A01la_scatterplot

Private template

generated from BioDataScience-Course/A02la_scatterplot

1 Branch 1 Tag Go to file Go to file Code

phgrosjean warning=FALSE for f...

54f74a1 · 2 years ago 9 Commits

R	Partial ver...	2 years ago
figu...	Version 20...	2 years ago
tests	warning=F...	2 years ago
.Rb...	Initial com...	2 years ago
.giti...	Version 20...	2 years ago
.urc...	warning=F...	2 years ago
.urc...	warning=F...	2 years ago
A01...	First draft ...	2 years ago
DES...	Version 20...	2 years ago
LIC...	Version 20...	2 years ago
REA...	Version 20...	2 years ago
urc...	warning=F...	2 years ago

README License

SDD I Module 1 : Premiers graphiques en nuage de points

About

Exercice graphique nuages de points du module 1 de SDD I

Readme
View license
Activity
Custom properties
0 stars
2 watching
1 fork
Audit log

Releases 1

Version for academic year 20... Latest
1 minute ago

Packages

No packages published
Publish your first package

Contributors 2

ean Philippe Grosjean

nEngels Guyliann Engels

Languages

R 83.0% Makefile 17.0%

Ce projet nécessite d'avoir assimilé l'ensemble des notions du premiers modules du cours de Science des données biologiques 1. Son template est ici : https://github.com/BioDataScience-Course/A01Ia_scatterplot.

Objectifs

Ce projet est *individuel* et *cadré*. Vous allez :

- cloner un projet hébergé dans GitHub et l'utiliser dans RStudio (Saturn Cloud)
- découvrir comment un projet RStudio se structure dans un dépôt GitHub
- exécuter des instructions R dans un script R
- découvrir le format R Markdown avec un document Quarto
- réaliser des graphiques en nuage de points avec R
- interpréter ces graphiques
- apprendre à utiliser les tests diagnostics pour vérifier votre travail
- synchroniser vos modifications avec GitHub (commit/pull/push)

Cela fait beaucoup de chose, mais nous procéderons par étapes.

Les données employées dans le cadre de cette étude proviennent de recherches sur la croissance de l'oursin *Paracentrotus lividus* Lamarck (1816) en élevage. Vous trouverez des informations importantes depuis R dans la page d'aide de ce jeu de données comme ceci (entrez ces instructions à la **Console** et validez-les par la touche `Entrée`) :

```
SciViews::R()
.?urchin_bio
```

Consignes

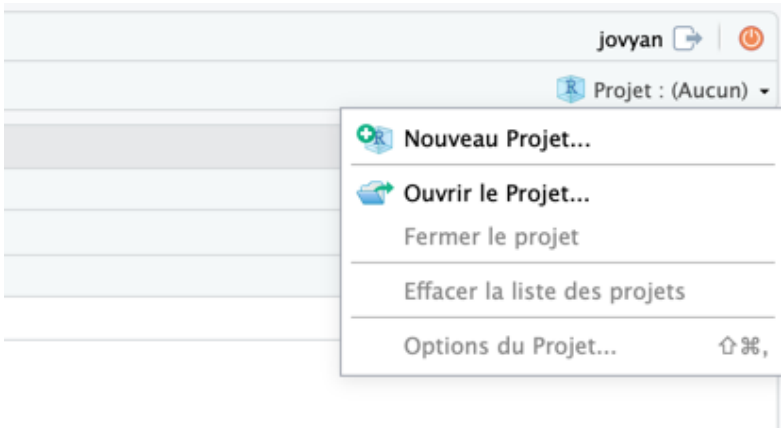
Suivez les explications ci-dessous, repérez les différents éléments dans votre logiciel et effectuez les exercices demandés à l'intérieur des différents fichiers...

