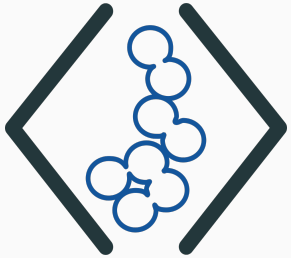


Paquetes en R (I)

Gonzalo García-Donato

Jueves, 8 de junio de 2017

Universidad de Castilla-la Mancha

Índice de contenidos

Introducción

Motivación

- Un paquete de R es un conjunto de funciones y datos junto con documentación para su uso.

Motivación

- Un paquete de R es un conjunto de funciones y datos junto con documentación para su uso.
- Normalmente asimilamos los paquetes a proyectos de cierta envergadura pensando en hacer útil nuestro trabajo para muchos otros.

Motivación

- Un paquete de R es un conjunto de funciones y datos junto con documentación para su uso.
- Normalmente asimilamos los paquetes a proyectos de cierta envergadura pensando en hacer útil nuestro trabajo para muchos otros.
- ¿Porqué paquetes en un curso de reproducibilidad?

Motivación

- Un paquete de R es un conjunto de funciones y datos junto con documentación para su uso.
- Normalmente asimilamos los paquetes a proyectos de cierta envergadura pensando en hacer útil nuestro trabajo para muchos otros.
- ¿Porqué paquetes en un curso de reproducibilidad?

Motivación

- Un paquete de R es un conjunto de funciones y datos junto con documentación para su uso.
- Normalmente asimilamos los paquetes a proyectos de cierta envergadura pensando en hacer útil nuestro trabajo para muchos otros.
- ¿Porqué paquetes en un curso de reproducibilidad?

La construccion y uso de nuestros propios paquetes es muy efectivo para armonizar procedimientos dentro de un grupo de trabajo, ya que:

- permiten compartir facilmente un entorno particular de funciones y datos

Motivación

- Un paquete de R es un conjunto de funciones y datos junto con documentación para su uso.
- Normalmente asimilamos los paquetes a proyectos de cierta envergadura pensando en hacer útil nuestro trabajo para muchos otros.
- ¿Porqué paquetes en un curso de reproducibilidad?

La construccion y uso de nuestros propios paquetes es muy efectivo para armonizar procedimientos dentro de un grupo de trabajo, ya que:

- permiten compartir facilmente un entorno particular de funciones y datos

```
library("pintamapas")
```

- su caracter de empaquetado y su identificacion por versiones posibilita la trazabilidad de los resultados, facilitando la reproducibilidad

```
packageVersion("pintamapas")
```

El nombre

Antes de empezar!

Es muy importante que elijamos un nombre para la criatura que debe evocar las funcionalidades contenidas. Algunos autores recomiendan no utilizar mayúsculas y evitar nombres muy largos.

El nombre

Antes de empezar!

Es muy importante que elijamos un nombre para la criatura que debe evocar las funcionalidades contenidas. Algunos autores recomiendan no utilizar mayúsculas y evitar nombres muy largos.

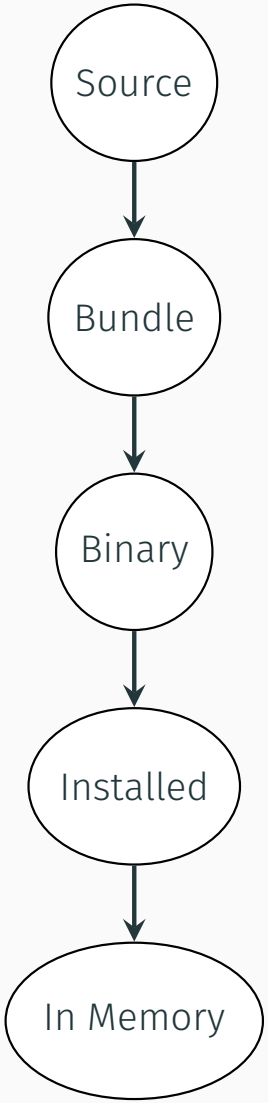
En esta sesión imaginaremos que el nombre elegido del paquete es, ”pintamapas”.

Los estados de un paquete

Los estados de un paquete

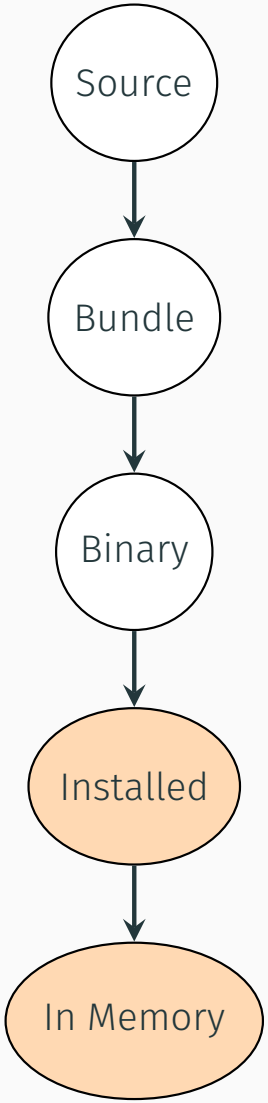
- Para entender el proceso de construcción de un paquete y, lo que es más importante, para compartir un paquete es importante conocer los estados por los que puede pasar: desde su estado de desarrollo (“Source”) a cargalo en memoria (“In memory”).

Los estados de un paquete



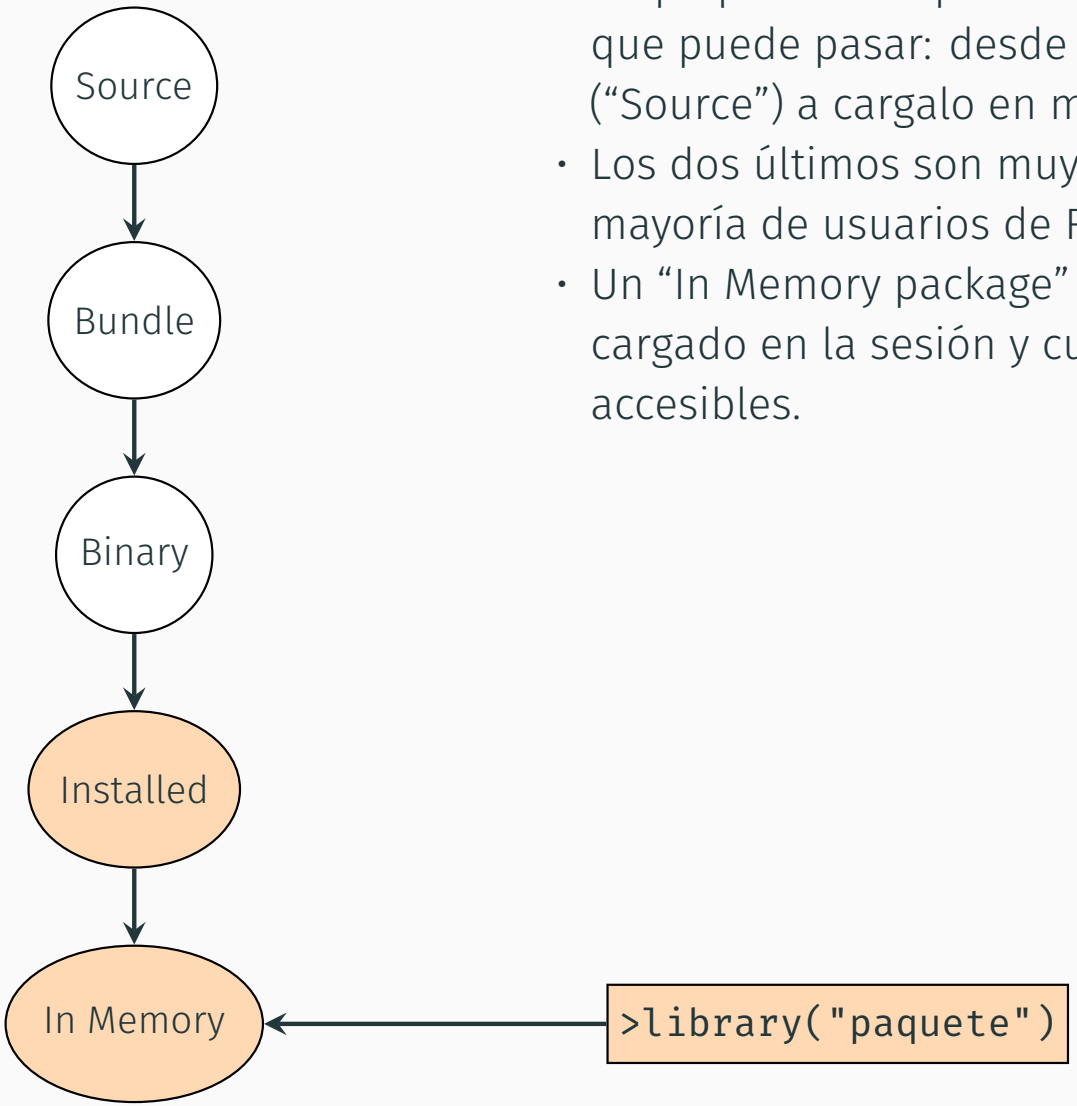
- Para entender el proceso de construcción de un paquete y, lo que es más importante, para compartir un paquete es importante conocer los estados por los que puede pasar: desde su estado de desarrollo (“Source”) a cargalo en memoria (“In memory”).

