

Introducción al software estadístico R. Instructivo del paquete IntRo

Nicolás Schmidt



R-package **IntRo**

Instructivo

Nicolás Schmidt

`nschmidt@cienciassociales.edu.uy`

<https://github.com/Nicolas-Schmidt/IntRo>

Índice

1	Introducción	2
2	Notificación de errores	2
3	Instalación del paquete IntRo	3
4	Versiones del paquete IntRo	3
5	Dinámica de la autoevaluación	4
6	Funciones del paquete	5
6.1	Argumento: <code>nombre</code> , <code>apellido</code> , <code>mail</code> , <code>enviar</code>	5
6.2	Argumento: <code>summary</code>	6
6.3	Argumento: <code>intentos</code>	7
7	Proceso de envío	7
8	Ejemplo de autoevaluación	10
9	Funciones adicionales del paquete	11
9.1	Ejemplos:	11

1. Introducción

Este es un instructivo de uso del paquete en desarrollo **IntRo**. Este paquete fue desarrollado para el curso Introducción al software estadístico R que se dicta en la Facultad de Ciencias Sociales de la Universidad de la República (Montevideo, Uruguay). Este paquete contiene un conjunto de funciones que permiten que el estudiante pueda autoevaluarse los ejercicios de los distintos módulos del curso.

El principal motivo del desarrollo de este paquete es didáctico. El inicio en el aprendizaje de un software estadístico y de un lenguaje de programación no es una tarea sencilla. El éxito del aprendizaje y los avances de autonomía en el uso de una herramienta de estas características dependen de muchos factores (cursos, estudio, dedicación...), pero, fundamentalmente debe ser acompañado de mucha práctica. Gran parte de los avances en esta tarea provienen de la práctica y experimentación con el uso del programa y el lenguaje.

Por este motivo es que este paquete obliga al estudiante a tener que usar R para hacer los ejercicios y para hacer la corrección de los mismos. También debe usar R para enviar por mail la corrección al docente. Esto significa que los estudiantes se van a familiarizar y van a practicar repetidamente la carga de paquetes, la instalación, el uso de funciones, la corrección de estilo, la construcción de objetos, la manipulación de objetos etc.

En este instructivo va a encontrar la mayor cantidad de explicaciones de posibles errores o situaciones en las que puede encontrar escollos para seguir avanzando en la autoevaluación de los ejercicios.

2. Notificación de errores

No todos los posibles errores que se puedan encontrar en el proceso de autocorrección de los ejercicios serán cubiertos en este instructivo. En tal sentido, todos los problemas con los que se encuentre debe notificarlos al docente por mail tratando de cubrir distintos aspectos relevantes para encontrar una solución rápida al error.

Siempre que vaya a reportar errores debe tratar de cubrir los siguientes aspectos en su reporte:

- Módulo del ejercicio sobre el que tiene problemas.

- Número del ejercicio.
- Enviar una copia del código que produce un error.
- Enviar una copia del mensaje del error.

Si al reportar un error todos estos ítem están completos el docente va a poder replicar lo que el estudiante está haciendo y encontrar con mayor facilidad el error y de ahí encontrar el origen del mismo.

3. Instalación del paquete **IntRo**

Este paquete no se encuentra en CRAN principalmente por dos razones. La primera y fundamental es que está en desarrollo. La segunda es que no justifica su ingreso al CRAN dado que es una herramienta para un curso en particular. Esto hace que para su instalación se deba recurrir a un repositorio público en GitHub donde está alojado el paquete.

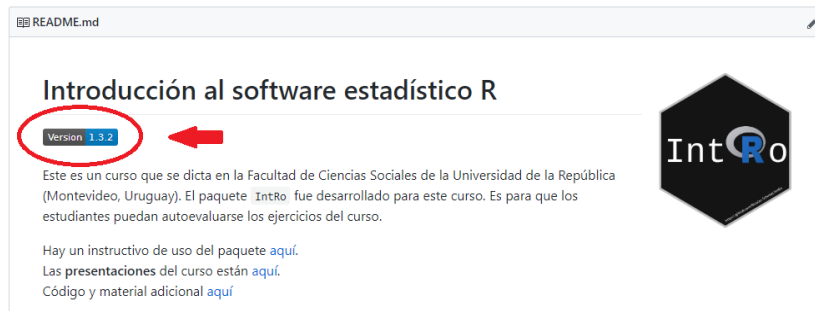
El proceso de instalación de este paquete es igual que para un paquete que se descarga desde el CRAN. Para los paquetes que se encuentran en Git se debe usar la función `install_github` del paquete `devtools`.

```
install.packages("devtools")
library(devtools)
install_github("Nicolas-Schmidt/IntRo")
library(IntRo)
```

