

Manejo y visualización de datos en R (Parte IV - Trabajo reproducible)

AUTHORS

Julen Astigarraga

AFFILIATIONS

Universidad de Alcalá, Forest Ecology and Restoration Group

PUBLISHED

Jan. 4, 2022

Contents

Introducción

Qué es R Markdown

 Fundamentos básicos de R Markdown

 Ejercicio 1

Qué es Git

Qué es GitHub

 Instalación de Git

 Ejercicio 2

 Repositorios y proyectos

 Ejercicio 3

 Flujo de trabajo en Git y GitHub

 ¿Cómo moverse de una zona a otra?

 Ejercicio 4

Algunos enlaces interesantes

Gran parte de la información de esta sesión ha sido obtenida de la nota ecoinformática: Astigarraga, J. & Cruz-Alonso, V. (en prensa). "¡Se puede entender cómo funcionan Git y GitHub!" Ecosistemas. Para más información ver: https://github.com/Julenasti/intro_git-github

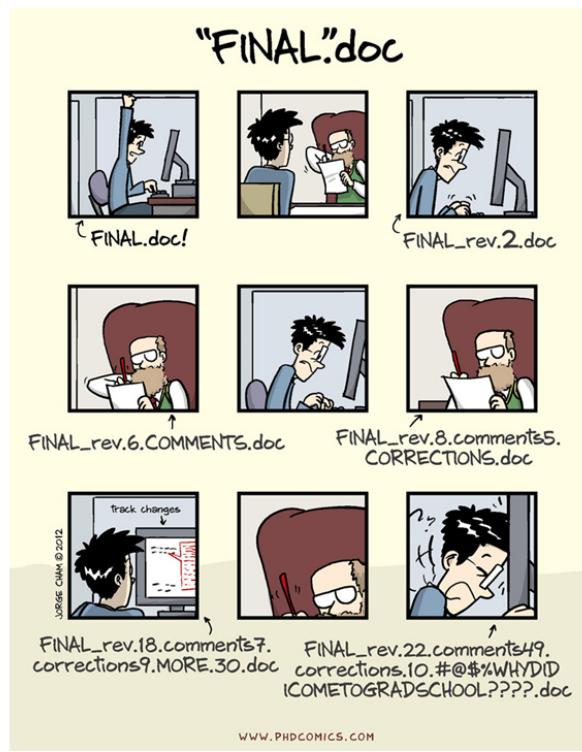
Introducción

La reproducibilidad, entendida como la capacidad que tienen algunos trabajos o proyectos de ser recreados de forma independiente a partir de los mismos datos y el mismo código que utilizó el equipo

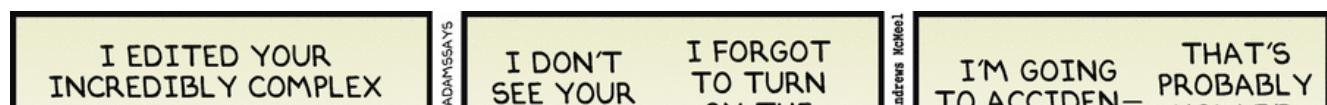
original ([The Turing Way Community 2021](#)), es una condición *sine qua non* del método científico ([Peng 2011](#)). Cada vez manejamos una mayor cantidad de datos lo que hace indispensable utilizar herramientas informáticas que garanticen la trazabilidad de todo el proceso de desarrollo de proyectos (desde su creación hasta su publicación), asegurando de esta manera su reproducibilidad ([Rodríguez-Sánchez et al. 2016](#)). En este sentido, es recomendable utilizar algún formato de archivo que permita integrar código y texto como R Markdown (<https://rmarkdown.rstudio.com>), combinada con alguna herramienta de control de versiones como Git (<https://git-scm.com/>) junto con plataformas en línea para albergar los proyectos como GitHub (<https://github.com/>), facilitando así el seguimiento de los proyectos y la coordinación entre colaboradores ([Blischak, Davenport, and Wilson 2016](#); [Galeano 2018](#); [Rodríguez-Sánchez 2020](#)). Aunque existen multitud de manuales disponibles gratuitamente sobre cómo utilizar Rmarkdown, Git y GitHub, estas herramientas son complejas y tienen una curva de aprendizaje pronunciada. El objetivo de la presente sesión es dar a conocer la estructura, funcionalidad y potencialidad de Git, así como su interacción con GitHub y Rmarkdown, para el trabajo en proyectos colaborativos.

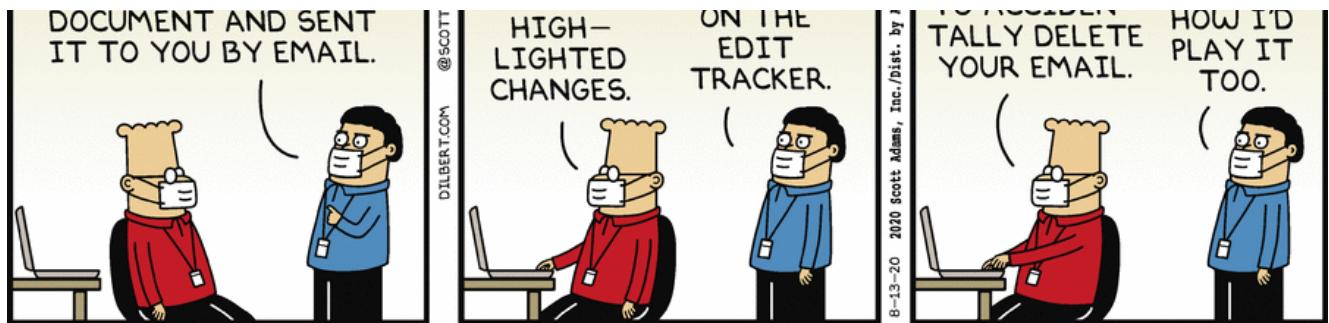
Por ejemplo, Git y GitHub nos pueden ayudar a solucionar algunos problemas comunes derivados de la creación de diferentes versiones que pueden ser un poco molestos:

- Sobreescritura de un archivo
- Versiones finales infinitas



- Trabajo por error en una versión que no era la final
- Creación de copias “en conflicto” cuando dos personas trabajan a la vez
- Ediciones sin control de cambios





Qué es R Markdown

R Markdown es un formato de archivo para crear documentos dinámicos con R. Un documento R Markdown está escrito en markdown (un formato de texto plano) y contiene partes de código de R (o algún otro lenguaje de programación) integrado. Fue diseñado para facilitar la reproducibilidad, ya que tanto el código de cálculo como el texto están en el mismo documento, y los resultados se generan automáticamente a partir del código fuente (<https://bookdown.org/yihui/rmarkdown/>). Como sus autores indican, R Markdown se apoya en knitr (<https://yihui.org/knitr/>) y Pandoc (<https://pandoc.org/>). knitr ejecuta el código informático integrado en Markdown y convierte R Markdown en Markdown. Pandoc convierte Markdown en el formato de salida que quieras (como PDF, HTML, Word, etc.). Además de que el trabajo sea reproducible hay múltiples razones para trabajar en R Markdown así como el aumento de tu eficiencia de trabajo (a medio-largo plazo) y la producción de documentos de alta calidad.

Fundamentos básicos de R Markdown

Para crear un archivo R Markdown: File -> New File -> R Markdown. Estos archivos tienen 3 componentes principales: (i) metadatos, (ii) texto, (iii) código.

- Metadatos: se escribe entre `---` (al comienzo del archivo). Utiliza la sintaxis de YAML. La sangría es importante!
- Texto: sintaxis Markdown
- 2 tipos de código:
 - Code chunk (trozo de código): se escribe entre ````{r}```; r indica el lenguaje (atajo: **Ctrl + Alt + I**)
 - Inline (en línea): se escribe entre `r`

Para compilar un archivo R Markdown se utiliza el botón de "Knit" de RStudio (atajo: **Ctrl + Shift + K**). También se puede hacer mediante código: `rmarkdown::render("intro_repro.Rmd," output_dir = "output")`

Para buscar ayuda: Help -> Markdown Quick Reference (italics, bold, headers, lists, links, images, R codes, tables, page break, superscripts...)

negrita, cursiva, subíndice₁, superíndice², codes, hypervínculo, notas al pie ¹

Títulos (# primer nivel; ## segundo nivel...), listas y sublistas (*, -, +),

"R Markdown es maravilloso"

— Julen Astigarraga

$$f(how) = \binom{do}{you} feel^{excellent} (1-p)^{n-k}$$

⚡ Cambia al **Visual markdown editing mode** mediante el botón del compás.

Para analizar las múltiples opciones de chunk (Ctrl + Alt + I):

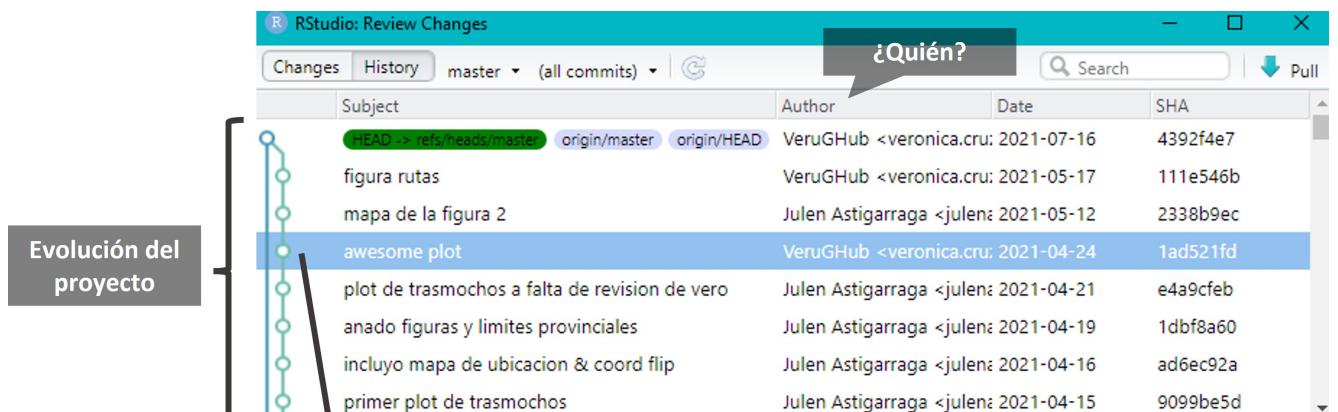
<https://bookdown.org/yihui/rmarkdown/r-code.html>

Ejercicio 1

Crea un archivo R Markdown, ponle un título, añade un texto, un plot (Ctrl + Alt + I) y compila al formato HTML ("Knit").