Il est également utile de vous familiariser avec les **aide-mémoires** suivants en parallèle (ils contiennent une foule d'information, mais essayer dans un premier temps d'y retrouver déjà ce que vous apprenez ici) :

- **RStudio IDE** (voir https://rstudio.github.io/cheatsheets/html/rstudio-ide.html)
- **Publish and Share with Quarto** (voir https://rstudio.github.io/cheatsheets/html/quarto.html)
- **rmarkdown** (voir https://rstudio.github.io/cheatsheets/html/rmarkdown.html)

Projet dans RStudio

Vous gérez les projets dans RStudio via un menu déroulant en haut à droite de la fenêtre. Ce menu est représenté par une petite boite bleue avec un "R" dedans (en dessous du nom d'utilisateur qui est toujours "jovyan" dans une machine Saturn Cloud et du bouton d'arrêt de session orange tout en haut à droite). Ce menu déroulant permet de créer un nouveau projet, ouvrir un projet existant ou naviguer rapidement entre les derniers projets qui ont été ouverts. Il affiche le nom du projet courant.

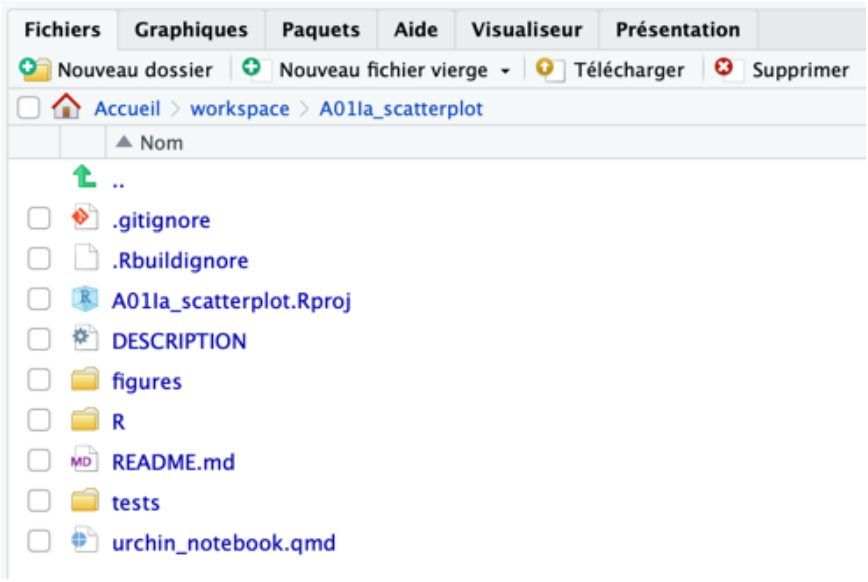


Normalement, vous l'avez déjà utilisé pour créer ce projet. Voyez maintenant ce que cela donne de refermer le projet avec le menu déroulant, et ensuite de le rouvrir. **Vérifiez toujours que RStudio est dans le bon projet en allant lire son nom en haut à droite de la fenêtre avant de travailler**. En effet, il peut arriver que, par distraction, on ouvre un fichier qui n'appartiennent pas au projet. On peut le faire, mais RStudio restera focalisé sur le projet ouvert (ou sur aucun éventuellement). Vous ne pourrez alors, par exemple, pas effectuer de commit, pull et push de vos modifications (voir plus loin).

Structure du projet : dossiers et fichiers

Nous conseillons de structurer un projet d'analyse de données dans GitHub en rangeant les différents fichiers dans des sous-dossiers. Des explications sur l'organisation d'un projet sont disponibles dans l'appendice B.1.1.2 Organisation d'un projet du cours en ligne.

Dans RStudio, vous visualisez le contenu d'un projet et vous naviguez dans ses sous-dossiers à partir de l'onglet **Fichiers** qui se trouve dans le panneau en bas à droite de la fenêtre.