Los estados de un paquete



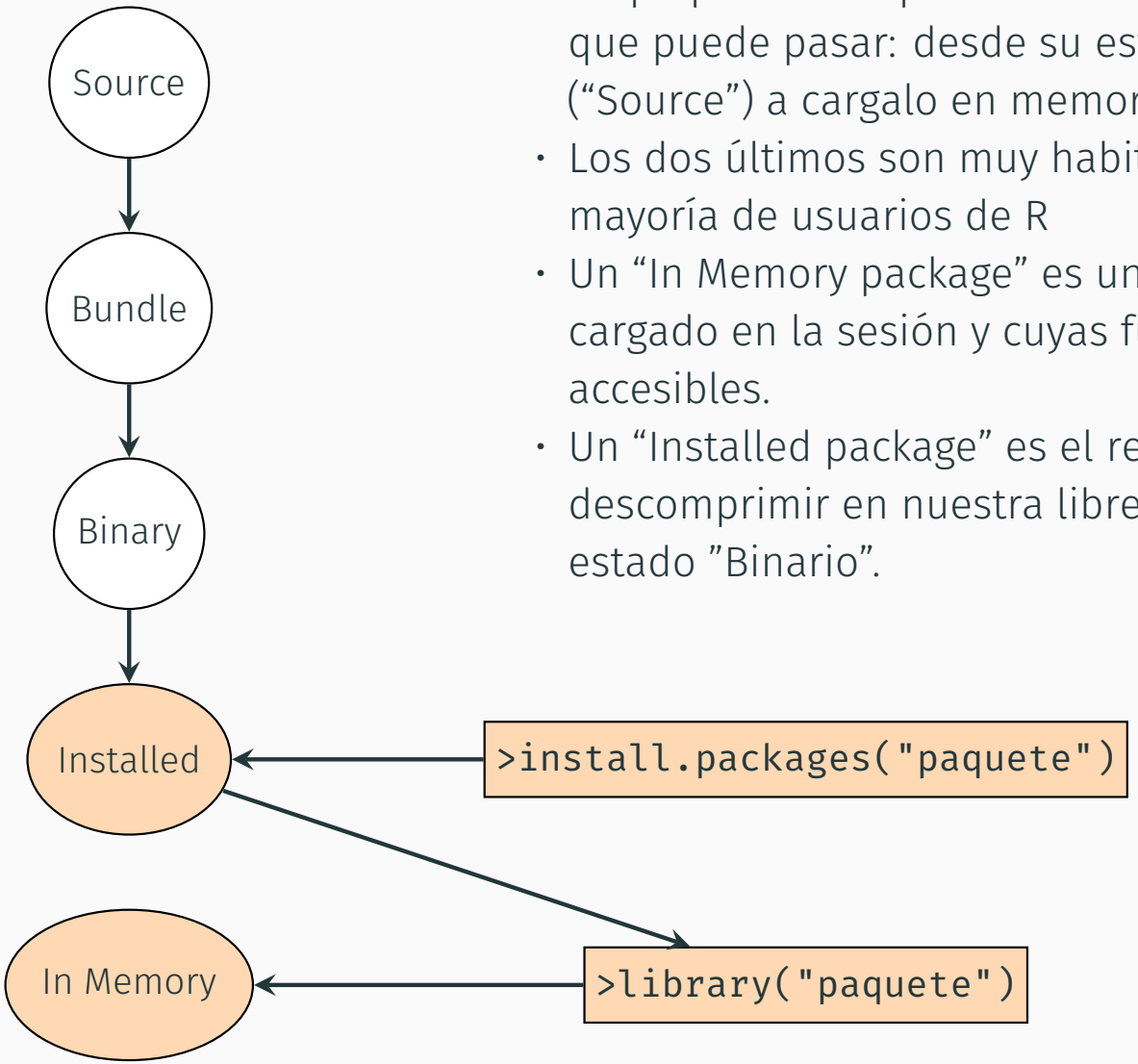
- Para entender el proceso de construcción de un paquete y, lo que es más importante, para compartir un paquete es importante conocer los estados por los que puede pasar: desde su estado de desarrollo (“Source”) a cargalo en memoria (“In memory”).
- Los dos últimos son muy habituales para la gran mayoría de usuarios de R

Los estados de un paquete



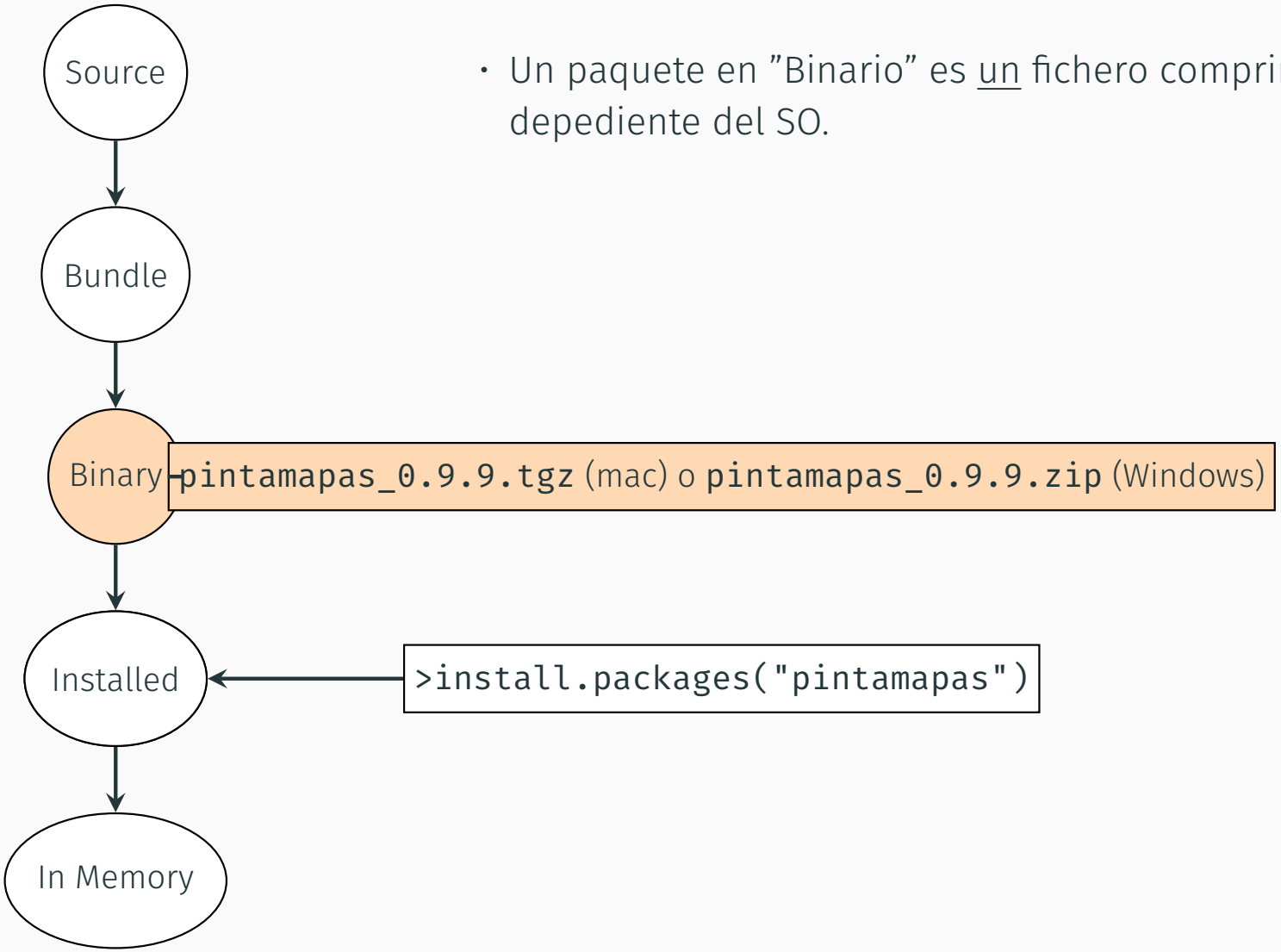
- Para entender el proceso de construcción de un paquete y, lo que es más importante, para compartir un paquete es importante conocer los estados por los que puede pasar: desde su estado de desarrollo (“Source”) a cargalo en memoria (“In memory”).
- Los dos últimos son muy habituales para la gran mayoría de usuarios de R
- Un “In Memory package” es un paquete que hemos cargado en la sesión y cuyas funciones y datos son accesibles.

