UNIVERSITÉ SORBONNE PARIS NORD

iame UMR 1137
Infection • Antimicrobials • Modelling • Evolution

RAPPORT DE STAGE

du 21 mai au 19 juillet 2024

Développement d'un package R pour la détection de sélection agissant sur les régions codantes

Naline AHOYO

Étudiante en M1 IBM

Julie Marin

Tutrice de stage

REMERCIEMENT

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réussite de ce stage.

Tout d'abord, je tenais premièrement à remercier ma tutrice de stage Madame Julie MARIN pour sa confiance, son accompagnement et ses précieux conseils tout au long de cette expérience. Ses retours constructifs m'ont permis de progresser tant sur le plan technique que professionnel.

Je remercie aussi vivement l'unité IAME de m'avoir accueillie au sein de leurs équipes pendant 8 semaines et notamment l'équipe EVRest pour leur bienveillance.

Plus particulièrement, je remercie Monsieur Thibaut MOREL-JOURNEL membre du projet, pour sa confiance et sa disponibilité. Ce fut une expérience très enrichissante, autant du point de vue personnel que professionnel.

Enfin, j'exprime toute ma reconnaissance à l'ensemble de l'équipe EVRest pour leur accueil bienveillant, leur disponibilité et leur soutien. Travailler dans un environnement aussi enrichissant et collaboratif a été une véritable opportunité pour moi. Je ressors de cette expérience avec plus de connaissances et surtout, une meilleure façon d'appréhender le travail dans le domaine informatique.

RESUME

Au cours de mon stage chez IAME, j'ai eu l'opportunité de participer activement au développement d'un package en R dédié à l'analyse de séquences génétiques. Ce projet m'a permis de développer des compétences solides en programmation, en particulier dans la création, l'organisation et la maintenance d'un package. J'ai également approfondi ma compréhension des concepts liés à la gestion des données et aux algorithmes d'analyse. Durant cette période, j'ai rencontré plusieurs défis, tels que la gestion des erreurs, l'optimisation du code et la compatibilité entre différentes versions de R. Ces difficultés m'ont amené à améliorer mes compétences en résolution de problèmes et à mieux comprendre les meilleures pratiques de développement. Ce stage a été une expérience formatrice, me préparant efficacement pour de futurs projets techniques et professionnels.

ABSTRACT

During my internship at IAME, I had the opportunity to actively contribute to the development of an R package designed for the analysis of genetic sequences. This project allowed me to build strong programming skills, particularly in the creation, organization, and maintenance of a package. I also deepened my understanding of data management and analysis algorithms. Throughout the internship, I encountered several challenges, such as error handling, code optimization, and ensuring compatibility across different R versions. These obstacles helped me enhance my problem-solving skills and grasp best practices in software development. Overall, this internship was a formative experience that has effectively prepared me for future technical and professional projects.

Table des matières

INTRODUCTION

Etant en première année de master Informatique Biomédicale à l'Université Sorbonne Paris Nord, j'ai eu l'opportunité de réaliser un stage de huit semaines dans ce domaine afin de travailler concrètement sur des notions étudiées au cours de cette année scolaire et en découvrir de nouvelles. En effet du 21 mai au 19 juillet 2024, j'ai eu la chance de travailler au sein de l'unité IAME (**Infection, Antimicrobiens, Modélisation, Évolution)** une unité appartenant à l'INSERM situé sur le campus de l'Université de Paris à la Faculté de médecine (site Bichat).

IAME est une unité qui allie recherche fondamentale et recherche clinique avec pour but de lutter contre les maladies infectieuses. Elle regroupe environ 200 personnes réparties en plusieurs équipes. Lors de mon stage j'ai rejoint l'équipe EVRest dirigée par le professeur Erick Denamur, également directeur principale de l'unité. J'ai choisi de faire mon stage avec cette équipe car la mission proposée éveillait particulièrement mon intérêt.

Ma mission consistait à développer un package R pour la détection de la sélection naturelle agissant sur les régions codantes des séquences d'ADN. Plus spécifiquement, ce package permettra le calcul du ratio (Ka/Ks) pour des séquences codantes alignées. En plus du développement du package, j'ai implémenté une nouvelle fonction afin de faciliter et d'améliorer l'utilisation du package, la majorité des fonctions à implémenter étant déjà écrites à mon arrivée.

