w
16:27:57 up 129 days, 5:28, 27 users, load average: 0,20, 0,25, 0,23
USER     TTY      FROM          LOGIN@        IDLE       JCPU      PCPU      WHAT
klein    pts/5    10.21.129.115  lun.17       1:38m     10.57s    9.93s    qrsh -pe ompi 8
escobar  pts/7    10.23.128.31   14:37       46:05     0.22s    0.09s    ssh node20
daron    pts/8    10.21.141.158  mer.12       1:17m     3:43      10.21s   -bash
tranchan pts/9    ngo34-1-78-210-1 09:16       31:01     1.69s    1.55s    qrsh -pe ompi 12
```





Comment suivre l'activité sur un serveur ?

ps

liste les processus en train de tourner

ps -aux

affiche la liste de tous les processus associés à chaque utilisateur

```
[tranchant@node10 ~]$ ps aux | head -4
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
trancha+  1272  0.0  0.0 116768  3376 pts/2      Ss  09:52   0:00 -bash
trancha+  3753  0.0  0.0 139512  1680 pts/2      R+  10:34   0:00 ps au
mariac  26118 197  9.1 4598024 4514192 pts/0 RNl+ 07:34 356:07 sniffles ...
```

Etat du processus

- R running
- S sleeping
- T Stopped
- Z Zombie



Comment suivre l'activité sur un serveur ?

top

liste les processus en train de tourner

```
top - 16:44:51 up 156 days, 23:10, 1 user, load average: 10,37, 9,80, 9,71
Tasks: 200 total, 3 running, 197 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,0 us, 0,1 sy, 88,5 ni, 11,5 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 65774384 total, 42442784 free, 1907228 used, 21424372 buff/cache
KiB Swap: 8388604 total, 5512296 free, 2876308 used. 62871460 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
18905	daron	30	10	916508	307308	976	R	960,8	0,5	51:38,57	admixture
3446	daron	30	10	1130556	937640	2584	R	100,0	1,4	308:00,92	treemix
19307	tranchat+	20	0	146164	2124	1424	R	0,3	0,0	0:00,02	top
22389	root	20	0	0	0	0	S	0,3	0,0	0:00,17	kworker/10:2

c → Afficher la commande complète en exécution

v → Afficher en mode arborescence

l → Afficher l'activité CPU (une ligne/CPU)

u → Faire une recherche sur un utilisateur en particulier

i → Ne pas afficher les tâches inactives (idle)

q → pour quitter



Comment supprimer un processus ?

kill -9 PID

tuer un processus

```
[tranchant@master0 ~]$ ps aux | grep "tranchant"
tranchant    20999  0.0  0.0 116748  3532 pts/1      Ss+   13:24    0:00 -bash
tranchant    21669  0.0  0.0 176384 22752 pts/1      R     13:33    0:00 perl
toggleGenerator.pl -d /data3/projects/riceAnnot/TOG5681/Illumina/
[tranchant@master0 ~]$ kill -9 21669
```



Practice

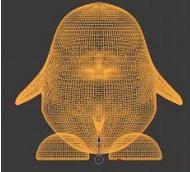
w
ps
kill
top

3

Go to [Practice 3](#) on our github



Lancer plusieurs commandes simultanément



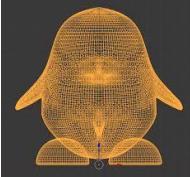
Comment lancer plusieurs ps en même temps ?

Lancer un processus en “arrière plan”

& Lancer un processus en arrière plan *cmd1 &*

nohup “Détacher” le processus de la console.
Fonctionne même quand la console est fermée, si deconnexion *nohup cmd1 &*

jobs Connaitre les processus qui tournent en arrière-plan *jobs*



Comment lancer plusieurs ps en même temps ?

Lancer plusieurs commandes en une ligne

- ; cmd2 exécutée une fois la cmd1 finie *cmd1 ; cmd2*

- && cmd2 exécutée uniquement si cmd1 correctement finie *cmd1 && cmd2*

```
wget linux.tar.gz && tar -zxvf linux.tar.gz
```



Practice

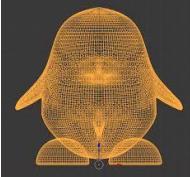
&&

4

Go to Practice 4 on our github



Expression Régulière (ER)



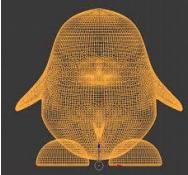
Commande grep

grep

pour rechercher un motif dans un fichier

grep [options] motif [file1, ...]