Los estados de un paquete



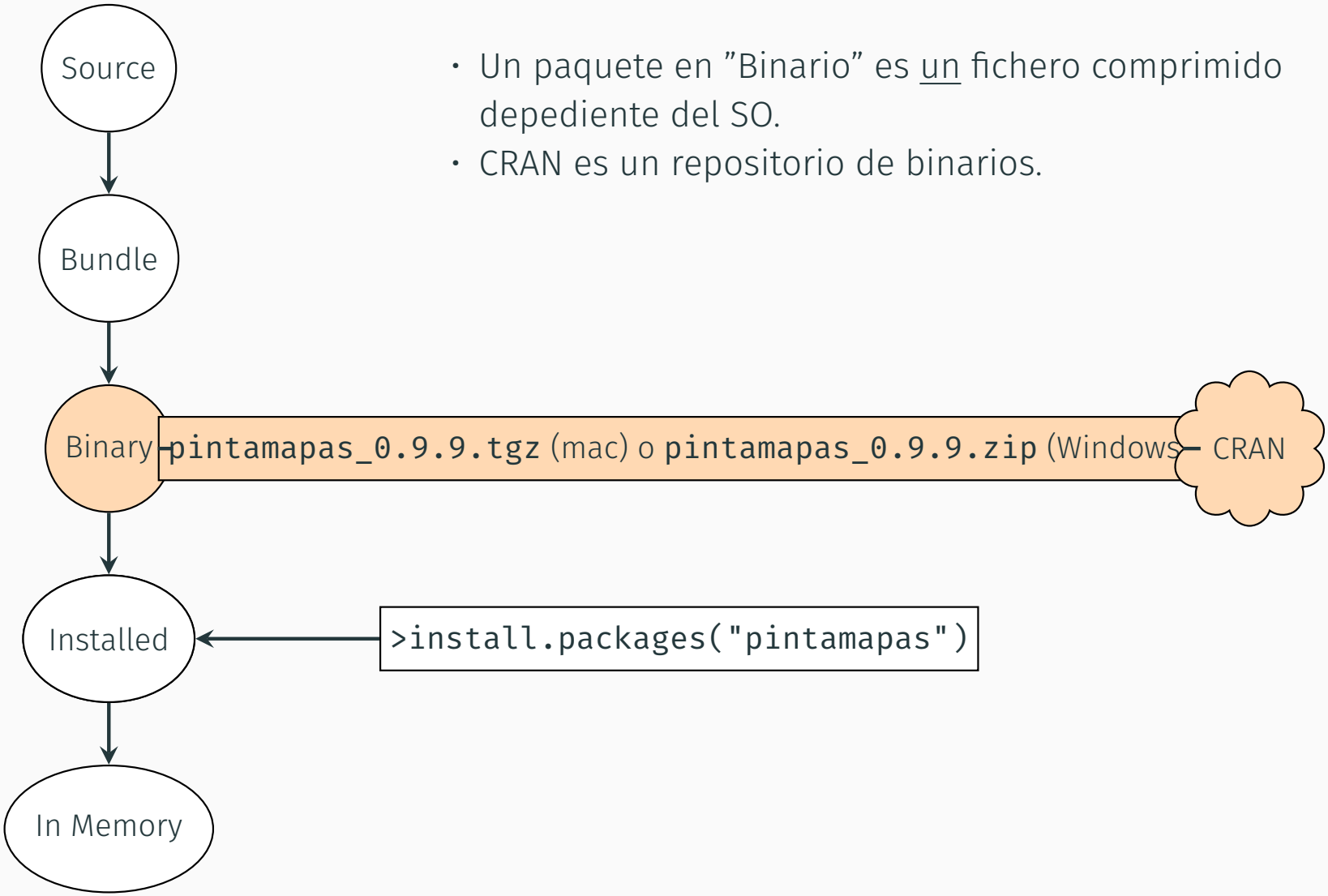
- Para entender el proceso de construcción de un paquete y, lo que es más importante, para compartir un paquete es importante conocer los estados por los que puede pasar: desde su estado de desarrollo (“Source”) a cargalo en memoria (“In memory”).
- Los dos últimos son muy habituales para la gran mayoría de usuarios de R
- Un “In Memory package” es un paquete que hemos cargado en la sesión y cuyas funciones y datos son accesibles.
- Un “Installed package” es el resultado de descomprimir en nuestra librería de R un paquete en estado ”Binario”.

El estado “Binary” y instalando desde CRAN



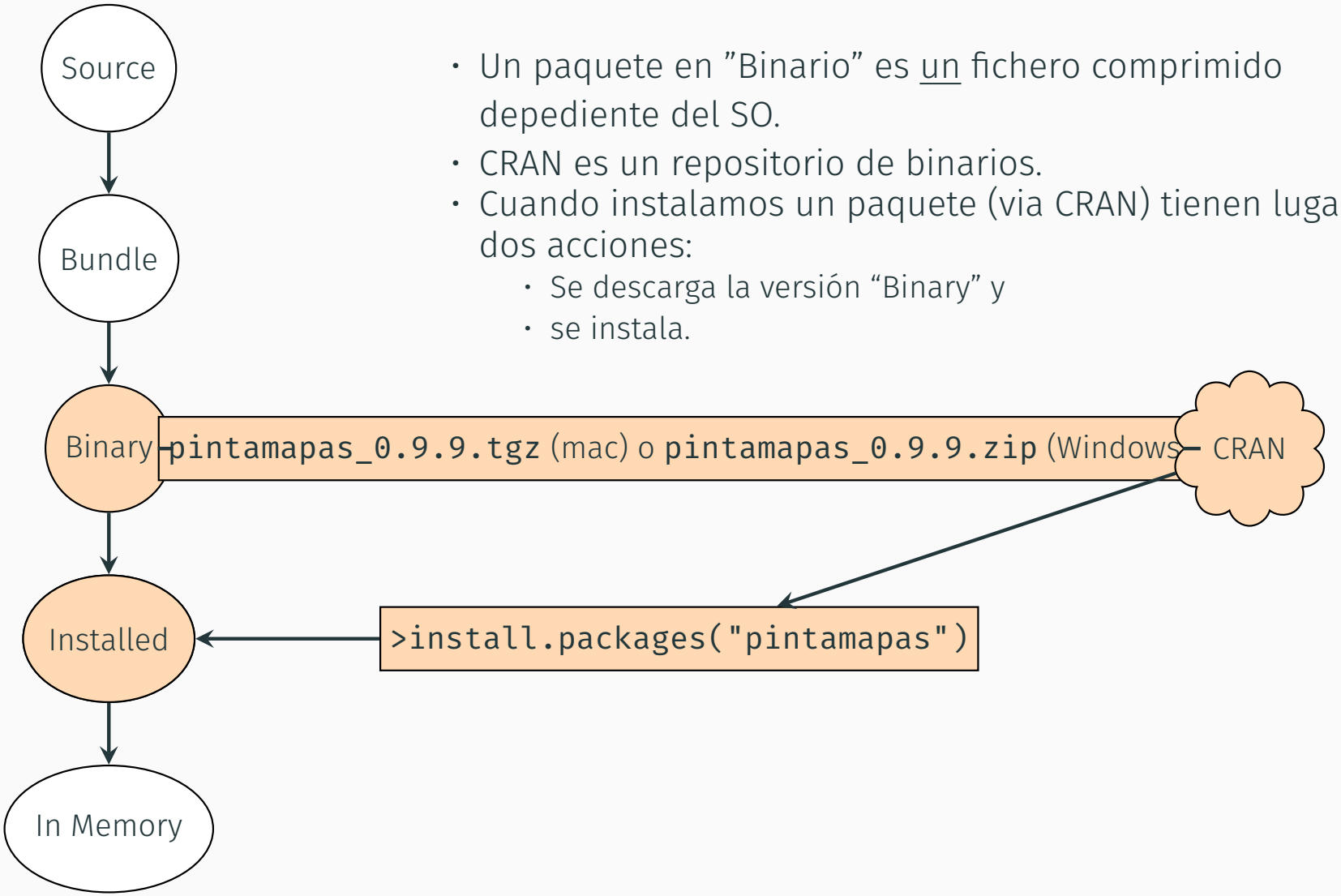
- Un paquete en ”Binario” es un fichero comprimido depediente del SO.