Au sein de l'équipe EVRest, en charge de ce projet, j'ai été encadrée par Madame Julie MARIN, avec qui je m'entretenais de manière quotidienne et par Monsieur Thibaut MOREL-JOURNEL hebdomadairement en cas de besoin. Ma tutrice de stage me transmettait des missions et m'aiguillait en cas de problème.

Afin que vous compreniez mieux le contexte de mon stage ainsi que ma place dans la conception du package R, je présenterai dans un premier mon organisme de stage (IAME) puis dans un second temps, le projet informatique réalisé pour vous faire part de mon travail.

1. Présentation de l'organisme d'accueil et du projet

1.1. Présentation de l'organisme d'accueil

Nom de l'organisme : IAME (Infection, Antimicrobials, Modelling, Evolution)

Historique : L'unité IAME a été créé en 2014, ses tutelles sont l'Institut national de la santé et de la recherche médicale (INSERM, l'Université Paris Cité et l'Université Sorbonne Paris Nord. Elle dépend de deux départements de l'Alliance française pour les sciences de la vie et la santé (AVIESAN), les maladies infectieuses et la santé publique. Elle est située principalement sur le campus de l'Université Paris Cité à la Faculté de médecine (site Bichat) au nord de Paris et reliée à plusieurs CHU à proximité. Elle est dirigée par **Erick Denamur** (directeur) et **France Mentré** (directrice adjointe).

Mission et vision : L'objectif principal de l'unité IAME est de comprendre les mécanismes sous-jacents à la propagation des infections et au développement de la résistance aux antimicrobiens. IAME s'est consacrée à l'étude des interactions entre les agents pathogènes, aux traitements antimicrobiens, et la modélisation de leur évolution. Cette unité valorise la recherche interdisciplinaire, l'innovation et la collaboration pour trouver des solutions durables aux problèmes de santé publique.

Structure organisationnelle : IAME est composée de plusieurs équipes spécialisées dans différents aspects de la recherche sur les infections et les antimicrobiens. Les équipes EVRest, BIPID et DeSCID ont été créées avec IAME en 2014, et EPIC a été récemment créée (avril 2024) en tant qu'équipe ATIP/AVENIR de l'INSERM par d'anciens membres de BIPID.

1.2. Objectifs du stage : développement d'un package R pour le calcul du ratio Ka/Ks

Dans le cadre de mon stage au sein de l'équipe EVRest, j'ai entrepris le développement d'un package R destiné au calcul du ratio Ka/Ks. L'objectif principal du stage était de créer un package, nommé Ratio, permettant de calculer le ratio Ka/Ks pour des séquences codantes alignées. Le ratio Ka/Ks est une mesure essentielle pour détecter la sélection naturelle dans les gènes, en comparant le taux de substitutions non-synonymes (Ka) au taux de substitutions synonymes (Ks).

1.3. Détection de la sélection naturelle avec le calcul du ratio Ka/Ks

Concept de la sélection naturelle : La sélection naturelle est un processus par lequel certaines caractéristiques héréditaires deviennent plus ou moins fréquentes dans une population en fonction de l'avantage qu'elles confèrent en termes de survie et de reproduction. Ce mécanisme est un pilier de la théorie de l'évolution, permettant de comprendre comment les organismes s'adaptent à leur environnement.

Applications pratiques : Le ratio Ka/Ks est utilisé pour détecter la sélection naturelle dans divers contextes, comme l'étude des gènes responsables de la résistance aux antimicrobiens, l'évolution des pathogènes, et l'adaptation des organismes à de nouveaux environnements. Par exemple, il peut identifier des gènes sous sélection positive chez les bactéries résistantes aux antibiotiques, fournissant des cibles potentielles pour de nouveaux traitements.

Le ratio Ka/Ks : Le ratio Ka/Ks compare le taux de substitutions non synonymes (Ka) au taux de substitutions synonymes (Ks) dans les gènes codants. Les substitutions non synonymes entraînent des changements dans les acides aminés des protéines, tandis que les substitutions synonymes n'ont pas d'effet sur la séquence protéique.

- **Ka/Ks > 1 :** Indique une sélection positive, suggérant que les mutations sont bénéfiques et favorisées.
- **Ka/Ks = 1 :** Indique une évolution neutre, où les mutations se fixent par dérive génétique.
- **Ka/Ks < 1 :** Indique une sélection négative, où les mutations sont délétères et éliminées.

