Talleres I

Sesión: Reproducibilidad de la investigación

Profesor: Javier Lorenzo Rodríguez

Fecha: 2023-12-12

Elementos a reportar para realizar un trozo de código reproducible

```
Para que un trozo de código sea reproducible bien por mi, o por vuestro/a
tutor/a, un compañero/a de trabajo/estudio o si subís alguna duda a
plataformas de apoyo como [stack overflow](https://stackoverflow.com)
podéis seguir las siguientes recomendaciones:
```

- El objetivo último es garantizar que la información provista permita que:
  - Se pueda reproducir el trozo de texto
  - Se pueda verificar y replicar el error
  - Identifique **claramente** el problema que queréis transmitir
  - Utilizar las expresiones y la información **mínima necesaria** para entender el problema.
- Por tanto el código y la información a presentar deberá ser:
  - **mínima:** Breve descripción del problema. Se puede utilizar un ejemplo básico para ello. Esto es:
    - incorporar un vector muestra del dataframe con el que se está trabajando (no es necesario todo el data set)
    - paquete con el que se está trabajando (exclusivamente)
    - nombres descriptivos para los objetos y las funciones
    - En caso de trabajar con simulaciones no olvidéis incluir .seed(123) para garantizar la replicabilidad de los datos
  - **Dividir hasta identificar:** Si no está claro el origen del problema, intenta dividir los pasos, verificar que está correcto hasta encontrar el origen del fallo.
  - Si el problema viene de información descargada, incluir la ruta completa y el origen del fallo
- Información imprescindible para reproducir:
  - Describir el problema: Especificar el código de error, la línea del código que lo produce o el resultado no deseado y cuál sería el deseado.
  - Incorporar la información acerca de la versión de R que se está utilizando (primer output de la consola cuando abrimos R)
    - `sessionInfo()` o `rstudioapi::versionInfo()`
  - Verificar que el paquete utilizado no está deprecado para la versión que se utiliza `packageVersion("name of the package")`
  - Verificar que el comando ejecutado está asociado al paquete que queremos utilizar (el falso amigo del autocompletado de Rstudio)

Por qué es importante la reproducibilidad?

Dicho de manera sencilla, un estudio científico es reproducible si se dispone del código capaz de recrear todos los resultados a partir de los datos originales (Peng 2011).

- El trabajo científico se basa en la máxima de la aportación de conocimiento acumulativo: esto es, toda investigación, por pequeña que sea, debe aportar una información original añadida al conocimiento sobre el campo de estudio.
- Debe garantizar la veracidad de dicho avance a través del empleo del método científico (sistematización, rigor, modelos teóricos y empíricos testados)
- Debe garantizar la interpretación de los resultados: convenciones internacionales, notaciones, standares...
- La reproducibilidad es por tanto una garantía de transparencia y calidad: los artículos reproducibles están mejor blindados frente a errores, y cuando los contienen son detectados y corregidos más fácilmente [Hayden, 2015].
- Además, la reutilización de código pre-existente por parte de otros autores contribuye a acelerar el progreso científico.
- Se debe replicar para saber qué se ha hecho, cómo se ha hecho para conocer qué se puede hacer más, para mejorar, criticar, avanzar. (reverse ingeniering)
- Debe entenderse como una oportunidad de mejorar nuestra manera de hacer ciencia y la contribución de nuestros trabajos al avance científico general.
- En el campo de las ciencias sociales ha dejado de ser algo voluntario para convertirse en una obligación, el hacer público no sólo los resultados sino el proceso.
- Compartir el conocimiento es la base de la ciencia (en teoría)

A que llamamos reproducibilidad?

- El texto del artículo viene acompañado de código (texto interpretable por un ordenador) que permite recrear exactamente a partir de los datos originales todos los resultados y figuras incluidos en el artículo [Peng,2011; Marwick,2016].
- Concepto distinto a repetibilidad, que se refiere a la posibilidad de replicar el mismo estudio (con nuevos datos) a partir de la información proporcionada en el artículo.
- La reproducibilidad se relaciona principalmente con la transparencia, trazabilidad, y completitud del protocolo seguido para llegar a unos resultados concretos a partir de un conjunto de datos determinado.
- Además, dicha transparencia le da un sello de calidad al trabajo y facilita su aceptación, incrementando su impacto posterior en términos de citas y reconocimiento [Piwowar,2007; Vandewalle,2012].
- La existencia de un código ordenado y bien estructurado permitirá además su reutilización en proyectos posteriores, ahorrando tiempo y esfuerzos al equipo de investigación [@Garijo2013]. Además, compartir públicamente el código con el que generamos unos resultados puede ayudarnos a identificar errores (idealmente antes de su publicación) y abrir nuevas líneas de colaboración [@Hampton2015a].