El estado “Binary” y instalando desde CRAN



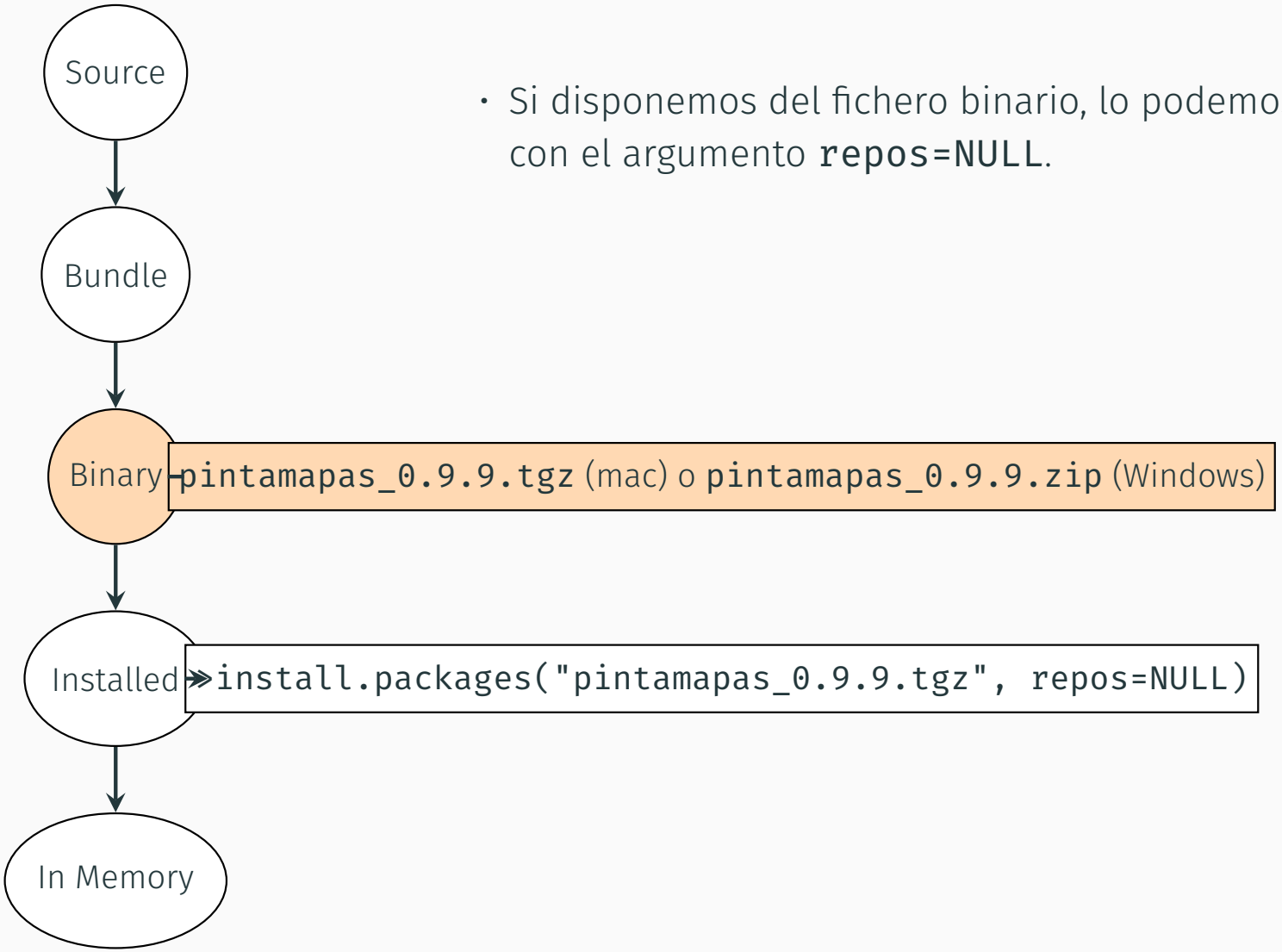
- Un paquete en ”Binario” es un fichero comprimido depediente del SO.
- CRAN es un repositorio de binarios.

El estado “Binary” y instalando desde CRAN



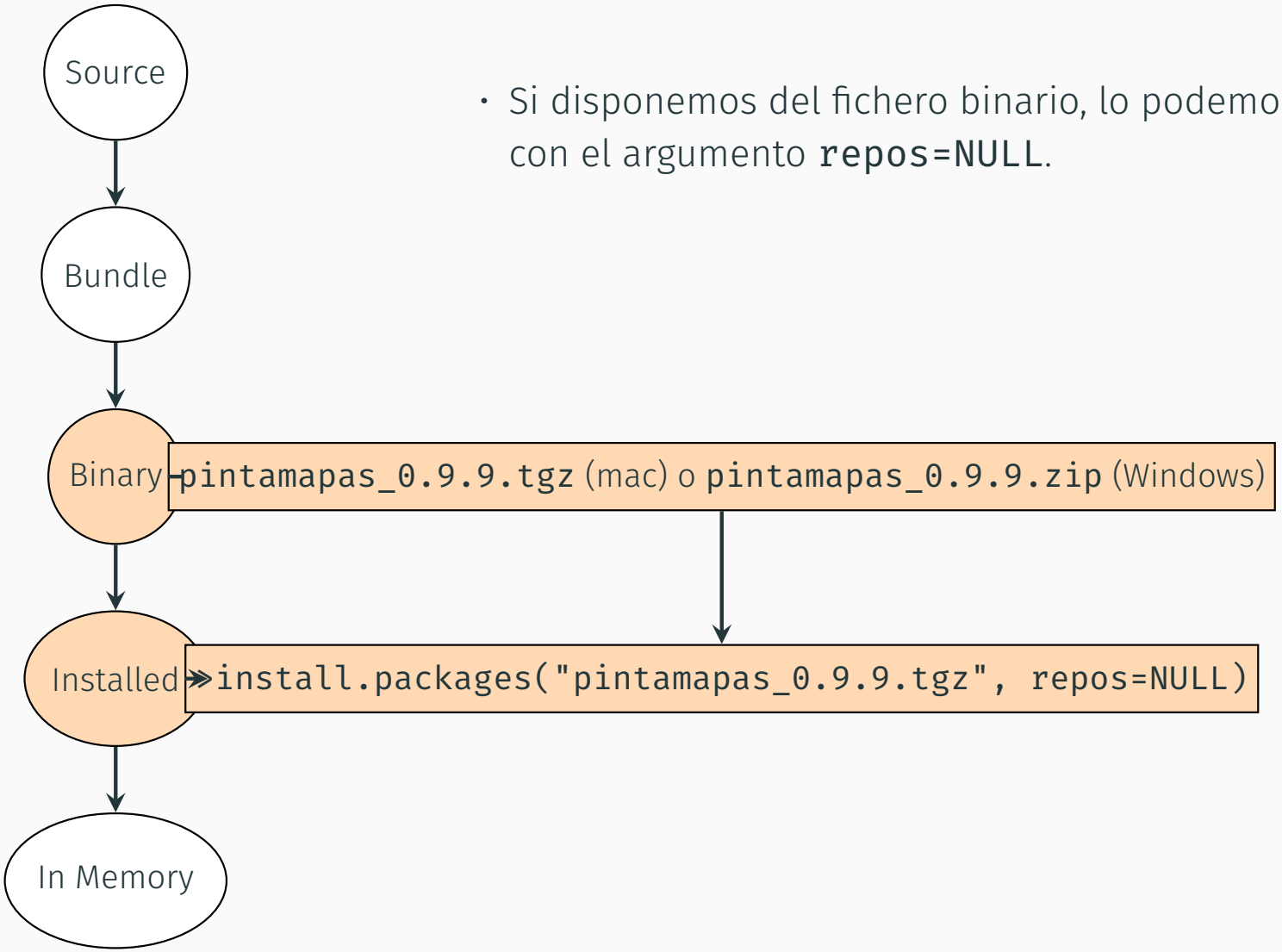
- Un paquete en ”Binario” es un fichero comprimido depediente del SO.
- CRAN es un repositorio de binarios.
- Cuando instalamos un paquete (via CRAN) tienen lugar dos acciones:
  - Se descarga la versión “Binary” y
  - se instala.