Option	Description
-i	Recherche le motif sans tenir compte de la casse
-c	Compte le nombre de lignes dans lesquelles le motif a été trouvées
-v	Affiche seulement les lignes sans le motif



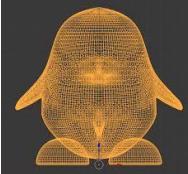
Commande grep

grep

pour rechercher un motif dans un fichier

```
[tranchant@node10 Bank]$ grep "gene" all.gff3 | head -3
Chr1 MSU_osa1r7      gene 2903 10817      .      +      .
ID=LOC_Os01g01010;Name=LOC_Os01g01010;Note=TBC%20domain%20containing%20protein%2C%20expressed
Chr1 MSU_osa1r7      gene 11218     12435      .      +      .
ID=LOC_Os01g01019;Name=LOC_Os01g01019;Note=expressed%20protein
Chr1 MSU_osa1r7      gene 12648     15915      .      +      .
ID=LOC_Os01g01030;Name=LOC_Os01g01030;Note=monocopper%20oxidase%2C%20putative%2C%20expressed

[tranchant@node10 Bank]$ grep "gene" all.gff3 | tail -3
ChrSy    MSU_osa1r7    mRNA 589676    589999      .      +      .
ID=ChrSy.fgenesh.mRNA.89;Parent=ChrSy.fgenesh.gene.89;Name=ChrSy.fgenesh.mRNA.89
ChrSy    MSU_osa1r7    CDS   589676    589999    11.35      +      0
ID=ChrSy.fgenesh.CDS.327;Parent=ChrSy.fgenesh.mRNA.89;score=11.35
ChrSy    MSU_osa1r7    exon  589676    589999    11.35      +      .
ID=ChrSy.fgenesh.exon.327;Parent=ChrSy.fgenesh.mRNA.89;score=11.35
```



Commande grep

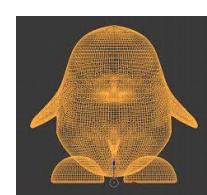
grep

pour rechercher un motif dans un fichier

```
[tranchant@node10 Bank]$ grep "gene" all.gff3 | head -3
Chr1 MSU_osa1r7      gene 2903 10817      .      +      .
ID=LOC_Os01g01010;Name=LOC_Os01g01010;Note=TBC%20domain%20containing%20protein%2C%20expressed
Chr1 MSU_osa1r7      gene 11218     12435      .      +      .
ID=LOC_Os01g01019;Name=LOC_Os01g01019;Note=expressed%20protein
Chr1 MSU_osa1r7      gene 12648     15915      .      +      .
ID=LOC_Os01g01030;Name=LOC_Os01g01030;Note=monocopper%20oxidase%2C%20putative%2C%20expressed

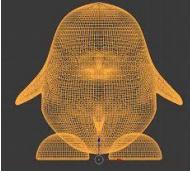
[tranchant@node10 Bank]$ grep "gene" all.gff3 | tail -3
ChrSy    MSU_osa1r7    mRNA 589676     589999      .      +      .
ID=ChrSy.fgenesh.mRNA.89;Parent=ChrSy.fgenesh.gene.89;Name=ChrSy.fgenesh.mRNA.89
ChrSy    MSU_osa1r7    CDS   589676     589999     11.35      +      0
ID=ChrSy.fgenesh.CDS.327;Parent=ChrSy.fgenesh.mRNA.89;score=11.35
ChrSy    MSU_osa1r7    exon  589676     589999     11.35      +      .
ID=ChrSy.fgenesh.exon.327;Parent=ChrSy.fgenesh.mRNA.89;score=11.35
```

grep -e "gene\s" all.gff3



Expression régulière ou rationnelle

Chaine de caractères qui décrit une ensemble de chaînes de caractères possibles permettant de faire des sélections



