Recherche Opérationnelle et Optimisation
TP1 : premiers pas avec R et RStudio

Responsable : Julien Ah-Pine

M1 Informatique 2017/2018

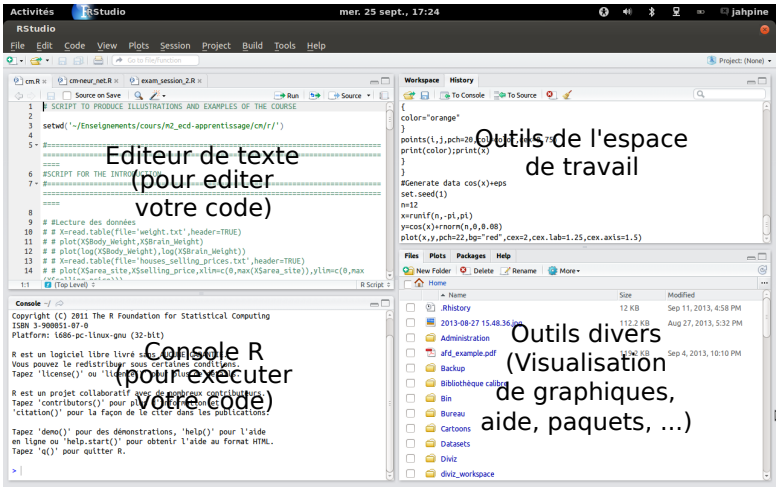
1 R et Rstudio

R (`http://www.r-project.org/`) est un langage de programmation interprété et un environnement pour la statistique computationnelle et la visualisation graphique de données. Il est un dérivé du langage S développé par John Chambers et dédié à l'origine aux statisticiens. Il existe de nombreuses boîtes à outils (paquets, en anglais "packages") qui permettent à la communauté scientifique de mettre en oeuvre des méthodes statistiques à la classiques et nouvelles. Mais le succés de R en fait un outil de plus en plus utilisé par des chercheurs, ingénieurs, étudiants provenant d'autres domaines scientifiques comme l'apprentissage automatique ("machine learning"), la fouille de données ("data-mining"), l'économétrie, la biométrie ... et également la théorie des graphes. R est ainsi aujourd'hui un langage qui est utilisé de façon générale pour du "data science". Vous pourrez utiliser la "reference card" suivante pour vous aider à trouver des commandes élémentaires de R : `http://www.sites.univ-rennes2.fr/laboratoire-statistique/PAC/doc/refcard.pdf`.

R peut être utilisé par défaut via une console d'évaluation intéractive (console REPL "read-evaluate-print-loop" ou "shell"). Mais RStudio (`http://www.rstudio.com/`) est un environnement de développement dédié à R. Cet environnement permet de travailler avec R et ses graphiques de façon plus intéractive, d'écrire du code R avec un éditeur de texte proposant de nombreuses fonctionnalités, d'organiser votre code et de gérer plusieurs projets à la fois ... Il est également multiplateforme (Windows, Linux, Mac). R et RStudio sont open-source ! Vous pouvez donc les installer gratuitement sur votre machine personnelle.

2 Interface de RStudio

1. Lancez RStudio (Menu `Démarrer` ...)
2. L'interface de RStudio se compose principalement de 4 fenêtres (si vous en avez trois tapez `Ctrl + Shift + n`) :



3 Création et manipulation de vecteurs, listes et matrices

3. Dans la **console** tapez :

```
#Création d'un vecteur représentant les sommets
> X=c(1,2,3,4,5)
```

Puis tapez suivi de `Entrée` :

```
> X
```

4. Dans l'**éditeur de texte** tapez :

```
#Création du vecteur de sommets en utilisant une séquence
1:5
seq(1,5,1)
```

Puis sélectionnez (avec la souris ou le clavier à l'aide de la touche `Shift`) les trois dernières lignes et faites `Ctrl + Entrée`. Vous pouvez donc entrez des commandes dans l'éditeur de texte et exécutez celles-ci par ligne ou par bloc. Les lignes sélectionnées sont alors envoyées à la console qui les exécute.

5. Enregistrez le fichier texte sur votre disque et donnez lui le nom de `tp1.R`. Ce fichier est un script dans lequel vous pourrez enregistrer toutes vos commandes et exécuter celles-ci ultérieurement. Ainsi, sauvegardez régulièrement ce fichier en tapant `Ctrl + s`. Par ailleurs, vous remarquerez que le caractère `#` permet d'entrer des commentaires. N'hésitez pas à vous en servir pour décrire les commandes ou faire référence à l'exercice que vous êtes en train de traiter ! Dans la suite, vous écrirez donc vos commandes dans l'éditeur de texte et vous les exécuterez comme indiqué précédemment.