Como la función `install_github` pertenece al paquete `devtools` es que debemos instalar ese paquete y llamarlo a la sesión de trabajo en la que estamos trabajando para poder usar dicha función.

4. Versiones del paquete **IntRo**

Como ya se dijo, el paquete **IntRo** está en contante desarrollo ya que sigue las necesidades de un curso que es dinámico. En tal sentido en la parte superior izquierda de la portada de la pagina inicial del repositorio del curso se va a encontrar con un ícono que identifica la version actual del paquete.



Se recomienda tener al momento de realizar la autoevaluación la última versión. Hay distintas maneras de verificar la versión de un paquete instalado.

```
packageVersion("IntRo")  
  
## [1] '1.3.5'
```

Si se trabaja en RStudio también se puede consultar en el desplegable de los paquetes instalados. Se busca el paquete y además de una breve descripción aparece la versión instalada del paquete.

Si al verificar la versión del paquete se verifica que está trabajando en una versión anterior debe volver a instalar el paquete desde el repositorio como se detalló en la sección anterior. Si quiere eliminar el paquete antes de volver a instalar la nueva versión lo puede hacer con la función `remove.packages()`. Sea cual sea la opción que utilice (actualizar o eliminar + actualizar) recuerde verificar que se instaló la versión correcta.

5. Dinámica de la autoevaluación

Los estudiantes van a tener cuatro repartidos de ejercicios. Cada uno de los repartidos contiene ejercicios dedicados a cada una de las cuatro estructuras de datos básicas de R dados en el curso: `vector()`, `matrix()`, `data.frame()` y `list()`. Estos se deben realizar en R y van a tener la posibilidad de autocorregirse y de modificar y arreglar los ejercicios todas las veces que entiendan necesario hacerlo.

Una vez que el estudiante considere que ya no quiere avanzar más con la autocorrección deberá enviar el resultado del repartido por mail desde R. El

estudiante va a conocer la nota final del ejercicio una vez que envíe el repartido. Al enviar el repartido el estudiante va a recibir un mail con el resultado y el mismo mail lo va a recibir el docente.

6. Funciones del paquete

El paquete tiene una función para cada módulo. Las funciones llevan el nombre de `auto_eval_`^{*i*}. Según el módulo es el nombre de la función.

Módulo	Función
vectores	<code>auto_eval_vector()</code>
matrices	<code>auto_eval_matrix()</code>
data.frame	<code>auto_eval_df()</code>
listas	<code>auto_eval_list()</code>

Todas las funciones tienen los mismos seis argumentos:

```
auto_eval_i(nombre = NULL,
            apellido = NULL,
            mail = NULL,
            intentos = 1,
            enviar = FALSE,
            summary = TRUE)
```

6.1. Argumento: nombre, apellido, mail, enviar

Los primeros tres argumentos son los necesarios para identificar el proceso de envío del repartido al docente para la obtención de la nota final. Los tres argumentos son de tipo `character`, esto significa que deben ir entre comillas:

```
# EJEMPLO
auto_eval_i(nombre = "Su_NOMBRE",
            apellido = "Su_APELLIDO",
            mail = "Su_EMAIL",
            intentos = 1,
            enviar = FALSE,
            summary = TRUE)
```

En estos tres argumentos el estudiante debe cargar sus datos: su *nombre*, su *apellido* y su *email*. Recuerde que usted va a enviar un mail desde su casilla de correo electrónico. De manera conjunta usted va a recibir un correo con la nota final y el docente va a recibir una copia de ese correo.