Expression régulière ou rationnelle

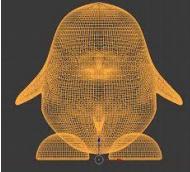
Chaine de caractères qui décrit une ensemble de chaînes de caractères possibles permettant de faire des sélections

ER *basiques* - ERb

*vi
grep
sed*

ER *étendues* - ERe

*grep -E
sed -E
awk*

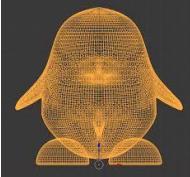


Expression régulière ou rationnelle

Chaine de caractères qui décrit une ensemble de chaînes de caractères possibles permettant de faire des sélections

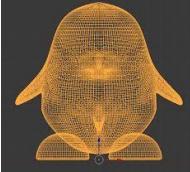
Communes aux ERs basiques et étendues

^	début de ligne	LOC1
\$	fin de ligne	$LOC1\$$
.	n'importe quel caractère	$^L.C1$
*	0 à n fois	$ATCA*T$
[...]	plage de caractères permis	[ATGC]
[^...]	plage de caractères interdits	[^ATGC]



Expression régulière ou rationnelle

- [0-9] N'importe quel chiffre
- [a-z] N'importe quelle lettre en minuscule
- [^A-Z] N'importe quel caractère excepté une lettre en majuscule
- [a-zA-Z] N'importe quelle lettre en minuscule ou majuscule



Expression régulière ou rationnelle

Chaine de caractères qui décrit une ensemble de chaînes de caractères possibles permettant de faire des sélections

Communes aux ERs basiques (vi, grep, sed)

\{n\}	n repetitions du caractère placé devant
\{n,x\}	entre n et x fois le caractère précédent
\{n,\}	au minimum n fois le caractère précédent
\{,n\}	au maximum n fois le caractère précédent
\(ERb\)	mémorisation d'une ER basique
\1 \2	rappel de mémorisation



Practice

grep

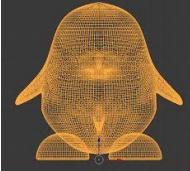
5

Go to [Practice 5](#) on our github



Des commandes pour rechercher et modifier des fichiers

commande sed



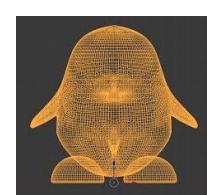
sed, Stream EDitor

SYNTAX

sed : “Stream Editor”
pour rechercher et modifier une ligne

Syntax : sed *OPTIONS* 'operation' *inputfile*

Option	Description
-n	écrit seulement les lignes spécifiées sur la sortie standard
-e	permet de spécifier les commandes à appliquer sur le fichier.
-s	Consider files as separate rather than as a single continuous long stream.
-iSUFFIX	edit files in place (makes backup if SUFFIX is supplied)

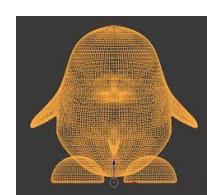


Sélection et affichage de lignes dans un fichier

par numero de ligne

*sed -n 'line **P**' inputFile*

Line number	<i>sed -n '5p' all.gff3</i>	affiche la 5ème ligne
\$	<i>sed -n '\$p' *.fastq</i> <i>sed -n -s '\$p' *.fastq</i>	affiche la dernière ligne
first~step	<i>sed -n '1~4p' ir.fastq</i>	affiche à partir de la ligne 1 , toutes les 4 lignes



Sélection et affichage de lignes dans un fichier *par numero de ligne*

*sed -n 'line **P**' inputFile*

```
sed -n "4p; 7p" test.txt                                            # affiche ligne 4 et 7
```

```
sed -n "4,7 p" file.fastq                                        # affiche ligne 4 à 7
```

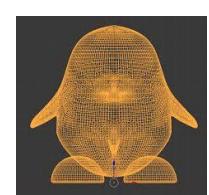


Practice

printing with sed

6

Go to [Practice 6](#) on our github



Suppression de lignes dans un fichier

par numero de ligne

*sed 'line **d**' inputfile*

```
sed "2d; 4d" test.txt                                    # supprime ligne 2 et 4  
sed "2,4 d" test.txt                                    # supprime ligne 2 à 4  
sed '2~4d' irigin1_1.fastq
```

