
**FACULTAD DE CIENCIAS AGRARIAS**

**UNIVERSIDAD NACIONAL DE ROSARIO**

# MEJORA Y ACTUALIZACIÓN DEL PAQUETE DE R: CleanBSequences

**DRA. FLORENCIA I. POZZI**

**TRABAJO FINAL PARA OPTAR AL TITULO DE ESPECIALISTA EN BIOINFORMÁTICA**

**DIRECTOR: DRA. SILVINA A. FELITTI**

**2022**

**MEJORA Y ACTUALIZACIÓN DEL PAQUETE DE R: CleanBSequences**

BIÓL. FLORENCIA I. POZZI
Bióloga– Universidad Nacional de Córdoba

Este Trabajo Final es presentado como parte de los requisitos para optar al grado académico de Especialista en Bioinformática, de la Universidad Nacional de Rosario y no ha sido previamente presentada para la obtención de otro título en esta u otra Universidad. El mismo contiene los resultados obtenidos en investigaciones llevadas a cabo en la Facultad de Ciencias Agrarias de la UNR, durante el período comprendido entre el año 2019 y 2022, bajo la dirección de la Dra. Silvina A. Felitti.

Nombre y firma del autor

Nombre y firma del Director

Nombre y firma del Co - Director

Defendida: .....de 20__.

.

## AGRADECIMIENTOS

Quiero agradecer a mi directora, la Dra. Silvina Felitti, por brindarme la posibilidad de realizar este trabajo final, por su honorable dirección, transmisión de conocimientos, predisposición, amabilidad, positivismo, y sobre todo por su excelencia como persona.

A la Facultad de Ciencias Agrarias de la Universidad Nacional de Rosario (FCA-UNR) y al Instituto de Investigaciones en Ciencias Agrarias de Rosario (IICAR) por permitirme disponer de sus instalaciones.

A cada uno de los que pertenecen a la sala de estudios de postgrado y a mis compañeros de la especialización, porque todos han contribuido a mi formación durante la realización de este trabajo final.

A mis compañeras de la cátedra de Microbiología.

A mí querida familia en este plano: Mamá, Gusti y Ago.

A mi sostén desde hace tantos años: Marcelo

A la razón de mi vida: Emma

Y sobre todo a mi PAPÁ…

A todos, muchas gracias!!

## PUBLICACIONES Y PRESENTACIONES A CONGRESOS

Los resultados informados en este trabajo de tesis fueron volcados en las siguientes publicaciones y reuniones científicas:

POZZI, FLORENCIA I.; FELITTI, SILVINA A. (2021) Prediction of gene silencing in *Arabidopsis thaliana* using decision trees and support vectors machines algorithms. XI Congreso Argentino de Bioinformática y Biología Computacional.

POZZI, FLORENCIA I.; GREEN, GISELA Y.; BARBONA, IVANA G.; RODRÍGUEZ, GUSTAVO R.; FELITTI, SILVINA A. (2020) CleanBSequences: an efficient curator of biological sequences in R. MOLECULAR GENETICS AND GENOMICS.

POZZI, FLORENCIA I; FELITTI, SILVINA A. (2020) Development of new function to improve and update the R package: CleanBSequences 1st Latin American Congress of Women in Bioinformatics and Data Science.

POZZI, FLORENCIA I; GREEN, GISELA Y.; RODRÍGUEZ, GUSTAVO R.; FELITTI, SILVINA A. (2018) Desarrollo de una función en R para agilizar y automatizar la limpieza de secuencias obtenidas por la técnica de cDNA-AFLP. Conferencia Latinoamericana sobre Uso de R en Investigación + Desarrollo.

# ABREVIATURAS Y SÍMBOLOS

AFLP: Polimorfismo en la longitud de los fragmentos amplificados, del inglés *amplified fragment length polymorphism*.
cDNA-AFLP: Polimorfismo en la longitud de los fragmentos amplificados en ADN copia, del inglés *cDNA amplified fragment length polymorphism*.
CRAN: del inglés *Comprehensive R Archive Network*.
MSAP: Polimorfismos amplificados sensibles a la metilación, del inglés *Methylation-sensitive amplified polymorphisms*.
PCR: Reacción en cadena de la polimerasa, del inglés del inglés *polymerase chain reaction*.
PF: Cebador directo, del inglés *Primer Forward*.
PR: Cebador reverso, del inglés *Primer Reverse*.

**RESUMEN**

Este trabajo presenta un nuevo método y herramienta mejorada y actualizada para resolver un problema común de los biólogos moleculares y genetistas que utilizan marcadores moleculares en sus investigaciones y desarrollos científicos: la curación de secuencias. Los estudios ómicos realizados por biólogos moleculares y genetistas suelen implicar el uso de marcadores moleculares. AFLP, cDNA-AFLP y MSAP son ejemplos de marcadores que brindan información a nivel de genómica, transcriptómica y epigenómica, respectivamente. Estos tres tipos de marcadores moleculares usan adaptadores que son la plantilla para la amplificación por PCR. Las secuencias de los adaptadores tienen que ser eliminadas para el análisis de los resultados. Dado que en estos estudios se suele obtener un gran número de secuencias, esta limpieza de los datos podría demandar mucho tiempo y trabajo. Para automatizar este trabajo, previamente se creó un paquete R, llamado CleanBSequences cuya versión inicial fue 0.4.0, que permitía curar las secuencias de forma masiva, rápida, sin errores y que se pueden usar sin conexión. La curación se realizaba alineando los primers forward y/o reverse o los extremos de los vectores de clonación con las secuencias a eliminar. Después del alineamiento, se generaban nuevas subsecuencias sin fragmentos biológicos no deseados por el usuario, es decir, secuencias necesarias para las técnicas. A partir del uso de dicha herramienta se detectaron ciertos errores y mejoras a ser incluidos en una nueva versión del paquete. Se planteó como objetivo de trabajo mejorar y actualizar el paquete CleanBSequences en CRAN, con lo cual, los errores fueron subsanados y las mejoras incluidas en una nueva versión del paquete, para ello se trabajó sobre las funciones preexistentes (OnePrimerRemove y TwoPrimersRemove) y se generaron nuevas funciones (DNAStringSetOPR y DNAStringSetTPR). La nueva versión del paquete pasó por todos los chequeos correspondientes y fue publicada en CRAN como CleanBSequences 1.4.0. En conclusión, se logró mejorar y actualizar el paquete CleanBSequences en CRAN.

**PALABRAS CLAVE:** CRAN, Marcadores Moleculares, Secuencias Biológicas.

## ABSTRACT

This work presents a new method and an improved and updated tool to solve a common problem of molecular biologists and geneticists who use molecular markers in their scientific research and development: sequence curation. Omic studies conducted by molecular biologists and geneticists usually involve the use of molecular markers. AFLP, cDNA-AFLP, and MSAP are examples of markers that render information at the genomics, transcriptomics, and epigenomics levels, respectively. These three types of molecular markers use adaptors that are the template for PCR amplification. The sequences of the adaptors must be eliminated for the analysis of the results. Since a large number of sequences are usually obtained in these studies, data cleaning is a time-consuming and labor-intensive step. To automate this work, an R package was previously created, called CleanBSequences whose initial version was 0.4.0, which allowed curating sequences massively, quickly, without errors and that can be used offline. Curing was performed by aligning the forward and/or reverse primers or ends of cloning vectors with the sequences to be removed. After the alignment, new subsequences were generated without biological fragments unwanted by the user, that is, sequences necessary for the techniques. By using this tool errors and improvement opportunities were detected. The objective of the work was to improve and update the CleanBSequences package in CRAN, Errors were fixed and the improvements were included in a new version of the package. The pre-existing functions (OnePrimerRemove and TwoPrimersRemove) were modified and new functions (DNAStringSetOPR and DNAStringSetTPR) were developed. The new version of the package passed all the corresponding checks and was published on CRAN as CleanBSequences 1.4.0. In conclusion, it was possible to improve and update the CleanBSequences package in CRAN.

**KEYWORD:** CRAN, Molecular Markers, Biological Sequences.

# INTRODUCCIÓN

## Marcadores moleculares utilizados para el estudio de diferentes niveles ómicos

Los estudios ómicos realizados por biólogos moleculares y genetistas suelen implicar el uso de diversos tipos de marcadores moleculares. Son ejemplo de marcadores moleculares que brindan información a nivel de genoma, transcriptoma y epigenoma: 1- los *Amplified fragment length polymorphisms* (AFLP), los cuales son utilizados en estudios de diversidad genética dentro y entre poblaciones (genómica) (Montaño-Pérez et al., 2006; Mecchia et al., 2007), 2- los AFLP-*based transcript profiling* (cDNA-AFLP) empleados para comparar la expresión génica en diferentes tejidos y condiciones experimentales (transcriptómica) (Vuylsteke et al., 2007; Xiao et al., 2009; Hsu et al., 2008; Pereira da Costa et al., 2018; Felitti et al., 2015; Depetris et al., 2018; Pozzi et al., 2019) y 3- los *Methylation-sensitive amplified polymorphisms* (MSAP), marcadores moleculares que informan sobre las modificaciones epigenéticas (epigenómica) (Albertini y Marconi 2014; Yaish et al., 2014; Gimenez et al., 2016). Todas estas metodologías no necesitan conocimiento previo del genoma de la especie de estudio y pueden obtenerse a partir de su utilización más de 4000 fragmentos en el mismo experimento/momento, los cuales pueden ser secuenciados (Amini et al., 2016; Ke et al., 2017). Los protocolos para los tres tipos de marcadores anteriormente nombrados (AFLP, cDNA-AFLP y MSAP), requieren de la utilización de adaptadores a los fragmentos que fueron previamente digeridos con enzimas de restricción. Además, tienen en común la amplificación de ADN utilizando cebadores complementarios a las secuencias de los adaptadores y que cada fragmento que resulte de interés debe ser re-amplificado (Felitti et al., 2015; Depetris et al., 2018) o clonado para su posterior secuenciación (Mecchia et al., 2007; Ochogavía et al., 2011; Amini et al., 2016; Gimenez et al., 2016; Hiki et al., 2017; Ke et al., 2017). Por último, para poder realizar el análisis de las secuencias, las mismas deben ser curadas eliminando el segmento que corresponde a los adaptadores (se obtienen secuencias curadas). El procesamiento de las secuencias re-amplificadas es manual o, en general, usando *softwares* de licencia paga (Depetris et al., 2018) y para las secuencias clonadas suelen utilizarse *softwares* como VecScreen de NCBI (https://www.ncbi.nlm.nih.gov/tools/vecscreen/), el cual requiere conexión a internet además de que las secuencias deben ser curadas individualmente (Ochogavía et al., 2011; Soresi et al., 2015; Gimenez et al., 2016). En cualquiera de los dos casos, el curado de secuencias se realiza individualmente para cada secuencia (una secuencia a la vez). Si se considera que miles de

secuencias son generadas por una amplia comunidad de genetistas y biólogos moleculares que utilizan marcadores moleculares y que deben curarse de forma rápida y correcta, es de suma importancia el desarrollo de nuevas herramientas que den respuesta a esta problemática, facilitando así el trabajo de los investigadores. También es fundamental que el desarrollo de estas nuevas herramientas se lleve a cabo con lenguajes de programación ampliamente distribuidos, que no requieran conexión a internet, que sean gratuitos y de código abierto. Esto último permite a todos los miembros de la comunidad científica tener acceso a los nuevos desarrollos y adaptarlos a su conveniencia (Pozzi et al., 2020).

