imarpe: un paquete para la automatización de gráficos, tablas y reportes usando R

Criscely Luján-Paredes

24 de Julio de 2017

Resumen

Este documento es una introducción al uso del paquete `imarpe`, el cuál proporciona herramientas para la elaboración de gráficas, tablas y reportes que se realizan de manera rutinaria en las investigaciones producidas por el Instituto del Mar del Perú (IMARPE). El objetivo principal de `imarpe` es que los usuarios trabajen en R sin requerir conocimientos avanzados de programación, ahorrando tiempo en el procesamiento de información.

Palabras clave: R, imarpe, automatización, reportes, figuras, tablas.

Introducción

`imarpe` es un paquete implementado en R cuya principal función es automatizar el procesamiento de información así como automatizar la elaboración de gráficas, tablas, y reportes que son necesarios en el trabajo rutinario para el personal del Instituto del Mar del Perú (IMARPE). El diseño de este paquete es flexible, permitiendo al usuario realizar cambios sobre los parámetros de los resultados ya tendrán a su disposición el código en R que genera los productos; adicionalmente, el diseño de este paquete permite trabajar con funciones genéricas (e.g. plot, summary, print) sobre cada clase de datos, facilitando de esta manera su utilización incluso con conocimientos mínimos de R.

Dentro de las principales ventajas que proporciona `imarpe` están:

- No requiere que el usuario posea conocimientos avanzados de programación y/o R,
- Permite ahorrar tiempo en el procesamiento de la información,
- Brinda al usuario un procesamiento automatizado de la información,
- Permite que el usuario realice cambios sobre los parámetros de las salidas (tablas, figuras y reportes.)
- Proporciona gráficas de buena calidad así como la capacidad de realizar análisis reproducibles.

Programación orientada a objetos

R tiene tres sistemas orientados a objetos (S3, S4 y R5), y todos trabajan con los conceptos "clase" y "método". Una clase se define como un tipo de objeto, describiendo que propiedades posee, como funciona y como este objeto se puede relacionar con otros tipos de objetos; de esta manera, cada objeto debe poseer una clase. Por otro lado, un método se define como aquella función que está asociada a un tipo particular de objetos.

S3 implementa un estilo de programación orientada a objetos llamada función genérica OO (o en inglés, generic-function OO). En el funcionamiento de este sistema se realizan cálculos a través de métodos y a través de un tipo especial de función llamada "función genérica", la cual decide qué método llamar.

El uso principal de S3 en R es a través de los métodos print, summary and plot. Estos métodos permiten tener una función genérica por ejemplo de print, que mostraría el objeto de una manera especial.

Instalación de imarpe

Para usar imarpe se debe tener instalado R, y se recomienda usar la interface gráfica de R Studio. Para la instalación de todo paquete de R se require contar con internet y luego de verificar esto, desde R Studio se debe intalar y cargar el paquete devtools, de la siguiente manera:

```
install.packages("devtools")
library("devtools")
```

La instalación de imarpe se realizará directamente de la cuenta github del Instituto del Mar del Peru (IMARPE) https://github.com/imarpe, lugar donde se trabaja en las mejoras y actualizaciones de los paquetes en R desarrollados por el personal del IMARPE.

En esta web también se encuentra imarpe, y su instalación se realizará usando las siguientes líneas de código. La primera línea de código instala el paquete y debe ser corrida por única vez, sin embargo, cada vez que se haga una actualización del paquete, se deberá correr nuevamente para cargar de manera automática la nueva versión de imarpe. La segunda línea de código se encargará de cargar el paquete la instalado, esta debe ser ejecutada cada vez que se quiera usar el paquete.

```
install_github("imarpe/imarpe")
library(imarpe)
```

Estructura de imarpe

imarpe cuenta con el desarrollo del módulo de pesquería. Sin embargo, el objetivo es extender el desarrollo del paquete a tres módulos más, los cuales estén conformados por los modulos de biología, oceanografía y cruceros hidroacústicos.