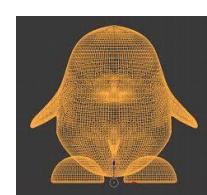


Practice

deleting with sed

7

Go to [Practice 7](#) on our github



sed , Stream EDitor

ER : PRINT/DELETE LINE

Sélection de lignes dans un fichier

par motif

sed 'ER' inputFile

```
sed '/^#/d' test.sed
```

```
sed -n '/^Bonjour/p; /^Au revoir/p' test.sed
```

```
sed -n '/^Bonjour/,/4.$/p' test.sed
```

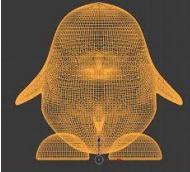


Practice

sed using ER

8

Go to Practice 8 on our github



Substitution/Remplacement dans lignes

Sélection de lignes dans un fichier vérifiant une expression régulière
ET appliquant une modification ou un traitement

```
sed "s/motif recherché/nouveau motif/" file
```

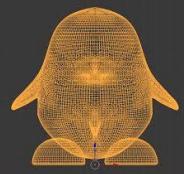
substitution

séparateur

motif recherché

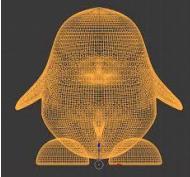
nouveau motif

fichier à parser



Sed : Quelques exemples

Example	Description
sed "s/day/night/" file	Change la 1ère occurrence de “day” par “night” par ligne
sed "s/linux/LINUX/2" file	Change la 2ème occurrence de “linux” par “LINUX” par ligne
sed "s/ [lL]inux/LINUX/g" file	Change toutes occurrences de “linux” par “LINUX”



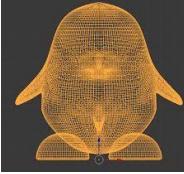
Sélection et Substitution de lignes dans un fichier

par motif

```
sed 's/ / /'    inputfile  
sed 'y/éè/ee/' inputfile
```

```
sed -n '2~4s/T/u/p;' irigin1_1.fastq
```

```
sed -n '2~4y/Tt/Uu/p;' irigin1_1.fastq
```



sed : rechercher et modifier une ligne

Selection de lignes dans un fichier vérifiant une expression régulière
ET appliquant une modification ou un traitement

```
sed "s/[0-9] [0-9] */nouveau motif/" file
```

substitution

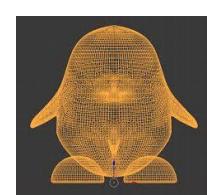
motif recherché

nouveau motif

fichier à parser

Recherche une chaîne de caractères
commençant par un chiffre suivi par 0 ou plusieurs nombres

=> Chaîne de caractère enregistrée dans la variable

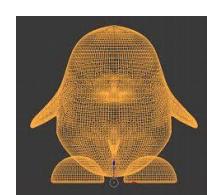


Sed : Quelques exemples

Chr1	2903
Chr1	3354
Chr1	4357
Chr1	5457



Chr1	2903	2903
Chr1	3354	3354
Chr1	4357	4357
Chr1	5457	5457

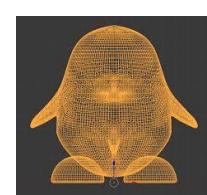


Sed : Quelques exemples

Chr1 2903		Chr1 2903 2903
Chr1 3354		Chr1 3354 3354
Chr1 4357		Chr1 4357 4357
Chr1 5457		Chr1 5457 5457

sed 's/\s[0-9]/& &/'*

Pattern trouvé sauvegardé
dans le caractère spécial &



Sed : Quelques exemples

```
echo abcd123 | sed 's/\([a-z]*\).*/\1/'  
abcd
```

```
echo abcd123 | sed -E 's/( [a-z]* ) .*/\1/'  
abcd
```

```
echo abcd123 | sed -E 's/( [a-z]* )(.* )/\2 \1/'  
123 abcd
```