El argumento `enviar` es de tipo `logical` por lo que solo admite dos valores: `TRUE` o `FALSE`. Por defecto la función tiene `FALSE` por lo que es el estudiante el que debe activar el proceso de envío poniendo `TRUE` cuando lo entienda necesario.

```
# EJEMPLO
auto_eval_i(nombre = "Su_NOMBRE",
            apellido = "Su_APELLIDO",
            mail = "Su_EMAIL",
            intentos = 1,
            enviar = TRUE,
            summary = TRUE)
```

Debe ser cuidadoso con esto ya que una vez que fue enviado el repartido (si bien puede volver a enviarlo) solo será evaluado el primer envío.

6.2. Argumento: `summary`

El valor que admite este argumento es de tipo `logical` y por defecto es `TRUE`. Simplemente controla la impresión en pantalla del estado de situación del repartidos. Si es `TRUE` se va a imprimir una tabla en la consola que va a reportar el estado de situación de cada uno de los ejercicios del repartido. Los estados posibles son 1- Correcto, 2- Incorrecto y 3- Incompleto.

Un ejemplo de cómo se ve la tabla de resumen es el que aparece abajo. Los valores pueden ser 0 o 1.

```
##
## =====
##          RESULTADO DE LOS EJERCICIOS
## =====
##
##          Correcta Incorrecta Incompleta
## Ejercicio 1          0          1          0
## Ejercicio 2          0          0          1
## Ejercicio 3          1          0          0
## Ejercicio 4          1          0          0
```

## Ejercicio 5	1	0	0
## Total	3	1	1

En el ejemplo que aparece el estudiante tiene 3 respuestas correctas, 1 respuesta incorrecta y 1 ejercicio que aún no está hecho.

Debe saber que puede ir corrigiendo a medida que va realizando los ejercicios y no necesariamente debe tener todos los ejercicios hechos para ir a la autoevaluación. Recuerde que la autoevaluación persigue un fin didáctico y como tal debe interactuar con la función la cantidad de veces que entienda necesario hacerlo. Todos los ejercicios tienen mensajes de advertencia que en caso de cometer errores van a ayudarlo a encontrar soluciones.

6.3. Argumento: intentos

En este argumento es donde entra la honestidad y el pacto que se desea hacer con los estudiantes. En este argumento que admite números enteros el estudiante debe declarar la cantidad de veces que intentó autoevaluarse. Evidentemente, a medida que aumentan los intentos disminuye la nota final ya que hay computado un pequeño detractor.

Este valor es completamente subjetivo y debe ser lo más honesto posible. Si usted corrió la función muchas veces pero solo para evaluar ejercicios distintos, deberá usted mismo establecer el número que le parece que corresponde.

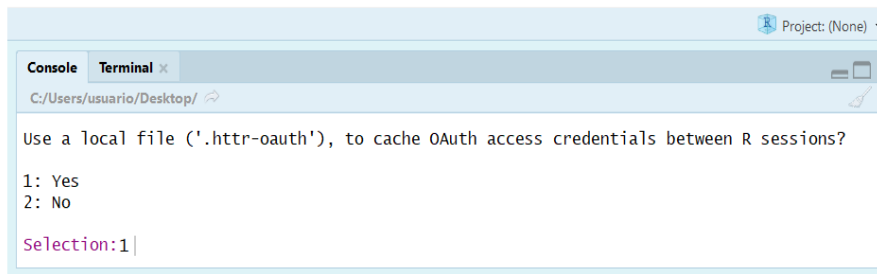
7. Proceso de envío

El proceso de envío se debe activar en el momento en el que el estudiante o bien terminó con todos los ejercicios o no quiere avanzar más de lo que ya avanzó.

Para realizar un envío se deben completar los argumentos: **nombre**, **apellido**, **mail** y poner **TRUE** en el argumento **enviar**. Debe ser cuidadoso con los problemas de tipeo en la redacción de su mail.

