Índice

¿Qué es la reproducibilidad?

¿Qué es la reproducibilidad?

"reproducibility refers to the ability of a researcher to duplicate the results of a prior study using the same materials as were used by the original investigator. That is, a second researcher might use the same raw data to build the same analysis files and implement the same statistical analysis in an attempt to yield the same results. . . . Reproducibility is a minimum necessary condition for a finding to be believable and informative."

NSF 2015, citado en Goodman, Fanelli, and Ioannidis 2016, Science Translational Medicine Vol. 8, Issue 341, pp. 341ps12

¿Qué es la reproducibilidad?

- es la capacidad de recrear o producir de nuevo los resultados de una investigación **dada la misma data**
- el "reproductor" puede ser uno mismo en el futuro u otro investigador
- distinto a la replicabilidad: capacidad de un investigador de duplicar los resultados de un estudio si se siguen los mismos procedimientos pero se recaban nuevos datos
- requiere buenas prácticas de organización y distribución del material de investigación

¿Qué es la reproducibilidad?

¿Por qué mantener la reproducibilidad en nuestras investigaciones?

- es requisito mínimo para que los resultados sean entendibles y confiables
- facilita el trabajo con coautores
  - la investigación con coautores es cada ves más común
- aumenta la eficiencia y escalabilidad del trabajo propio
  - los proyectos son cada ves más complejos y necesitan mayor minuciosidad del investigador
  - dado los largos tiempos de investigación, es muy probable tener que realizar revisiones en el futuro, cuando uno no se acuerda de todo lo que hizo

¿Qué es la reproducibilidad?

- veremos dos estrategias para mantener la reproducibilidad en nuestras investigaciones
  - automatización y organización de directorios
  - control de versión (con Github Desktop)

Organización de directorios

Organización de directorios

Supongamos que acabamos de obtener datos de un experimento. El instinto nos llevará a analizar los datos de la siguiente manera:

- abrir programa de análisis
- importar datos
- realizar la regresión
- copiar y pegar los resultados para escribir el reporte

Organización de directorios

Esa es la antítesis de la reproducibilidad

- no queda registro alguno de cómo se obtuvieron los resultados
- para hacer una revisión en el futuro, se deben repetir todos los pasos

Organización de directorios

La solución es escribir scripts

- son programas escritos para la ejecución automatizada de tareas
- debemos utilizarlos para todo lo que podamos
  - cargar los datos
  - limpiar los datos
  - analizar los datos

Organización de directorios

Ejemplo de script

Organización de directorios

El objetivo es crear un proceso automatizado que empiece tomando los datos originales (fuente), ejecute tareas programadas (scripts) y produzca los resultados reportados en la investigación.

Al ser automatizado, el proceso es reproducible.