Practice

sed

9

Go to [Practice 9](#) on our github



Des commandes pour rechercher et modifier des fichiers

commande awk



awk

awk: Langage pour manipuler un fichier ligne par ligne

- Nom des auteurs : “Aho, Weinberger, and Kernighan”



awk: Langage pour manipuler un fichier ligne par ligne

- Nom des auteurs : “Aho, Weinberger, and Kernighan”
- Un langage de programmation qui permet facilement de manipuler des fichiers tabulés (blast, sam, vcf) et d'extraire une partie des données
- Un langage utilisé pour rechercher des motifs et pour effectuer des opérations, des actions associées.



awk: Langage pour manipuler un fichier ligne par ligne

Principales caractéristiques d'awk

- Fichier en entrée tabulé
 - Comme tout langage de programmation, awk a des variables et peut appliquer des conditions
 - awk peut faire des opérations sur les nombres et les chaînes de caractères
 - awk peut générer et afficher des données/rapports suite à des manipulations



awk

awk: Langage pour manipuler un fichier ligne par ligne

Syntax : awk [-F] 'program' file

Option	Description
-F	Donne la nature des séparateurs de champs



awk

awk: Langage pour manipuler un fichier ligne par ligne

Syntax : awk [-F] 'program' file

Option	Description
-F	Donne la nature des séparateurs de champs

Variables prédéfinies utilisées par awk

Variable	Description
\$0	ligne entière
NR	Numéro de la ligne lue
NF	Nombre de champs dans la ligne



awk

awk voit le fichier en entrée
comme des enregistrements et des champs

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

File: contact.txt



awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r�aron@videotron.ca

File: contact.txt

```
awk '{print $0}' contact.txt
```

```
Helene 56 edu hcyr@sun.com
jean 32 ri jeanc@inexpress.net
julie 22 adm juliem@sympatico.ca
michel 24 inf michel@uqo.ca
richard 25 inf r aron@videotron.ca
```

Affiche chaque
ligne



awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

File: contact.txt

```
$awk '{print NR, $1, $2}' contact.txt
```

1 Helene 56
2 jean 32
3 julie 22
4 michel 24
5 richard 25

Affiche
le num  ro de la ligne lue
Puis le 1  er champ
puis le 2  e champ du fichier
tabul  



awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

```
$awk '{print $1,$2};  
END { print NR "lignes lues en tout" }' contact.txt
```

Helene 56

Jean 32

Julie 22

Michel 24

Richard 25

5 lignes lues en tout

Instruction ex  ut  e une fois le fichier lu dans son int  gralit  



awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

```
$awk '{print $1,$3; somme+=$2}  
END { print "Somme des ages  gale   ", somme }' contact.txt
```

Helene edu
jean ri
julie adm
michel inf
richard inf

Somme des ages  gale   159

On ajoute l' ge (\$2)   la variable somme   chaque ligne lue

Puis on affiche la somme calcul e   la fin de la lecture du fichier



awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

File: contact.txt

```
$awk '{somme+=$2}  
END { print " Age moyen = ", somme/NR }' contact.txt
```

Age moyen = 31,8

On ajoute l' ge (\$2)   la variable **somme**   chaque ligne lue

Puis on affiche la moyenne une fois le fichier lu



awk

awk: Langage pour manipuler un fichier ligne par ligne

avec une liste d'instructions et de **conditions aussi**

Condition {Instr-1; Instr-2; ...; Instr-n}

```
awk '$2 > 24 && $2 < 50 { print "Age de ", $1, "compris  
entre 24 et 50, égal à ", $2 }' contact.txt
```

L age d Helene est supérieur à 24 et égal à 56
L age d jean est supérieur à 24 et égal à 32
L age d richard est supérieur à 24 et égal à 25

Avec 2
conditions



awk

awk: Langage pour manipuler un fichier ligne par ligne

```
awk '$3 == "inf" {print $0}' contact.txt
```

```
michel 24 inf michel@uqo.ca
richard 25 inf rcaron@videotron.ca
```

```
$awk '/j/ {print $0}' contact.txt
```

```
jean 32 ri jeanc@inexpress.net
julie 22 adm juliem@sympatico.ca
```



awk

awk: Langage pour manipuler un fichier ligne par ligne

```
awk '{print $1, $2-10}' contact.txt
```

```
Helene 46  
Jean 12  
Julie 12  
Michel 14  
Richard 15
```

```
awk '$2 > 30 && $3 == "ri" {print $0}' contact.txt
```

```
jean 32 ri jeanc@inexpress.net
```