Módulo Pesquería:

Esta sección trabaja sobre información pesquera, orientada al análisis de las variables: captura y desembarque, esfuerzo y captura por unidad de esfuerzo (cpue), reproduciendo tablas, gráficas y reportes de manera automatizada. Así mismo, el Programa de Observadores a Bordo (Programa de Bitácoras de Pesca) del IMARPE también cuenta un subsección de trabajo dentro del módulo de pesquería, el cual permite obtener las principales gráficas y tablas que son requeridas en los reportes periódicos que emiten.

Las principales funciones con las que cuenta imarpe son 4 y las explicaremos al detalle en las siguientes sub-secciones.

```
getFishingData
getBitacoraData
getMainResults.bitacora
getDailyReport
```

1. Análisis de información pesquera

1.1 Uso de getFishingData

Esta función es usada para dos tipos de clases de datos: fishery y cpue. La primera clase de datos contiene variables pesqueras que en este caso incluye a los desembarques (landing) y al esfuerzo (effort). La segunda clase de datos incluye a la captura por unidad de esfuerzo, de ahi el nombre de la clase.

imarpe incluye una base de datos interna (fisheryData) que será usada para ejemplificar el uso de esta función. Sin embargo, toda base de datos debe poseer información sobre el tiempo (año, mes y dia), sobre la

especie en estudio, sobre el tipo de flota y puertos, la información de la captura (o desembarque) así como del esfuerzo pesquero, la información de posición (latitud y longitud) es opcional.

El nombre de las columnas debe ser el mismo al de la base de datos de ejemplo, sin embargo el uso de mayúsculas y minúsculas será resuelto internamente.

En la siguiente demostración del uso de `getFishingData`, la función analiza los desembarques de la base de datos contenida en el archivo .csv (archivo delimitado por comas) para obtener como producto i) un archivo de clase data.frame con la base de datos analizada por puerto en resolución temporal diaria, ii) una lista con las principales características de la base de datos analizada, y iii) un resumen de la información por tipo de flota en resolución temporal mensual.

Adicionalmente, la ayuda de esta función puede ser visualizada para una mejor comprensión de los parámetros de la función.

```
# Cargar la base de datos
fisheryData = system.file("extdata", "fisheryData.csv", package = "imarpe")

# Para cargar la ayuda de la función
?getFishingData

# Usar getFishingData con la base de datos ejemplo (fisheryData) de caballa
landing  = getFishingData(file = fisheryData, type = "fisheryinfo", varType = "landing",
                          sp = "caballa")

class(landing)
dataBase = landing$data
info = landing$info
fleet = landing$fleeTable

# Para analizar un periodo de tiempo de la base de datos
landing  = getFishingData(file = fisheryData, type = "fisheryinfo", varType = "landing",
                          sp = "caballa", start = "2009-04-10", end = "2009-08-30")

# Para analizar la información de un puerto específico
landing  = getFishingData(file = fisheryData, type = "fisheryinfo", varType = "landing",
                          sp = "caballa", start = "2009-04-10", end = "2009-08-30",
                          port = "PAITA")
```

De manera similar al análisis de los desembarques se analizará la información del esfuerzo pesquero continuando con el uso de la función `getFishingData` así como de la base de datos interna (`fisheryData`).

Para el análisis de esta variable (esfuerzo) el tipo de información (parámetro `type`) sigue siendo `fisheryInfo` pero el tipo de variable (parámetro `varType`) será `effort`; y se debe precisar que tipo de esfuerzo pesquero se analizará en el parámetro `efforType`. En el siguiente ejemplo se usó capacidad de bodega, pero podría ser número de viajes, número de anzuelos así como número de embarcaciones (ver ayuda de la función `getFishingData` para más información).

```
# Para analizar la información de esfuerzo con la base de datos ejemplo (fisheryData)
# de caballa
effort  = getFishingData(file = fisheryData, type = "fisheryinfo", varType = "effort",
                         sp = "caballa", efforType = "capacidad_bodega")

class(effort)
dataBase = effort$data
info = effort$info
fleet = effort$fleeTable
```

Finalmente, el análisis de la captura por unidad de esfuerzo (cpue) se analizará en la siguiente sección. Aquí los parámetros `type` y `varType` deben ser iguales a `cpue`, y se debe precisar que tipo de esfuerzo se usará (en `efforType`) para calcular la cpue.

```
# Para analizar la información de cpue con la base de datos ejemplo (fisheryData)
# de caballa
cpue  = getFishingData(file = fisheryData, type = "cpue", varType = "cpue",
                       sp = "caballa", efforType = "capacidad_bodega")

class(cpue)
dataBase = cpue$data
info = cpue$info
fleet = cpue$fleeTable
```

Para los dos últimos ejemplo, de esfuerzo y cpue, también se pueden usar los parámetros `start` y `end` para hacer un análisis sobre un periodo de tiempo específico, así como el parámetro `port`, para analizar la información de un determinado puerto.