**Software R**

R es un *software* de código abierto y gratuito que cuenta con el apoyo de R *Foundation for Statistical Computing* (https://www.r-project.org/foundation/). Permite el desarrollo de algoritmos que favorecen el análisis rápido y preciso de los datos obtenidos por secuenciación u otro método molecular de forma automatizada (Sihaloho, 2015). La elaboración de estos algoritmos requiere de una exhaustiva búsqueda bibliográfica y conocimiento de las funciones, paquetes y dependencias involucradas (Pozzi et al., 2020). El éxito del proyecto R puede explicarse debido al sistema de paquetes R, definiéndose como paquete: "extensión del sistema base de R con código, datos y documentación en formato estandarizado" (Leisch, 2009). Este sistema de paquetes permite una extensión fácil, transparente y multiplataforma del sistema base de R, además de ser una forma cómoda de mantener las colecciones de funciones y conjuntos de datos de R, permitiendo la distribución de metodología estadística a otros.

A su vez, RStudio es un conjunto integrado de herramientas que fueron diseñadas para ayudar a ser más productivo el lenguaje R. Consta de una consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo (https://www.rstudio.com/).

**Antecedentes Previos**

Previamente se desarrolló y depositó en CRAN un paquete de R llamado CleanBSequences versión 0.4.0 (https://mran.microsoft.com/snapshot/2020-10-09/web/packages/CleanBSequences/index.html) el cual permitía curar secuencias biológicas generadas por marcadores moleculares de distintos niveles ómicos, de forma gratuita, masiva,

sin errores y sin conexión a internet. Dicho paquete constaba de dos funciones: OnePrimerRemove y TwoPrimersRemove, en el primer caso, OnePrimerRemove, los parámetros eran la secuencia a curar y el primer reverse (PR), todos ellos objetos del tipo DNAString. Dicha función generaba la reversa complementaria para el PR, luego buscaba una expresión regular comparando la secuencia de interés con el PR, marcaba la posición donde comenzaba dicha expresión regular, se pasaba ese valor de posición a un valor numérico y generaba y devolvía una subsecuencia que tenía como comienzo la letra inicial de la secuencia de interés y como finalización la letra del inicio de la expresión regular obtenida entre la secuencia de interés-PR, restándole una unidad. En la función TwoPrimersRemove los parámetros eran la secuencia a curar y los primers forward (PF) y reverse, todos ellos objetos de tipo DNAString. Dicha función generaba la complementaria para el PF y la reversa complementaria para el PR, luego buscaba una expresión regular comparando la secuencia de interés con el PF y con el PR, marcaba las posiciones donde comenzaban dichas expresiones regulares, se pasaban esos valores de posición a valores numéricos y generaba y devolvía una subsecuencia que tenía como comienzo la letra final de la expresión regular obtenida entre secuencia de interés-PF sumándole una unidad y como finalización la letra del inicio de la expresión regular obtenida entre la secuencia de interés-PR restándole una unidad (Figura 1). A partir del desarrollo y utilización de dicho paquete mediante el empleo de secuencias a curar provenientes de la técnica de cDNA-AFLP (Pozzi et al., 2019), se logró detectar ciertas falencias y posibles mejoras las cuales fueron desarrolladas en el presente trabajo a fin de disminuir el esfuerzo y el tiempo de los usuarios del paquete en el curado de secuencias, además de mejorar la estética de los resultados obtenidos.

```
#' @examples
#' SEQ4 = DNAString("AATCG")
#' SEQ5 = DNAString("CG")
#' OnePrimerRemove (SEQ4,SEQ5)
#' @import Biostrings
#' @export

OnePrimerRemove = function (SEQ,PrimerOPR){
  PrimerRRA = reverseComplement(PrimerOPR)
  Pos3 = regexpr(PrimerRRA,SEQ)
  nmatchPos3= as.numeric(Pos3)
  Pos4=nchar(SEQ)
  Pos5=Pos4-(Pos4-1)
  nmatchPos6= as.numeric(Pos5)
  A=nmatchPos6
  B=nmatchPos3-1
  Subseq= subseq(SEQ, start=A, end=B)
  return(Subseq)
}
```

```
#' @examples
#' SEQ1 = DNAString("AATCG")
#' SEQ2 = DNAString("TT")
#' SEQ3 = DNAString("CG")
#' TwoPrimerRemove (SEQ1,SEQ2,SEQ3)
#' @import Biostrings
#' @export

TwoPrimerRemove = function (SEQs,PrimerF,PrimerR){
  PrimerFC = complement(PrimerF)
  PrimerRRC = reverseComplement(PrimerR)
  Pos = regexpr(PrimerFC,SEQs)
  Pos1 = regexpr(PrimerRRC,SEQs)
  length= attr(Pos,"match.length")
  nlength=as.numeric(length)
  nmatchPos=as.numeric(Pos)
  nmatchPos1= as.numeric(Pos1)
  A=nmatchPos+nlength
  B=nmatchPos1-1
  Subseq= subseq(SEQs, start=A, end=B)
  return(Subseq)
}
```

**Figura 1.** Funciones OnePrimerRemove y TwoPrimersRemove del paquete CleanBSequences 0.4.0.

## OBJETIVO GENERAL

Mejorar y actualizar el paquete CleanBSequences en CRAN.

## OBJETIVOS ESPECÍFICOS

1. Incorporar nuevas funciones y realizar mejoras en funciones ya existentes del paquete CleanBSequences.

2. Construir una nueva versión del paquete CleanBSequences en RStudio (*Build*).

3. Depositar la nueva versión del paquete CleanBSequences en CRAN.

# MATERIALES Y MÉTODOS

## Identificación y detalle de las falencias y mejoras de la nueva versión del paquete

A partir del uso y prueba del paquete CleanBSequneces versión 0.4.0 (Pozzi et al., 2020) se detectaron diferentes falencias y posibles mejoras que debieron ser realizadas previo a la actualización y depósito del paquete en CRAN. Como falencias se puede nombrar la reversa del PF, ya que en las secuencias obtenidas dicho primer se encuentra "en directo", no requiriendo dicha transformación, por lo cual debió subsanarse el error, a fin de evitar problemas a nivel de usuario. Por otro lado, se identificaron como mejoras que fueron incorporadas en la nueva versión del paquete: 1- La lectura de un archivo multifasta, que contiene todas la secuencias a curar 2- La generación de *max mismatch* que permite obtener alineamientos positivos entre un 80% y 100% de homología entre secuencias: de esta manera se generan alineamientos teniendo en cuenta distintos porcentajes de match mínimos entre los *primers* y las secuencias a curar (la versión 0.4.0 del paquete requería de un porcentaje de match del 100% para un alineamiento positivo). 3- función que muestre el alineamiento. Es decir, que muestre la posición de los *primers* en la secuencia a curar, permitiendo así una mejora estética del paquete.

## Protocolo de actualización del paquete CleanBSequences y su depósito en CRAN

La actualización del paquete y su depósito en CRAN se llevó a cabo según lo descripto por Wickham (2020).

*1- Preparación de la PC*

Para poder comenzar con la actualización del paquete se requirió tener previamente preparada la PC con la descarga de softwares y paquetes de R que permitan el desarrollo de paquetes y su documentación. Por lo cual, primero se descargó R (versión 4.2.0- para Windows) desde la página https://cran.r-project.org/bin/windows/base/, luego se descargó RStudio (versión 2022.02.2+485- para Windows), desde la página https://www.rstudio.com/products/rstudio/download/. Una vez instalados los softwares anteriores, se descargó Rtools 4.2 (https://cran.r-project.org/bin/windows/Rtools/) que son un conjunto de herramientas para el desarrollo de paquetes de R con Windows. Seguido se descargaron los paquetes de R:

1- devtools 2.4.3 (https://cran.r-project.org/web/packages/devtools/index.html): Herramientas que facilitan el desarrollo de paquetes de R (Wickham et al., 2021).

2- roxygen2 7.1.2 (https://cran.r-project.org/web/packages/roxygen2/index.html): Genera la documentación del paquete R como la documentación Rd y el archivo NAMESPACE (Wickham et al., 2021).

3- knitr 1.39 (https://cran.r-project.org/web/packages/knitr/index.html): Herramienta de uso general para la generación de informes en R (Xie et al., 2022).

4- Biostring 2.62.0 (https://www.bioconductor.org/packages/release/bioc/html/Biostrings.html): Un eficiente manipulador de secuencias biológicas. Permite contener secuencias en memoria, contiene algoritmos de alineamiento de secuencias y otras utilidades, para una manipulación rápida de grandes secuencias biológicas o conjuntos de secuencias biológicas (Pages et al., 2022). Se descargó debido a que es el paquete del cual depende CleanBSequences.

*2- Construcción de una nueva versión de CleanBSequences en RStudio*

Para la generación de una nueva versión del paquete CleanBSequences en RStudio se creó un nuevo proyecto siguiendo los siguientes pasos:

File→ New Project→ New Directory→R Package (Figura 2).

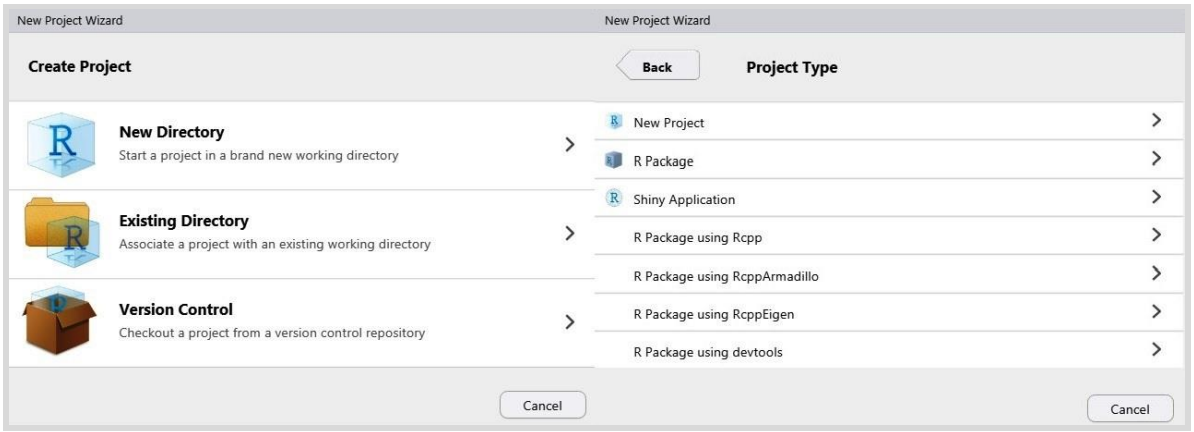



**Figura 2.** Pasos a seguir para la generación de la nueva versión del paquete CleanBSequences.

Seguido a lo anterior se completaron los casilleros para el nombre del paquete y se adicionaron los archivos .R del paquete en su versión anterior (0.4.0): OnePrimerRemove.R y TwoPrimersRemove.R → Create Project (Figura 3).