Méthodologies de calcul : Illustration du calcul du Ka/Ks pour détecter la sélection naturelle

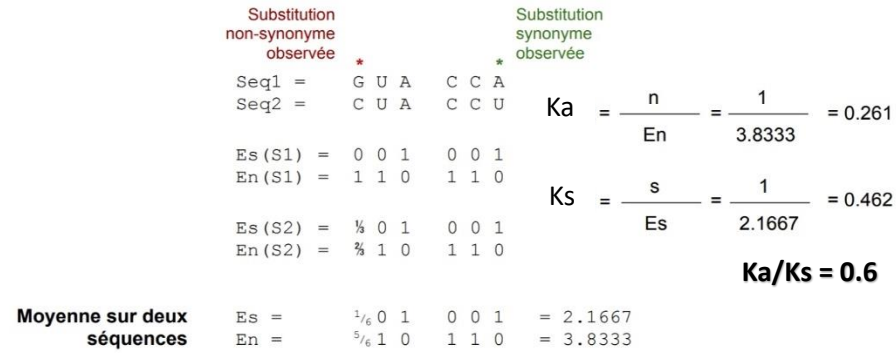


Figure 1. Calcul du ratio Ka/Ks pour détecter la sélection naturelle

Voici une explication détaillée des différentes parties de la figure 1:

1. **Séquence 1 : GUA**

   Le premier codon de seq1 (**GUA**) code pour la valine (Val). Lorsqu'on effectue la substitution des nucléotides en 1ere position on obtient :

   - UUA = Leu
   - CUA = Leu
   - AUA= Ile.

Une substitution à la 1ère position ne donne aucun codon synonyme à la valine. On a donc 3/3 (*i.e.* 1) possibilités d'observer une mutation non-synonyme (En(S1)) et 0 possibilité d'obtenir une mutation synonyme (Es (S1))

Lorsqu'on fait la substitution des nucléotides en 2nd position on observe le même résultat que pour la 1ère position. Quant à la substitution des nucléotides de la 3ème position on obtient :

- GUU = **Val**
- GUC = **Val**
- GUG = **Val.**

Toutes les substitutions de la 3ème position sont synonymes, elles correspondent à la valine. On a donc 3/3 possibilités d'observer une mutation synonyme (Es (S1)) et 0 possibilité d'observer une mutation non-synonyme (En (S1))

Le même processus de calcul est effectué pour le deuxième codon de la séq 1 (**CCA).**

2. **Séquence 2 : CUA**

La séq 2 (**CUA)** est un codon qui code pour la Leucine (Leu). Lorsqu'on effectue la substitution des nucléotides en 1ere position on obtient :

- AUA = Ile
- GUA = Val
- UUA= **Leu.** Pour cette substitution en 1ère position on a donc 2/3 possibilités d'observer une mutation non-synonyme (En (S2)) et 1/3 possibilité d'observer une mutation synonyme (Es (S2))

Lorsqu'on fait la substitution des nucléotides en 2nd position on a 0 possibilité d'observer une mutation synonyme (Es (S2)) et 3/3 possibilités d'observer une mutation non-synonyme (En (S2)). Quant à la substitution des nucléotides de la 3eme position on obtient :

- CUG = **Leu**
- CUC = **Leu**
- CUU= **Leu.**

Toutes les substitutions de la 3ème position donnent un codon synonyme à la **Leucine**. On a donc 3/3 possibilités d'observer une mutation synonyme (Es (S2)) et 0 possibilité d'observer une mutation non-synonyme (En (S2)).

Le même processus de calcul est effectué pour le deuxième codon (**CCU)** de la séquence 2.

3. **Calcul de Es (Nombre de sites synonymes possibles) et En (Nombre de sites non-synonymes possibles) :**

Pour chaque position on calcule la moyenne sur les deux séquences et on additionne le résultat :

- Es = 1/6 + 0 + 1 + 0 + 0 + 1 = 2.1667
- En = 5/6 + 1 + 0 + 1 + 1 + 0 = 3.8333

4. **Calcul du ratio Ka/Ks :**

Pour faire le calcul du Ka/Ks on utilise également **n** (nombre de mutation non-synonymes observés) et **s** (nombres de mutation synonymes observés). Dans notre exemple de calcul le nombre mutation synonyme observée est de 1 (en vert) et le nombre de mutation non-synonyme observée est de 1 (en rouge).