- La reproducibilidad no es una cualidad binaria sino un gradiente que va desde trabajos totalmente irreproducibles (que sólo contienen el texto, tablas y figuras finales) a estudios perfectamente reproducibles donde la integración de texto, código y datos permite regenerar fácilmente el resultado final a partir de los datos originales [e.g. Goring 2013; Fitz John 2014].

- La publicación de los datos y/o el código empleado para el análisis contribuyen a mejorar la reproducibilidad.
- Igualmente, la existencia de un sistema de control de versiones (como git) permite reconstruir perfectamente la historia del proyecto.
- Finalmente, en el extremo del gradiente de reproducibilidad se encuentran los documentos dinámicos (por ejemplo, Rmarkdown) que integran perfectamente texto, datos y código ejecutable. In Rodríguez-Sánchez et al. 2016
- El software de código abierto y los nuevos repositorios están ayudando mucho a la replicabilidad, a la reproducibilidad y al avance del conocimiento científico compartiendo no sólo los resultados sino el proceso de investigación. Algo extendido en otras disciplinas pero relativamente reciente en las Ciencias Sociales.

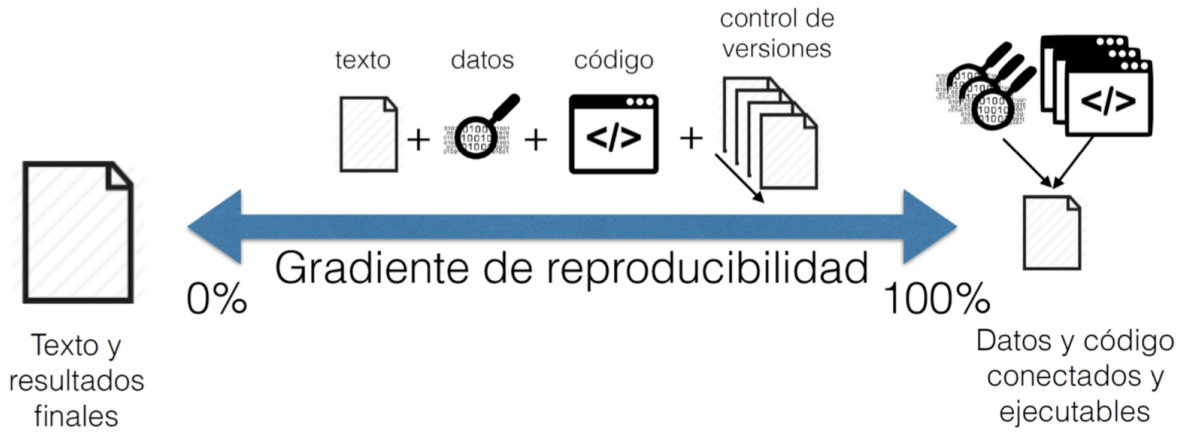

Figure 1: alt text

Breve esquema de flujo de trabajo

- En primer lugar, los datos se recogen según un protocolo bien diseñado, se documentan con metadatos, se someten a un control de calidad (mediante funciones de código), y se almacenan en un repositorio de datos en la nube.
- Después procederíamos al análisis, siempre guardando la información en códigos. El análisis propiamente dicho se puede realizar mediante documentos de Rmarkdown que integran texto, código y resultados (tablas y figuras). En Rodríguez-Sánchez et al. 2016

Flujo de trabajo en un proyecto de investigación

Una buena organización ayuda no sólo a la reproducibilidad del estudio sino a la eficiencia del propio investigador y al trabajo en equipo:

- Sistema claro y aceptado de etiquetado de carpetas, archivos etc.
  - documento con el protocolo compartido por el equipo (.txt suficiente)
- Sistematización de almacenamiento de datos
  - Repositorio público (abierto o cerrado)
  - Local (servidor de empresa, compartido por link, etc)
  - Repositorios de datos tras publicación:
    - GESIS
    - arxiv
    - harvard ### Creación de proyectos utilizando Rstudio:
- En repositorio público
- Con control de versión
- En local con acceso a link compartido
- Diferentes opciones (pantalla Rstudio)

Creación de flujo de trabajo:

```
{r} dir.create(paste0(getwd(), "/figures"), showWarnings = F) dir.create(paste0(getwd(),
"/processed-data"), showWarnings = F)   dir.create(paste0(getwd(), "/raw-data"), showWarnings
```

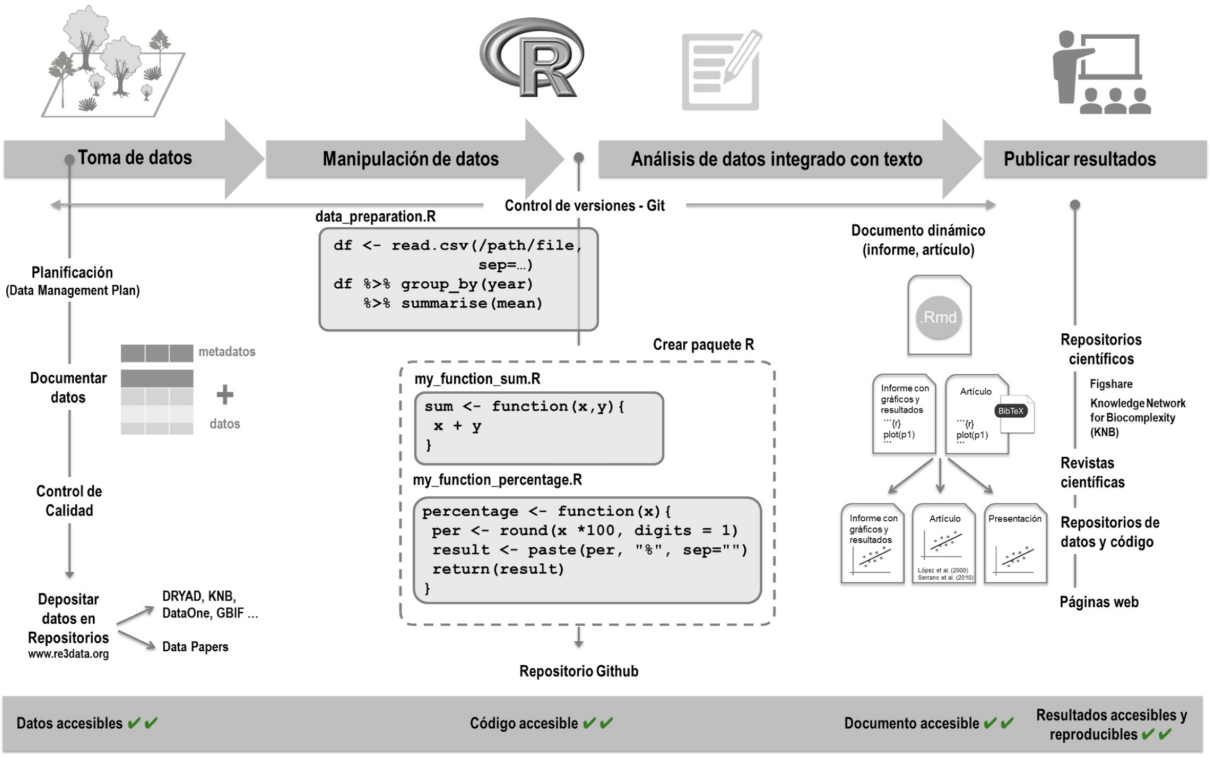

Figure 2: alt text

`= F)  dir.create(paste0(getwd(), "/scripts"), showWarnings = F)  dir.create(paste0(getwd(), "/manuscript"), showWarnings = F)` Y tantos como nos apetezca o consideremos conveniente. Creando un proyecto siempre llamaremos al directorio del proyecto y a cualquier carpeta, pudiendo llamar el script o la data utilizando simplemente `/raw-data/cand.tl`