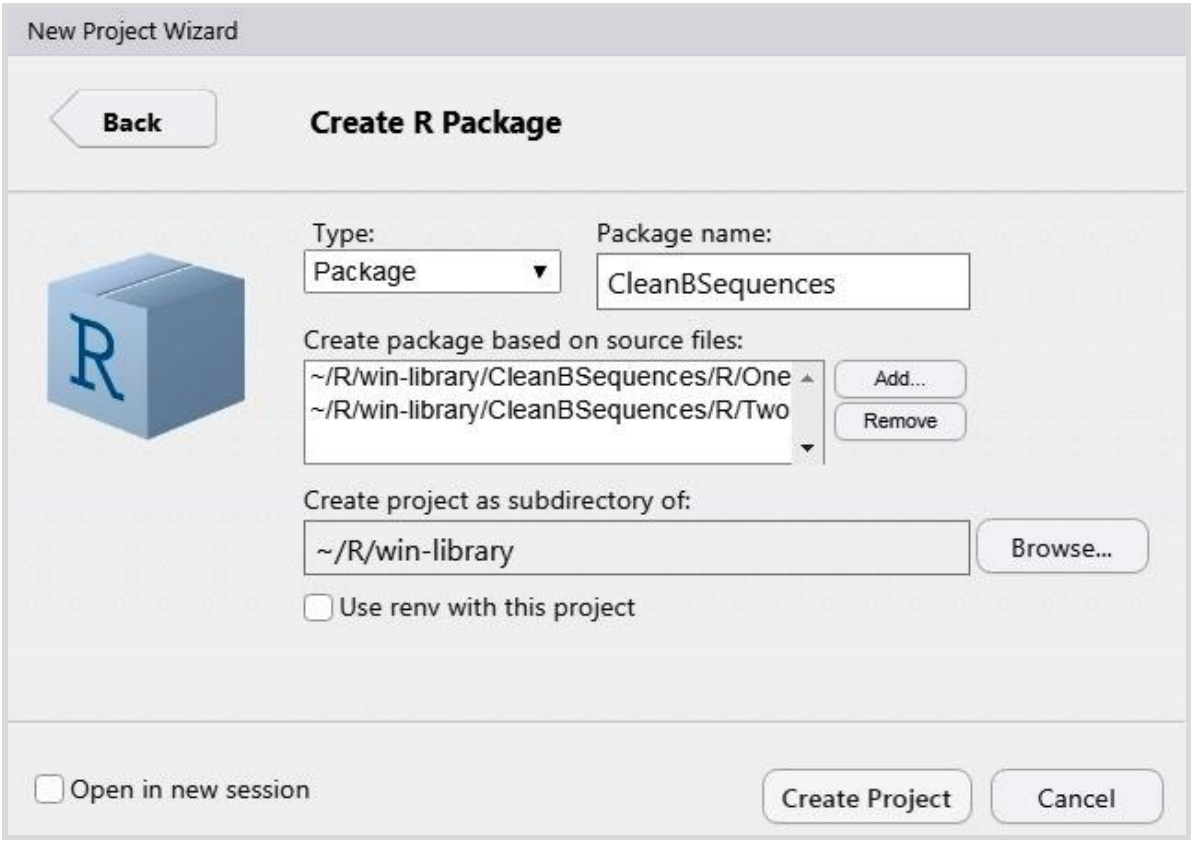



**Figura 3.** Creación del paquete de R CleanBSequences en RStudio.

Una vez creado el proyecto se debió completar el archivo DESCRIPTION (Figura 4) y se comenzó con la modificación de los archivos .R del nuevo proyecto (Figura 1). Además de corregir errores y actualizar las funciones preexistentes en el paquete, se generaron nuevas funciones, distintas a las anteriores, las cuales fueron generadas siguiendo los pasos descriptos a continuación: File→New File→R Script. Se les asignó como nombre: DNAStringSetOPR y DNAStringSetTPR y se guardaron dentro de la carpeta R del proyecto creado como archivos .R.

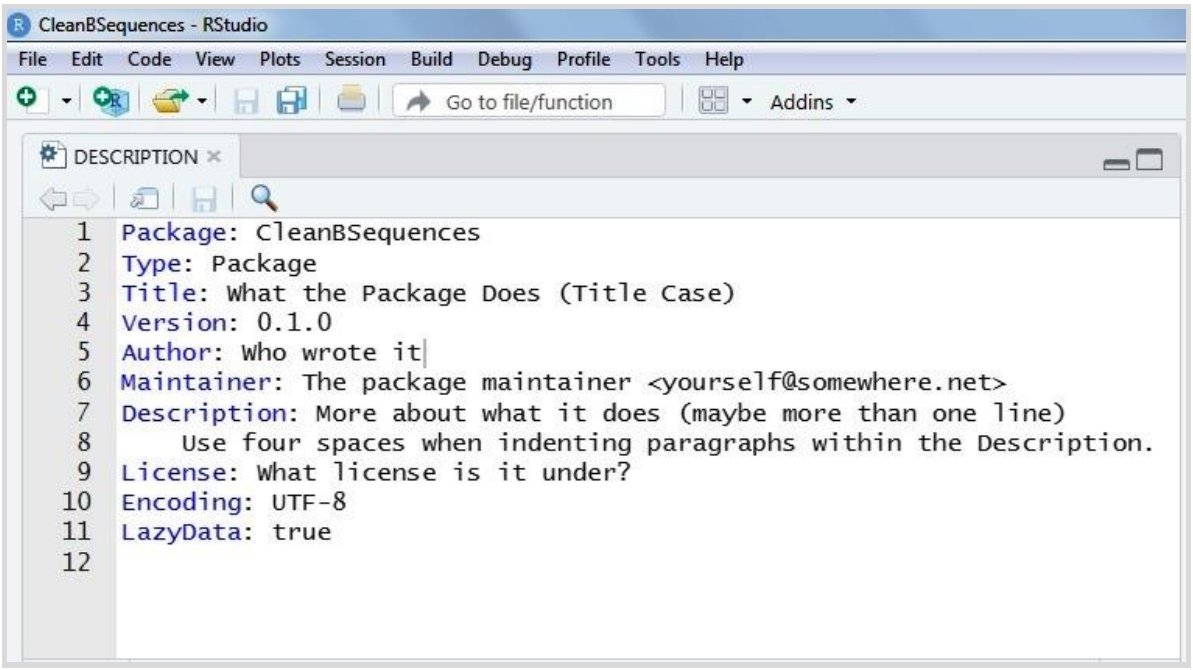



**Figura 4.** Archivo DESCRIPTION.

Por último, se creó la carpeta "inst" dentro de la carpeta que contiene todos los archivos del paquete. Esta carpeta "inst", permite guardar archivos que serán utilizados como datos de pruebas (datos externos) cuando se ejecutan los ejemplos del paquete, así los datos se cargan y pueden ser utilizados como parte de la prueba. Dentro de la carpeta "inst" se generó la carpeta "sequences" y dentro de ella se crearon dos archivos fasta para ser utilizados como ejemplos durante la ejecución de las pruebas: SeqInputOPR.fasta (para ser utilizado como entrada por la función DNAStringSetOPR) y SeqInputTPR.fasta (para ser utilizado como entrada por la función DNAStringSetTPR) (Figura 5).

Las secuencias utilizadas para los ejemplos de las funciones fueron generadas a partir del análisis de la expresión diferencial de genes de endospermo en desarrollo de *Paspalum notatum* (*P. notatum*), mediante la técnica cDNA-AFLP. En la Tabla 1 puede observarse las secuencias a curar utilizadas en cada una de las funciones generadas, además de las secuencias de los primers forward y reverse complementarios a los adaptadores de las enzimas *Taq*I y *Cvi*AII, respectivamente (Pozzi et al., 2019).

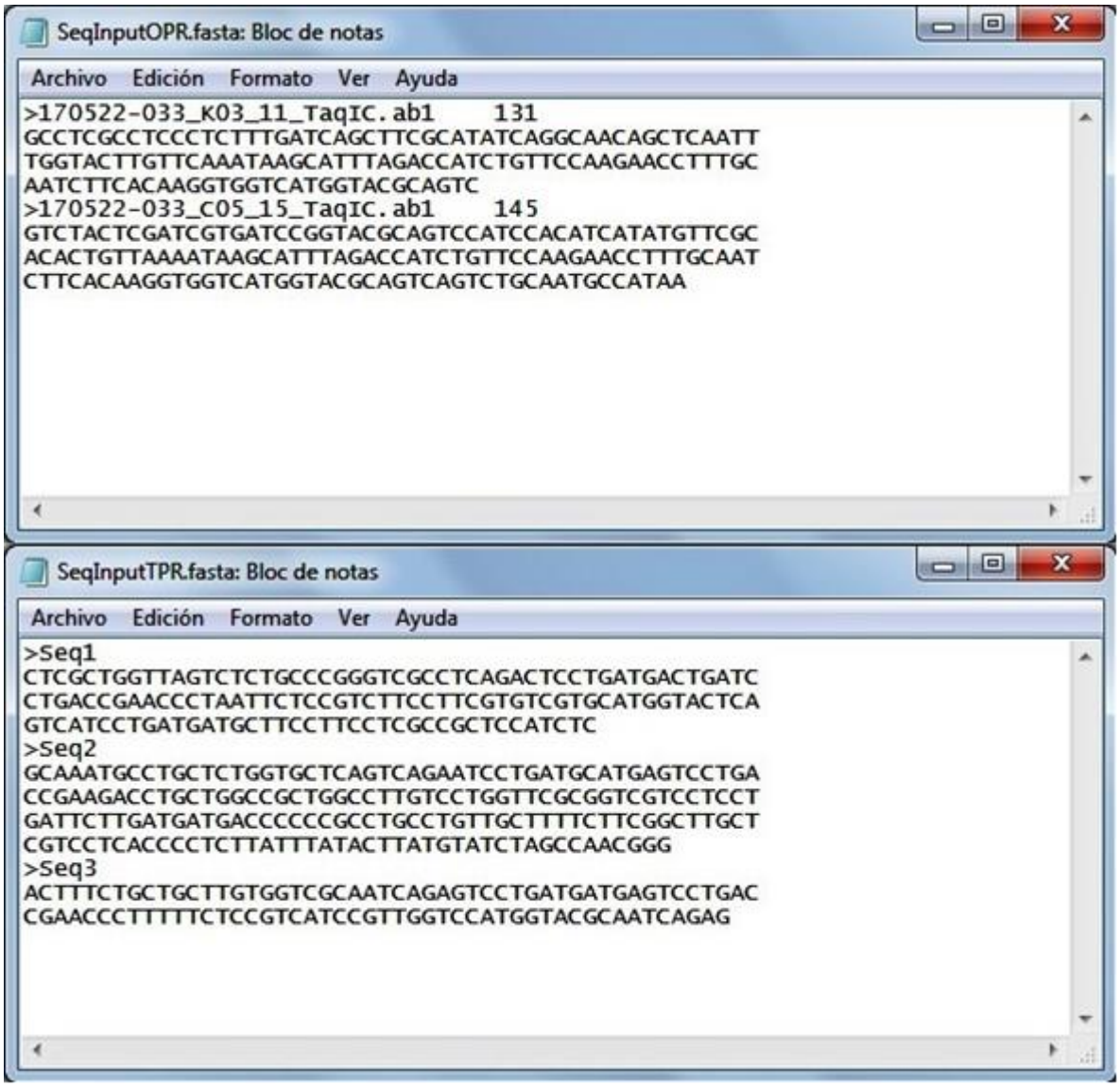



**Figura 5.** Archivos .fasta generados e incorporados en la carpeta "sequences", dentro de la carpeta "inst".

**Tabla 1.** Secuencias utilizadas como entrada en las funciones de CleanBSequences.

| **Secuencias a ser curadas** | |
|---|---|
| **DNAStringSet**[1] | |
| **DNAStringSetOPR**[2] | |
| *170522-033_K03_11_TaqIC.ab1 131* | GCCTCGCCTCCCTCTTTGATCAGCTTCGCATATCAGGCAACAGCTCAATTTGGTACTTGTTCAAATAAGCATTTAGACCATCTGTTCCAAGAACCTTTGCAATCTTCACAAGGTGGTCATGGTACGCAGTC |
| *170522-033_C05_15_TaqIC.ab1 145* | GTCTACTCGATCGTGATCCGGTACGCAGTCCATCCACATCATATGTTCGCACACTGTTAAAATAAGCATTTAGACCATCTGTTCCAAGAACCTTTGCAATCTTCACAAGGTGGTCATGGTACGCAGTCAGTCTGCAATGCCATAA |
| **DNAStringSetTPR**[2] | |
| *Seq1* | CTCGCTGGTTAGTCTCTGCCCGGGTCGCCTCAGACTCCTGATGACTGATCCTGACCCGAACCCTAATTCTCCGTCTTCCTTCGTGTCGTGCATGGTACTCAGTCATCCTGATGATGCTTCCTTCCTCGCCGCTCCATCTC |
| Seq2 | GCAAATGCCTGCTCTGGTGCTCAGTCAGAATCCTGATGCATGAGTCCTGACCGAAGACCTGCTGGCCGCTGGCCTTGTCCTGGTTCGCGGTCGTCCTCCTGATTCTTGATGATGACCCCCCGCCTGCCTGTTGCTTTTCTTCGGCTTGCTCGTCCTCACCCCTCTTATTTATACTTATGTATCTAGCCAACGGG |
| *Seq3* | ACTTTCTGCTGCTTGTGGTCGCAATCAGAGTCCTGATGATGAGTCCTGACCGAACCCTTTTTCTCCGTCATCCGTTGGTCCATGGTACGCAATCAGAG |
| **DNAString**[1] | |
| **OnePrimerRemove**[2] | |
| *170522-033_K03_11_TaqIC.ab1 131* | GCCTCGCCTCCCTCTTTGATCAGCTTCGCATATCAGGCAACAGCTCAATTTGGTACTTGTTCAAATAAGCATTTAGACCATCTGTTCCAAGAACCTTTGCAATCTTCACAAGGTGGTCATGGTACGCAGTC |
| **TwoPrimersRemove**[2] | |
| *Seq3* | ACTTTCTGCTGCTTGTGGTCGCAATCAGAGTCCTGATGATGAGTCCTGACCGAACCCTTTTTCTCCGTCATCCGTTGGTCCATGGTACGCAATCAGAG |
| **Secuencia de Primer** | |
| *PF* | GATGAGTCCTGACCGAA |
| *PR* | GACTGCGTACCATGC |

[1]Tipo de objeto. [2] Nombre de la función del paquete.