Enfin, le ratio Ka/Ks est ensuite calculé. Dans le cas de l'exemple on obtient : Ka/Ks = 0.6

2. Présentation du Package R

2.1. Description générale du package `Ratio`

Au cours de mon stage, j'ai développé le package Ratio en langage R. À l'exception d'une fonction que j'ai moi-même implémentée (décrite ci-dessous), les autres fonctions étaient déjà écrites. Mon travail a consisté à transformer un ensemble de fonctions préexistantes en un package R structuré, documenté et validé, prêt à être utilisé.

Le package **Ratio** est conçu pour faciliter le calcul du ratio Ka/Ks, qui est un indicateur clé de l'action de la sélection naturelle sur les gènes. Un ratio Ka/Ks supérieur à 1 suggère une sélection positive, un ratio égal à 1 indique une évolution neutre, et un ratio inférieur à 1 indique une sélection purifiante.

Le package Ratio fournit des outils pour :

- Lire des séquences codantes alignées à partir de fichiers FASTA.
- Calculer le nombre de substitutions synonymes et non-synonymes.
- Calculer les ratios Ka/Ks pour évaluer la pression de sélection sur les gènes.
- Visualiser les résultats pour faciliter l'interprétation.

2.2. Fonctionnalités principales du package

Le package `Ratio` offre plusieurs fonctionnalités principales qui simplifient le processus pour le calcul et l'analyse des ratios Ka/Ks. Voici les principales fonctionnalités détaillées :

Lecture de séquences FASTA : Le package permet de lire des fichiers FASTA, qui sont couramment utilisés pour stocker des séquences nucléotidiques ou protéiques. La fonction utilisé « **read_fasta »** permet de lire des séquences codantes alignées à partir de fichiers FASTA et de les préparer pour le calcul des ratios.

Calcul des substitutions synonymes et non-synonymes : Le cœur du package Ratio réside dans sa capacité à calculer les substitutions synonymes (Ks) et non-synonymes (Ka). La fonction « **syn_or_nonsyn** » détermine si une substitution est synonyme ou non synonyme en comparant les codons d'une séquence donnée aux codons de la séquence de référence.

Calcul des ratios Ka/Ks : Une fois le nombre de substitutions calculé, le package fournit une fonction pour calculer le ratio Ka/Ks. La fonction « **kaks »** calcule les ratios Ka/Ks en utilisant les substitutions identifiées et les substitutions possibles pour chaque type de substitution.

3. Méthodologie de Développement

3.1. Création de la structure du package

La création du package R nécessite une organisation méthodique et structurée des fichiers et des répertoires pour assurer la modularité et la facilité d'utilisation. Voici les étapes que j'ai suivies pour créer la structure du package Ratio :

- **Initialisation du package :**
  - ✓ Utilisation de la fonction **'' usethis ::create_package(Ratio)''** pour créer la structure de base du package.
  - ✓ Cette commande génère les dossiers et fichiers de base nécessaire, tels que **DESCRIPTION, NAMESPACE, et le répertoire R/.**
- **Description et Données :**
  - ✓ Modification du fichier DESCRIPTION pour inclure les données du package : nom, version, auteur, et une brève description.
  - ✓ Ajout des dépendances nécessaires dans le champ **Imports.** (Annexe 2)
- **Organisations des Dossiers :**
  - ✓ **R/** : Contient les scripts R définissant les fonctions du package.
  - ✓ **man/** : Contient la documentation des fonctions, générée automatiquement avec roxygen2.
  - ✓ **data/** : Utilisé pour stocker les jeux de données internes du package.
  - ✓ **inst/extdata/** : Contient les fichiers de données externes nécessaires pour les exemples et tests. (Annexe 3)

3.2. Implémentation des fonctions principales

Voici un résumé des principales fonctions implémentées :

Fonction fasta_single_line

Au cours de mon stage, j'ai eu à développer la fonction '**fasta_single_line'** moi-même. Cette fonction a été conçue pour transformer des fichiers FASTA en format single-line, où chaque séquence est sur une seule ligne. La fonction lit un fichier FASTA, élimine les retours à la ligne à l'intérieur des séquences et sauvegarde le fichier en format single-line. Elle est particulièrement utile pour préparer les fichiers de séquences avant de les utiliser dans les analyses plus complexes, comme le calcul des ratios Ka/Ks.