El script: información básica, flujo de trabajo y estructura

Recordad, reproducible, intuitivo y completo. ### Formatos - Rscript - Rmkd - md - *Consejos:* - emplead un script para cada dataset que vayáis a utilizar - Incluir toda la información - Podéis trabajar con un script individual para cada sección del proyecto y llamadlo como tal - Incluir en el script final **SOLO** lo que merece la pena para la reproducibilidad del análisis - Convertir ese script en bonito o en un markdown. (a gusto del consumidor o dependiendo del producto final del estudio)

Información general

- Cabecera del script debe contener:
  - Nombre del proyecto
  - Nombre del autor del script
  - Fecha de la última revisión del script
  - Sumario de las tareas que nos vamos a encontrar
  - Opcional como cabecera: dónde se aloja la información básica
  - Opcional como cabecera: librerías a instalar (si no están en CRAN, incluid la URL)
- Cómo hacer la cabecera:

  - paquete bannercommenter
  - Añadir en addins little boxes
  - `install.packages("devtools") devtools::install_github("ThinkRstat/littleboxes")`
  - Utilizar los snippets de Rstudio o de cualquier editor de texto (pantalla Rstudio)
    * Abrir Rstudio
    * Herramientas – Opciones globales – Codigo – TabEditing–Snippets–Habilitar edicion Snippets

Cargar las librerías

- Para que se entienda lo que vamos a hacer y se pueda replicar exactamente:
  - Conviene ponerlas todas al comienzo del script, salvo que tengan sentido más tarde. En cualquier caso especificar.
  - Incluso se pueden comentar brevemente el para qué se utilizan (esto sólo en las más específicas o si se ha desarrollado un paquete propio)
- Formas de hacerlo:
  - Clásica: `library(tidyr)` se puede añadir # Limpieza de datos
  - De forma más segura: `{r} packages = c("dplyr","evaluate","ggplot2","ggraph","ggrepel", "haven","igraph","knitr", "lubridate","psych","quanteda","readr","reshape", "rlang","rmarkdown","scales", "stopwords","tidyselect","tidytext","tidyverse","tm", "utf8","zoo") ## Now load or install&load all package.check <- lapply(packages, FUN = function(x) { if (!require(x, character.only = TRUE)) { install.packages(x, dependencies = TRUE) library(x, character.only = TRUE) } })`
  - Esto nos permite garantizar que todos los paquetes necesarios se cargan en la sesión y, en caso de no estar instalados los instala y carga.
  - Las funciones aparecen por defecto en la estructura del script. También conviene: cargarlas al inicio del script con una sección específica que lo detalle o b) tenerlas en scripts separados y llamarlas si son necesarias.
- Otra forma es añadir los paquetes y funciones más utilizadas en el snippet de la cabecera, así se ejecutan automáticamente una vez que abrimos un script (solo en caso de aquellas más utilizadas)
- **NO ACONSEJABLE:**"Cargar todos los paquetes como si no hubiera mañana"
  - Sobrecarga la memoria de Rstudio y a veces puede bloquearlo
  - Sobre todo si los cargamos desde local (poco recomendable)
  - El autocompletado y muchas librerías que hacen lo mismo pueden llevar a error.
  - Si queremos garantizar que estamos utilizando el paquete que queremos: `nombredelibreria::funcion`

Cuerpo del script

- Alojamiento de los datos
  - si trabajas con stwd()
    * Es una ruta local que tienes que cambiar cada vez que abres tu script
    * No lo hace en absoluto replicable porque yo no tengo acceso a tu máquina local
    * Si vuelves dentro de un mes y has cambiado el directorio estás perdido. Tienes que volver a rescribir todas las dependencias
    * Si tienes archivos en distintos subdirectorios tendrás que especificarlo cada vez
    * **TIP** No muy aconsejable utilizar `attach()` en cualquier caso
  - **NO GUARDAR ESPACIO DE TRABAJO** (Podéis deshabilitar esta función en global options para no sentiros tentados de hacerlo) #### Manipulaciones posteriores de limpieza y debugging #### Secciones del script: (En Rstudio)