Qué es Git

Git es un sistema avanzado de control de versiones (como el "control de cambios" de Microsoft Word) distribuido ([Blischak, Davenport, and Wilson 2016](#); [Ram 2013](#)). Git permite "rastrear" el progreso de un proyecto a lo largo del tiempo ya que hace "capturas" del mismo a medida que evoluciona y los cambios se van registrando. Esto permite ver qué cambios se hicieron, quién los hizo y por qué, e incluso volver a versiones anteriores. Además, Git facilita el trabajo en paralelo de varios participantes. Mientras que en otros sistemas de control de versiones (p. ej. Subversion (SVN, <https://subversion.apache.org/>) o Concurrent Versions System (CVS, <http://cvs.nongnu.org/>) hay un servidor central y cualquier cambio hecho por un usuario se sincroniza con este servidor y de ahí con el resto de usuarios, Git es un control de versiones distribuido que permite a todos los usuarios trabajar en el proyecto paralelamente e ir haciendo "capturas" del trabajo de cada uno para luego unirlos. Otras alternativas de control de versiones distribuido comparables a Git son Mercurial (<https://www.mercurial-scm.org/>) o Bazaar (<https://bazaar.canonical.com/>), pero Git es con diferencia el más utilizado.



The screenshot shows a GitHub commit history for a repository named '3R/3.2021.3-trasmochos.R'. The commit details are as follows:

- SHA:** 1ad521fd0c849003cd77edeb0cb2e811d501ebb2
- Author:** VeruGHub
- Date:** 2021-04-24 17:54
- Subject:** awesome plot
- Parent:** e4a9cfeb41f8e661e4c2630b6edd8ad331d826d8

Annotations with arrows point to specific fields:

- An arrow points from the question '¿Dónde?' to the 'Author' field.
- An arrow points from the question '¿Cuándo?' to the 'Date' field.
- An arrow points from the question '¿Qué?' to the 'Subject' field.
- An arrow points from the question '¿Dónde?' to the file name '3R/3.2021.3-trasmochos.R'.

The commit message content is:

```
@@ -137,8 +137,8 @@ table(tree_plot_sp$sppcompa234)
137 137
138 138 # tambien agrupo los quercus
139 139 plot_sp <- tree_plot_sp %>%
140 140 filter(!c(sppcompa234 == "055" |
141 141 sppcompa234 == "255")) %>%
140 140 filter(sppcompa234 != "055" &
141 141 sppcompa234 != "255") %>%
142 142 mutate(sppcompa234 = case_when(
143 143 sppcompa234 == "071" ~ "Fagus sylvatica",
144 144 sppcompa234 == "072" ~ "Castanea sativa",
```

Ejemplo de un proyecto rastreado por Git con indicaciones de cómo se registran los cambios y la evolución del proyecto, el autor o autora de los cambios (¿quién?), el momento en que se han registrado (¿cuándo?), en qué documentos o líneas se han producido cambios (¿dónde?) y qué ha cambiado (¿qué?)

Qué es GitHub

GitHub es un servidor de alojamiento en línea o repositorio remoto para albergar proyectos basados en Git que permite la colaboración entre diferentes usuarios o con uno mismo (Galeano 2018; Perez-Riverol et al. 2016). Un repositorio es un directorio donde desarrollar un proyecto que contiene todos los archivos necesarios para el mismo. Aunque existen distintos repositorios remotos (p. ej. GitLab, <https://gitlab.com/>, o Bitbucket, <https://bitbucket.org/>) con funcionalidad similar, GitHub es hoy en día el más utilizado. GitHub registra el desarrollo de los proyectos de manera remota, permite compartir proyectos entre distintos usuarios y proporciona la seguridad de la nube entre otras funciones. Cuando se trabaja en proyectos colaborativos, la base de la interacción entre Git y GitHub es que todos los colaboradores de un proyecto están de acuerdo en que GitHub contiene la copia principal del proyecto, es decir, GitHub contiene la copia centralizada del control de versiones distribuido o descentralizado.

The screenshot shows a GitHub user profile for 'Julenasti'. The main sections include:

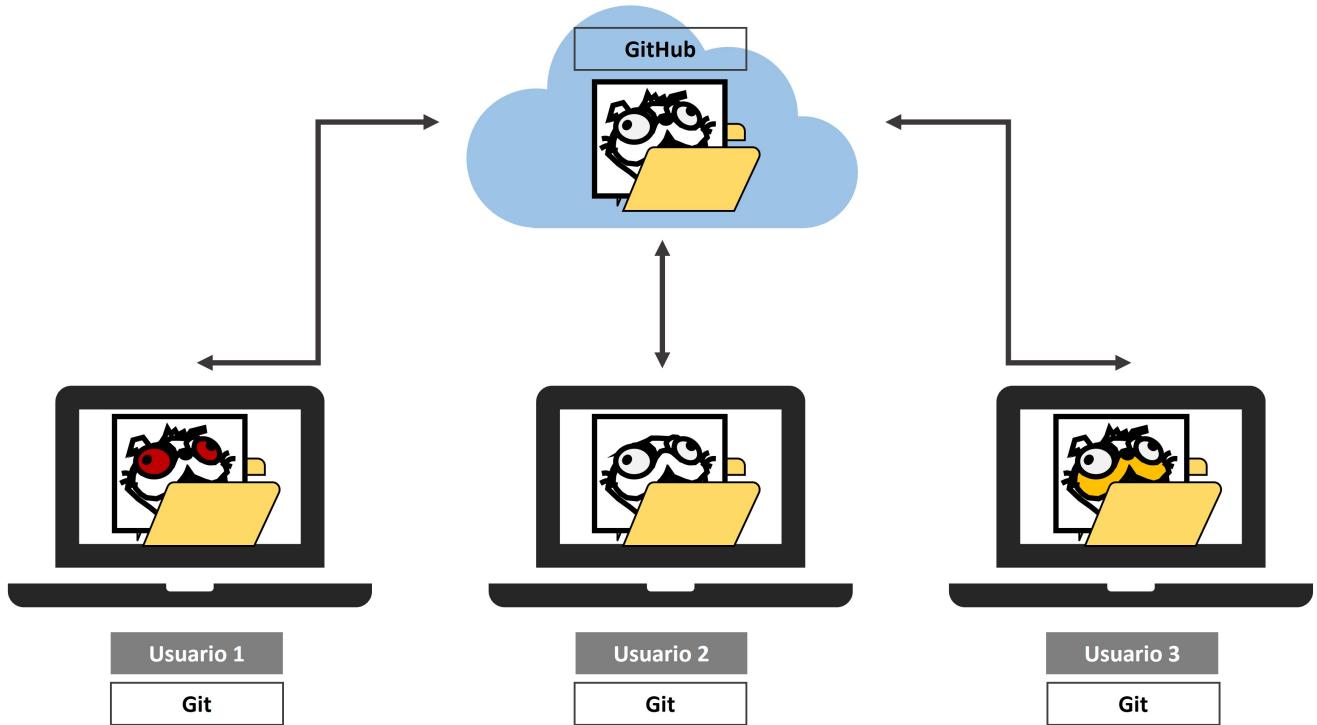
- Repositories:** Shows a list of repositories like 'species-ranges', 'intro-git-github', 'julenasti', 'beech-man', 'intro_tidyverse-dplyr', 'IDAR-IFN', and 'climate-variables'.
- Recent activity:** Shows recent actions taken by the user or others, such as pushes to 'Julenasti/julenasti' and deployments to GitHub pages.
- Explore repositories:** Lists other repositories the user has interacted with, including 'TheoreticalEcology/ecodata', 'luukvdmeer/sfnetworks', and 'datalorax/equatiomatic'.
- User stats:** Shows the user is signed in as 'Julenasti' with a profile picture, and provides links to 'Your profile', 'Your repositories', 'Your codespaces', 'Your projects', 'Your stars', 'Your gists', 'Feature preview', 'Help', 'Settings', and 'Sign out'.