El estado “Binary”: una primera forma de compartir



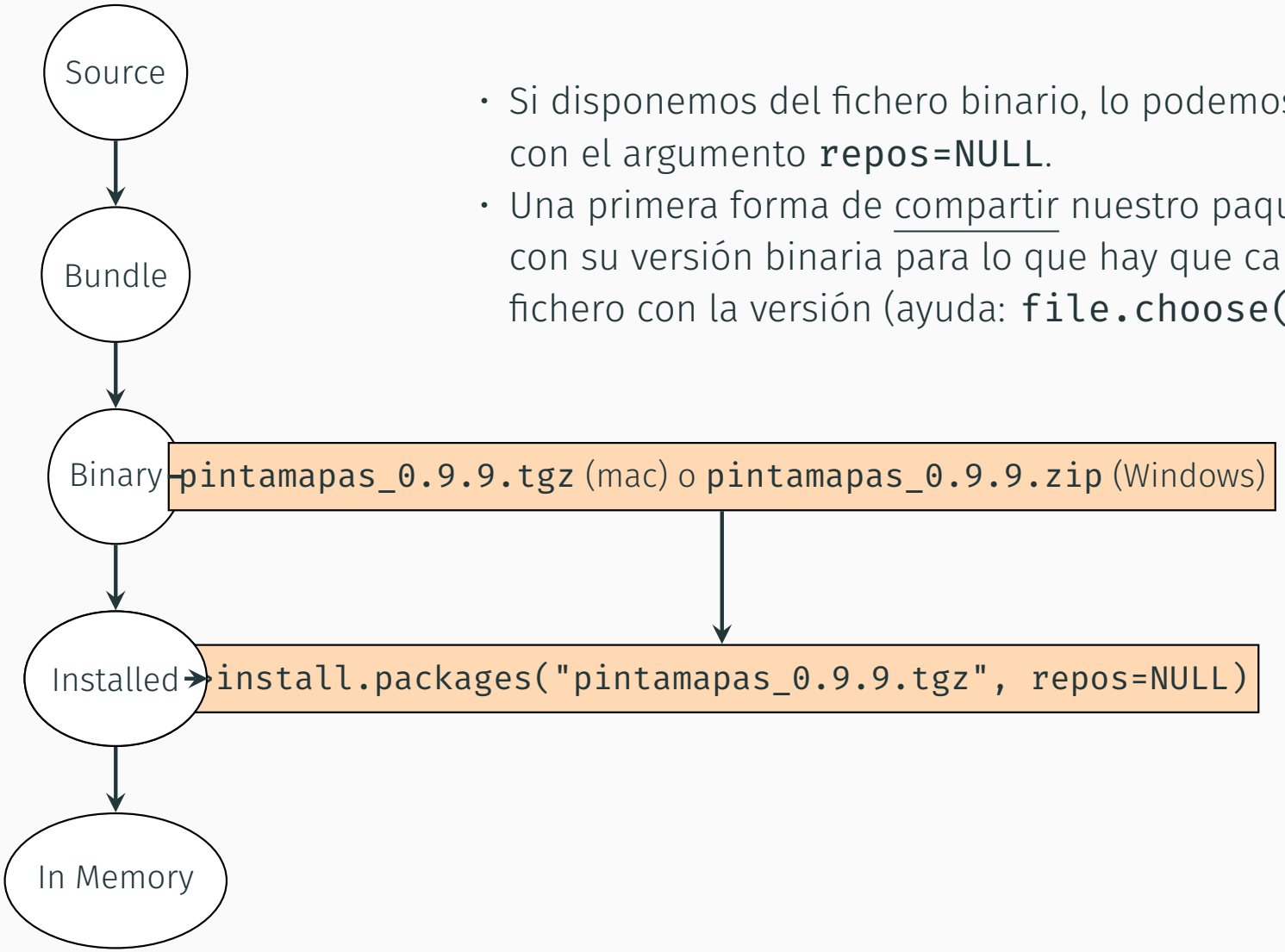
- Si disponemos del fichero binario, lo podemos instalar con el argumento `repos=NULL`.

El estado “Binary”: una primera forma de compartir



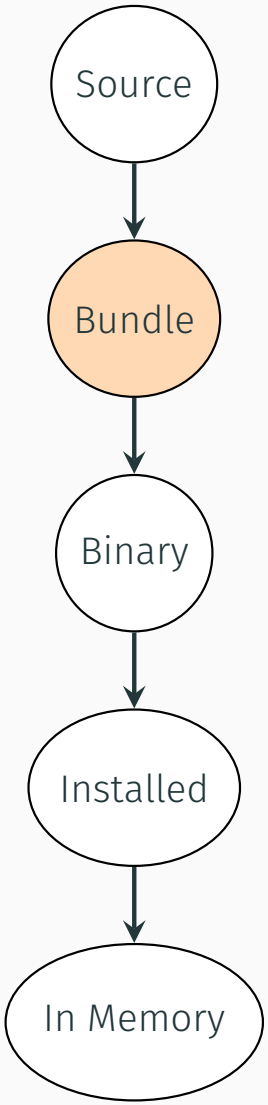
- Si disponemos del fichero binario, lo podemos instalar con el argumento `repos=NULL`.

El estado “Binary”: una primera forma de compartir



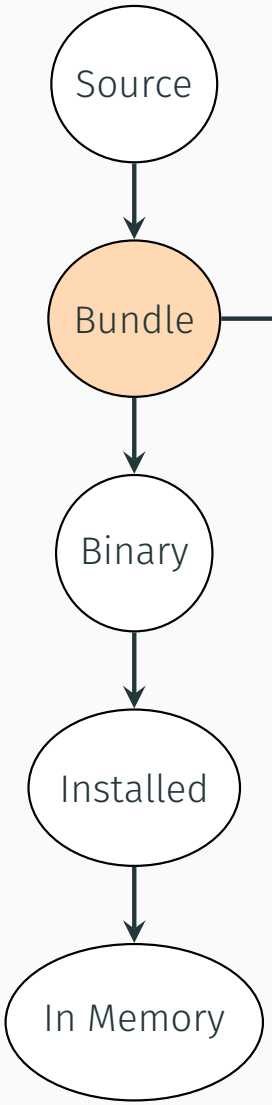
- Si disponemos del fichero binario, lo podemos instalar con el argumento `repos=NULL`.
- Una primera forma de compartir nuestro paquete es con su versión binaria para lo que hay que cargar el fichero con la versión (ayuda: `file.choose()`).

El estado “Bundle”: una segunda forma de compartir



- Un paquete en estado “Bundle” (excepto por ficheros ignorados explicitamente) es un fichero que comprime el paquete en estado “Source”.