On y retrouve généralement :

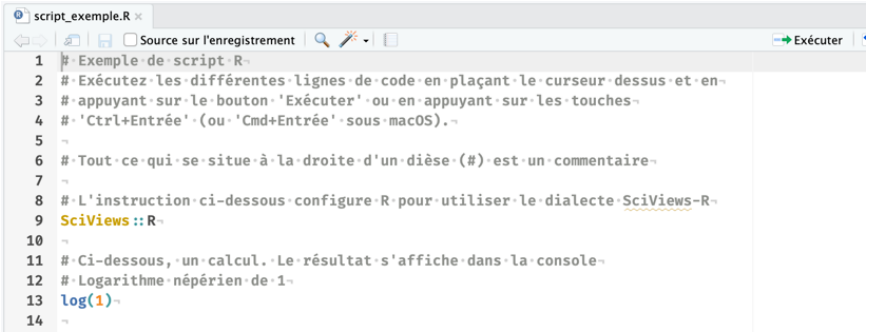
- un fichier `README.md` qui présente le contenu du dépôt (que vous lisez en ce moment)
- un fichier `DESCRIPTION` qui présente des métadonnées relatives au projet.
- dans le cas de RStudio, un fichier `.Rproj`, ici `A01Ia_scatterplot.Rproj` qui ouvre et paramètre le projet dans RStudio. Vous ne l'éditez jamais directement.
- un fichier `.gitignore` pour un dépôt git qui indique les fichiers qui ne doivent pas être repris dans les versions (fichiers temporaires, versions finales des rapports, par exemple). **Vous ne devez pas y toucher.** (si un fichier `.Rbuildignore` est aussi présent, ignorez-le également).
- éventuellement, un dossier `data/` qui contient les données, par exemple, des fichiers au format CSV (ici, il n'y en a pas car nous utiliserons des données qui sont invluses dans un package R)
- en option, un dossier `R/` qui rassemble des fichiers composés d'une suite d'instructions pour effectuer des

analyses, dont l'extension est `.R`. Ce sont les "scripts R" que vous allez découvrir tout au long du cours. Des explications complémentaires se trouvent dans l'appendice B.1.2 Scripts R dans RStudio du cours en ligne.

- éventuellement un dossier `tests/` auquel il ne faut rien toucher. Dans ce cas, une batterie de tests vous est fournie pour vous permettre de vérifier que vous avez réalisé tous les exercices demandés correctement.
- un dossier `figures` qui contient les images et schémas à inclure dans vos documents.
- enfin, un ou plusieurs fichiers `.qmd` qui permettent de créer vos bloc-notes, vos rapports ou vos présentations. Ce sont les fichiers clés du projet. Dans le cas présent, vous avez `urchin_notebook.qmd`.

Fichier R (script R)

Dans ce projet simple, nous avons un script R d'exemple (`script_exemple.R`) dans le sous-dossier `R`. Celui-ci ne fait rien d'utile pour l'analyse centrale du projet concernant l'oursin violet. Il sert uniquement à découvrir à quoi cela ressemble et comment RStudio les gèrent. Ce fichier est éditable, mais vous n'allez rien à y modifier pour cet exercice. Vous allez juste exécuter les différentes instructions qui s'y trouvent.



Pour cela, vous placez le curseur sur la ligne à exécuter (vous commencez au début du document) sans rien sélectionner. Ensuite, vous appuyez sur le bouton **Exécuter** dans la barre d'outil de la zone d'édition (ou utilisez le raccourci clavier `CTRL + Entrée` sous Windows ou `CMD + Entrée` sous macOS). Vous voyez une ou plusieurs lignes qui sont recopiées dans la **Console** (dans le panneau en bas à gauche), et ensuite du texte apparaît relatif au résultat de l'exécution de l'instruction. Ces instructions sont exécutées dans le logiciel R et c'est donc l'onglet **Console** qui vous donne accès au moteur de calcul de R.

Tout ce qui suit un dièse `#` est du commentaire qui n'est pas interprétés par R. Vous l'utiliserez pour documenter vos instructions (c'est important de le faire !)

Vous pouvez aussi entrer directement des instructions dans la **Console**. Essayez, par exemple `1 + 1` et tapez ensuite sur la touche `Entrée`, encore appelée `Retour chariot` à la droite de votre clavier, représentée par une flèche qui pointe vers le bas puis vers la gauche. R effectue le calcul et répond `[1] 2`. Le résultat est bien "2". Le "[1]" entre crochet indique que c'est le premier (et seul) élément d'un vecteur. Si votre instruction génère plusieurs nombres, c'est pratique. Essayez par exemple `1:100 * 2` pour comprendre l'intérêt de ces indices entre crochets devant la réponse (cette fois-ci, un vecteur de cent nombres est renvoyé).