A screenshot of the GitHub homepage. At the top, there's a banner for 'fordsbif Deploy to GitHub pages'. Below it, a notification from 'github-actions' bot pushing to 'Julenasti/Julenasti' repository yesterday. A link to 'output' is shown. The main content area shows a profile picture of a person wearing a mask, followed by the user's name 'Julen Astigarraga' and bio.

Página inicial de GitHub

A screenshot of a GitHub user profile page for 'Julen Astigarraga'. The profile includes a circular photo of a person wearing a mask, the user's name 'Julen Astigarraga', and their bio: 'PhD student at the University of Alcalá'. It also shows 15 followers, 16 following, and a link to their LinkedIn profile. The GitHub interface features a dark theme with various sections like Overview, Repositories, Projects, and Packages.

Perfil de GitHub



Interacción entre Git y GitHub. Git, al ser un control de versiones distribuido, permite que todos los usuarios trabajen paralelamente sin interferir en el trabajo de los demás. Luego cada usuario sincroniza su trabajo con la copia principal del proyecto ubicado en GitHub

Instalación de Git

En este punto es necesario que tengas instalada la versión más reciente de R (<https://cloud.r-project.org/>), RStudio (<https://www.rstudio.com/products/rstudio/download/>), Git (<https://happygitwithr.com/install-git.html>) y una cuenta en GitHub (<https://github.com/>) creada.

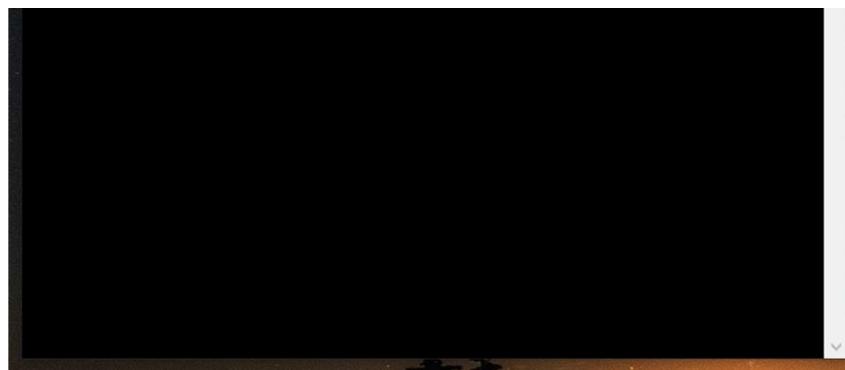
Ejercicio 2

En el *shell*, preséntate a Git ([Chapter 7: Git-Intro](#))