Objectif : Convertir les fichiers FASTA multi-lignes en format single-line, facilitant ainsi la manipulation et l'analyse des séquences.

Description :

- Lit les séquences d'un fichier FASTA d'entrée.
- Convertit chaque séquence multi-lignes en une seule ligne.
- Écrit les séquences converties dans un fichier de sortie.

Fonction kaks

La fonction Kaks constitue le cœur du package. Elle prend en charge l'ensemble du processus de calcul des ratios Ka/Ks pour les séquences codantes alignées. Cette fonction principale fait appel à plusieurs fonctions auxiliaires, chacune jouant un rôle spécifique dans le calcul du ratio.

Objectif : Calculer les ratios de substitutions non-synonymes (Ka) et synonymes (Ks) entre des paires de séquences pour analyser les pressions sélectives sur les gènes.

Description :

- Charge les tables de données nécessaires.

- Convertit les séquences FASTA multi-lignes en single-line.
- Génère les combinaisons de paires de séquences à comparer.
- Parallélise le calcul des ratios Ka/Kspour chaque paire de séquences.
- Compte les substitutions synonymes et non-synonymes et calcule les ratios Ka/Ks.
- Compile les résultats dans un tableau final.

Autres Fonctions auxiliaires

En plus des fonctions principales, plusieurs fonctions auxiliaires ont été développées pour assister dans le traitement et l'analyse des séquences. Ces fonctions jouent un rôle crucial en supportant les calculs et en assurant la précision des résultats finaux. Voici une description des principales fonctions utilitaires implémentées :

- **Fonction comb_tab :** Génère toutes les combinaisons de paires de séquences à comparer.
- **Fonction read_codons :** Lit et segmente les séquences en codons.
- **Fonction codon_counter :** Compte le nombre de chaque codon dans une séquence et ajoute les informations synonymes et non-synonymes.
- **Fonction find_mismatches :** Identifie les positions des codons différents entre deux séquences.
- **Fonction syn_or_nonsyn :** Détermine si une substitution est synonyme ou non-synonyme.
- **Fonction compute_kaKs :** Calcule les ratios Ka/Ks à partir des tableaux de mutations.
- **Fonction final_KaKs :** Agrège les résultats et calcule le ratio final Ka/Ks pour chaque paire de séquences.

Pour une meilleure compréhension et pour des exemples concrets d'implémentation, toutes ces fonctions sont illustrées en annexe de ce rapport (Annexes 1-3).

3.3. Gestion des dépendances

La gestion des dépendances est essentielle pour garantir que le package Ratio fonctionne correctement dans divers environnements et configurations R. Voici les pratiques suivies :

- **Déclaration des dépendances**
  - ✓ Les dépendances essentielles ont été déclarées dans le fichier DESCRIPTION sous les champs Imports ;
    Exemple : Imports :
    seqinr,
    foreach,
    doParallel
- **Utilisation des dépendances**

- ✓ Les fonctions des packages importés sont appelées en utilisant la notation '' **::**'' pour éviter les conflits de noms et assurer une utilisation explicite des dépendances.
  Exemple : stats::complete.cases

❖ **Test de compatibilité**

- ✓ Les tests unitaires incluent des vérifications pour s'assurer que les dépendances fonctionnent correctement avec les fonctions du package Ratio.
- ✓ Les tests ont été exécutés sur différentes versions de R pour vérifier la compatibilité ascendante. Personnellement, j'ai effectué les tests sur la **version 4.3.2** de R sur Windows, tandis que ma tutrice de stage a vérifié la fonctionnalité sur sa propre installation, qui utilise la **version 4.2.3** sur Mac. Ces tests ont permis de garantir que le package fonctionne correctement sur plusieurs versions de R, assurant ainsi une robustesse et une portabilité accrues.

