Description: Code R pour l'analyse des données d'échantillonage

Contents

1 Accès rapide aux descriptions des fonctions R

1.1 Invertébrés

1.2 Poissons

```
poissons.tableau.brut(save=FALSE)
BioDens.sp.poissons()
poissons.ts1(AS="A",save=FALSE)
poissons.ts2(AS="A",save=FALSE)
poissons.p3(quel.graph="all",save=FALSE)
```

1.3 LIT

```
LIT.tableau.brut(save=FALSE, AS="pas de filtre")
LIT.resume(yy=2011, ff="Coraux_Gen", AS="A", save=FALSE)
LIT.ts1(AS="A")
LIT.ts2(AS="A")
LIT.bp1(yy=2011, ff2="Coraux_Gen", AS="A")
```

2 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
> print("hop")
```

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.
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 ...`).

[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.

[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).

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/Especе)
      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")}
```

Modifier les requêtes lancées par défaut

Le fichier `GS_CodesInvertebres_Launch.r` doit être édité pour modifier les requêtes lancées par défaut.

3 Description des fonctions pour invertébrés, poissons et LIT

3.1 Invertébrés

`inv.biodiv(AS="pas de filtre",qunit="St",wC="all",save=FALSE)`:
Inv_IndexBiodivPar + qunit + filtre station + filtre taxo
Tableau des indices de biodiversité (moyenne et écart type) et richesse spécifique (via `inv.RichSpecifique()`) par station (valeur par défaut, peut être changée en modifiant l'argument `qunit`). Le tableau produit est utilisé par d'autres fonctions, entre autres `inv.biodiv.geom()`.

`inv.biodiv.geom(AS="A",save=FALSE)`:
Inv_IndexBiodiv + géomorphologie/impact + filtre station + filtre taxo
Tableau résumé des moyennes (et ET) sur station par géomorphologie, et par géomorphologie/type d'impact.

`inv.sprich.tbl(AS="A",grtax="Groupe",save=FALSE)`: *Inv_NumEspeceParGeomorph + groupe taxo + filtre station + filtre taxo*
Tableau du nombre d'espèces pour chacun des membres de l'aggrégation taxonomique spécifiée (`grtax`),

proportion des espèces contenues dans chaque membre, et nombre total d'espèces observées par campagne/géomorphologie.

`sprich.by.aggrtaxo(AS="A", grtax="Groupe", save=FALSE)`:
Inv_NumEspeceParAggrTaxon + groupe taxo + filtre station + filtre taxo
Tableaude la richesse spécifique (nombre d'espèces) pour chaque membre du l'aggrégation taxonomique choisie (`grtax`). Par exemple si `grtax = "S`$_{Groupe}$", *letableaucontiendrapourchaquestationlenombred'espcepourchacu groupestaxonomiquesobservssurlesstationscontenuesdanslefiltre.*

`inv.RichSpecifique <- function(AS="A", aj.impact=FALSE, aggr="St")`:
Tableau interne non sauvegardé mais utilisé en information complémentaire pour d'autres tableaux (`inv.biodiv()`, `inv.biodiv.geom()`). Richesse spécifique par niveau taxonomique (nombre de genres, familles, sous-groupes, groupes) par station ou transect, avec option d'ajouter la moyenne (ET) de la RS par géomorphologie (`aggr="geom"`) et zone d'impact (`aj.impact = TRUE`).

inv.dens.tbl inv.dens.geom inv.dens.tbl.parT inv.graph.TS() inv.graph.TS.top10

3.2 Poissons

`poissons.tableau.brut(save=FALSE)`:
GS_Poissons_TableauDonneesBrutes_ + date + .csv
Reformattage du **tableau** de données brutes initiales en incluant les valeurs de densité et de biomasse pour chaque observation.

`BioDens.sp.poissons()`:
Tableau préparatoire (non sauvegardé) calculant les densités et biomasses par campagne, transect et espèce en se basant sur l'objet `ds.calc` produit par la fonction `poissons.tableau.brut()`. Le tableau produit par cette fonction (nommé `BDtable` dans l'environnement R) est utilisé par les fonctions `TS1.poissons()` et `TS2.poissons()`.