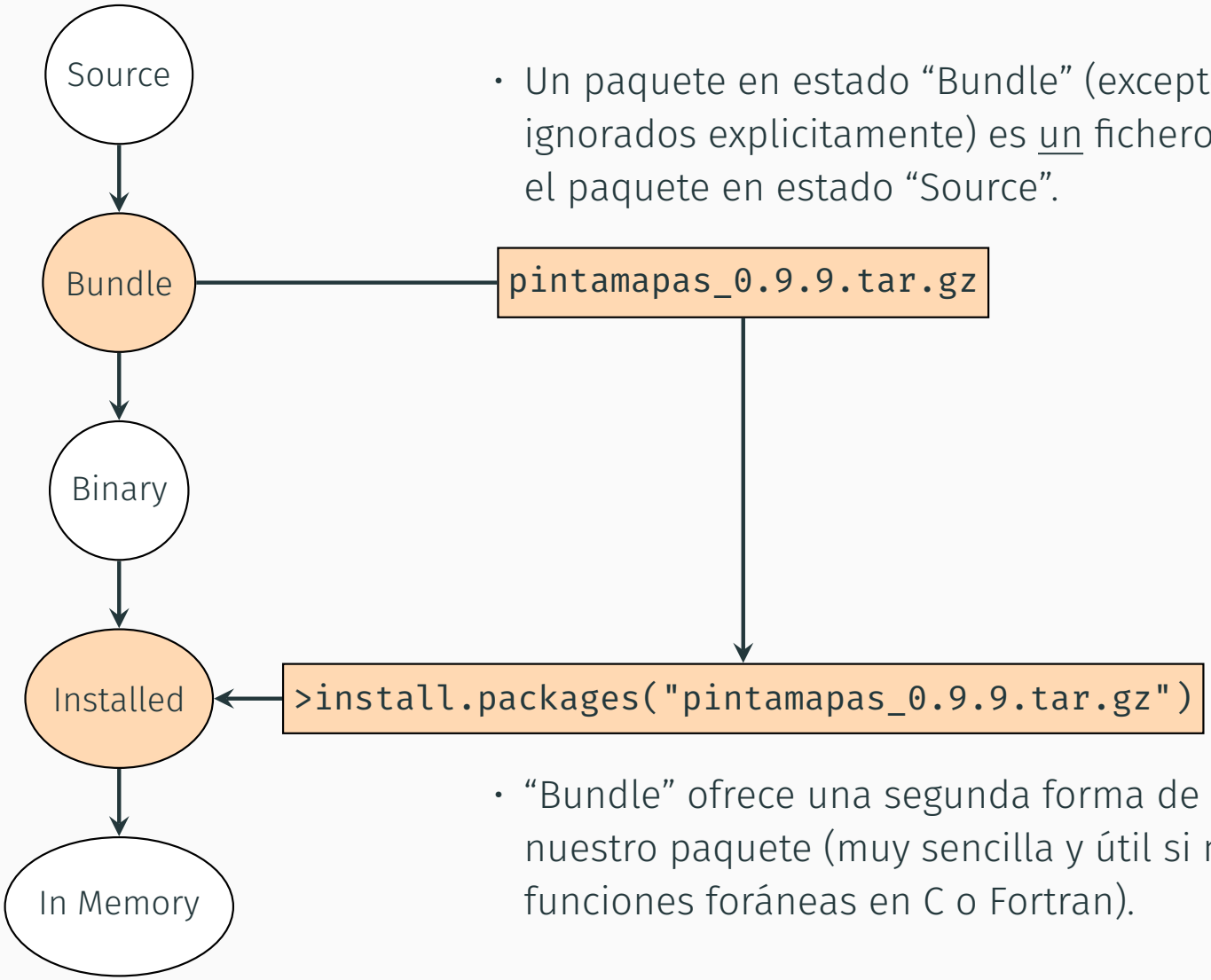
El estado “Bundle”: una segunda forma de compartir



- Un paquete en estado “Bundle” (excepto por ficheros ignorados explicitamente) es un fichero que comprime el paquete en estado “Source”.

`pintamapas_0.9.9.tar.gz`

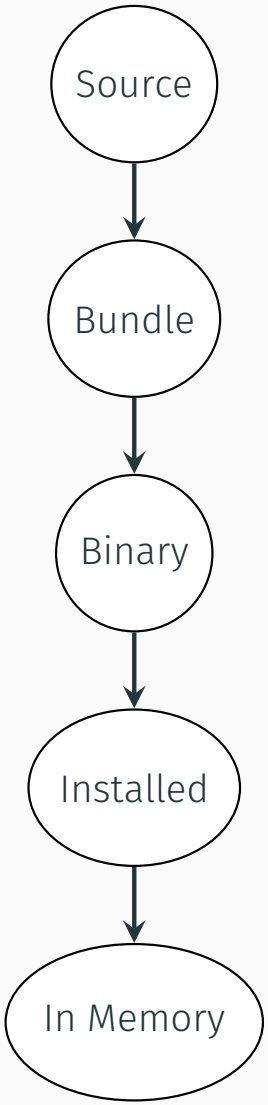
El estado “Bundle”: una segunda forma de compartir



- Un paquete en estado “Bundle” (excepto por ficheros ignorados explicitamente) es un fichero que comprime el paquete en estado “Source”.

- “Bundle” ofrece una segunda forma de compartir nuestro paquete (muy sencilla y útil si no tenemos funciones foráneas en C o Fortran).

Desde “Source” a “‘bundle” o “‘binary”

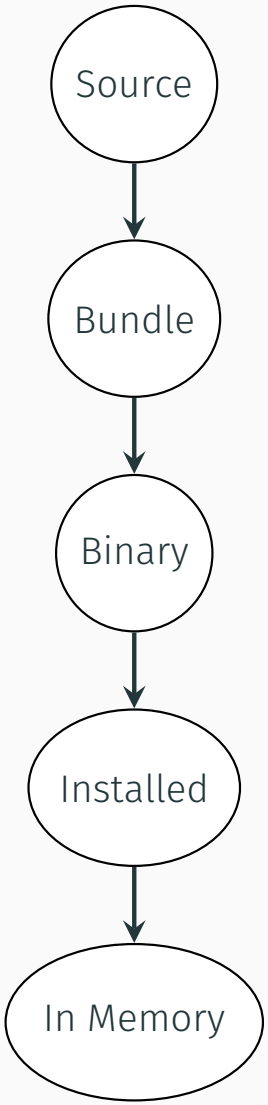


- A partir de un paquete en “Source”, podemos obtener un “Binary” con

```
devtools::build("pintamapas", binary = TRUE)
```

Para ésto hay que tener instalada Rtools.exe en Windows o Xcode en mac. Se crea un fichero con extensión (.tgz o .zip).

Desde “Source” a “‘bundle” o “‘binary”



- A partir de un paquete en “Source”, podemos obtener un “Binary” con

```
devtools::build("pintamapas", binary = TRUE)
```

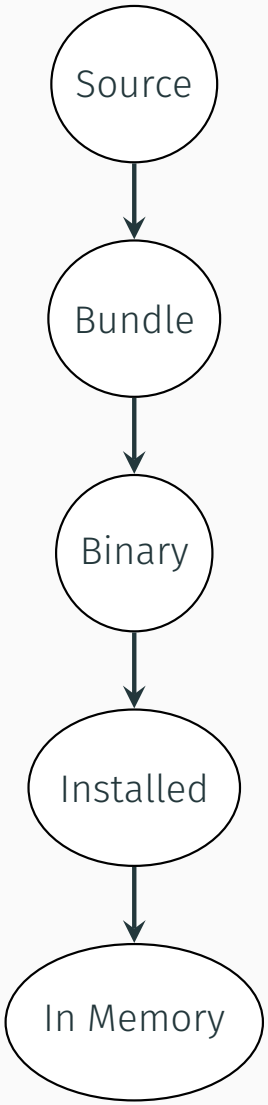
  Para ésto hay que tener instalada Rtools.exe en Windows o Xcode en mac. Se crea un fichero con extensión (.tgz o .zip).

- o un “Bundle” con

```
devtools::build("pintamapas")
```

  y se crea un fichero con extensión (.tar.gz)

Desde “Source” a “‘bundle” o “‘binary”



- A partir de un paquete en “Source”, podemos obtener un “Binary” con

```
devtools::build("pintamapas", binary = TRUE)
```

  Para ésto hay que tener instalada Rtools.exe en Windows o Xcode en mac. Se crea un fichero con extensión (.tgz o .zip).