Lorsque vous aurez exécuté toutes les instructions de `script_exemple.R` vous pouvez le refermer et passer à l'étape suivante.

Fichier Quarto

Ouvrez le fichier `urchin_notebook.qmd`. Un document Quarto contient trois régions distinctes :

- un préambule, encore appelé **entête YAML** (du nom du langage utilisé pour y encoder des informations)
- les **zones de texte** mises en forme à l'aide du Markdown
- les **zones de code** R, encore appelées **morceaux** (ou **chunks** en anglais)

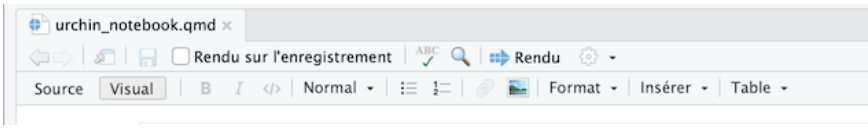
Le préambule débute *toujours* par `---` et se termine *toujours* par `---`. On y indique des informations sous la forme `clé: valeur`. Il ne faut pas d'espace entre la clé et les deux points, par contre il faut un espace entre les deux points et la valeur qui peut être indiquée entre doubles guillemets s'il s'agit de texte. Dans le document, vous éditez l'entête YAML en y indiquant votre nom d'auteur.

```yaml
---
title: "Biométrie des oursins violets"
author: "___"
date: "`r Sys.Date()`"
format:
  html:
    code-fold: true
    code-tools: true
    toc: true
editor: visual
lang: fr
---
```

Les zones de texte respectent les conventions du langage Markdown.

- En mode éditeur visuel (bouton **Visual** just au dessus du texte à gauche enfoncé), les balises de formatage du texte sont cachée et vous avez un rendu type (titres en police plus grande, gras, italique, images insérées, ...), mais pas encore un rendu final qui dépendra du format choisi et d'autres paramètres indiqués éventuellement dans le préambule.

- En mode éditeur source (bouton **Source** enfoncé), les balises de formatage Markdown du texte sont clairement visibles. Les lignes sont numérotées dans la marge à gauche. C'est pratique pour retrouver un endroit dans le document en cas d'erreur qui renseigne un numéro de ligne. Laissez toujours une ligne vide entre les différents éléments du texte, comme un titre, un paragraphe, un morceau R, ainsi qu'après les `---` de la fin d'entête YAML lorsque vous travaillez dans ce mode.



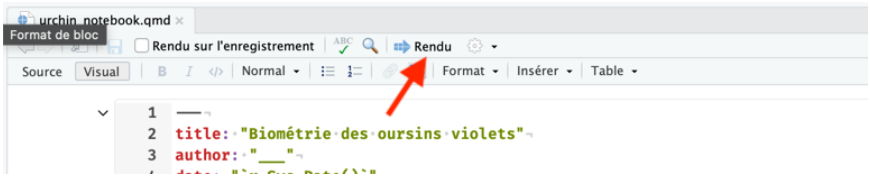
Les zones dédiées au code sont balisées par ```` ```{r} ```` pour des morceaux R (ou simplement `{r}` en mode éditeur visuel), et sont terminées par trois apostrophes inverses (```` ```) lorsqu'elles sont visualisées en mode éditeur source. Dans tous les cas, les morceaux R apparaissent sur un fond de couleur différente (par exemple, grisé si le fond est blanc avec le thème choisi) et un ou deux boutons verts apparaissent en haut à droite du morceau pour contrôler l'exécution des instructions qu'il contient.

```r
{r setup, include=FALSE}
# Nécessaire pour les tests SDD, ne pas utiliser dans un "vrai" projet
if (!"tools:tests" %in% search())
  source(here::here("tests/tools_tests.R"), attach(NULL, name = "tools:tests"
))