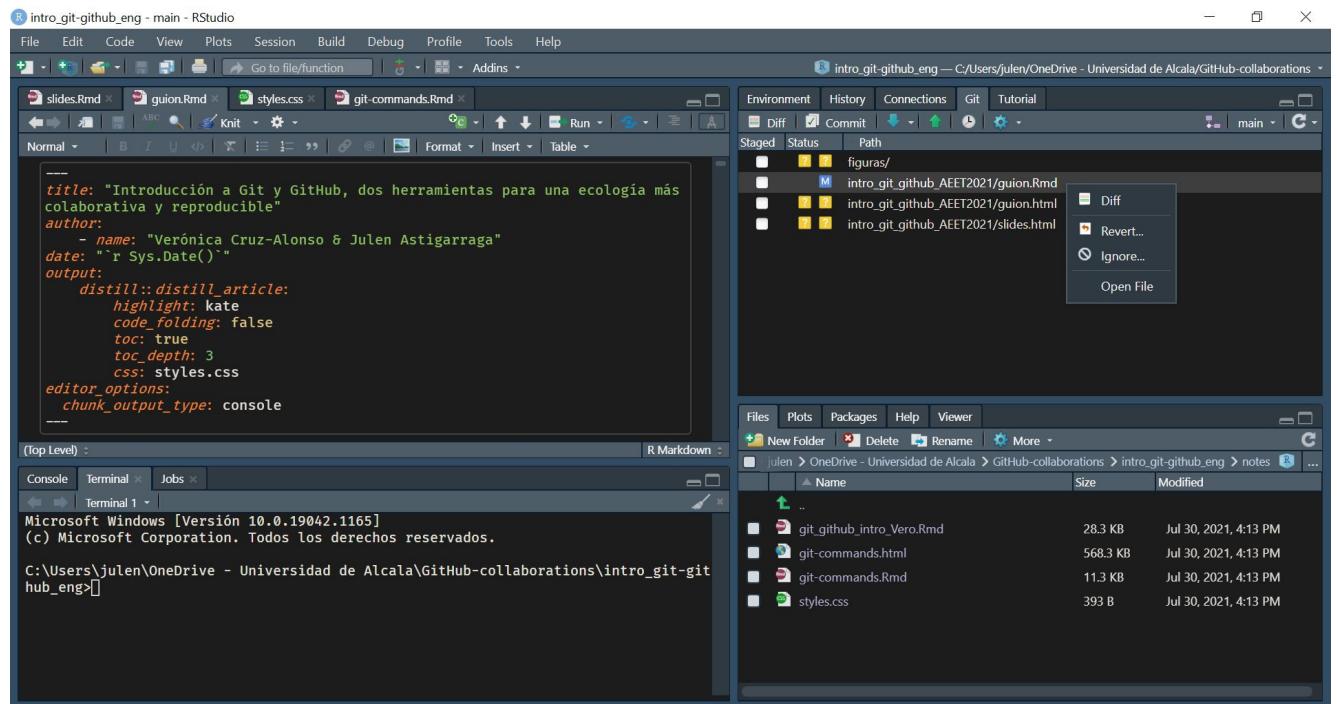
⚡ ¿Qué es el *shell*? El *shell* (o terminal) es un programa en tu ordenador cuyo trabajo es ejecutar otros programas (ver <https://happygitwithr.com/shell.html#shell>). En esta sesión aprenderemos a trabajar principalmente desde la línea de comandos del *shell* aunque también veremos cómo hacerlo a través de un cliente como RStudio (<https://www.rstudio.com/>), que incorpora una pestaña llamada "Git" que facilita la transición entre zonas de trabajo ya que contiene funcionalidades básicas de Git. Usar un cliente como RStudio es recomendable para usuarios noveles de Git (ver <https://happygitwithr.com/rstudio-git-github.html>).



```
MINGW64:/c/Users/julen
julen@MSI MINGW64 ~
$ |
```



Terminal de Git



A través de RStudio

Tools -> Shell

```
git config --global user.name 'Nombre Apellido'
```

```
git config --global user.email 'nombre@ejemplo.com'
```

Comprueba que has instalado Git correctamente:

```
git --version
```

Para ver el usuario utilizado para configurar Git:

```
git config user.name
```

Para ver a qué cuenta de correo está asociado Git:

```
git config user.email
```

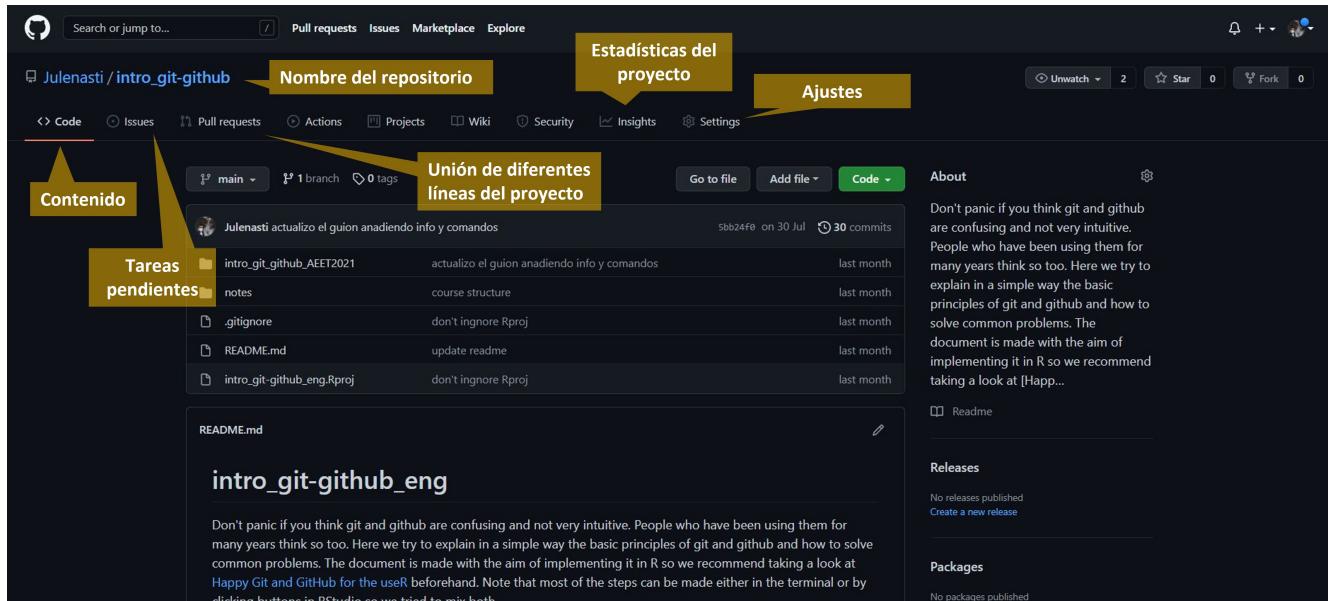
Para ver tanto el usuario como el correo asociado:

```
git config --global --list
```

Repositorios y proyectos

Un repositorio es como un “contenedor” donde desarrollar un proyecto.