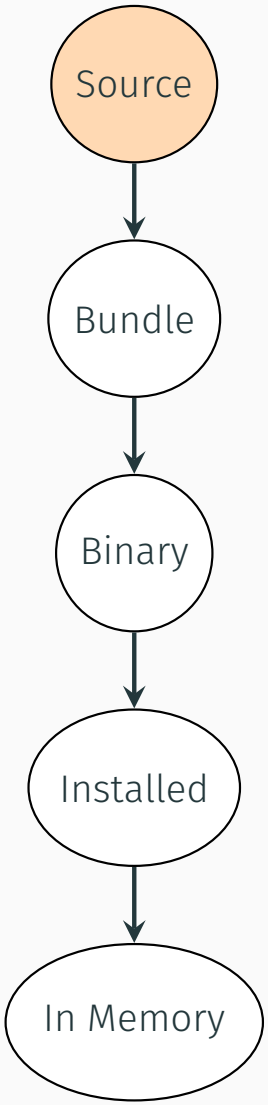
- o un “Bundle” con

```
devtools::build("pintamapas")
```

  y se crea un fichero con extensión (.tar.gz)

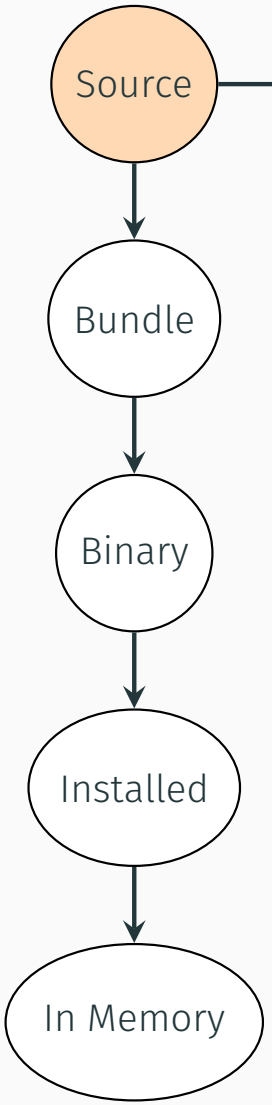
- Y ya estaríamos preparados para mandarselo a los compañeros!

“Source”: el estado más importante para el desarrollo



- “Source” es el paquete en su estado de desarrollo.

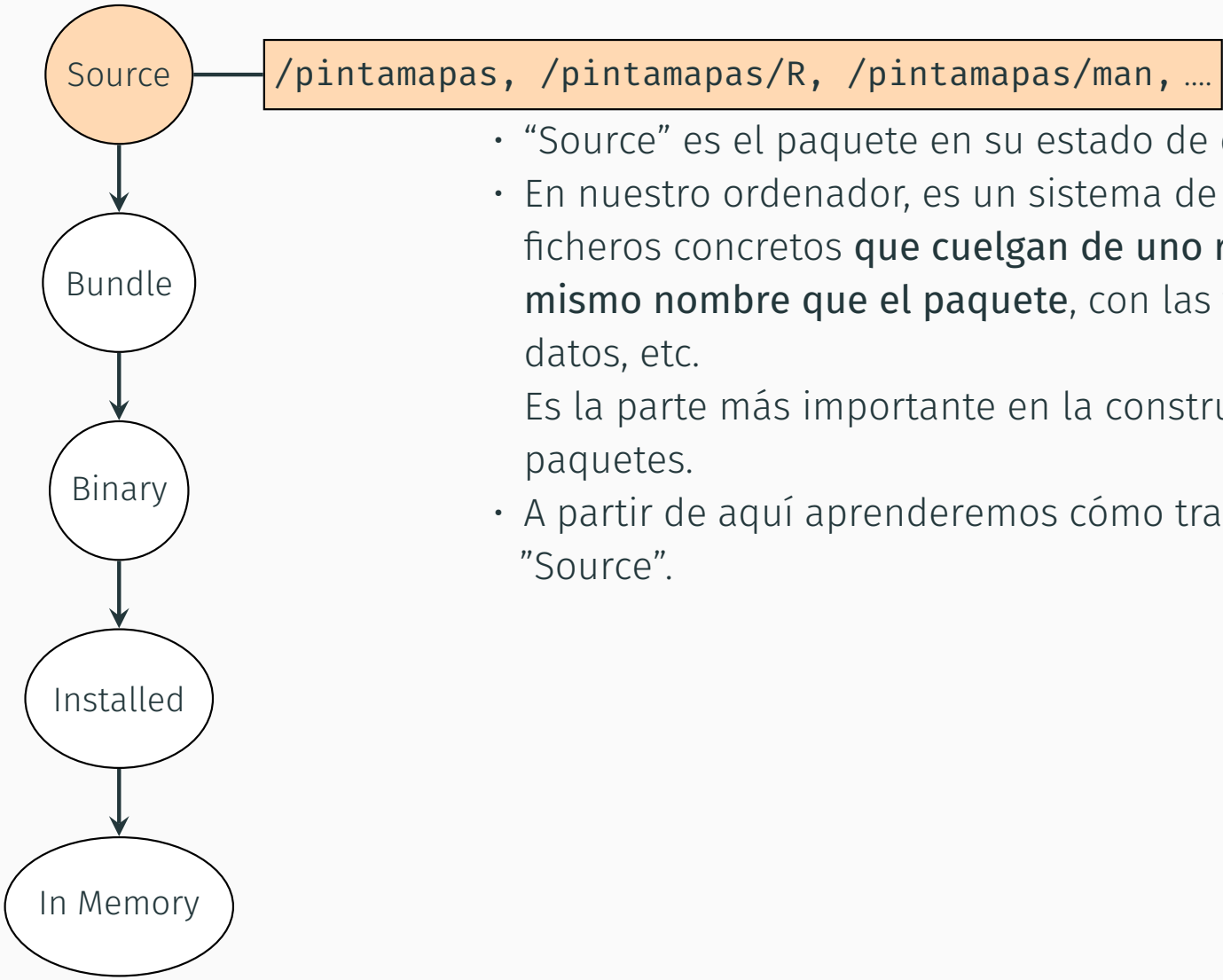
“Source”: el estado más importante para el desarrollo



`/pintamapas, /pintamapas/R, /pintamapas/man, ....`

- “Source” es el paquete en su estado de desarrollo.
- En nuestro ordenador, es un sistema de directorios y ficheros concretos **que cuelgan de uno raiz con el mismo nombre que el paquete**, con las funciones, los datos, etc.
  Es la parte más importante en la construcción de paquetes.

“Source”: el estado más importante para el desarrollo



- “Source” es el paquete en su estado de desarrollo.
- En nuestro ordenador, es un sistema de directorios y ficheros concretos **que cuelgan de uno raiz con el mismo nombre que el paquete**, con las funciones, los datos, etc.
  Es la parte más importante en la construcción de paquetes.
- A partir de aquí aprenderemos cómo trabajar el estado ”Source”.

Desarrollo de un paquete

Estructura de directorios en “Source”

Cuidado

La estructura de directorios que ha de tener un paquete en estado “Source” es muy particular y no admite variaciones. No seguirla escrupulosamente nos llevará a que no podamos construir el “Binary”.

La mínima expresión de un paquete lo componen:

- el directorio /pintamapas/R con ficheros de extensión .R con los scripts de las funciones de R.

Estructura de directorios en “Source”

Cuidado

La estructura de directorios que ha de tener un paquete en estado “Source” es muy particular y no admite variaciones. No seguirla escrupulosamente nos llevará a que no podamos construir el “Binary”.

La mínima expresión de un paquete lo componen:

- el directorio /pintamapas/R con ficheros de extensión .R con los scripts de las funciones de R.
- dos archivos ASCII en /pintamapas:
  - el archivo DESCRIPTION, con metadatos acerca del propio paquete (nombre, autores, dependencias a instalar junto al paquete...),
  - El archivo NAMESPACE, que establece qué funciones serán accesibles para el usuario y las dependencias de funciones de otros paquetes.

Primeros pasos con `devtools`

`devtools`

La estructura y mantenimiento de un paquete en desarrollo se puede hacer “a mano”, aunque el paquete `devtools` está específicamente diseñado para ayudarnos en estas tareas.

Primeros pasos con `devtools`

`devtools`

La estructura y mantenimiento de un paquete en desarrollo se puede hacer "a mano", aunque el paquete `devtools` está específicamente diseñado para ayudarnos en estas tareas.

En concreto el comando `create` construye los mínimos componentes que antes mencionabamos y que debemos usar para iniciar el paquete:

```
devtools::create("pintamapas")
```

- Ahora dedicaremos un tiempo a estos mínimos componentes creados.

El directorio R

Es un directorio en el que incluiremos ficheros de extensión .R con las funciones que queremos ofrecer en nuestro paquete.

El directorio R

Es un directorio en el que incluiremos ficheros de extensión .R con las funciones que queremos ofrecer en nuestro paquete.