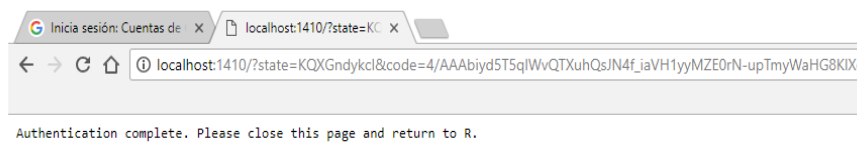
Para poder enviar un mail desde R desde su dirección de mail deberá habilitarlo. El proceso es muy sencillo y se describe a continuación.

Lo primero que les va a aparecer es esto:



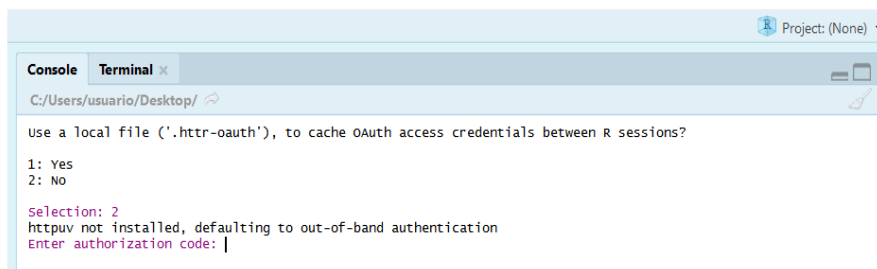
```
Project: (None)
Console Terminal x
C:/Users/usuario/Desktop/
Use a local file ('.httr-oauth'), to cache OAuth access credentials between R sessions?
1: Yes
2: No
Selection: 1 |
```

Si optan por la opción 1 los va a direccionar a su mail y ahí le dan permiso al paquete para que envíe el mail. Una vez confirmen eso en una pagina web les va a aparecer lo siguiente:



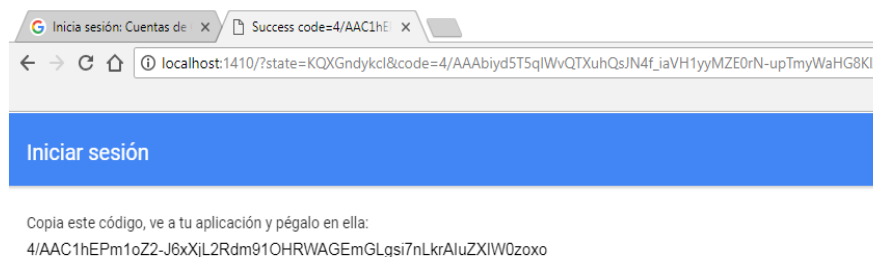
Esto indica que el proceso fue correcto.

Si optan por la opción 2:



```
Project: (None)
Console Terminal x
C:/Users/usuario/Desktop/
Use a local file ('.httr-oauth'), to cache OAuth access credentials between R sessions?
1: Yes
2: No
selection: 2
httpuv not installed, defaulting to out-of-band authentication
Enter authorization code: |
```

Van a ser redireccionados a una pagina web para habilitar el permiso de enviar mail desde su correo y les va a aparecer el código:

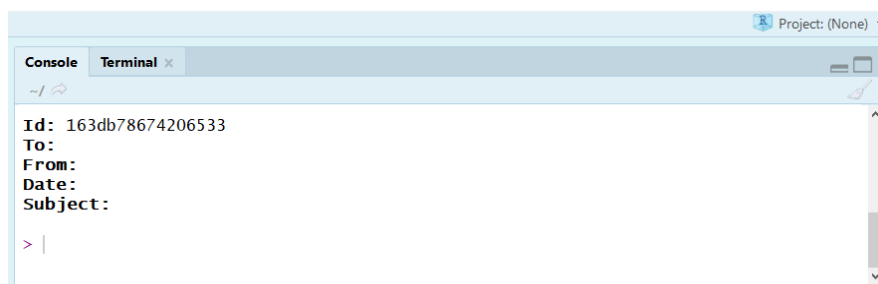


Cuando les aparece el código lo copian y lo pegan en la consola.



```
Project: (None)
Console Terminal x
C:/Users/usuario/Desktop/
Use a local file ('.httr-oauth'), to cache OAuth access credentials between R sessions?
1: Yes
2: No
Selection: 2
httpuv not installed, defaulting to out-of-band authentication
Enter authorization code: 4/AACiHEPm1oZ2-36xxjLZrdm91OHRWAGemGLgsi7nLkfrAIuZXIW0zoxo
```

Marcando cualquiera de las dos opciones el resultado final es el mismo. Les tiene que aparecer esto:



```
Project: (None)
Console Terminal x
~/
Id: 163db78674206533
To:
From:
Date:
Subject:
> |
```

Esto significa que el mail fue enviado exitosamente¹. Si aparece el prompt ('>'), y no hay mensaje de error, R está pronto para recibir otra orden!