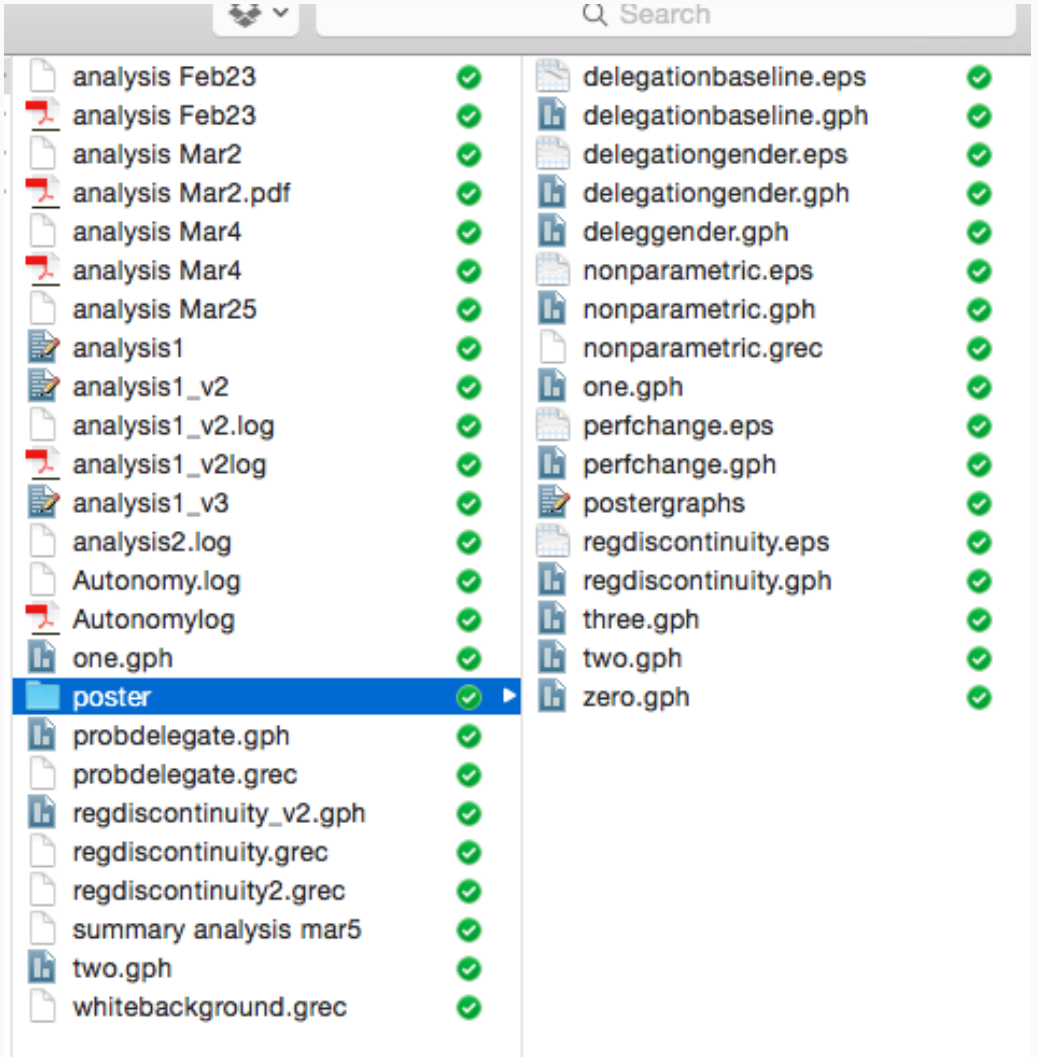
Organización de directorios

Crear tal sistema requiere organización para evitar confusión sobre

- ¿qué archivo depende de cuál?
- si actualizo los datos, ¿el script corre correctamente?
- si corro el script, reemplazo los datos, pero cometo un error, ¿pierdo los datos?

Organización de directorios

Caos



Organización de directorios

Código inutilizable después de todo el tiempo invertido

```
. do "/var/folders/n8/ydk3gnf112ngh7f11gxn_m900000gn/T//SD01180.000000"

. use "./dataset.dta"
file ./dataset.dta not found
r(601);

end of do-file

r(601);

.
```

Organización de directorios

Recomendaciones para manejar el flujo de trabajo:

- Scott Long: *The Workflow of Data Analysis Using Stata*
- Gentzkow y Shapiro: A Practitioner's Guide *video*, *texto*
- *Guía de Software Carpentry*

Organización de directorios

Tips #1: Tratar la data como solo lectura ("read only")

- no editar manualmente (ej. excel, editor Stata)
- no reescribir
- esto mantiene la integridad de la data y permite saber cómo la fuente ha sido modificada (mediante scripts) para producir otra data