- Para el mantenimiento del paquete es recomendable agrupar las funciones atendiendo a su finalidad en diferentes ficheros. P. ej.,:
  1. `fit_model.R`: función principal de ajuste.
  2. `plot_model.R`: producción de diversas figuras.
  3. `summary_model.R`: presentación de resultados selectos.

Jordi nos habló acerca de buenas prácticas en la redacción de código.

El directorio R

Es un directorio en el que incluiremos ficheros de extensión .R con las funciones que queremos ofrecer en nuestro paquete.

- Para el mantenimiento del paquete es recomendable agrupar las funciones atendiendo a su finalidad en diferentes ficheros. P. ej.,:
  1. `fit_model.R`: función principal de ajuste.
  2. `plot_model.R`: producción de diversas figuras.
  3. `summary_model.R`: presentación de resultados selectos.

Jordi nos habló acerca de buenas prácticas en la redacción de código.

- es recomendable terminar las funciones principales con la instrucción `on.exit()`.

El directorio R

Es un directorio en el que incluiremos ficheros de extensión .R con las funciones que queremos ofrecer en nuestro paquete.

- Para el mantenimiento del paquete es recomendable agrupar las funciones atendiendo a su finalidad en diferentes ficheros. P. ej.,:
  1. `fit_model.R`: función principal de ajuste.
  2. `plot_model.R`: producción de diversas figuras.
  3. `summary_model.R`: presentación de resultados selectos.

Jordi nos habló acerca de buenas prácticas en la redacción de código.

- es recomendable terminar las funciones principales con la instrucción `on.exit()`.

Cuidado

No uses JAMÁS las instrucciones `source()`, `library()` `require()` en el desarrollo de paquetes.

El archivo DESCRIPTION

Es un fichero ASCII con metadatos acerca del paquete.

El archivo DESCRIPTION

Es un fichero ASCII con metadatos acerca del paquete.Su contenido ha de editarse a mano y conviene tenerlo actualizado conforme se desarrolle el paquete.

El archivo DESCRIPTION

Es un fichero ASCII con metadatos acerca del paquete.Su contenido ha de editarse a mano y conviene tenerlo actualizado conforme se desarrolle el paquete.

Hay muchos campos disponibles, los que aparecen con `create` son:

- `Package`: pintamapas
- `Title`: explica qué hace el paquete en una única línea.
- `Version`:
- `Authors@R`: identifica a los autores del paquete.
- `Description`: explica lo que hace el paquete en un párrafo, extendiendo así el campo título.
- `Depends`:
- `License`:
- `Encoding`: identifica el tipo de codificación de caracteres. Conviene utilizar UTF-8.
- `LazyData`: ¿deben cargarse las funciones y datos al ejecutar `library(pintamapas)`, o se pueden cargar conforme se necesite? Conviene marcar la opción `true`.

Version

- El campo `Version`:
  - marca el estado de desarrollo del paquete.
  - influye en las dependencias de otros paquetes.
- En el desarrollo de software existe un sistema de codificación de versiones estándar, el cual también es empleado por R. Siguiendo este sistema, la mínima expresión de una versión válida consiste en dos o tres números enteros separados por un punto.
  - Estos números reflejan los cambios sufridos por el software siguiendo el esquema

    cambio fundamental.cambio menor.parche frente a bug
  - Por ejemplo, una versión `2.8.20` indica que el software ha sufrido un cambio fundamental (posible incompatibilidad con la versión `1.xx.xx`), dentro del cual ha habido 8 cambios menores y 20 *bugs* han sido corregidos.
  - Lo habitual es que al software que se lance al público se le otorgue la versión `1.0.0`.
- También podemos encontrarnos con un cuarto número, que indica avances en una versión en desarrollo.
  - Este número empieza en 9000. P. ej., la versión `0.9.2.9000` es la primera versión en desarrollo de la versión estable `0.9.2`.

License

- En el campo `License` debemos identificar el tipo de licencia que da cobertura legal a nuestro paquete.
- R es un proyecto de código abierto, lo que quiere decir que cualquier persona es libre de ejecutar, copiar, distribuir, estudiar, cambiar o mejorar el software. Así pues, no tiene mucho sentido emplear licencias privativas para nuestro paquete.
- Existe una gran variedad de licencias de código abierto. Lo realmente importante de este tipo de licencias es que añaden una cláusula del tipo "*haz lo que quieras, pero si algo sale mal la culpa no será mía*". Aquí solo presentamos las más relevantes (dispones de más información en este enlace):

Licencia	Permisividad	Aplicabilidad	Restricciones
MIT	Elevada	Cualquier ámbito	Ninguna
GNU GPLv3	Moderada–elevada	Especial en patentes	Sobre obra derivada

Depends

- Suele ser habitual que en nuestras funciones incorporemos llamadas a funciones de otros paquetes
  - nuestro paquete depende de otros para poder funcionar

Carlos os hablará de esto con más detalle.

- Aspecto básico: asegurar que los paquetes de los que dependemos se instalen junto al nuestro para poder funcionar correctamente.
  - Campo `Imports`: refleja qué paquetes deben instalarse junto a este.
  - Campo `Depends`: indica qué paquetes han de instalarse junto al nuestro y además los carga.
- Opción adicional: quizá nuestro paquete pueda tomar funcionalidades de otros paquetes, aunque estos no son un requisito para su funcionamiento.
  - Campo `Suggest`: indica algunos paquetes sugeridos cuya instalación es opcional (depende del usuario).

DESCRIPTION: un ejemplo

```
Package: purrr
Title: Functional Programming Tools
Version: 0.2.2.2
Authors@R: c(
    person("Lionel", "Henry", , "lionel@rstudio.com", c("aut", "cre")),
    person("Hadley", "Wickham", , "hadley@rstudio.com", "aut"),
    person("RStudio", role = "cph")
    )
Description: Make your pure functions purr with the 'purrr' package. This
    package completes R's functional programming tools with missing features
    present in other programming languages.
License: GPL-3 | file LICENSE
LazyData: true
Imports: magrittr (>= 1.5), Rcpp, lazyeval (>= 0.2.0), tibble
Suggests: testthat, covr, dplyr (>= 0.4.3)
URL: https://github.com/hadley/purrr
BugReports: https://github.com/hadley/purrr/issues
RoxygenNote: 6.0.1
NeedsCompilation: yes
Packaged: 2017-05-10 14:14:02 UTC; lionel
Author: Lionel Henry [aut, cre],
  Hadley Wickham [aut],
  RStudio [cph]
Maintainer: Lionel Henry <lionel@rstudio.com>
Repository: CRAN
Date/Publication: 2017-05-11 18:22:22 UTC
Built: R 3.4.0; x86_64-pc-linux-gnu; 2017-05-12 08:24:59 UTC; unix
```

Archivo NAMESPACE

Es un archivo algo técnico.

Mínima descripción: importa funciones de otros paquetes, importa código foráneo, define clases y exporta las funciones de nuestro paquete.

Este archivo no se edita a mano:

- puede generar problemas (olvidos, repeticiones, errores ortográficos).

Existen dos paquetes de R diseñados para facilitar, a su vez, la creación de paquetes.

- `roxygen2`
- `devtools`

Además, podemos sacar provecho de dos viejos conocidos:

- `knitr`
- `rmarkdown`

NAMESPACE: un ejemplo BayesVarSel

```
# Generated by roxygen2: do not edit by hand

export(BMAcoeff)
export(Btest)
export(Bvs)
export(GibbsBvs)
export(Jointness)
export(PBvs)
export(histBMA)
export(plotBvs)
export(predictBvs)
import(MASS)
import(mvtnorm)
import(parallel)
importFrom(grDevices,gray)
importFrom(grDevices,gray.colors)
importFrom(graphics,axis)
importFrom(graphics,barplot)
importFrom(graphics,hist)
importFrom(graphics,image)
importFrom(graphics,layout)
importFrom(graphics,par)
importFrom(graphics,plot)
importFrom(graphics,text)
importFrom(stats,as.formula)
importFrom(stats,density)
importFrom(stats,lm)
importFrom(stats,quantile)
importFrom(stats,rbinom)
importFrom(stats,runif)
importFrom(utils,read.table)
importFrom(utils,write.table)
useDynLib(BayesVarSel)
```

Muchas gracias por la atención

Referencias bibliográficas

Referencias bibliográficas (1/1)

Wickham, H. (2015a). *Advanced R*. Boca Raton, FL: CRC.
Wickham, H. (2015b). *R Packages*. Sebastopol, CA: O'Reilly.