Lo que usted esta haciendo al mandar la evaluación es conectar su mail a R para poder mandar un mail. Por esta razón es que le pide que autentifique la cuenta. En este envío el docente está en copia para que también le llegue la nota final de cada repartido de ejercicios que usted manda. Esto también se puede verificar en su bandeja de mails enviados. Ahí va a encontrar el mail que usted mismo se envió con copia al docente.

¹Si no les aparece algo similar a lo de la imagen deben verificar que el mail lo recibieron. En caso de que no aparezca lo de la imagen ni reciban el mail deben notificar al docente.

8. Ejemplo de autoevaluación

```
# -----  
# MODULO VECTORES  
# -----  
library(IntRo) # cargo el paquete del curso  
# Ejercicio 1:  
vec1 <- seq(2, 20, 2)  
  
# evaluo el ejercicio  
auto_eval_vector()  
  
##  
## =====  
##          RESULTADO DE LOS EJERCICIOS  
## =====  
##  
##          Correcta Incorrecta Incompleta  
## Ejercicio 1          1          0          0  
## Ejercicio 2          0          0          1  
## Ejercicio 3          0          0          1  
## Ejercicio 4          0          0          1  
## Ejercicio 5          0          0          1  
## Ejercicio 6          0          0          1  
## Ejercicio 7          0          0          1  
## Ejercicio 8          0          0          1  
## Ejercicio 9          0          0          1  
## Ejercicio 10         0          0          1  
## Ejercicio 11         0          0          1  
## Ejercicio 12         0          0          1  
## TOTAL                1          0         11
```

Resuelvo que voy a enviar el resultado del repartido. Entonces cargo los argumentos `nombre`, `apellido`, `email` con mis datos y ejecuto la función:

```
auto_eval_vector(nombre = "Nicolas",  
                  apellido = "Schmidt",  
                  mail = "nicoschlab@gmail.com",  
                  intentos = 3,  
                  enviar = TRUE,  
                  summary = TRUE)
```

9. Funciones adicionales del paquete

El paquete cuenta con dos funciones adicionales de utilidad. Estas funciones son: `rfunction()` y `ci()`. La primera de ellas tiene como objetivo estimular la lectura de código fuente y la experimentación con funciones nuevas. Esta función devuelve una función aleatoria de las que tiene un paquete en particular que se tenga instalado o por defecto una función del paquete `base`.

La segunda función es un atajo para verificar las reglas de coerciones implícitas de R en relación a los tipos de datos. Por defecto devuelve la coerción de los tipos de un vector atómico.

9.1. Ejemplos:

```
library(IntRo)
rfunction()

##
## - The function of the day is: bindingIsLocked
## - Package: base --> 1224 functions
##
## - Package Description: Base R functions.

rfunction(package = "stats")

##
## - The function of the day is: rexp
## - Package: stats --> 447 functions
##
## - Package Description: R statistical functions.

ci()

##          logical  integer  double   raw      complex  character
##          -----  -
## logical  | logical
## integer  | integer  integer
## double   | double   double   double
## raw       | logical  integer  double   raw
## complex  | complex  complex  complex  complex  complex
## character | character character character character character character

ci(c("integer", "double", "logical"))

##          integer double  logical
##          -----  -
## integer  | integer
## double   | double  double
## logical  | integer double  logical
```

NICOLAS SCHMIDT. Docente e Investigador en el Instituto de Ciencia Política (Facultad de Ciencias Sociales, Universidad de la República, Uruguay). Magíster en Ciencia Política por la Universidad de la República y Licenciado en Ciencia Política por la misma Universidad. Mail: nschmidt@cienciassociales.edu.uy