Ces commandes peuvent être utilisées avec en entrée la sortie standard ou un fichier tabulé (comme .gff, fichier blast m8 , .vcf)



Practice

awk

10

Go to [Practice 10](#) on our github



awk - fonctions

Manipulation de chaîne de caractères

`length(myText)`

longueur de myText

`substr(myText,start,length)`

Extrait la sous chaîne de la chaîne `myText` à partir de la position `start` sur une longueur `length`

`tolower(myText)`

Modifie la casse de myText en minuscule

`toupper(myText)`

Modifie la casse de myText en majuscule

`split`

0 ou 1 occurrence du caractère/groupe de caractère précédent

`gsub(search,replace,var)`

`sub(ER,replace, var)`



awk - fonctions

Manipulation de nombres

`int(myNb)`

partie entière de myNb

`log(myNb)`

logarithme de myNb

`sqrt(myNb)`

racine carée de myNb



Fusionner des fichiers : la commande join



Commande join

join fichier1 fichier2

```
:~$ cat fichier1
1 Bash
2 Python
3 Perl
4 Java
5 C++
:~$ cat fichier2
1 sympa
2 cool
3 no comment
4 pffff
5 ouille
```

```
:~$ join fichier1 fichier2
1 Bash sympa
2 Python cool
3 Perl no comment
4 Java pfff
5 C++ ouille
```



Commande join

Fusionner en précisant les colonnes communes :

```
join -1 2 -2 1 fichier1 fichier3
```

Préciser les colonnes à afficher :

```
join -1 2 -2 1 fichier1 fichier3 -o 2.1,2.2
```