4. Documentation et Tests

4.1. Documentation avec roxygen2

Pour garantir que le package Ratio soit bien documenté et facilement compréhensible par les utilisateurs, j'ai utilisé le package **roxygen2** de R pour générer automatiquement la documentation des fonctions. Roxygen2 permet d'écrire la documentation directement dans les fichiers de code source sous forme de commentaires structurés. Voici les étapes que j'ai suivies pour documenter les fonctions :

Ajout de commentaires roxygen2 : J'ai ajouté des blocs de commentaires au-dessus de chaque fonction en utilisant la syntaxe roxygen2. Chaque bloc de commentaires comprend :

- Un titre
- Une description de la fonction
- Les arguments de la fonction et leur description
- La valeur retournée par la fonction (Annexe 1)

4.2. Vérification et validation du package

Pour m'assurer que le package Ratio respecte les standards de qualité et de conformité de CRAN, j'ai effectué une série de vérifications et de validations :

1. **Vérification du package :**
   - J'ai utilisé la commande devtools::check() pour effectuer une vérification complète du package. Cette commande vérifie la conformité du package avec les règles de CRAN, incluant la documentation, les tests, et la structure du package.
2. **Résolution des erreurs et avertissements :**
   - J'ai analysé les messages d'erreurs, de notes et d'avertissements générés par devtools::check() et j'ai apporté les corrections nécessaires pour assurer que le package passe toutes les vérifications sans problème.
3. **Validation finale :**

- Une fois toutes les erreurs et avertissements corrigés, j'ai effectué une validation finale en testant le package dans différents environnements R et pour différents systèmes d'exploitation pour m'assurer de sa portabilité et de sa robustesse.

5. Difficultés rencontrées et solutions apportées

Au cours du développement de mon package Ratio j'ai rencontré plusieurs défis techniques qui ont nécessité des solutions innovantes et une approche minutieuse pour assurer l'efficacité du package.

Problèmes avec les dépendances :

L'un des principaux défis a été la gestion des dépendances. Plusieurs packages R nécessaires pour le fonctionnement de Ratio avaient des versions incompatibles entre elles. Pour résoudre ce problème, j'ai dû spécifier des versions précises de chaque package dans le fichier DESCRIPTION et m'assurer que ces versions étaient installées avant de tester et de construire le package.

Erreurs lors de l'utilisation de `devtools::check` :

L'exécution de `devtools::check` a révélé plusieurs erreurs et avertissements, principalement liés à la documentation et aux dépendances. Pour résoudre ces problèmes, j'ai passé en revue chaque message d'erreur, ajusté la documentation en conséquence, et vérifié que toutes les dépendances étaient correctement déclarées.

Gestion des messages d'erreur :

Pour améliorer la convivialité du package et faciliter le débogage, j'ai dû écrire des arguments spécifiques dans le code afin de générer des messages d'erreur explicites. Cela permet aux utilisateurs de comprendre plus facilement ce qui ne va pas et comment rectifier les problèmes.

Problème de lecture des fichiers FASTA :

Initialement, la lecture des fichiers FASTA présentait des difficultés, notamment avec les séquences multi-lignes. La solution a été de développer la fonction fasta_single_line pour convertir les fichiers en format single-line, simplifiant ainsi le traitement des séquences.

Ces difficultés ont non seulement mis à l'épreuve mes compétences en développement de packages R, mais elles m'ont également permis d'acquérir une expérience précieuse dans la gestion des dépendances et le débogage approfondi.