Respecto a los métodos, se construyeron cinco métodos para cada una de las clases de datos `fishery` y `cpue`. Estos son: `print`, `summary`, `print.summary`, `plot` y `report` y las ayudas para cada método debe ser llamada por el método seguido por un punto (".") y luego la clase de datos correspondiente.

Métodos de la clase `fishery`

```
# Revisar la ayuda de los métodos de la clase fishery
?print.fishery
?summary.fishery
?print.summary.fishery
?plot.fishery
?report.fishery
```

La descripción de cada método se dará a continuación:

- print.fishery: muestra un resumen del contenido de información de la base de datos de clase fishery. Esta información incluye: (1) el nombre de la base de datos utilizada; (2) el número de registros que poseen los datos; el periodo de tiempo analizado ((3) meses y (4) años); (5) el número de puertos que poseen los datos; (6) la especie analizada en la base de datos y (7) el tipo de variable, en este caso sería `landing` o `effort`.
- summary.fishery: proporciona una lista con: (1) el tipo de variable que fue analizada; (2) una base de datos diaria por puerto de la variable analizada; (3) una base de datos total en escala temporal diaria; una base de datos por puerto; (4) una base de datos en escala temporal mensual; (5) y una base de datos en escala temporal anual.
- print.summary.fishery: este método permite visualizar los productos del método `summary.fishery`.
- plot.fishery: este método permite realizar siete tipos de gráficos con datos de la clase `fishery`.
- report.fishery: este método exporta un reporte en formato pdf de la base de datos analizada. Internamente este método distingue que tipo de variable ha sido analizada y reproduce un reporte con un formato para los desembarques y otro para el esfuerzo.

Una breve ejemplificación del uso de los métodos de la clase `fishery` se muestra en la siguiente sección:

```
# Cargar la base de datos
fisheryData = system.file("extdata", "fisheryData.csv", package = "imarpe")

# Crear un objeto de la clase fishery usando la base de datos ejemplo (fisheryData) de #caballa
landing  = getFishingData(file = fisheryData, type = "fisheryinfo", varType = "landing",
```

```
                          sp = "caballa")

#Algunos ejemplos del uso de los métodos
print(landing)

sumLanding = summary(landing)
print(sumLanding)

sumLanding = summary(landing, language = "english")
print(sumLanding, language = "english")

plot(landing)

report(landing, daysToPlot = "15")
```

Métodos de la clase cpue

```
# Revisar la ayuda de los métodos de la clase cpue
?print.cpue
?summary.cpue
?print.summary.cpue
?plot.cpue
?report.cpue
```

La descripción de cada método se dará a continuación:

- print.cpue: muestra un resumen del contenido de información de la base de datos de clase cpue. La información que proporciona este método es similar a la que proporciona `print.fishery`, sin embargo puede revisar la ayuda de `print.cpue` para mayor información.
- summary.cpue: proporciona una lista con: (1) el tipo de esfuerzo que fue usado para estimar la cpue; (2) una base de datos diaria por puerto; (3) una base de datos total en escala temporal diaria; una base de datos por puerto; (4) una base de datos en escala temporal mensual; (5) y una base de datos en escala temporal anual.
- print.summary.cpue: este método permite visualizar los productos del método `summary.cpue`.
- plot.cpue: este método permite realizar seis tipos de gráficos con datos de la clase `cpue`.
- report.cpue: este método exporta un reporte en formato pdf de la base de datos analizada.