Para crear un repositorio en GitHub damos a “+ New repository.” Aquí se indica el nombre, una pequeña descripción, y si quieres que sea público o privado. Se recomienda iniciar el repositorio con un archivo “README” (*Initialize this repository with a README*) para recoger cualquier información esencial para el uso del repositorio (estructura, descripción más detallada del contenido, etc.).



Repositorio en GitHub destacando algunas pestañas importantes

En R, creamos un nuevo proyecto y lo conectamos al repositorio: File -> New project -> Version control -> Git -> copiar el URL del repositorio que hemos creado de GitHub (está en la página principal de nuestro repositorio, en “clone or download”). Seleccionamos el directorio donde queremos guardar el proyecto y pulsamos en “Create project.”

Si vamos al directorio seleccionado, encontraremos la carpeta conectada a Git y GitHub que hemos creado en nuestro ordenador. Podemos copiar aquí todos los archivos que nos interesan para el proyecto (datos, imágenes, etc).

Para más información sobre cómo clonar el repositorio en GitHub (repositorio remoto) en nuestro ordenador (repositorio local) ver <https://happvaitwithr.com/rstudio-ait-aitgithub.html> para hacerlo desde file:///C:/Users/julen/OneDrive/Escritorio/GitHub-col/intro_git-github/doctordado-uah/intro_repro.html

RStudio; y ver [Galeano \(2018\)](#) para hacerlo mediante la línea de comandos.

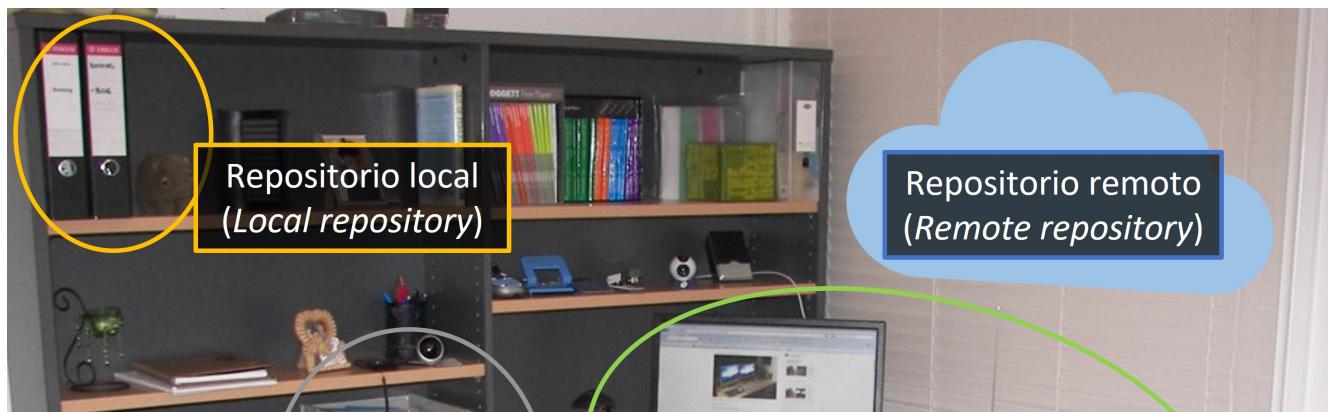
Ejercicio 3

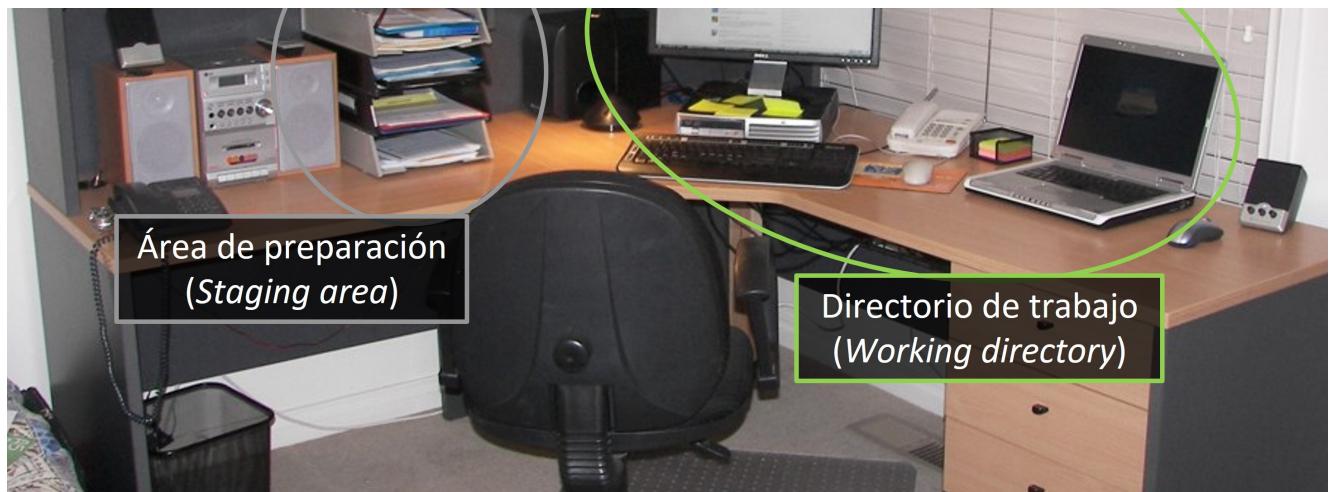
1. Crea un repositorio en GitHub y conecta a un nuevo proyecto de RStudio (esto generará un repositorio (carpeta) en tu ordenador en la ubicación que le hayas especificado)
2. Copia el archivo de R Markdown que has creado en el Ejercicio 1 en el directorio de trabajo (es decir, copia el archivo R Markdown y guárdalo dentro del repositorio que has creado)
3. En RStudio ve a la pestaña Git (donde está el *environment*) para ver todos los documentos que han sido identificados por Git