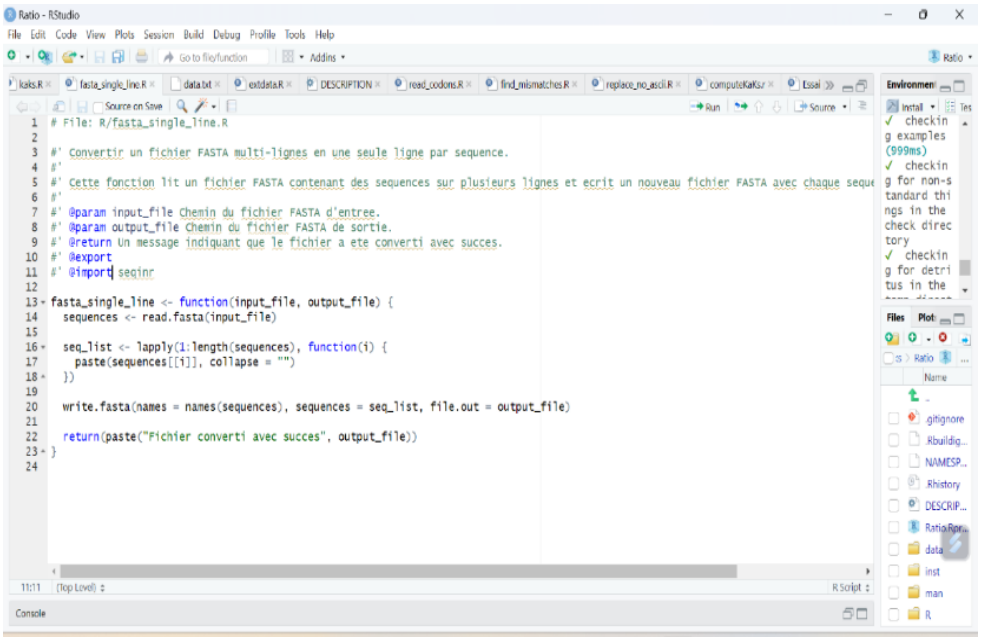
CONCLUSION

En conclusion ce stage m'a permis de développer des compétences techniques en programmation et de comprendre en profondeur le processus de création d'un package R. J'ai été confronté à plusieurs défis, tels que la gestion des fichiers de données et la compatibilité du code, ce qui m'a permis de renforcer mes capacités en résolution de problèmes.

Grâce à l'accompagnement de ma tutrice et de l'équipe EVRest, j'ai pu améliorer mon sens de l'organisation et du travail en équipe. Cette expérience m'a préparée pour des projets futurs et ouvert des perspectives d'améliorations pour le package développé.