*3- <u>Construcción y verificación del paquete</u>*

Una vez desarrollado todo lo necesario para construir el paquete, se procedió a configurar las herramientas de construcción del mismo. Para ello se siguieron los siguientes pasos:

Build→ Configure Build Tools→ Configure→ Para el caso del uso de roxygen2 para generar archivos, se seleccionaron los casilleros Rd file (Contiene la ayuda de las funciones contenidas en el paquete), Collated Field, NAMESPACE file (contiene las importaciones, es decir el nombre del paquete del cual CleanBSequences utiliza funciones; y las exportaciones que son las funciones que están disponibles para ser utilizadas por otros paquetes) y Vignettes (guía del paquete, describe el problema que resuelve el paquete y le muestra al lector como resolverlo). Sumado a lo anterior, se indicó que automáticamente cuando corra la construcción del paquete, se ejecute R CMD Check (verifica que un paquete de R funcione correctamente), se construya el paquete fuente (directorio de archivos con una estructura específica y con extensión .tar.gz) (es el que se sube a CRAN) y el paquete binario (es específico de la plataforma, .zip en Windows y son construidos y desarrollados por CRAN) y por último que se reconstruya el paquete, incluyendo la actualización de toda la documentación, se instale en la librería, para luego reiniciar el paquete construido/actualizado. Además, se seleccionó como opciones adicionales de los testeos que evalúe el paquete como CRAN (--as-cran) siendo esto último una recomendación cada vez que se quiera subir un paquete en dicha plataforma (Figura 6).

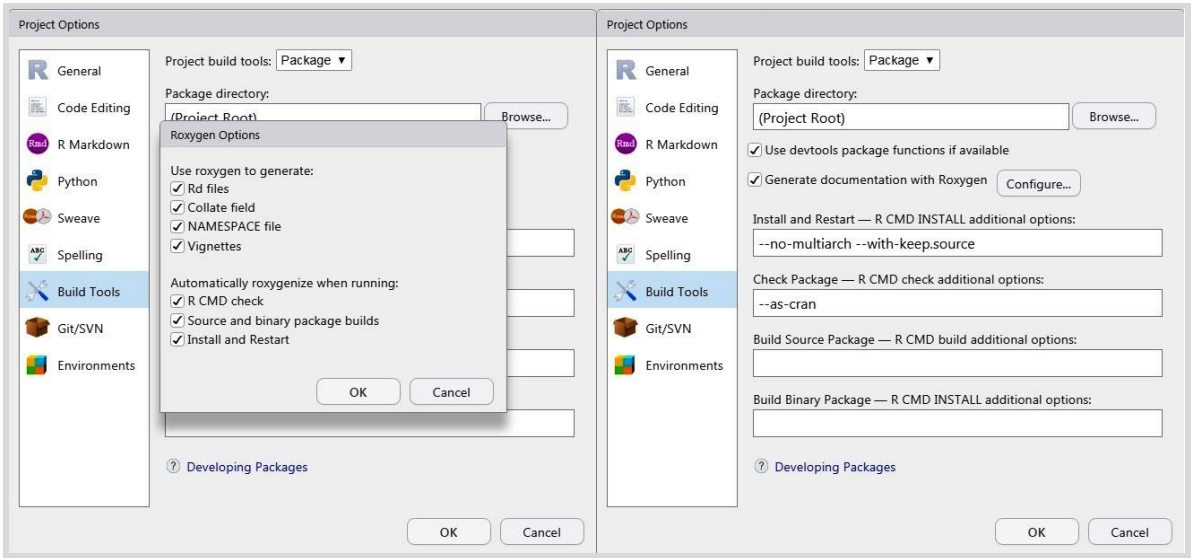



**Figura 6.** Opciones de configuración de construcción del paquete seleccionadas.

El siguiente paso fue construir el paquete, siguiendo la secuencia: Build→ Clean and Rebuild. Y por último chequear que no hubiese problemas con el mismo mediante los pasos: Build→Check Package. Cuando ya no hubo errores, notas ni advertencias se construyó el paquete fuente de la siguiente manera: Build→Build Source Package.

*4- Depósito del paquete en CRAN*

El paquete fuente se subió a CRAN accediendo al sitio web https://cran.r-project.org/submit.html y completando los datos requeridos por el sistema y aceptando las políticas de CRAN (Figura 7).

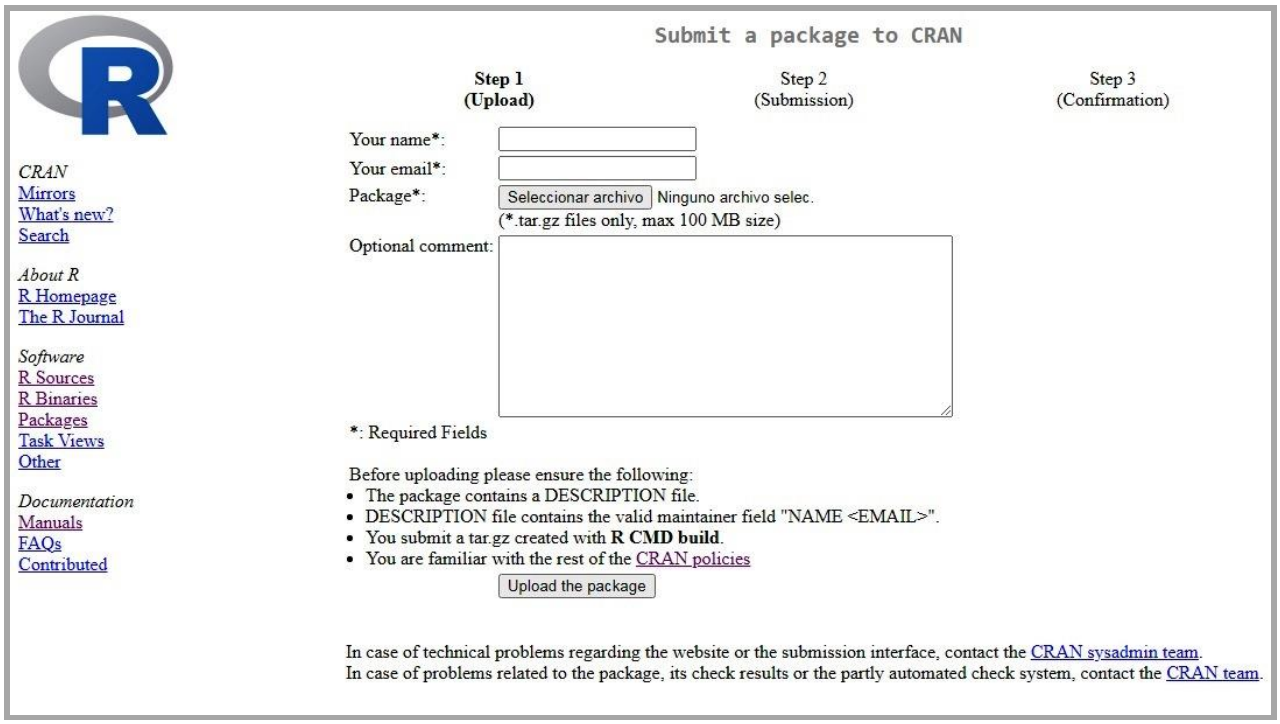



**Figura 7.** Formulario para subir un paquete R en CRAN.

## RESULTADOS Y DISCUSIÓN

A partir de la detección de errores a nivel de usuario y las posibles mejoras a introducir en la nueva versión del paquete, se modificaron y actualizaron las funciones presentes en la versión anterior (0.4.0): OnePrimerRemove y TwoPrimersRemove, eliminando la complementaria del PF. Además, se generaron 2 nuevas funciones: DNAStringSetOPR y DNAStringSetTPR. A continuación, se explican sus nuevas funcionalidades:

OnePrimerRemove: Los parámetros son SEQs y PrimerR, ambos objetos de tipo DNAString. La función primero genera la reversa complementaria del PR, ya que con dicha transformación podemos encontrar el primer en la secuencia a curar. Como el criterio de la función es poder obtener un porcentaje de alineamiento entre el primer y la secuencia a curar mínimo del 80%: Primero obtiene el largo en caracteres de PR, para luego determinar el 80% del total de caracteres de dicho primer. Seguido a lo anterior, obtiene el número de letras que corresponderían al 20% restante, lo pasa a un valor numérico y ese valor es el que utiliza en el parámetro max.mismatch al momento de llevar a cabo el alineamiento bajo la función matchPattern del paquete Biostring. Dado un alineamiento positivo, realiza la subsecuencia a fin de poder extraer la secuencia curada. Para ello, especifica como comienzo de la subsecuencia la letra inicial de la secuencia a curar y como finalización, el comienzo del alineamiento entre PR-Secuencia a curar y le resta una unidad (-1). Muestra la subsecuencia y la escribe en un archivo temporario tipo fasta, el cual estará disponible en caso de que el usuario así lo requiera. Por último, mediante la función *pairwiseAlignment* del paquete Biostring, imprime en la consola el alineamiento entre el PR y la secuencia a curar.