Flujo de trabajo en Git y GitHub

Git es capaz de rastrear todos los archivos contenidos en un repositorio. Para comprender cómo Git registra los cambios y cómo podemos compartir dichos cambios con nuestros colaboradores es importante entender cómo se estructura Git y cómo se sincroniza con GitHub. Hay cuatro "zonas" de trabajo:

1. **Directorio de trabajo (*working directory*):** es donde se está trabajando. Esta zona se sincroniza con los archivos locales del ordenador.
2. **Área de preparación (*staging area o Index*):** es la zona intermedia entre el directorio de trabajo y el repositorio local de Git. Es la zona de borradores. El usuario debe seleccionar los archivos que se van a registrar en la siguiente "captura" de Git.
3. **Repositorio local (*local repository o HEAD*):** es donde se registran todos los cambios capturados por Git en tu ordenador.
4. **Repositorio remoto (*remote repository*):** es donde se registran todos los cambios capturados por Git en la nube (GitHub).





Graphical representation of the different working areas in Git and GitHub: working directory, staging area or Index, local repository or HEAD, and remote repository. Background image from Philip Brookes (<https://creativecommons.org/licenses/by-nc-nd/2.0/legalcode>)

¿Cómo moverse de una zona a otra?

Al principio todos los cambios realizados están en amarillo porque Git no sabe que hacer con ellos. Estamos en el directorio de trabajo y puede que no nos interese guardar todos los cambios para el futuro.

Para añadir un cambio del directorio de trabajo al área de preparación hay que utilizar `git add`. Este comando indica a Git que se quieren incluir las actualizaciones de algún archivo en la próxima "captura" del proyecto y que Git las registre. Sin embargo, `git add` no afecta al repositorio local.

- `git add <nombre de archivo>`: añade una actualización de algún archivo del directorio de trabajo al área de preparación.

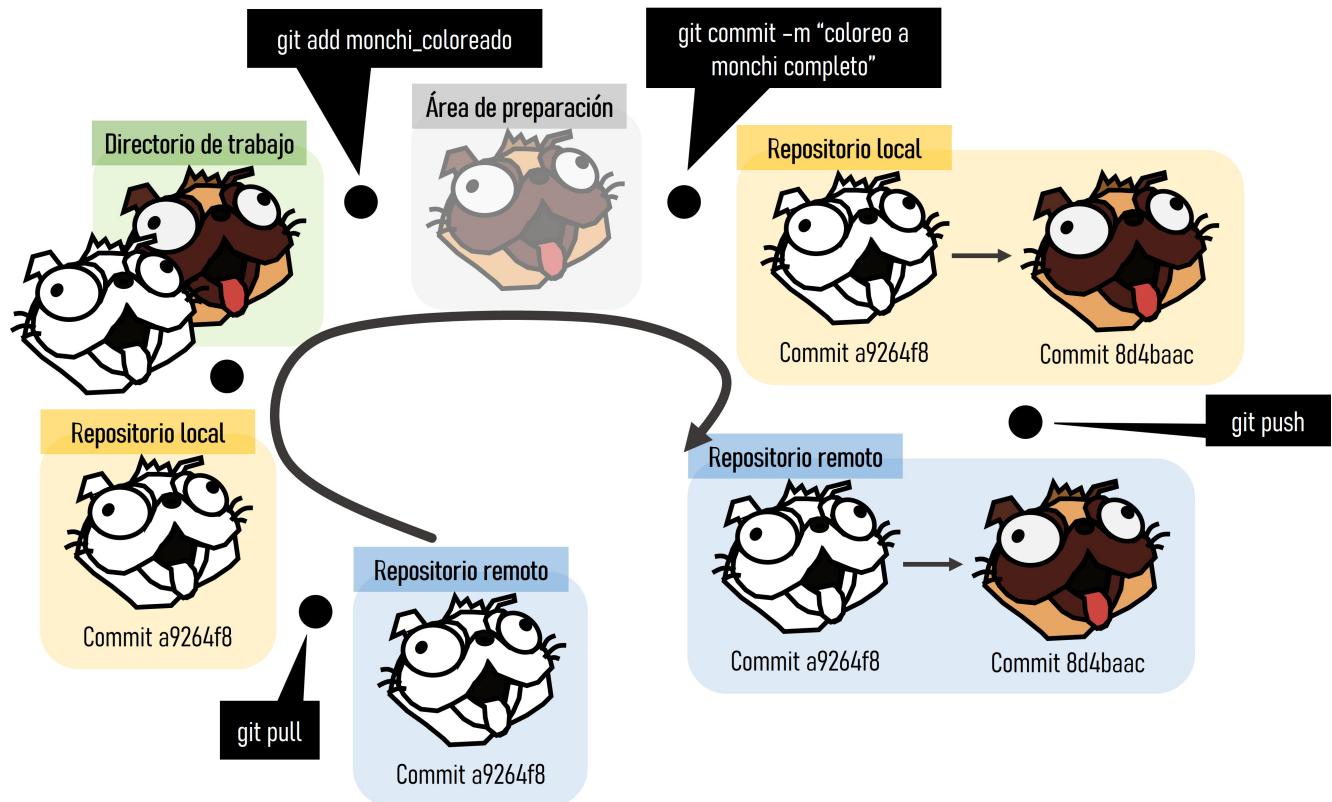
Para ver el estado del directorio de trabajo y del área de preparación se utiliza `git status`. Este comando permite ver qué archivos están siendo rastreados por Git, qué cambios han sido añadidos al área de preparación (*staged*) y qué archivos están siendo registrados por Git.