`poissons.ts1(AS="A",save=FALSE)`:
GS_Poissons_TS1_ + filtre + date + .csv
Tableau synthèse contenant la moyenne et l'écart type pour la densité, la biomasse, la richesse spécifique et la taille moyenne, calculés pour les catégories suivantes: toutes espèces confondues, commerciales, carnivores, herbivores, piscivores, planctonophages, sédentaires, territoriales, mobiles, très mobiles et ciblées (ou non) par la pêche en Nouvelle-Calédonie.

`poissons.ts2(AS="A",save=FALSE)`:
GS_Poissons_TS2_ + filtre + date + .csv
`Tableau` synthèse contenant pour chaque espèce la densité moyenne sur toutes les années, pour chaque année, sur toutes les stations et pour chaque station.

`poissons.p3(quel.graph="all",save=FALSE)`:
GS_PoissonsST_ [valeur représentée].[catégorie de poissons] + date + .pdf
Cette fonction produit des **graphiques** représentant des séries temporelles de densité, biomasse ou richesse spécifique pour les catégories de poissons définies dans l'objet `graph.key$df.id`. Par défaut le filtre annuel est appliqué. Cette fonction est seulement executée dans le code `GS_CodePoissons_Launch.r` si la valeur de l'objet `graph.poissons` est changée à `TRUE`.

3.3 LIT

`LIT.tableau.brut(save=FALSE, AS="pas de filtre")`:
GS_LIT_TableauBrut_ + filtre + date + .csv
Tableau formatté des données de couverture brutes (une rangée par campagne/transect) – ce tableau est sauvegardé dans l'environnement R sous `LITbrut` et est utilisé par les autres fonctions LIT.

`LIT.resume(yy=2011, ff="Coraux_Gen", AS="A", save=FALSE)`:
GS_LIT_Tableau1A_ + code catégorie de coraux + filtre + year + date + .csv
Tableau des couvertures moyennes (et SE) pour chaque géomorphologie selon l'année et la catégorie de substrat spécifiées (par défaut, 2011 et types de substrat généraux).

`LIT.ts1(AS="A")`:
Tableau 1: *GS_LIT_SerieTempTable_ + filtre + _GeoMorphImpact_ + date + .csv*
Tableau 2: *GS_LIT_SerieTempTable_ + filtre + _GeoMorphImpact_DiffCouv_ + date + .csv*
Figures: *GS_LIT_SerieTempTable_ + filtre + _GeoMorphImpact_ + type de coraux + géomorphologie + date + .pdf*
Deux tableaux sont produits. Le premier contient la moyenne (et SE) des couvertures pour chaque campagne, géomorphologie et zone d'impact, le deuxième contient la différence entre les couvertures par géomorphologie des zones impactées et non-impactées. Des **graphiques** sont également produits et comparent pour chaque géomorphologie et catégorie de substrat les tendances de couvertures moyennes par campagne entre les zones impactées et non-impactées.

`LIT.ts2(AS="A")`:
Tableau: *GS_LIT_SerieTempTable_ + filtre + _GeoMorphImpact_bySt_ + date + .csv*
Figure: *GS_LIT_SerieTemp_ + filtre + _GeoMorphImpact_bySt_ + type de coraux + géomorphogie + catégorie d'impact + date + .pdf*
Tableau des couvertures moyennes (et SE) pour chaque type de substrat par station et campagne. Cette fonction produit aussi des **graphiques** comparant la couverture moyenne de chaque type de substrat entre les stations de la même géomorphologie et zone d'impact.

`LIT.bp1(yy=2011, ff2="Coraux_Gen", AS="A")`:
Nom de fichier: *GS_LIT_Hist1_ + year + code catégorie de coraux + .pdf*
Graphiques en barre comparant la couverture moyenne (et SE) à l'intérieur d'une catégorie de substrat et pour l'année spécifiée.

Note: les catégories de substrats sont définies dans l'objet `coraux.fig`, créer par la fonction `prep.analyse()`.

4 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.