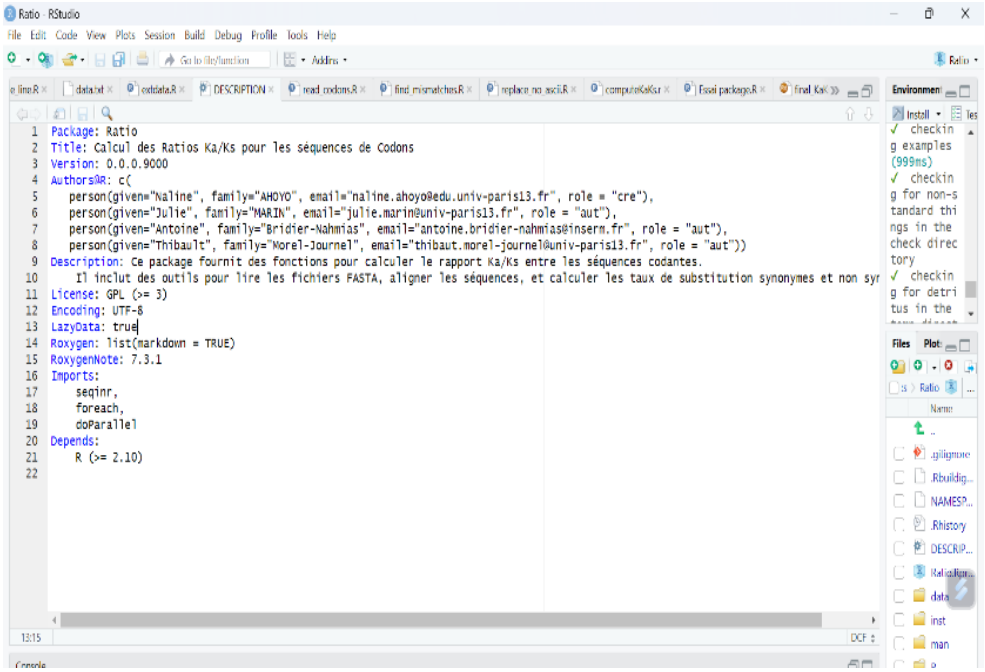
ANNEXES

1. Documentation avec Roxygen2



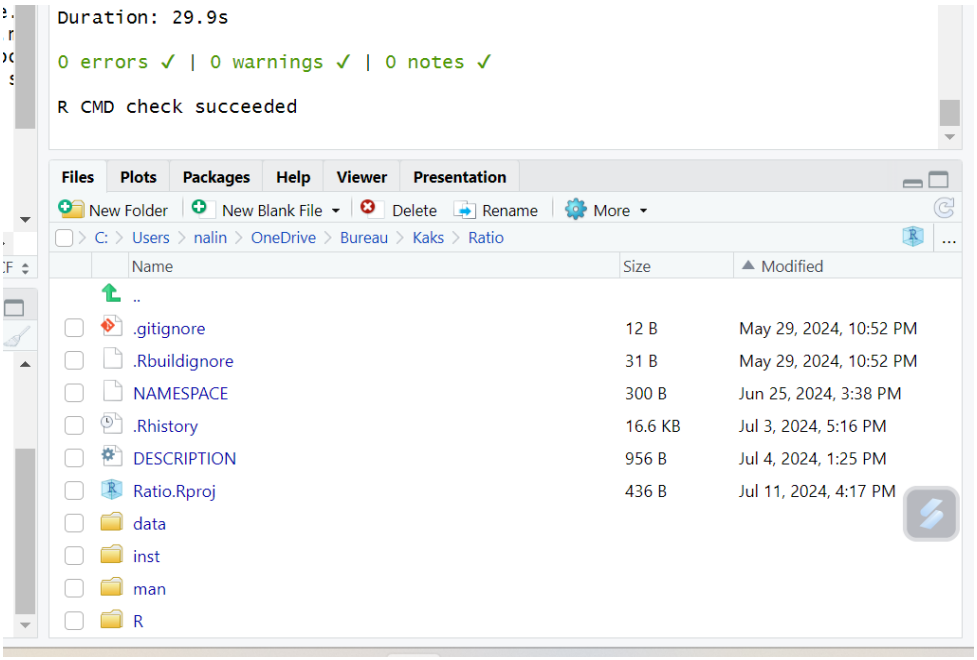
```
# File: R/fasta_single_line.R

#' Convertir un fichier FASTA multi-lignes en une seule ligne par sequence.
#'
#' Cette fonction lit un fichier FASTA contenant des sequences sur plusieurs lignes et ecrit un nouveau fichier FASTA avec chaque seque
#'
#' @param input_file Chemin du fichier FASTA d'entree.
#' @param output_file Chemin du fichier FASTA de sortie.
#' @return Un message indiquant que le fichier a ete converti avec succes.
#' @export
#' @import seqinr

fasta_single_line <- function(input_file, output_file) {
  sequences <- read.fasta(input_file)

  seq_list <- lapply(1:length(sequences), function(i) {
    paste(sequences[[i]], collapse = "")
  })

  write.fasta(names = names(sequences), sequences = seq_list, file.out = output_file)

  return(paste("Fichier converti avec succes", output_file))
}
```

2. Description et Données



```
Package: Ratio
Title: Calcul des Ratios Ka/Ks pour les séquences de Codons
Version: 0.0.0.9000
Authors@R: c(
    person(given="Naline", family="AHOYO", email="naline.ahoyo@edu.univ-paris13.fr", role = "cre"),
    person(given="Julie", family="MARIN", email="julie.marin@univ-paris13.fr", role = "aut"),
    person(given="Antoine", family="Bridier-Nahmias", email="antoine.bridier-nahmias@inserm.fr", role = "aut"),
    person(given="Thibault", family="Morel-Journel", email="thibaut.morel-journel@univ-paris13.fr", role = "aut"))
Description: Ce package fournit des fonctions pour calculer le rapport Ka/Ks entre les séquences codantes.
    Il inclut des outils pour lire les fichiers FASTA, aligner les séquences, et calculer les taux de substitution synonymes et non syn
License: GPL (>= 3)
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.3.1
Imports:
    seqinr,
    foreach,
    doParallel
Depends:
    R (>= 2.10)
```

3. Organisation des dossiers


Duration: 29.9s
0 errors ✓ | 0 warnings ✓ | 0 notes ✓
R CMD check succeeded
Files
Plots
Packages
Help
Viewer
Presentation
New Folder
New Blank File
Delete
Rename
More
C: Users nalin OneDrive Bureau Kaks Ratio
Name
Size
Modified
..
.gitignore 12 B May 29, 2024, 10:52 PM
.Rbuildignore 31 B May 29, 2024, 10:52 PM
NAMESPACE 300 B Jun 25, 2024, 3:38 PM
.Rhistory 16.6 KB Jul 3, 2024, 5:16 PM
DESCRIPTION 956 B Jul 4, 2024, 1:25 PM
Ratio.Rproj 436 B Jul 11, 2024, 4:17 PM
data
inst
man
R