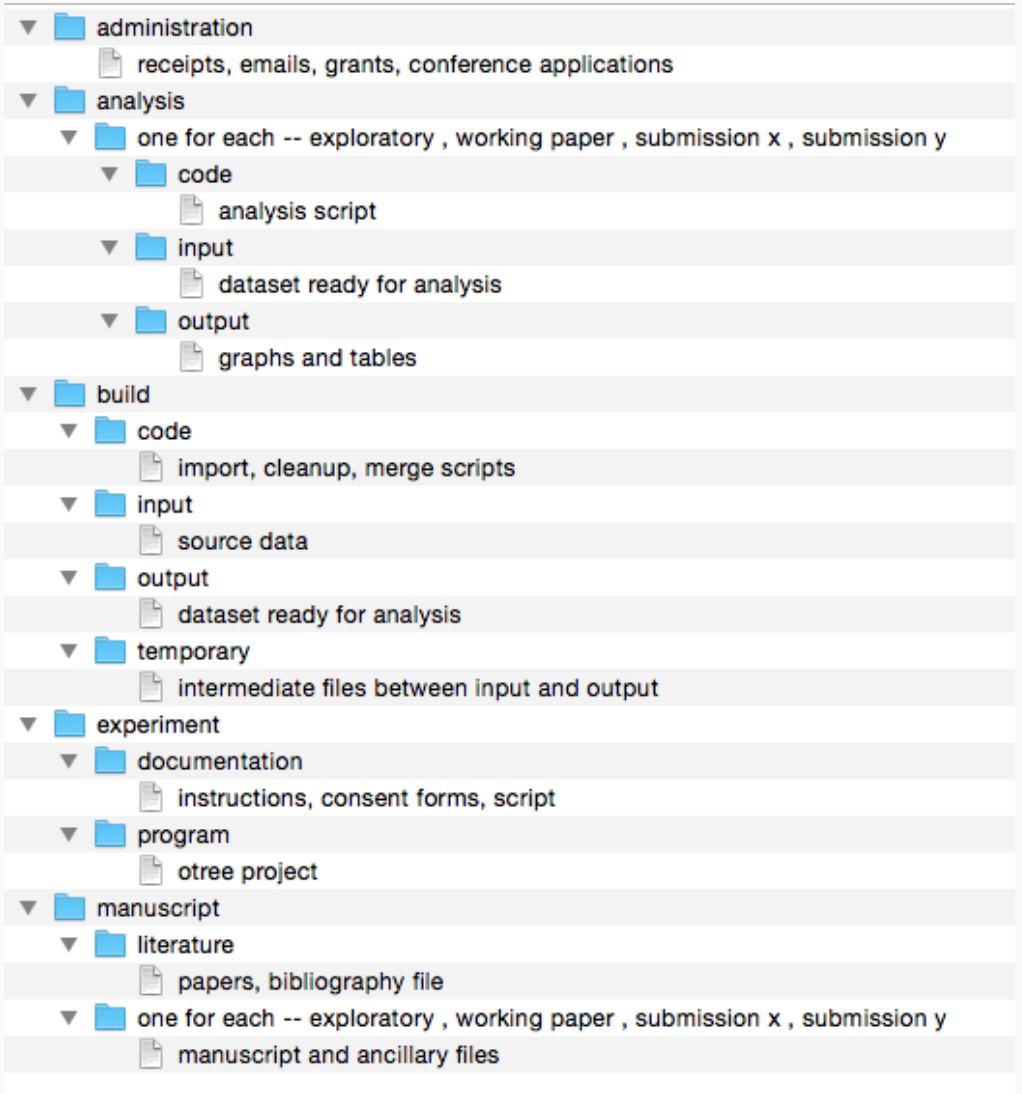
Organización de directorios

Tips #2: Crear subdirectorios distintos para limpieza y análisis de datos

- scripts de limpieza toman la fuente, la limpian (agregan/remueven variables, combinan bases), producen una nueva data lista para análisis, y la guardan como un archivo distinto
- scripts de análisis toman la data lista para análisis y aplican el análisis
    - pueden editar la data pero nunca la guardan
    - producen y guardan gráficas y tablas

De esta forma se reproduce todo el proceso desde la fuente al resultado, y se puede reproducir el análisis sin tener que re-ejecutar la limpieza

Organización de directorios: sugerencia



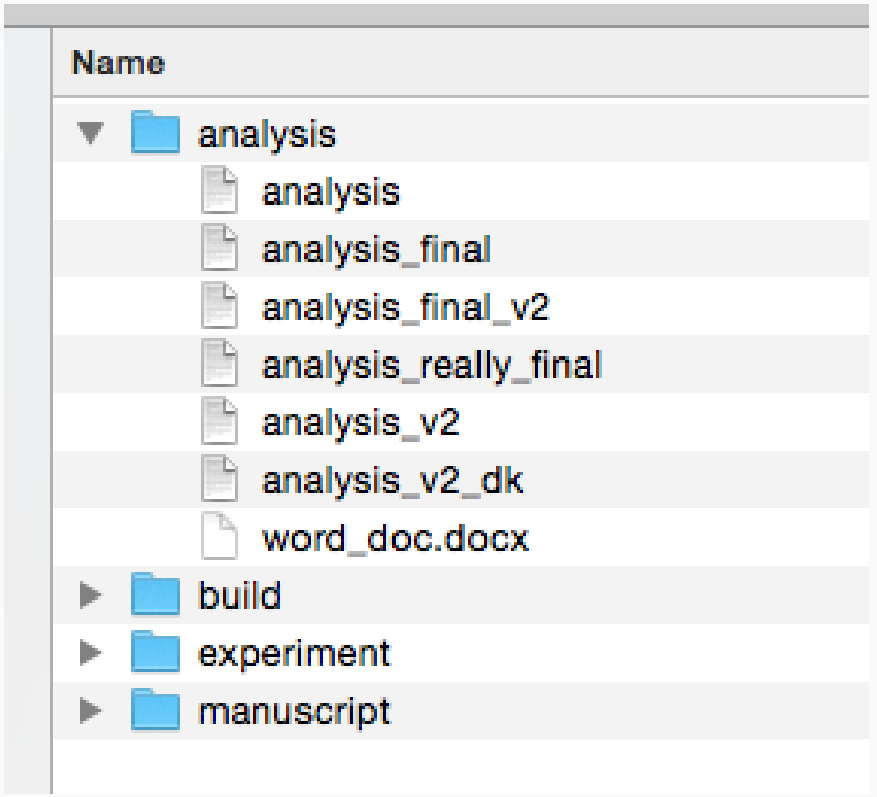
Control de versión con Git usando Github Desktop

Control de versión con Git usando Github Desktop

- la automatización y organización de directorios son pasos hacia mayor reproducibilidad
  - remueven las intervenciones manuales al proyecto
  - dejan registro de todos las acciones
  - compartimentan el proyecto, aclaran las dependencias entre archivos y permiten compartir/publicar solo ciertas partes del proyecto
- pero aun manteniedo la automatización y organización, caos dentro de cualquier directorio puede inutilizar todo el proceso

Control de versión con Git usando Github Desktop

¿¿¿Cuál es el análisis final, y en qué varía de los demás???



