Code R pour l'analyse des données KNS

1 Survol

Réglages initiaux

Pour lancer le code et produire tous les tableaux/graphiques, commencez par créer deux dossiers à l'emplacement de votre choix:

1. un dossier où les analyses R sont faites [`dossier.R`], et qui contiendra les fichiers R requis par les analyses
2. un dossier où les dernières versions des bases de données sont sauvegardées `[dossier.DB]` (ce dossier doit être synchronisé avec DropBox)

Une fois ces dossiers créés, placez dans le fichier `dossier.R` le contenu du dossier `GS_KNS_Code-R` (disponible sous `DropBox/KNS_GINGER-SOPRONER`).

Ouvrez le fichier `GS_MotherCode.r` [1] et définissez la valeur des objets `dossier.R` et `dossier.DB` en fonction de l'emplacement choisi pour ces dossiers. Pour voir un exemple du format accepté par R pour la définition de l'emplacement de dossiers, tapez dans la console R:

```
> getwd() # montre le working directory
```

Le symbole `#` dénote le début d'un commentaire non-exécuté par R.

Note: Lorsque non-existants, une fonction dans `GS_MotherCode.r` crée automatiquement les trois dossiers suivants dans le `dossier.R`: *Tableaux*, *Graphiques* and *Data*.

Routine de lancement des sorties de base

Dans le fichier `GS_MotherCode.r`, s'assurez que les variables sont bien définies dans la première section. Notamment, les variables `sorties.INV`, `sorties.LIT` et `sorties.POISSONS` devraient avoir la valeur `TRUE` si vous voulez produire les sorties associées, ou `FALSE` dans le cas opposé. La variable `filtre.sur.especes` contrôle les espèces incluses dans l'analyse: lorsque sa valeur est `FALSE` toutes les espèces sont incluses (voir prochaine section).

Vérifiez aussi que l'objet `tourner.tout` (défini dans `GS_MotherCode.r`) a la valeur `TRUE` si vous voulez tout tourner immédiatement, ou `FALSE` si vous voulez charger les fonctions sous R et tourner certains aspects de l'analyse manuellement.

[1] Un fichier .r peut être modifié sous NotePad ou l'éditeur compris avec R. Si vous programmez plus souvent, *TINN-R* et *R-Studio* sont deux alternatives disponibles gratuitement. Ces logiciels permettent non seulement d'éditer sous R, mais aussi d'interagir plus facilement avec la console.

Ouvrez R [2], et chargez le fichier `GS_MotherCode.r` avec la commande `source()`. En début de session seulement, donnez l'emplacement complet de la fonction `GS_MotherCode.r` (donc remplacez ... par l'emplacement du `dossier.R`). .

```
> source(".../GS_KNS_MotherCode.r")
```

Note: La première fois que vous lancerez le code, il vous faudra installer certaines librairies R (vous aurez comme message d'erreur: `there is no package called ...`).

Important!! Vous devez charger le fichier `GS_MotherCode.r` au moins une fois en début de session, car ce fichier installe toutes les fonctions nécéssaires à la production des sorties et défini plusieurs variables utilisées dans l'analyse.

Rajouter un filtre sur les espèces

Si vous désirez inclure ou exclure de l'analyse certains groupes taxonomiques, mettez l'option `filtre.sur.especes = TRUE`. Lorsque `GS_MotherCode.r` est lancé, R vous demandera automatiquement de spécifier les valeurs du filtre sur espèces (voir exemple ci-bas). Pour changer ces valeurs manuellement, vous pouvez également lancez vous même la fonction:

```
> filtre.especes("oui")

   Inclure ou exclure?
      Inclure

   Unité taxomique?  (Groupe/Sous-Groupe/Famille/Genre/Espece)
      Grp2

   Nom?
      Crustaces, Mollusques, Echinodermes
```

Attention: lorsque `filtre.sur.especes = FALSE`, lancer `GS_KNS_MotherCode.r` ôtera automatiquement le filtre enregistré au préalable. Pour ôter *temporairement* ce comportement, vous pouvez rajoutez un # devant la ligne qui exécute cette fonction, en prenant bien soin de l'ôter avant de quitter R pour qu'elle soit active lors de la prochaine session.

```
#if(!filtre.sur.especes){filtre.especes()}else{filtre.especes("oui")}
```

[2] Une courte introduction aux concepts de bases de R peut se trouver dans les Sections 1 et 2 du document *SimpleR*, disponible à l'adresse suivante: `cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf` (voir aussi les autres sections qui survolent plusieurs applications statistiques).

2 Concepts utiles sous R

Fonctions

Le terme *fonction* sous R dénomme un objet contenant une ou plusieurs lignes de codes qui performent une tâche spécifiée par le programmeur (sous Excel on appellerait ça une macro). Une fonction est très utile lorsqu'on veut répéter les mêmes lignes de code en ne changeant qu'un seul aspect à chaque fois, ou lorsqu'on veut pouvoir lancer une analyse complexe en ne tapant qu'une seule commande. Par exemple si je veux faire le même graphique pour les Crustacés, Mollusques et Échinodermes, je pourrais taper les même ligne de codes 3 fois en changeant les termes 'Crustaces', Mollusques', etc., ou bien je pourrais écrire une fonction qui prend en *argument* le groupe (taxonomique). Par exemple:

```
> fonction.exemple <- function(x) x * 2
> fonction.exemple(4)

[1] 8
```

`function()` est une fonction de base R utilisée pour indiquer qu'on défini un objet de type "function", qu'il s'appelle "fonction.exemple", qu'il prend en argument "x" et qu'il multiplie cet argument par la valeur 2.

```
> fonction.exemple()

Error in x * 2 : x is missing
```

Ici "x" n'a pas de valeur par défaut, donc R retourne un message d'erreur. Lorsqu'on définit une valeur par défaut pour `x` (e.g. `x=3`, voir ci-bas), lancer `fonction.exemple()` ne retourne plus de message d'erreur, mais bien la valeur 6.

```
> fonction.exemple <- function(x=3) x * 2
> fonction.exemple()

[1] 6
```

Pour voir les valeurs par défaut définies dans les fonctions, vérifiez si les arguments sont suivis par un =, ou pas. Si vous lancez une fonction et ne définissez pas un des arguments, R prendra la valeur définie par défaut ou donnera un message d'erreur (si absente). A faire attention, donc: lorsque vous lancez des fonctions manuellement il faut s'assurer que les valeurs définies correspondent bien à ce que vous voulez produire.