TwoPrimersRemove: Los parámetros son SEQs, PrimerF y PrimerR, todos objetos de tipo DNAString. Como el criterio de la función es obtener un porcentaje de alineamiento mínimo del 80%: primero determina el largo en caracteres de PF, luego obtiene el 80% del total de caracteres de dicho primer. Seguido a lo anterior, calcula el número de letras que corresponderían al 20% restante, lo pasa a un valor numérico y ese valor es el que utiliza en el parámetro max.mismatch (en “m1a”) al momento de llevar a cabo el alineamiento bajo la función matchPattern del paquete Biostring. Seguido a lo anterior, genera la reversa complementaria del PR. Obtiene el largo en caracteres de PR, para luego determinar el 80% del total de caracteres de dicho primer. Calcula el número de letras que corresponderían al 20% restante, lo pasa a un valor numérico y ese valor es el que utiliza en el parámetro max.mismatch (en “m2a”). Dado un alineamiento positivo, realiza una subsecuencia a fin de

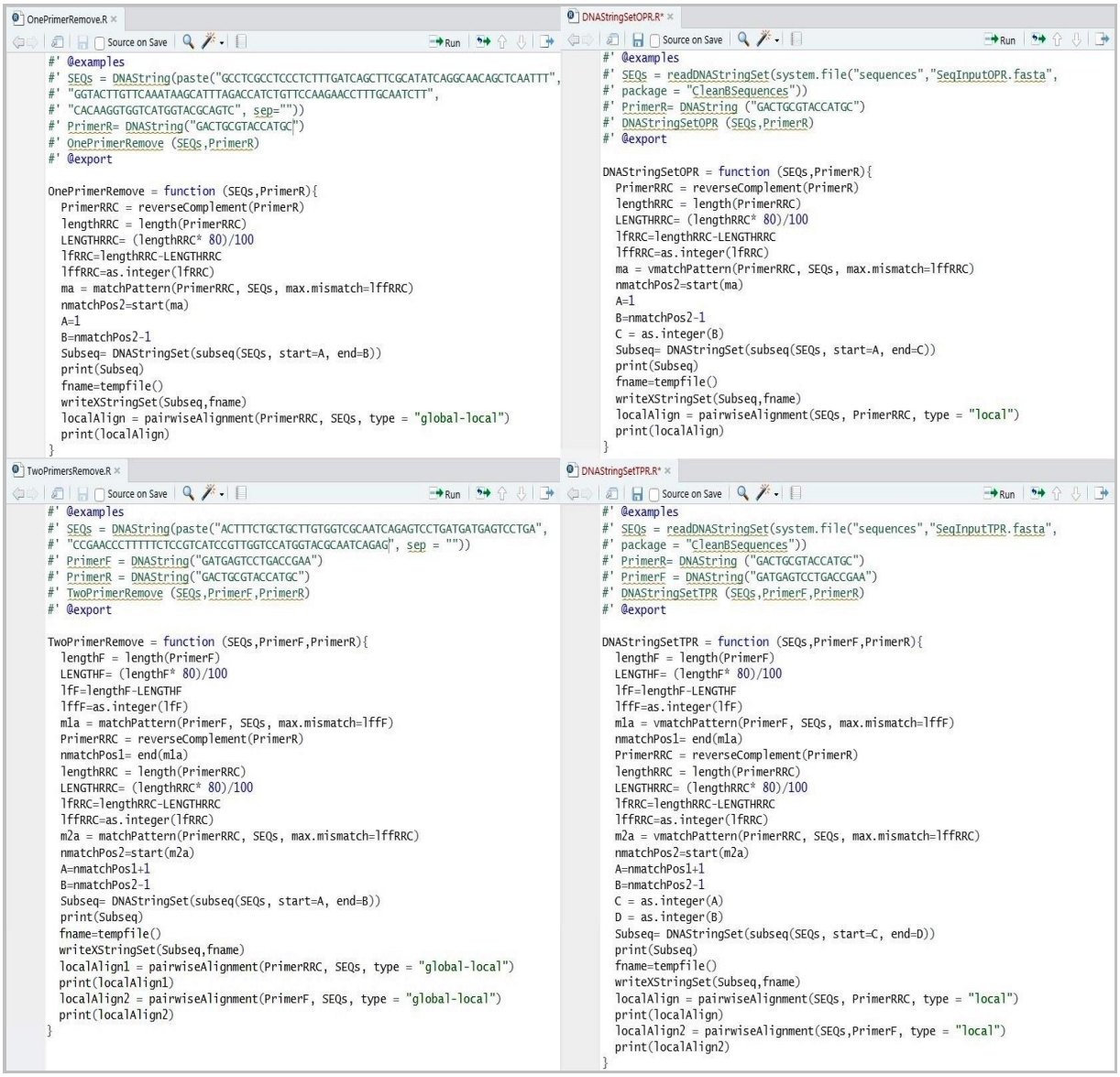
poder extraer la secuencia curada. Para ello, posiciona como comienzo de la subsecuencia la letra final del alineamiento entre PF-Secuencia a curar, y le suma una unidad (+1) y como finalización, el comienzo del alineamiento entre PR-Secuencia a curar y le resta una unidad (-1). Muestra la subsecuencia y la escribe en un archivo temporario tipo fasta, que estará disponible para el usuario en el caso de que lo necesite. Por último, mediante la función pairwiseAlignment del paquete Biostring, imprime en la consola el alineamiento entre: PF-Secuencia a curar y PR-Secuencia a curar.

DNAStringSetOPR: Los parámetros son SEQs y PrimerR, el primero es un objeto de tipo DNAStringSet y el segundo de tipo DNAString. Para este caso se incluyeron en el paquete dos secuencias en un archivo tipo fasta, y se planteó en el ejemplo las líneas de comandos para poder llamarlo a fin de ser utilizado como entrada de la función. La función DNAStringSetOPR primero genera la reversa complementaria del PR, ya que con dicha transformación podemos encontrar el primer en las secuencias a curar. Como el criterio de la función es poder obtener un porcentaje de alineamiento entre el primer y las secuencias a curar mínimo del 80%: Primero obtiene el largo en caracteres de PR, para luego determinar el 80% del total de caracteres de dicho primer. Seguido a lo anterior, obtiene el número de letras que corresponderían al 20% restante, lo pasa a un valor numérico y ese valor es el que utiliza en el parámetro max.mismatch al momento de llevar a cabo el alineamiento bajo la función vmatchPattern del paquete Biostring. Dado alineamientos positivos, realiza las subsecuencias a fin de poder extraer las secuencias curadas. Para ello, especifica, para cada una de las secuencias utilizadas, como comienzo de la subsecuencia la letra inicial de la secuencia a curar y como finalización, el comienzo del alineamiento entre PR-Secuencia a curar y le resta una unidad (-1). Muestra las subsecuencias y las escribe en un archivo temporario tipo fasta, disponible para el usuario. Por último, mediante la función pairwiseAlignment del paquete Biostring, imprime en la consola el alineamiento entre el PR y las secuencias a curar.

DNAStringSetTPR: Los parámetros son SEQs, PrimerF y PrimerR, el primero es un objeto de tipo DNAStringSet y los últimos dos de tipo DNAString. Para este caso, también se brindó en el paquete secuencias (3) en archivo tipo fasta, y se planteó en el ejemplo las líneas de comandos para que sea utilizado como entrada de la función. La función DNAStringSetTPR primero determina el largo en caracteres de PF, luego obtiene el 80% del total de caracteres de dicho primer. Seguido a lo anterior, calcula el número de letras que corresponderían al 20% restante, lo pasa a un valor numérico y ese valor es el que utiliza en el parámetro max.mismatch (en “m1a”) al momento de llevar a cabo el alineamiento bajo la función

vmatchPattern del paquete Biostring. Seguido a lo anterior, genera la reversa complementaria del PR. Obtiene el largo en caracteres de PR, para luego determinar el 80% del total de caracteres de dicho primer. Calcula el número de letras que corresponderían al 20% restante, lo pasa a un valor numérico y ese valor es el que utiliza en el parámetro max.mismatch (en "m2a"). Dado alineamientos positivos, realiza una subsecuencia, para cada secuencia a curar dentro del archivo, a fin de poder extraer la secuencia curada. Para ello, posiciona como comienzo de la subsecuencia la letra final del alineamiento entre PF-Secuencia a curar, y le suma una unidad (+1) y como finalización, el comienzo del alineamiento entre PR-Secuencia a curar y le resta una unidad (-1). Muestra las subsecuencias y las escribe en un archivo temporario tipo fasta, que estará disponible para el usuario si así lo requiere. Por último, mediante la función pairwiseAlignment del paquete Biostring, imprime en la consola el alineamiento entre: PF-Secuencia a curar y PR-Secuencia a curar.
La versión final de cada una de las funciones anteriormente descriptas se encuentra en la Figura 8. Mientras que en la Figura 9 pueden observarse las salidas de cada una de las funciones desarrolladas para la nueva versión del paquete CleanBSequences.

Los resultados obtenidos, en general, pudieron validarse comparando las salidas con los resultados obtenidos con el programa Sequencher 4.1.4 (Figura 10). Hubo una diferencia de salida entre el programa y el paquete de R, más específicamente en la secuencia Seq1. Esto podría deberse a que en el alineamiento entre la secuencia de interés y el PF hay un GAP (hueco) de 3 bases (GATGAC[GAT]TCCTGACCGAA) y las funciones del paquete que buscan un patrón para alinear, como matchPattern y vmatchPattern, no consideran GAPs. Esta diferencia podría considerarse como una mejora a incorporar en una nueva versión futura del paquete CleanBSequences, aunque vale la pena aclarar que estos errores pueden evidenciarse y salvarse al momento de mirar los alineamientos en la salida de la función del paquete de R cuando se aplica pairwiseAlignment.



OnePrimerRemove.R

```
#' @examples
#' SEQs = DNAString(paste("GCCTCGCCTCCCTCTTTGATCAGCTTCGCATATCAGGCAACAGCTCAATTT",
#' "GGTACTTGTTCAAATAAGCATTTAGACCATCTGTTCCAAGAACCTTTGCAATCTT",
#' "CACAAGGTGGTCATGGTACGCAGTC", sep=""))
#' PrimerR= DNAString("GACTGCGTACCATGC")
#' OnePrimerRemove (SEQs,PrimerR)
#' @export

OnePrimerRemove = function (SEQs,PrimerR){
  PrimerRRC = reverseComplement(PrimerR)
  lengthRRC = length(PrimerRRC)
  LENGTHRRC= (lengthRRC* 80)/100
  lfRRC=lengthRRC-LENGTHRRC
  lffRRC=as.integer(lfRRC)
  ma = matchPattern(PrimerRRC, SEQs, max.mismatch=lffRRC)
  nmatchPos2=start(ma)
  A=1
  B=nmatchPos2-1
  Subseq= DNAStringSet(subseq(SEQs, start=A, end=B))
  print(Subseq)
  fname=tempfile()
  writeXStringSet(Subseq,fname)
  localAlign = pairwiseAlignment(PrimerRRC, SEQs, type = "global-local")
  print(localAlign)
}
```

DNAStringSetOPR.R*

```
#' @examples
#' SEQs = readDNAStringSet(system.file("sequences","SeqInputOPR.fasta",
#' package = "CleanBSequences"))
#' PrimerR= DNAString ("GACTGCGTACCATGC")
#' DNAStringSetOPR (SEQs,PrimerR)
#' @export

DNAStringSetOPR = function (SEQs,PrimerR){
  PrimerRRC = reverseComplement(PrimerR)
  lengthRRC = length(PrimerRRC)
  LENGTHRRC= (lengthRRC* 80)/100
  lfRRC=lengthRRC-LENGTHRRC
  lffRRC=as.integer(lfRRC)
  ma = vmatchPattern(PrimerRRC, SEQs, max.mismatch=lffRRC)
  nmatchPos2=start(ma)
  A=1
  B=nmatchPos2-1
  C = as.integer(B)
  Subseq= DNAStringSet(subseq(SEQs, start=A, end=C))
  print(Subseq)
  fname=tempfile()
  writeXStringSet(Subseq,fname)
  localAlign = pairwiseAlignment(SEQs, PrimerRRC, type = "local")
  print(localAlign)
}
```