```

Des commentaires sont mis à votre disposition dans le document Quarto. Vous verrez qu'il y a des commentaires distincts pour les consignes qui vous expliquent ce que vous devez faire. Les deux types (commentaire normaux et consignes) ne seront *pas* affichés dans le document final. **Vous ne devez pas y toucher.**

Suivez maintenant les consignes du document. Générez les tableaux et graphiques et complétez le document là où c'est demandé (vous devez compléter des instructions R et cocher des interprétations). Ensuite, compilez la version finale de votre bloc-notes (bouton **Rendu**). Si vous avez du mal avec le Markdown, retrouvez toutes les informations dans l'annexe B.1.3 R Markdown/R Notebook du cours en ligne et dans les aides-mémoires.



Vérifications (tests)

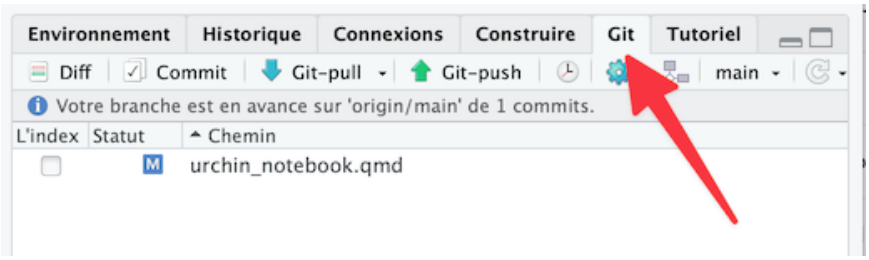
Dans le cas où votre projet contient un sous-dossier `tests/` et qu'un onglet **Construire** est présent dans le panneau en haut à droite, comme ici, cela signifie qu'une batterie de tests est disponible pour vérifier que les exercices sont bien réalisés en tout ou en partie. Ouvrez cet onglet et cliquez sur le bouton **Construire tout**. Les tests indiquent `[réussi]` si cela est correct ou `[échec]` / `[erreur]` en cas de réponse erronée ou d'exercice non réalisé. Pour les tests ratés, vous pouvez cliquer sur le lien et le fichier de test s'ouvre à la ligne correspondante. **Ne modifiez rien** dans ce fichier, mais lisez le commentaire juste en dessous du test pour déterminer quoi faire pour corriger le problème. Lorsque vous pensez avoir corrigé les problèmes, **refaites un rendu (bouton Rendu) du document modifié** et relancez les tests avec le bouton **Construire tout**. Recommencez jusqu'à avoir tous les tests `[réussi]`, ou en tous cas, le plus possible d'entre eux.

Une fois que vous êtes satisfait de votre travail, vous pouvez passer à l'étape suivante qui consiste à synchroniser vos modifications avec GitHub.

Commit et synchronisation des modifications avec GitHub

Avant toute chose, assurez-vous **toujours d'avoir enregistré tous vos documents**. Dans la zone d'édition, observez bien les différents onglets ouverts. Ils portent le nom des fichiers. Si ce nom est en rouge et suivi d'un astérisque, alors il y a des modifications non sauvegardées. Cliquez sur l'onglet et ensuite allez dans le menu **Fichier** -> **Enregistrer**. Repérez l'icône au début de cette entrée de menu. C'est une petite "disquette" bleue. C'est le support qui était utilisé au 20^e^ siècle pour enregistrer des documents informatiques, mais qui a disparu aujourd'hui. L'icône est restée, par contre, pour symboliser l'enregistrement d'un document ! Vous repérez ensuite la même icône dans la barre d'outils. C'est un raccourci utile pour enregistrer, de même que le raccourci `CTRL + S` (enfoncez la touche `CTRL`, puis `S` et enfin relâchez les deux pour activer ce **raccourci clavier**). Il est aussi renseigné à la droite de l'élément de menu (vous devez rouvrir le menu **Fichier** pour le voir). Avec un Mac, ce sera plutôt le raccourci `CMD + S`. Vous avez aussi une entrée de menu **Sauvegarder tout** qui est bien pratique si plusieurs documents sont en cours d'édition. Repérez aussi l'icône et le raccourci clavier correspondants.