Les fichiers doivent être triés au préalable



Practice

join

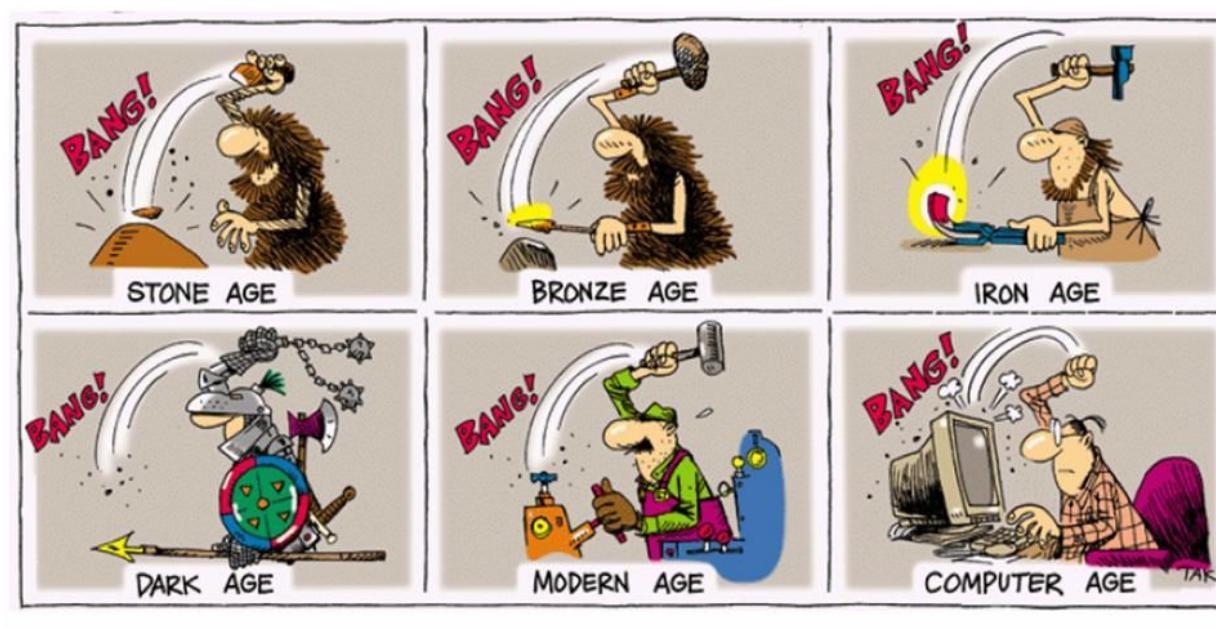
11

Go to [Practice 11](#) on our github



Pour automatiser le lancement de commandes

bash





Exécuter un script bash

sh nom_script.sh

```
[tranchant@node10 Bash]$ sh helloWorld.sh
```



Premier script en bash

- Toujours débuter par : `#!/bin/sh`
- Suivis par les instructions, une instruction par ligne
- **Chaque instruction doit se terminer par ;**
- N'hésitez pas à commenter votre script en plaçant un `#` devant votre commentaire



Premier script en bash

- Toujours débuter par : `#!/bin/sh`
- Suivis par les instructions, une instruction par ligne
- **Chaque instruction doit se terminer par ;**
- N'hésitez pas à commenter votre script en plaçant un `#` devant votre commentaire
 - Pour vous et vos collègues pour comprendre le code
 - ignore le texte placé après un `#`
 - Commentaires libres



Premier script en bash

- Pas d'accent
- Premières instructions

```
echo 'text';           pour écrire sur la sortie (écran)  
echo -e "text \n";  pour réaliser un saut de ligne
```



Modifier le script

P1.2

- Sauver le script ***helloWorld.sh*** sous un nouveau nom (ex : *helloWorld-v2.pl*)
- Modifier le code de ce nouveau script en affichant d'autres textes avec ***\n***
- Exécuter ce nouveau script



Modifier le script

P1.3

- Créer volontairement des erreurs dans votre code en retirant un ; puis un # et un “
- Observer les messages d'erreurs



Modifier le script

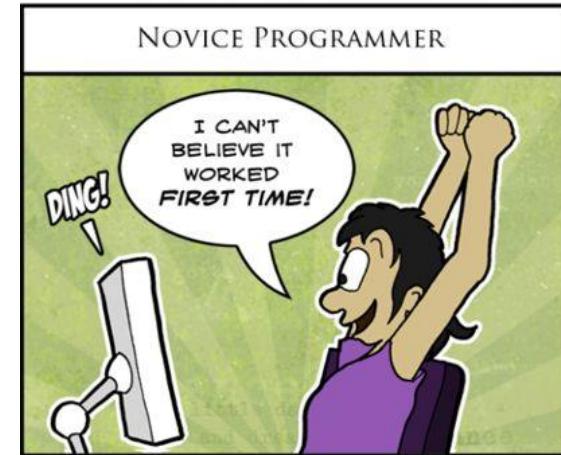
P1.3

- Créer volontairement des erreurs dans votre code en retirant un ; puis un # et un “ ”
- Observer les messages d'erreurs

Une des principales activités du programmeur est de « débugger »...

Souvent aussi longue qu'écrire le code !

Il faut donc s'entraîner à décoder les messages d'erreurs !





Boucle for



Exécuter une boucle

for...



- Pour parcourir une liste
- Exécuter les mêmes instructions sur chaque élément de la liste

```
for file in * ;  
do  
    instruction1  
    instruction2  
done
```



Practice

12

Go to [Practice 12](#) on our github



Les variables



Qu'est ce une variable ?

Variable...

```
nom = "Hello World";  
echo $nom;
```

« conteneur », « boîte » dans lesquels on peut stocker un objet, une information.

Règles

- Noms de variables uniquement avec des caractères *alpha-numériques* (*A-Z*, *a-z*, *0-9*) ou *underscore*
- **Sensible à la casse et pas d'espace**



Une variable variable...

Variable...

```
maVar= "Hello World!!!";  
echo $maVar;                      # Hello World  
echo ${maVar:6}                     # World!!!  
echo ${maVar:0:3};                  # Hel  
echo ${maVar:6:3};                  # Wor  
echo ${maVar: -2};                 # !!  
  
echo ${var%*};                     # Hello Worl  
echo ${var##Hello}                  # World!!!
```



Une variable variable...