TwoPrimersRemove.R

```
#' @examples
#' SEQs = DNAString(paste("ACTTTCTGCTGCTTGTGGTCGCAATCAGAGTCCTGATGATGAGTCCTGA",
#' "CCGAACCCTTTTTCTCCGTCATCCGTTGGTCCATGGTACGCAATCAGAG", sep = ""))
#' PrimerF = DNAString("GATGAGTCCTGACCGAA")
#' PrimerR = DNAString("GACTGCGTACCATGC")
#' TwoPrimerRemove (SEQs,PrimerF,PrimerR)
#' @export

TwoPrimerRemove = function (SEQs,PrimerF,PrimerR){
  lengthF = length(PrimerF)
  LENGTHF= (lengthF* 80)/100
  lfF=lengthF-LENGTHF
  lffF=as.integer(lfF)
  m1a = matchPattern(PrimerF, SEQs, max.mismatch=lffF)
  PrimerRRC = reverseComplement(PrimerR)
  nmatchPos1= end(m1a)
  lengthRRC = length(PrimerRRC)
  LENGTHRRC= (lengthRRC* 80)/100
  lfRRC=lengthRRC-LENGTHRRC
  lffRRC=as.integer(lfRRC)
  m2a = matchPattern(PrimerRRC, SEQs, max.mismatch=lffRRC)
  nmatchPos2=start(m2a)
  A=nmatchPos1+1
  B=nmatchPos2-1
  Subseq= DNAStringSet(subseq(SEQs, start=A, end=B))
  print(Subseq)
  fname=tempfile()
  writeXStringSet(Subseq,fname)
  localAlign1 = pairwiseAlignment(PrimerRRC, SEQs, type = "global-local")
  print(localAlign1)
  localAlign2 = pairwiseAlignment(PrimerF, SEQs, type = "global-local")
  print(localAlign2)
}
```

DNAStringSetTPR.R*

```
#' @examples
#' SEQs = readDNAStringSet(system.file("sequences","SeqInputTPR.fasta",
#' package = "CleanBSequences"))
#' PrimerR= DNAString ("GACTGCGTACCATGC")
#' PrimerF = DNAString("GATGAGTCCTGACCGAA")
#' DNAStringSetTPR (SEQs,PrimerF,PrimerR)
#' @export

DNAStringSetTPR = function (SEQs,PrimerF,PrimerR){
  lengthF = length(PrimerF)
  LENGTHF= (lengthF* 80)/100
  lfF=lengthF-LENGTHF
  lffF=as.integer(lfF)
  m1a = vmatchPattern(PrimerF, SEQs, max.mismatch=lffF)
  nmatchPos1= end(m1a)
  PrimerRRC = reverseComplement(PrimerR)
  lengthRRC = length(PrimerRRC)
  LENGTHRRC= (lengthRRC* 80)/100
  lfRRC=lengthRRC-LENGTHRRC
  lffRRC=as.integer(lfRRC)
  m2a = vmatchPattern(PrimerRRC, SEQs, max.mismatch=lffRRC)
  nmatchPos2=start(m2a)
  A=nmatchPos1+1
  B=nmatchPos2-1
  C = as.integer(A)
  D = as.integer(B)
  Subseq= DNAStringSet(subseq(SEQs, start=C, end=D))
  print(Subseq)
  fname=tempfile()
  writeXStringSet(Subseq,fname)
  localAlign = pairwiseAlignment(SEQs, PrimerRRC, type = "local")
  print(localAlign)
  localAlign2 = pairwiseAlignment(SEQs,PrimerF, type = "local")
  print(localAlign2)
}
```

**Figura 8.** Funciones del paquete CleanBSequences 1.4.0

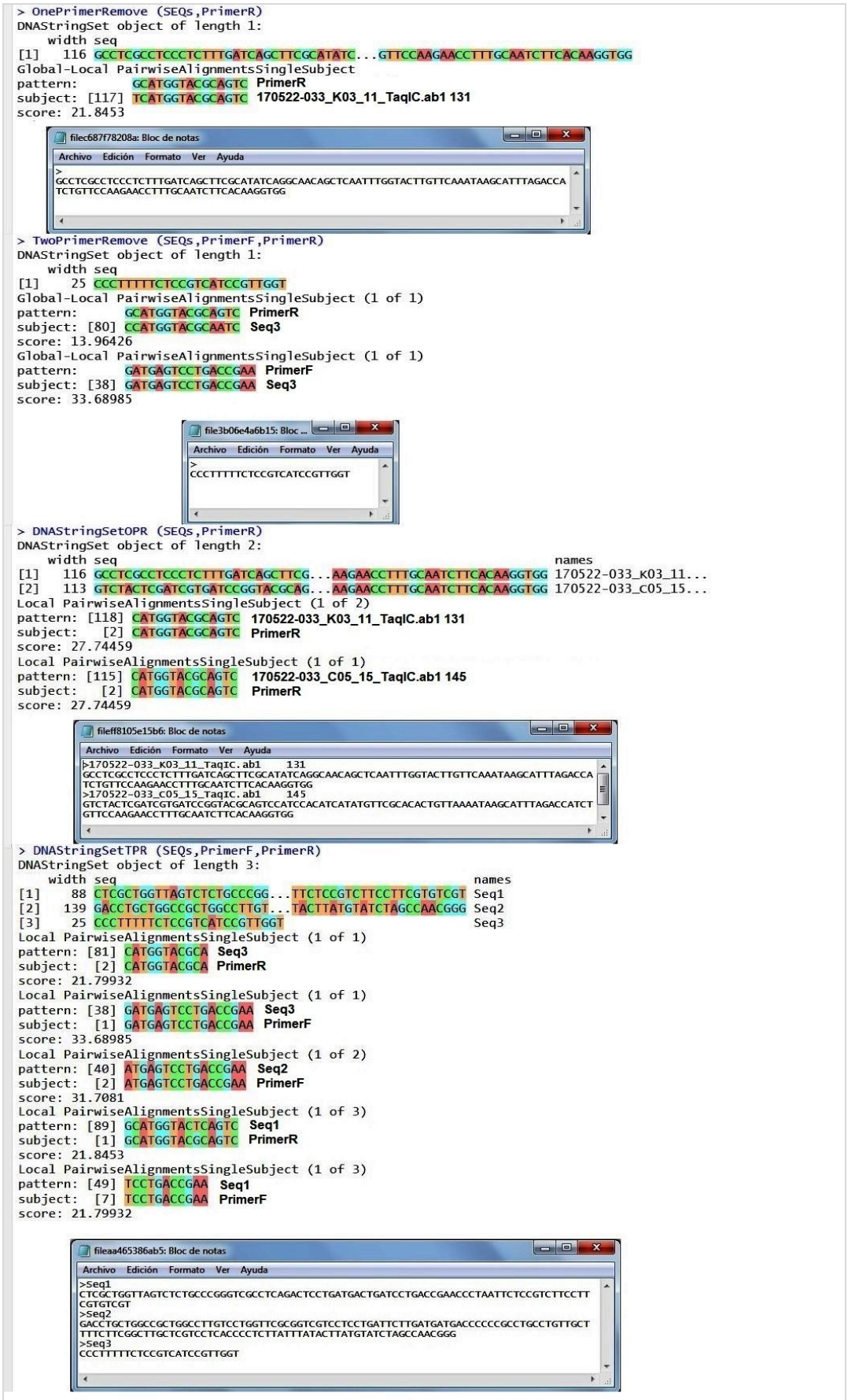



**Figura 9.** Salidas de cada una de las funciones desarrolladas para la nueva versión del paquete.

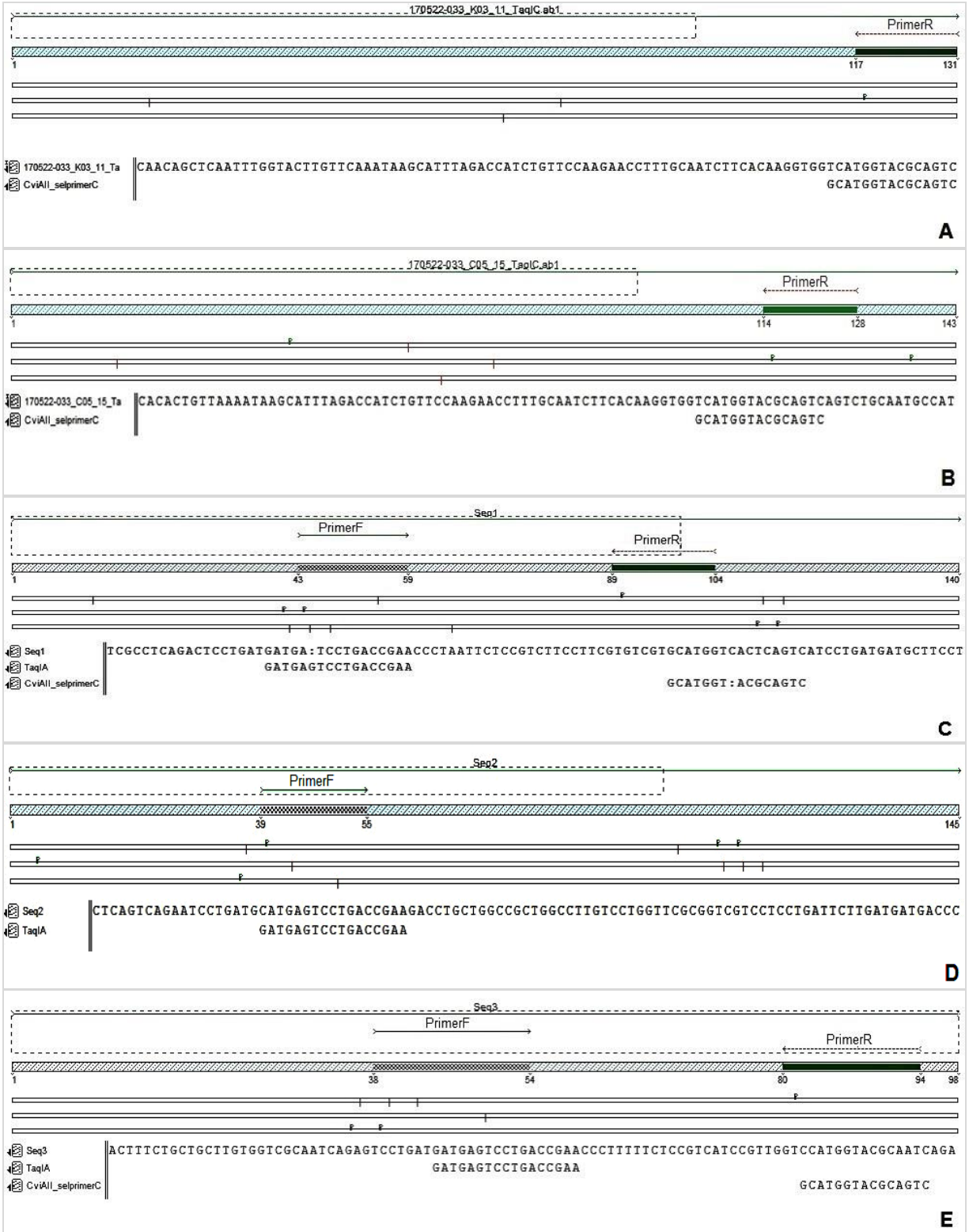



**Figura 10.** Salida del programa Sequencher 4.1.4. Secuencias: A- 170522-033_K03_11_TaqIC.ab1 131 (OnePrimerRemove y DNAStringSetOPR). B-170522-033_C05_15_TaqIC.ab1 145 (DNAStringSetOPR). C- Seq1 (DNAStringSetTPR). D- Seq2 (DNAStringSetTPR). E- Seq3 (TwoPrimersRemove y DNAStringSetTPR).

Los archivos finales DESCRIPTION, NAMESPACE pueden observarse en la Figura 11. Con la función OnePrimerRemove se ejemplifica en la Figura 12 un archivo .Rd generado con su respectiva visualización en la pestaña *Help*. Sumado a lo anterior, la versión final del fichero previo a la construcción del paquete se puede observar en la Figura 13.