- `## cualquier texto que termine con #### o ---- o ====`
- El truco es que la línea tiene que terminar con cuatro caracteres de los identificados
- Visualización de las secciones en el script (pantalla Rstudio)
- Esto nos permitirá navegar sencillamente entre el script, editarlo posteriormente en un md con muy poco recoding o convertirlo en un latex si nos apetece.
- Utilizad la opción "fold/unfold" para colapsar trozos de código por secciones cuando estéis trabajando en otra sección para evitar hacer scrolling

Comentarios

- Establecer una dinámica de comentarios:
  - Al final de la línea para explicar qué hace dicha línea si es interesante (#)
  - Una explicación de lo que hace el siguiente tramo de código (##)
  - Una reflexión general sobre el resultado producido (###)
  - El inicio de una nueva sección (####)

Tabulaciones

- Seguir un esquema estricto de tabulaciones o espacios
- Como normal general, cada paso, tras la coma o pipe
- Ser consistentes. Esto permite ejecutar automáticamente cada trozo de código anclado. Identificar errores por línea.
- Más fácil de leer para su trazabilidad y corrección.
- Permite añadir comentarios a cada línea en caso de ser necesario

Nombrar objetos

- cand_tl #correcto
- cand.tl # aceptable pero puede dar lugar a confusiones
- cand_TL # mal, combinar mayúsculas y minúsculas nunca una buena idea
- cand tl # no va a funcionar
- cand-tl # mal, os lo va a separar y si queréis copiar y pegar váis a tener que hacerlo dos veces
- df2 # poco identificable. Puede estar bien si queremos compartir el error, pero para nuestro propósito dentro de una semana no sabemos qué significa df2

Espaciado

- Incluir un espacio antes y después de (=, +, -, <-, etc.), excepto:
  - En paréntesis en las funciones
  - Cuando llaméis un paquete `devtools::whatever`
  - Añadir un espacio después de llaves: `if (y < 0 && debug) { message("Y is negative") }`

Editar tu script al final `{r} # Formatear tu viejo código añadiendo espacios y longitud máxima por línea install.packages("formatR") library("formatR") # Establece tu wd tidy_source("messy_s file = "tidy_script_2020-05-20.R", width.cutoff = 100) # Si no se especifica el archivo como nuevo, Rstudio lo sobreescribira. No olvides añadir width cutoff point. (100 es razonable) # Reformatea todos los escritorios de tu directorio # Establece tu wd donde corresponda # IMPORTANT: Esto sobreescribe los scripts, así que conviene tener un back up o hacerlo en un archivo nuevo tidy_dir(path="whatever/your/path/is", recursive = TRUE)`

`# recursive - si quieres que mire los scripts en distintos subdirectorios` ## Paquetes y Recursos a utilizar para la reproducción y elaboración de un proyecto de investigación que combine datos y análisis Aquí tenéis contenidos y paquetes que pueden facilitar la reproducibilidad de vuestras investigaciones: cran.r-project Los contenidos son sencillos: - Contenidos para transportar a editor de textos como Latex, Markdown reproducibles en HTML o PDF o incluso WORD (pandocs) - Para vincular vuestros proyectos en R con Git y GitHub + happygitwithr + rafalab.github como guía completa para reproducir investigación y resultados en Rstudio con GitHub y Markdown + education.github todos los repositorios gratuitos o de pago en los que poder albergar nuestrs datos, proyectos, código, etc. - Articulo recomendado para la reproducibilidad: Rodriguez-Sanchez - tutorial de tidy

Criterios/ requisitos de evaluación del script para talleres II

- **8** Que sea replicable y obtenga los resultados que presentáis en vuestro TFM o manuscirto (hasta donde llegue)
- **9** Que sea claro y se entienda cada paso dado
- **10** Que explique la decisión tomada y motive por qué (esto puede estar contenido en el script o en el manuscrito)
- **7** Que se replique pero tenga que añadir o modificar algo para poder ejecutarlo (una librería que falta, una manipulación no introducida, etc)
- **6** Que se replique pero tenga que incluir / modificar / intuir una buena cantidad de líneas en el código
- **5** que se replique pero no se van algunos resultados obtenidos por algún motivo
- **4** que no se pueda replicar

Materiales a incluir:

- Carpeta en Google Drive: con el raw data
- todos los archivos iniciales necesarios para ejecutar el script
- Rproject con repositorio público o en servidor
- Manuscrito o borrador en el que se visualicen los resultados

Preguntas y dudas generales