Variable...

```
file=BCU_AAOSW_3_1_C39R6ACX.bam  
echo $file                      # BCU_AAOSW_3_1_C39R6ACX.bam  
echo ${file:5}                    # AOSW_3_1_C39R6ACXX.bam  
  
echo ${file/.bam/.sam}           # BCU_AAOSW_3_1_C39R6ACXX.sam
```



Substitution au sein d'une variable

Variable...

```
maVar= "Hello World!!!";  
echo $maVar;           # Hello World!!!  
echo ${maVar/o/}       # Hell WorLd!!!  
echo ${maVar//o/};      # Hell WrLd!!!
```



Practice

for echo script

13

Go to [Practice 13](#) on our github



Practice

8

Go to [Practice 8](#) on our github



Arguments d'un script



Condition avec des nombres

- Transmettre au script des valeurs saisies dans la ligne de commande : arguments, paramètres
- Affectées aux variables réservées **1, 2, ...** et appellées **\$1, \$2, ...**

sh testNum.sh 25

```
#!/bin/bash

myNum=$1;

if [[ $myNum = 10 ]]
then
    echo "Egal a 10";
elif [[ $myNum -le 10 ]]
then
    echo "Inferieur ou egal a 10";
else
    echo "Superieur a 10";
fi
```



Les conditions



Condition avec une chaîne de caractère

Variable...

```
#!/bin/bash

myText="Hello world ! ";

if [[ $maVar = "Hello" ]]; then
    echo "Very Nice";
else
    echo "No nice";
fi

sh script.sh
```



Condition avec une chaîne de caractère

Variable...

```
#!/bin/bash

myText="Hello world ! ";

if [[ $maVar =~ "Hello" ]]; then
    echo "Very Nice";
else
    echo "No nice";
fi
```

```
sh script.sh          # Very Nice
```



Opérateur de comparaison

Nombres

\$a -eq \$b	\$a égal à \$b
\$a -ne \$b	\$a différent de \$b
\$a -lt \$b	\$a inférieur à \$b
\$a -gt \$b	\$a supérieur à \$b
\$a -le \$b	\$a inférieur ou égal à \$b
\$a -ge \$b	\$a supérieur ou égal à \$b



Condition avec des nombres

```
#!/bin/bash

myNum=18;

if [[ $myNum = 10 ]]
then
    echo "Egal a 10";
elif [[ $myNum -le 10 ]]
then
    echo "Inferieur ou egal a 10";
elif [[ $myNum -gt 10 ]]
then
    echo "Superieur a 10";
else
    echo "C'est quoi ce bins?";
fi
```



Condition avec des nombres

- Transmettre au script des valeurs saisies dans la ligne de commande : arguments, paramètres
- Affectées aux variables réservées **1, 2, ...** et appellées **\$1, \$2, ...**

sh testNum.sh 25

```
#!/bin/bash

myNum=$1;

if [[ $myNum = 10 ]]
then
    echo "Egal a 10";
elif [[ $myNum -le 10 ]]
then
    echo "Inferieur ou egal a 10";
else
    echo "Superieur a 10";
fi
```

Formateurs

- Christine Tranchant-Dubreuil
- Gautier Sarah
- Valérie Noël
- Ndomassi Tando
- Frédéric Mahé
- François Sabot



En informatique,
la pensée magique ne fonctionne pas !

Il faut pratiquer ... et ... *restez calme !*
à vous de jouer !

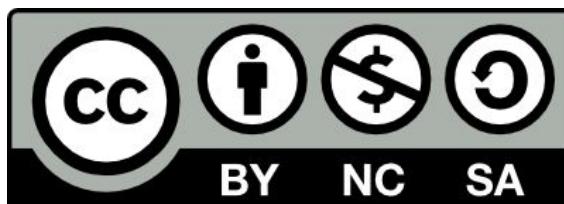


Copyright © Randy Glasbergen. www.glasbergen.com



Le matériel pédagogique utilisé pour ces enseignements est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions (BY-NC-SA) 4.0 International: <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Merci pour votre attention !



Le matériel pédagogique utilisé pour ces enseignements est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions (BY-NC-SA) 4.0 International:

<http://creativecommons.org/licenses/by-nc-sa/4.0/>