Une fois tous vos fichiers sauvegardés, vous allez réaliser successivement un **"commit"**, un **"pull"** et un **"push"** pour créer un point d'enregistrement de version (le "commit") et pour synchroniser vos changements avec le dépôt GitHub (le "pull" et le "push").



Pour cela, vous aller ouvrir l'onglet **Git** qui se trouve dans le panneau supérieur droit dans votre fenêtre. Cliquez ensuite sur le bouton **Commit** dans sa barre d'outils. Une fenêtre s'ouvre qui vous rappelle les modifications réalisées dans chaque fichier du projet. Sélectionnez les fichiers que vous souhaitez reprendre dans votre commit (mettez une coche dans la case devant le nom du fichier ; la plupart du temps, on sélectionne tout). Vous devez ajouter un **message informatif** à ce commit dans la partie de droite. Le message doit présenter de manière claire et concise ce que vous avez réalisé. Par exemple : "Ajout de l'auteur et commentaire du premier tableau." Cliquez enfin sur le bouton **Commit**.

Vous venez de réaliser votre premier commit ! Ne fermez pas encore la fenêtre. Cliquez maintenant sur la flèche bleue afin de faire un "pull" (= récupération des modifications qui ont été faites dans le dépôt GitHub) puis cliquez sur la flèche verte afin de faire un "push" (mise à jour de vos propres modifications dans le dépôt GitHub). Lisez bien les messages qui apparaissent. C'est là que les problèmes éventuels seront exposés, mais dans le cas présent, il ne devrait pas y en avoir. Vous pouvez fermer la fenêtre de commit à présent.

En pratique, ne soyez pas avare de **commit-pull-push**. Vous pouvez en faire autant que vous voulez. Dès que vous avez franchi une étape importante, qu'un exercice est fini, qu'un fichier est édité, etc., faites un commit-pull-push. Pensez toujours à bien vérifier tout de même que RStudio est dans le **bon projet** et que vos fichiers sont bien **sauvegardés**.

Rendez-vous maintenant dans GitHub afin de vérifier que vos modifications et votre commit sont bien répercutés dans le dépôt en ligne. Pour y accéder, retrouvez votre projet dans l'organisation GitHub https://github.com/BioDataScience-Course et ensuite, lisez ce qui apparaît en haut de la page : le texte de votre commit doit s'y trouver. Et si vous naviguez vers les fichiers qui ont été édités et que vous affichez leur contenu, vous devez y retrouver vos modifications.

Il est très important d'effectuer cette dernière vérification ! Chaque année, des étudiants perdent des points parce qu'il ont oublié de faire un dernier commit et/ou un push. Par conséquent, la fin de leur travail n'apparaît pas dans GitHub. Or la seule version que vos enseignants voient est celle dans GitHub. Votre travail local dans votre machine Saturn Cloud ne leur est pas accessible !

Une fois tout cela effectué, vous avez terminé et devriez être plus à l'aise dans l'utilisation de base de RStudio pour éditer des documents et les compiler en forme finale, ainsi que dans l'utilisation principale de git et GitHub via le clonage de projet, les commits, pulls et pushes.

*Une fois terminé, vous pouvez clôturer la session en cliquant sur le bouton orange tout en haut à droite de la fenêtre. Cela va refermer proprement votre machine Saturn Cloud. Dans la fenêtre Saturn Cloud, cliquez aussi sur le bouton **Stop** au niveau de votre ressource (prenez l'habitude de le faire toujours en fin de session... l'usage de vos machines virtuelles est facturé à l'heure).*

Utilisation de l'IA

Dans le cadre de votre travail, vous avez le droit d'être aidé par l'intelligence artificielle. Le chatbot SciViews est disponible dans votre RStudio sur Saturn Cloud via l'addin **Help**. Il répond aux questions concernant le langage R, les statistiques et la science des données.