Control de versión con Git usando Github Desktop

Solución: Control de versión

- es una herramienta para tomar "fotografías" del proyecto en los momentos que queramos
- registra los cambios hechos entre una fotografía y otra, así como quién los hizo y cuándo
- permite recobrar el proyecto a su estado en el momento de cualquier fotografía
- existen muchos sistemas de control de versión
    - usaremos Git

Control de versión con Git usando Github Desktop

Git vs. Github vs. Github Desktop

- Git
  - un sistema en particular de hacer control de versión
  - la manera más versatil y completa de manejarlo es a través del Shell
- Github
  - una compañía que alberga repositorios Git
- Github Desktop
  - una application creada por Github para manejar las funciones básicas de Git sin necesidad usar el Shell

Control de versión con Git usando Github Desktop

Breve introducción controlar las versiones de un proyecto de investigación con Git usando Github Desktop

1. crear una cuenta en Github
2. instalar Github Desktop
3. rastrear un directorio
4. publicar directorio en Github
5. obtener versiones anteriores del proyecto en Github

Control de versión con Git usando Github Desktop

1. Crear una cuenta en *Github*
   - Github ofrece repositorios privados por un cargo, pero estudiantes e investigadores pueden *solicitar obtenerlos gratis*

Control de versión con Git usando Github Desktop

2. Descárgate *Github Desktop* y sigue las instrucciones de instalación

- una vez instalada, abre la aplicación
- ingresa a Preferences/Accounts para vincular la applicación con tu cuenta de Github
- ingresa tu nombre y email de usuario en Preferences/Git
    - puedes ingresar un email proporcionado por Github y evitar hacer tu email público – para ver este email ingresa a tu cuenta en *Github*, presiona el ícono en la esquina superior izquierda y luego presiona Settings/Emails

Control de versión con Git usando Github Desktop

- crear carpeta "*mi_proyecto*" en Desktop
- crear simple archivo *.txt* y guardarlo en *mi_proyecto*
- en Github Desktop, presiona *Create New Repository*
- Name: *mi_proyecto*
- Local Path: la ubicación de Desktop
- *Create Repository*
- en History aparecerá el archivo *analisis.txt*, que ya está siendo rastreado
    - de hecho todo lo que hagamos en el directorio *mi_proyecto* será rastreado

Control de versión con Git usando Github Desktop

Los cambios no quedan guardados en el registro (historial) de "fotografías" del proyecto hasta que hagamos "commit". La creación del repositorio fue nuestro primer commit. Para hacer otro:

- hacer cambios a analisis.txt
- en la barra *Summary* en la sección *Changes* ingresar un mensaje corto pero descriptivo de los cambios al proyecto
  - ej: agregamos segunda linea en analisis.txt
- *Commit to master*
- Ahora *History* registra dos fotografías (commits)

Control de versión con Git usando Github Desktop

Podemos realizar múltiples cambios al proyecto antes de tomarle una nueva fotografía

- más cambios (borrar y añadir líneas) a analisis.txt
- agregar el subdirectorio *manuscript*
- agregar un documento word a este subdirectorio
    - los documentos binarios como .doc se rastrean al igual que los archivos de texto, pero Git no puede mostrarnos explíticamente las diferencias entre versions
- realizar un nuevo commit

Control de versión con Git usando Github Desktop

- hasta ahora el registro de las versiones del proyecto se encuentra únicamente en nuestra computadora local
- es recomendable trasladarlo también a un sitio remoto
  - para respaldar los datos
  - para compartir el proyecto con co-autores o el público en general
  - para acceder a versiones anteriores del proyecto sin usar el Shell

Control de versión con Git usando Github Desktop

Para mover ("push") el repositorio a Github desde Github Desktop:

- *Publish Repository*
- *Name*: mi_proyecto
- Ahora nuestro registro de todas las fotografías del proyecto se encuentra en Github
  - el historial de versiones se encuentra en la sección *Commits*
  - podemos descargar cualquier archivo o el proyecto entero en cualquiera de sus versiones
  - otras personas pueden copiar ("fork") nuestro proyecto

Control de versión con Git usando Github Desktop

Finalmente, para descargar ("fork") un repositorio de otra persona en Github desde Github Desktop:

- descargaremos la *sugerencia de estructura de directorios*
- Para descargarlo desde la página web:
  - Clone or download / Open in Desktop
- Para descargarlo desde Github Desktop:
  - *Clone a Repository*
  - URL: *https://github.com/davs-econ/research_project_structure*