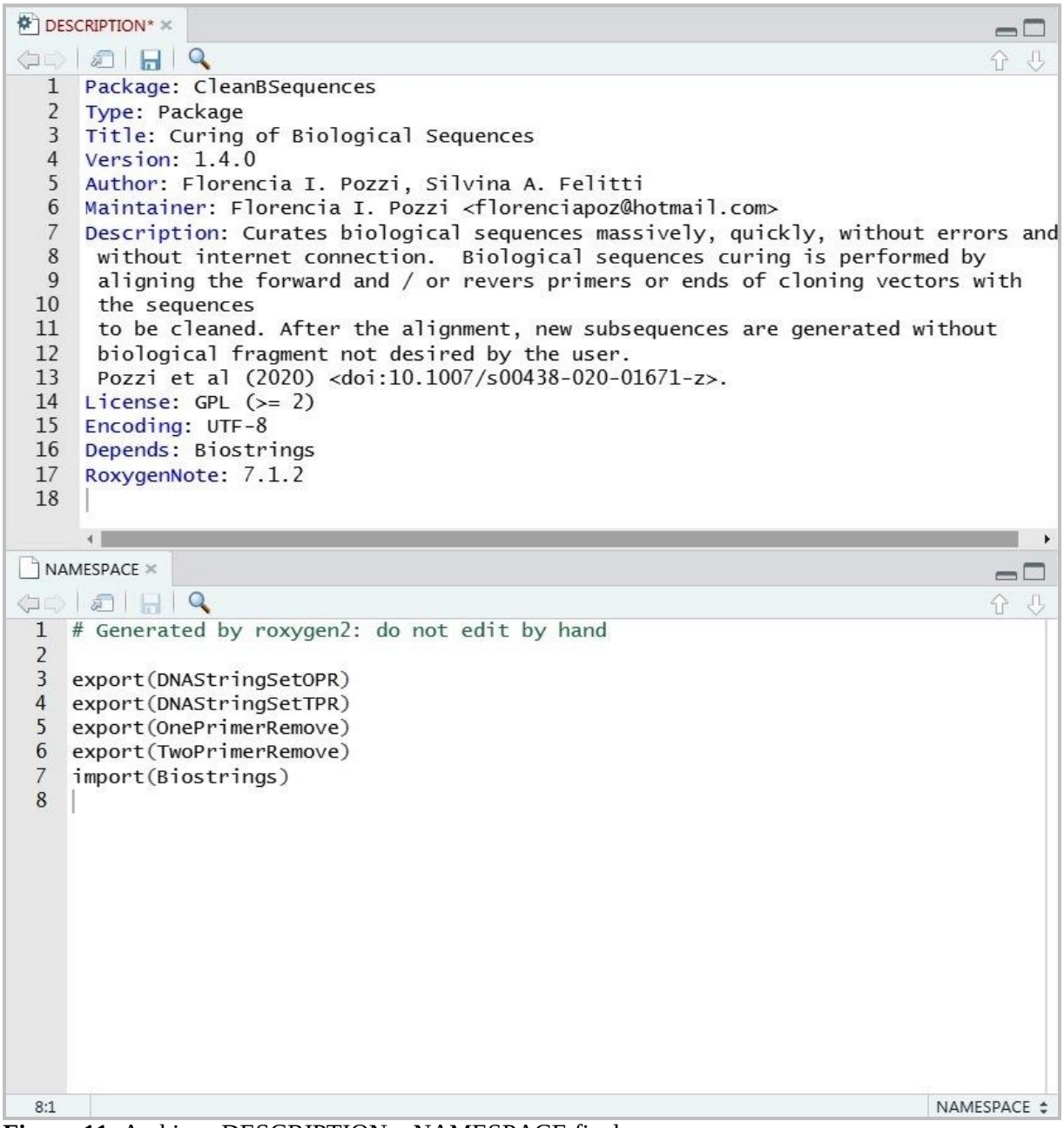



**Figura 11.** Archivos DESCRIPTION y NAMESPACE finales.

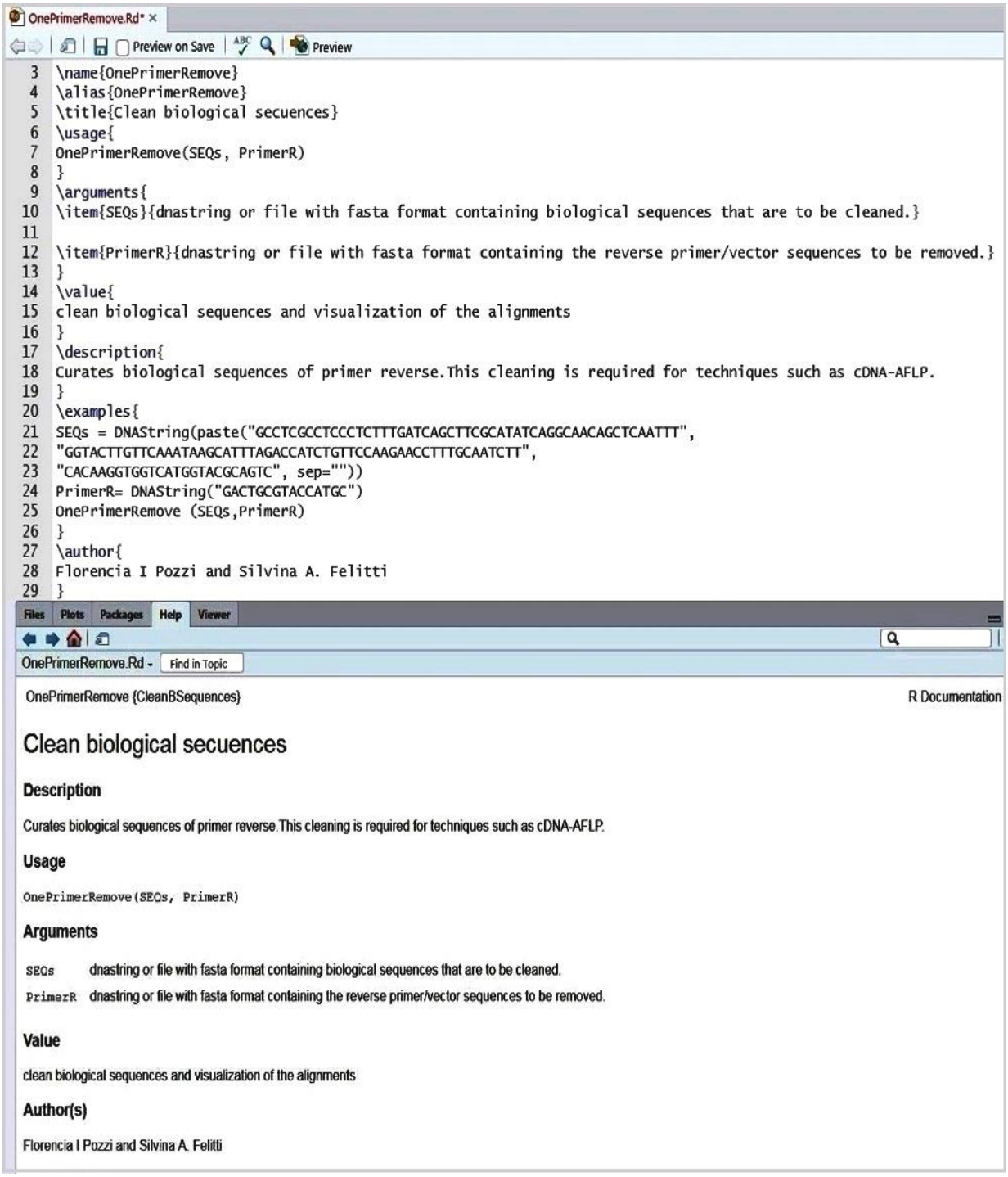



**Figura 12.** Archivo .Rd y su visualización en Help de la función OnePrimerRemove.

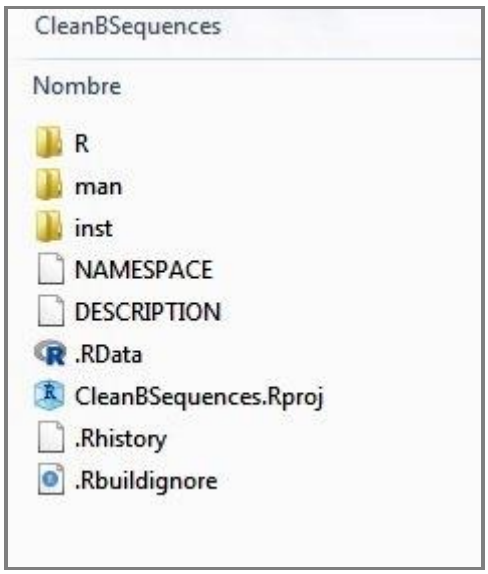



**Figura 13.** Fichero final para construcción del paquete CleanBSequences 1.4.0

Cómo se observa en la Figura 14 al momento de los chequeos del paquete no se obtuvieron errores, notas o advertencias. Durante el proceso de carga y depósito en CRAN, en una primera instancia, el paquete no pasó el chequeo automático. Solucionados los errores que se marcaron en el informe, el paquete pasó la etapa de chequeo automático a la de chequeo manual (realizada por un miembro del equipo CRAN). En esta instancia de evaluación, el paquete no pasó dos rondas consecutivas, debido a un error de la sintaxis específica para el DOI de una publicación informada en la descripción del paquete. Se subsanó el error y se volvió a subir el archivo tar.gz en CRAN. El día 27 de abril de 2022, el equipo CRAN informa su aceptación y publicación del paquete CleanBSequences versión 1.4.0 en CRAN: https://cran.r-project.org/web/packages/CleanBSequences/index.html (ver Anexo).

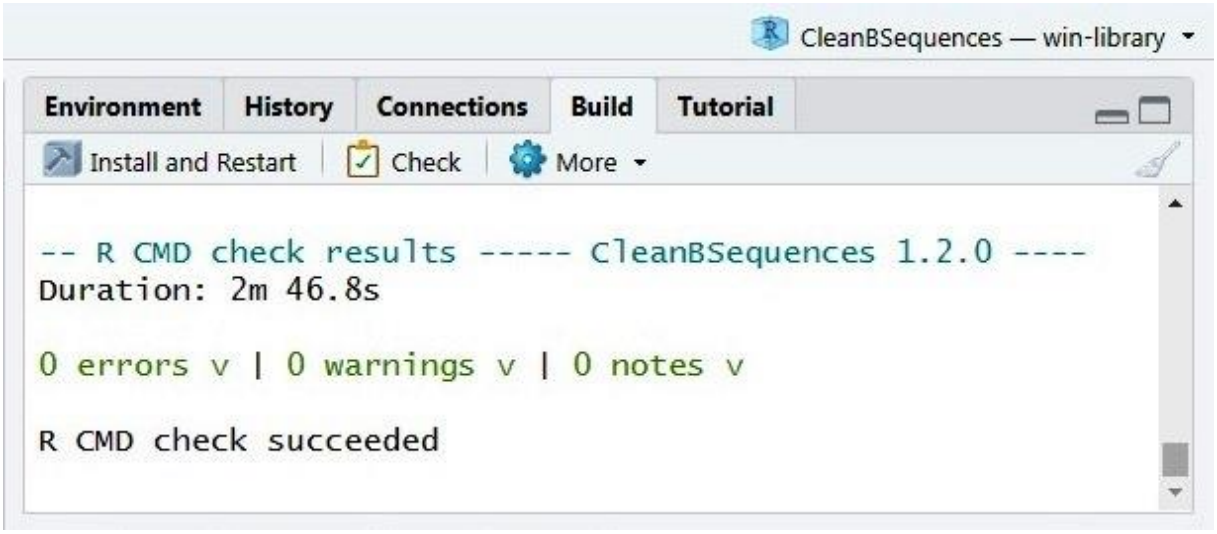



**Figura 14.** Resultado de los chequeos del paquete CleanBSequences 1.4.0 luego de ser construido.

## CONCLUSIONES

En base a los resultados obtenidos y enmarcados por el objetivo general y los objetivos específicos planteados, se concluye que:

Se logró detectar errores y plantear mejoras para una nueva versión del paquete de R previamente publicado como CleanBSequences 0.4.0. A partir de la sistematización de las mejoras a realizar, se trabajó sobre las mismas y se lograron incorporar a las funciones ya existentes en el paquete antes nombrado, además de crearse nuevas funciones. Se logró generar una nueva versión del paquete CleanBSequences siendo la actual 1.4.0. y la misma se subió y depositó en CRAN, pasando por todas las etapas de evaluación exitosamente.