Para registrar los cambios que nos interesen hay que utilizar `git commit`. Al ejecutar `git commit` se hace una "captura" del estado del proyecto. Junto con el *commit* se añade un mensaje con una pequeña explicación de los cambios realizados y por qué (p. ej. "incluyo las referencias en el formato de Ecosistemas"). Cada `git commit` tiene un SHA (Secure Hash Algorithm) que es un código alfanumérico que identifica inequívocamente ese *commit* (p. ej. 1d21fc3c33cxxc4aeb7823400b9c7c6bc2802be1). Parece difícil de entender, pero no te preocupes, sólo tienes que recordar los siete primeros dígitos 1d21fc3 (es broma). Con el SHA siempre se pueden ver los cambios que se hicieron en ese *commit* y volver a esa versión fácilmente.

- `git commit -m "mensaje corto y descriptivo"`

Por último, `git push` permite subir los cambios que hemos hecho a GitHub y quedarán visibles para nuestros colaboradores. Básicamente, `git commit` registra los cambios en el repositorio local y `git push` actualiza el repositorio remoto con los cambios y archivos asociados.

Cuando se retoma un proyecto tras horas, días o incluso meses, con `git pull` se descargan todas las actualizaciones que haya en GitHub (nuestras o de nuestros colaboradores), que se fusionarán (*merge*) con el último *commit* en nuestro repositorio local.



Flujo de trabajo en Git y GitHub mostrando las diferentes zonas de trabajo y los comandos utilizados para la transición de una zona de trabajo a otra

Ejercicio 4

- En el proyecto de cada equipo, guardad y subid a GitHub los cambios realizados en el Ejercicio 3 (`git add + git commit + git push`)

Captura de pantalla de la interfaz de GitHub:

- Rama actual:** main
- Contenido del repositorio con enlaces a cada archivo:** README.md, .gitignore, intro_git_github_eng.Rproj, notes, README.md, intro_git_github_AEET2021
- SHA y fecha del último commit:** 5bb24f0 on 30 Jul 30 commits
- Mensaje del último commit que modificó:** Julenasti actualizo el guion anadiendo info y comandos
- Botón:** Clonar o descargar el repositorio

intro_git-github_eng

ese archivo con enlace

Don't panic if you think git and github are confusing and not very intuitive. People who have been using them for many years think so too. Here we try to explain in a simple way the basic principles of git and github and how to solve common problems. The document is made with the aim of implementing it in R so we recommend taking a look at Happy Git and GitHub for the useR beforehand. Note that most of the steps can be made either in the terminal or by clicking buttons in RStudio so we tried to mix both.

No releases published
Create a new release

Packages
No packages published

Repositorio en GitHub destacando información importante

Algunos enlaces interesantes

R Markdown

- [R Markdown](#)
- [RStudio Cheat Sheets](#)

Ciencia reproducible

- [Ciencia reproducible: qué, por qué, cómo](#)

Control de versiones (Git)

- [Manual de referencia de Git](#)
- [Software Carpentry](#)
- [Atlassian Bitbucket](#)
- [Oh Shit, Git!?!?](#)
- [git - la guía sencilla](#)

Integrar Git, GitHub y RStudio

- [Happy Git and GitHub for the useR](#)

Enseñar y aprender con GitHub

- [GitHub Education para profesores e investigadores](#)



1. git commit



2. git push



3. leave building



Session Info

Footnotes

1. hello world [[←](#)]

References

Blischak, John D., Emily R. Davenport, and Greg Wilson. 2016. "A Quick Introduction to Version Control with Git and GitHub." *PLOS Computational Biology* 12 (1): e1004668. <https://doi.org/10.1371/journal.pcbi.1004668>.

Galeano, Javier. 2018. "¿Por qué usar GitHub? Diez pasos para disfrutar de GitHub y no morir en el intento." *Ecosistemas* 27 (2): 140–41. <https://doi.org/10.7818/ECOS.1604>.

Peng, Roger D. 2011. "Reproducible Research in Computational Science." *Science* 334 (6060): 1226–27. <https://doi.org/10.1126/science.1213847>.

Perez-Riverol, Yasset, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, et al. 2016. "Ten Simple Rules for Taking Advantage of Git and GitHub." *PLOS Computational Biology* 12 (7): e1004947. <https://doi.org/10.1371/journal.pcbi.1004947>.

Ram, Karthik. 2013. "Git Can Facilitate Greater Reproducibility and Increased Transparency in Science." *Source Code for Biology and Medicine* 8 (1). file:///C:/Users/julen/OneDrive/Escritorio/GitHub-col/intro_git-github/doctordado-uah/intro_repro.html

Medicine 8 (1): 7. <https://doi.org/10.1186/1751-0473-8-7>.

Rodríguez-Sánchez, Francisco. 2020. "Quince consejos para mejorar nuestro código y flujo de trabajo con R." *Ecosistemas* 29 (3): 2129–29. <https://doi.org/10.7818/ECOS.2129>.

Rodríguez-Sánchez, Francisco, Antonio Jesús Pérez-Luque, Ignasi Bartomeus, and Sara Varela. 2016. "Ciencia reproducible: qué, por qué, cómo." *Ecosistemas* 25 (2): 83–92. <https://doi.org/10.7818/ECOS.2016.25-2.11>.

The Turing Way Community. 2021. *The Turing Way: A Handbook for Reproducible, Ethical and Collaborative Research (1.0.1)*. Zenodo. <https://doi.org/10.5281/zenodo.5671094>.