6. Entrez et exécutez les commandes suivantes :

```
#Création d'une liste...
Gamma=list()
#... pour instancier l'ensemble des successeurs de chaque sommet
Gamma[[1]]=c(2)
Gamma[[2]]=c(1,3)
Gamma[[3]]=c(4,5)
Gamma[[4]]=c(5)
Gamma[[5]]=c(1)
```

7. Entrez et exécutez les commandes suivantes :

```
#Accès au contenu des différentes composantes d'une liste
Gamma[[2]]
Gamma[[2]][1]
Gamma[[2]][2]
```

8. Entrez et exécutez les commandes suivantes :

```
#Longueur d'un vecteur
N=length(X)
```

9. Déterminez la commande permettant de calculer à partir de `Gamma` le demi-degré extérieur d'un sommet

10. Entrez et exécutez les commandes suivantes les unes après les autres :

```
#Initialisation d'une matrice remplie de 0
B=matrix(0,ncol=N,nrow=N)
B
#Instanciation d'une matrice particulière
C=matrix(c(1,2,3,4,5,6),nrow=3)
C
```

```
#Autres façon de créer une matrice
rbind(c(1,4),c(2,5),c(3,6))
cbind(c(1,2,3),c(4,5,6))
```

11. Nous allons créer une **fonction** qui permet de convertir une liste `Gamma`, représentant l'application multivoque associée à un graphe, en une matrice d'adjacence. Pour cela, tapez tel quelles les lignes suivantes et instanciez la fonction en demandant à la console de l'interpréter (ie sélectionnez tous le bloc et tapez `Ctrl + Entrée`) :

```
Gamma_to_Adjacency = function(X,G)
  #Donne la matrice d'adjacence d'un graphe à partir de l'application multivoque G
  #INPUT:
  #X est l'ensemble des sommets
  #G est une liste de sous-liste donnant les successeurs de chaque sommet
  #OUTPUT:
  #A est la matrice d'adjacence (carrée binaire)
{
  N=length(X)#nombre de sommets
  print(paste("Le nombre de sommets du graphe est : ",N))#commande print pour afficher d
  A=matrix(0,ncol=N,nrow=N)#initialisation de la matrice d'adjacence
  for (i in 1:N)#parcours de tous les sommets i
  {
    d_i_p=length(G[[i]])#demi-degré extérieur de i
    print(paste("Le demi-degré extérieur su sommet ",i," est : ",d_i_p))
    if (d_i_p>0)
    {
      for (j in 1:d_i_p)#parcours de tous les successeurs de i
      {
                A[i,G[[i]][j]]=1#si j fait parti des successeurs alors A[i,j]=1
      }
    }
  }
  return(A)
}
```

12. Entrez et exécutez les commandes suivantes :

```
A=Gamma_to_Adjacency(1:5,Gamma)#Test de la fonction
A
#Extraction de vecteurs
A[,1]
A[2,]
#Recherche d'éléments
which(A[,1]==1)
```

13. Ecrivez une fonction `Adjacency_to_Gamma` qui prend en entrée une liste de sommets `X` et une matrice d'adjacence `A` et donne en sortie une liste `G` dont chaque composante est relative à un sommet de `X`, et donne la liste des successeurs de ce-dernier. Remarque : n'hésitez pas à utiliser l'aide accessible partir de l'onglet `help` dans la fenêtre en bas à droite de Rstudio.

4 Quelques opréations/fonctions utiles sur les matrices

14. Entrez et exécutez la commande suivante :

```
#Instanciation d'une matrice d'adjacence
B=rbind(c(0,1,1,0),c(0,0,0,1),c(0,0,0,1),c(0,0,0,0))
nrow(B)
```

```
ncol(B)
length(B)
```

Que fait chacune de ces commandes ?

15. Entrez et exécutez les commandes suivantes les unes après les autres :

```
B*B
B*matrix(1:16,nrow=4)
B%*%B
```

A quoi correspond chacune des opérations matricielles `*` et `%*%` ?

16. Entrez et exécutez les commandes suivantes les unes après les autres :

```
B%*%B>0
as.numeric(B%*%B>0)
C=matrix(as.numeric(B%*%B>0),nrow=4)
C
```

Que se passe t-il ? A quoi correspond `C` ? (Interprétez `C` telle la matrice d'adjacence d'un graphe et identifiez la situation lorsque dans `C` une valeur est non nulle).

17. Entrez et exécutez les commandes suivantes :

```
colSums(B)
rowSums(B)
B[-2,]
B[,-4]
B[-1,-1]
which(rowSums(B)==0)
B[1,which(B[1,]==1)]=10
```

Que fait chacune de ces commandes ?

5 Test de la présence d'un circuit

18. Déterminez à la main et en utilisant l'algorithme vu en cours, si les graphes orientés représentés par les matrices d'adjacence $\mathbf{A}$ et $\mathbf{B}$ suivantes possèdent un circuit :

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad ; \quad \mathbf{B} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

19. Ecrivez une fonction `Presence_Circuit` qui renvoie `1` si un graphe représenté par sa matrice d'adjacence `A` possède un circuit et `0` sinon.

20. Testez votre implémentation sur les exemples précédents et ensuite sur ceux du cours.