Debido a que la nueva herramienta actualizada no requiere de conexión a internet, sumado a que reduce el tiempo, el esfuerzo y los errores de trabajo se espera que facilite el trabajo de la comunidad científica que requiera trabajar con secuencias biológicas, ya que cuenta con distintas ventajas respecto a los programas disponibles. Sumado a lo anterior, resulta ser una herramienta flexible (gratuita y de código abierto) que tiene el potencial de ser utilizada en futuras mejoras para dar respuesta a nuevos problemas que surjan en dicha comunidad relacionados con el curado de secuencias.

## BIBLIOGRAFÍA

Albertini E and Marconi G (2014) Methylation-sensitive amplified polymorphism (MSAP) marker to investigate drought-stress response in Montepulciano and Sangiovese grape cultivars. Methods Mol Biol 1112: 151-64.

Amini S, Maali-Amiri R, Mohammadi R, Kazemi-Shahandashti SS (2016) cDNA-AFLP analysis of transcripts induced in chickpea plants by TiO2 nanoparticles during cold stress. Plant Physiol Biochem 111: 39-49.

Charif D, Clerc O, Frank C, Lobry JR, Necşulea A, Palmeira L, Penel S, Perrière G (2021) seqinr: Biological Sequences Retrieval and Analysis. R package version 4.2-8.

Depetris MB, Acuña CA, Pozz FI, Quarin CL, Felitti SA (2018) Identification of Genes Related to Endosperm Balance Number Insensitivity in *Paspalum notatum*. Crop Science 58: 813-822.

Felitti SA, Acuña CA, Ortiz JPA, Quarin CL (2015) Transcriptome analysis of seed development in apomictic *Paspalum notatum*. Ann Appl Biol 167: 36-54.

Gimenez MD, Yañez-Santos AM, Paz RC, Quiroga MP, Marfil CF, Conci VC, García-Lampasona SC (2016) Assessment of genetic and epigenetic changes in virus-free garlic (*Allium sativum L.*) plants obtained by meristem culture followed by in vitro propagation. Plant Cell Rep 35: 129–141.

Hiki K, Nakajima F, Tobino T (2017) Application of cDNA-AFLP to biomarker exploration in a non-model species *Grandidierella japonica*. Ecotoxicol Environ Saf 140: 206-213.

Hsu TW, Tsai WC, Wang DP, Lin S, Hsiao YY, Chen WH, Chen HH (2008) Differential gene expression analysis by cDNA-AFLP between flower buds of *Phalaenopsis Hsiang* Fei cv. H. F. and its somaclonal variant. Plant Science 175: 415-422.

Ke, L, Luo H, Zhang L, Zhang M, Yu X, Sun J, Sun Y (2017) Differential transcript profiling alters regulatory gene expression during the development of *Gossypium arboreum*, *G. stocksii* and somatic hybrids. Sci Rep. 3120.

Leisch F (2009) Creating R Packages: A Tutorial. https://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf

Mecchia MA, Ochogavía A, Pablo Selva J, Laspina N, Felitti S, Martelotto LG, Spangenberg G, Echenique V, Pessino SC (2007) Genome polymorphisms and gene differential expression in a 'back-and-forth' ploidy-altered series of weeping lovegrass (*Eragrostis curvula*). Journal of Plant Physiology 164: 1051-1061

Montaño-Pérez K, Villalpando E, Vargas-Albores F (2006) AFLP (Amplified Fragment Length Polymorphism) y su aplicación en acuicultura. Interciencia 31: 563-569.

Ochogavía AC, Seijo JG, González AM, Podio M, Duarte Silveira E, Machado Lacerda AL, Tavares de Campos Carneiro V, Ortiz JP, Pessino SC (2011) Characterization of retrotransposon sequences expressed in inflorescences of apomictic and sexual *Paspalum notatum* plants. Sexual plant reproduction 24: 231-246.

Pages H, Aboyoun P, Gentleman R, DebRoy S (2022) Biostrings: string objects representing biological sequences, and matching algorithms. R package version 2.62.0.

Pereira da Costa JH, Rodríguez GR, Picardi LA, Zorzoli R, Pratta GR (2018) Genome-wide expression analysis at three fruit ripening stages for tomato genotypes differing in fruit shelf life. Scientia horticulturae 229: 125-125.

Pozzi, FI, Green, GY, Barbona IG, Rodríguez GR, Felitti SA (2020) CleanBSequences: an efficient curator of biological sequences in R. Mol Genet Genomics 295: 837–841. https://doi.org/10.1007/s00438-020-01671-z

Pozzi FI, Pratta G, Acuña C, Felitti S (2019) Xenia in bahiagrass: Gene expression at initial seed formation. Seed Science Research 29(1): 29-37.

Sihaloho HF (2015) R and Its Applications in Ecological Research. Mar. Res. Indones. Indones 40: 33–39.

Soresi D, Carrera AD, Echenique V, Garbus I (2015) Identification of genes induced by *Fusarium graminearum* inoculation in the resistant durum wheat line Langdon (Dic-3A) 10 and the susceptible parental line Langdon. Microbiological research 177: 53-66.

Vuylsteke M, Peleman JD, Van Eijk MJT (2007) AFLPbased transcript profiling (cDNA-AFLP) for genome-wide expression analysis. Nature Protocols 2: 1399–1413.

Wickham H (2020). R Packages: Organize, Test, Document and Share your Code. Ed.O´reilly https://r-pkgs.org/index.html

Wickham H, Danenberg P, Csárdi G, Eugster M (2021) In-Line Documentation for R. R package version 7.1.2

Wickham H, Hester J, Chang W, Bryan J (2021). Tools to Make Developing R Packages Easier. R package version 2.4.3.

Xiao X, Li H, Tang C (2009) A silver-staining cDNA-AFLP protocol suitable for transcript profiling in the latex of *Hevea brasiliensis* (Para Rubber Tree). Molecular Biotechnology 42: 91–99.

Yaish MW, Peng M, Rothstein SJ (2014) Global DNA Methylation Analysis Using Methyl-Sensitive Amplification Polymorphism (MSAP). Methods Mol Biol 1062: 285-98.

Xie Y, Sarma A, Vogt A et al (2022) A General-Purpose Package for Dynamic Report Generation in R. R package version 1.39.

**ANEXO**

Documentación del paquete CleanBSequences 1.4.0 generado por CRAN

# Package 'CleanBSequences'

October 12, 2022

**Type** Package

**Title** Curing of Biological Sequences

**Version** 1.4.0

**Author** Florencia I. Pozzi, Silvina A. Felitti

**Maintainer** Florencia I. Pozzi <florenciapoz@hotmail.com>

**Description** Curates biological sequences massively, quickly, without errors and without internet connection. Biological sequences curing is performed by aligning the forward and / or revers primers or ends of cloning vectors with the sequences to be cleaned. After the alignment, new subsequences are generated without biological fragment not desired by the user.

Pozzi et al (2020) <doi:10.1007/s00438-020-01671-z>.

**License** GPL (>= 2)

**Encoding** UTF-8

**Depends** Biostrings

**RoxygenNote** 7.1.2

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-04-27 17:40:06 UTC

## R topics documented:

2 *DNAStringSetTPR*

DNAStringSetOPR *Curing of biological sequences*

**Description**

Curates biological sequences of primer reverse.This cleaning is required for techniques such as cDNA-AFLP.

**Usage**

DNAStringSetOPR(SEQs, PrimerR)

**Arguments**

| | |
|---|---|
| SEQs | file with fasta format containing biological sequences that are to be cleaned. |
| PrimerR | dnastring containing the reverse primer/vector sequences to be removed. |

**Value**

clean biological sequences and visualization of the alignments

**Author(s)**

Florencia I Pozzi, Silvina A. Felitti

**Examples**

```
SEQs = readDNAStringSet(system.file("sequences","SeqInputOPR.fasta", package = "CleanBSequences"))
PrimerR= DNAString ("GACTGCGTACCATGC")
DNAStringSetOPR (SEQs,PrimerR)
```

DNAStringSetTPR *Curing of biological sequences*

**Description**

Curates biological sequences of two restriction enzyme primers or cloning vectors.This cleaning is required for techniques such as cDNA-AFLP.This cleaning is required for techniques such as cDNA-AFLP.

**Usage**

DNAStringSetTPR(SEQs, PrimerF, PrimerR)

*OnePrimerRemove* 3

**Arguments**

SEQs file with fasta format containing biological sequences that are to be cleaned.

PrimerF dnastring containing the foward primer/vector sequences to be removed.

PrimerR dnastring containing the reverse primer/vector sequences to be removed.

**Value**

clean biological sequences and visualization of the alignments

**Author(s)**

Florencia I Pozzi, Silvina A. Felitti

**Examples**

```
SEQs = readDNAStringSet(system.file("sequences","SeqInputTPR.fasta", package = "CleanBSequences"))
PrimerR= DNAString ("GACTGCGTACCATGC")
PrimerF = DNAString("GATGAGTCCTGACCGAA")
DNAStringSetTPR (SEQs,PrimerF,PrimerR)
```

---

OnePrimerRemove *Clean biological secuences*

---

**Description**

Curates biological sequences of primer reverse.This cleaning is required for techniques such as cDNA-AFLP.

**Usage**

```
OnePrimerRemove(SEQs, PrimerR)
```

**Arguments**

SEQs dnastring or file with fasta format containing biological sequences that are to be cleaned.

PrimerR dnastring or file with fasta format containing the reverse primer/vector sequences to be removed.

**Value**

clean biological sequences and visualization of the alignments

**Author(s)**

Florencia I Pozzi and Silvina A. Felitti

**Examples**

```
SEQs = DNAString(paste("GCCTCGCCTCCCTCTTTGATCAGCTTCGCATATCAGGCAACAGCTCAATTT",
"GGTACTTGTTCAAATAAGCATTTAGACCATCTGTTCCAAGAACCTTTGCAATCTT",
"CACAAGGTGGTCATGGTACGCAGTC", sep=""))
PrimerR= DNAString("GACTGCGTACCATGC")
OnePrimerRemove (SEQs,PrimerR)
```

---

TwoPrimerRemove *Clean biological secuences*

---

**Description**

Curates biological sequences of two restriction enzyme primers or cloning vectors.This cleaning is required for techniques such as cDNA-AFLP.

**Usage**

```
TwoPrimerRemove(SEQs, PrimerF, PrimerR)
```

**Arguments**

| | |
|---|---|
| SEQs | DNAString containing biological sequences that are to be cleaned. |
| PrimerF | dnastring containing the foward primer/vector sequences to be removed. |
| PrimerR | dnastring containing the reverse primer/vector sequences to be removed. |

**Value**

clean biological sequences and visualization of the alignments

**Author(s)**

Florencia I Pozzi, Silvina A. Felitti

**Examples**

```
SEQs = DNAString(paste("ACTTTCTGCTGCTTGTGGTCGCAATCAGAGTCCTGATGATGAGTCCTGA",
"CCGAACCCTTTTTCTCCGTCATCCGTTGGTCCATGGTACGCAATCAGAG", sep = ""))
PrimerF = DNAString("GATGAGTCCTGACCGAA")
PrimerR = DNAString("GACTGCGTACCATGC")
TwoPrimerRemove (SEQs,PrimerF,PrimerR)
```

# Index