Una breve ejemplificación del uso de los métodos de la clase `cpue` se muestra en la siguiente sección:

```
# Cargar la base de datos
fisheryData = system.file("extdata", "fisheryData.csv", package = "imarpe")

# Crear un objeto de la clase cpue usando la base de datos ejemplo (fisheryData) de
#caballa
cpue  = getFishingData(file = fisheryData, type = "cpue", varType = "cpue",
                       sp = "caballa", efforType = "capacidad_bodega")

#Algunos ejemplos del uso de los métodos
print(cpue)

sumCpue = summary(cpue)
print(sumCpue)
```

```r
sumCpue = summary(cpue, language = "english")
print(sumCpue, language = "english")

plot(cpue)

report(cpue, daysToPlot = "1")
```

Uso de `combineFisheryVar`

Esta función permite combinar en un sólo objeto de clase `fishery` la información del desembarque y esfuerzo. El método `report.fishery` está disponible para ser usado en este tipo de objetos, permitiendo de esta manera la producción de un reporte unificado con la información de ambas variables pesqueras.

Adicionalmente, este tipo de objetos puede ser usado por el método `plot.fishery` usando el `ploType = "joined"`. Esto produce una gráfica en escala temporal diaria con la información de ambas variables (desembarque y esfuerzo), el desembarque en el eje vertical izquierdo y el esfuerzo en el eje vertical derecho.

```r
# Cargar la base de datos
fisheryData = system.file("extdata", "fisheryData.csv", package = "imarpe")

# Objeto de clase fishery para la variable tipo landing
landingOject = getFishingData(file = fisheryData, type = "fisheryinfo",
                              varType = "landing", sp = "caballa")

# Objeto de clase fishery para la variable tipo effort
effortObject = getFishingData(file = fisheryData, type = "fisheryinfo",
                              varType = "effort", sp = "caballa",
                              efforType = "viaje")

# Uso de la función combineFisheryVar
fisheryVar = combineFisheryVar(landing = landingOject, effort = effortObject)
class(fisheryVar)

# Gráfica
plot(fisheryVar, ploType = "plotJoined")

# report
report(fisheryVar, type = "joined")
report(fisheryVar, type = "joined", daysToPlot = "1")
```

1.2 Uso de `getDailyReport`

`getDailyReport` es usada para realizar un análisis de los desembarques pesqueros de anchoveta respecto a la biomasa estimada en las cuotas de captura del recurso. Esta función se encarga de descargar de manera automática los desembarques diarios de anchoveta que estan disponibles en la página web del Instituto del Mar del Perú (IMARPE) y en función a sus parámetros principales: fechas de análisis, información biológica de la especie, resultados del crucero para la estimación de la biomasa, la cuota oficial de biomasa; se realiza un análisis cuyos resultados serán guardados en un objeto de clase `fishingMonitoring`. Este objecto se guardará en el formato `RData` en el directorio donde se esté realizando el trabajo.

```r
# Revisar la ayuda de getDailyReport para más información
?getDailyReport
```

Método de la clase `fishingMonitoring`

La finalidad principal de `getDailyReport` es la construcción del reporte diario de la pesquería de anchoveta, por lo que el método construido para los objetos de la clase `fishingMonitoring` es `report.fishingMonitoring`.

```
# Revisar la ayuda del método de la clase fishingMonitoring.
?report.fishingMonitoring
```

A continuación se incluirá un ejemplo del uso de `getDailyReport`. Este código servirá para comprender el formato que deben tener los datos para poder obtener un objeto de clase `fishingMonitoring`. El siguiente código debe ser ejecutado sólo si se cuenta con los datos de entrada necesarios.

```
# Correr el siguiente código solo si se cuenta con la información necesaria.

# Lista de fechas
datesList = list(startDate = "2017-4-20",
                 endDate = "2017-6-19",
                 startExploringDate = "2017-4-22",
                 endExploringDate = "2017-4-25",
                 startSeasonDate = "2017-4-22",
                 endSeasonDate = "2017-7-31")

# Frecuencia simple por talla
simpleFreqSizes = "DataTallas_2017-I.csv"

# Archivo de crucero en formato RData
dataCruise = "cr_170304_det.RData"

#Valor de biomasa oficial (en toneladas)
officialBiomass = 7778463

# Parámetros a y b
a = 0.0034
b = 3.273

x = getDailyReport(datesList = datesList, simpleFreqSizes = simpleFreqSizes,
                   dataCruise = dataCruise, officialBiomass = officialBiomass,
                   a = a, b = b)
class(x)

report(x)
```