

Lec1: Intro & Git

Isidoro Garcia Urquieta

2023

Objetivos del curso

- ▶ Este es un curso de Advanced Analytics. El objetivo es proveerlos de las herramientas estadísticas y de programación desde el enfoque de economistas.

Objetivos del curso

- ▶ Este es un curso de Advanced Analytics. El objetivo es proveerlos de las herramientas estadísticas y de programación desde el enfoque de economistas.
- ▶ Al final de curso espero que puedan:

Objetivos del curso

- ▶ Este es un curso de Advanced Analytics. El objetivo es proveerlos de las herramientas estadísticas y de programación desde el enfoque de economistas.
- ▶ Al final de curso espero que puedan:
- ▶ Construir modelos predictivos de Regresión o Clasificación (Random Forests, LASSO, XgBoosting)

Objetivos del curso

- ▶ Este es un curso de Advanced Analytics. El objetivo es proveerlos de las herramientas estadísticas y de programación desde el enfoque de economistas.
- ▶ Al final de curso espero que puedan:
- ▶ Construir modelos predictivos de Regresión o Clasificación (Random Forests, LASSO, XgBoosting)
- ▶ Construir modelos de inferencia causal combinados con *Machine Learning*

Objetivos del curso

- ▶ Este es un curso de Advanced Analytics. El objetivo es proveerlos de las herramientas estadísticas y de programación desde el enfoque de economistas.
- ▶ Al final de curso espero que puedan:
- ▶ Construir modelos predictivos de Regresión o Clasificación (Random Forests, LASSO, XgBoosting)
- ▶ Construir modelos de inferencia causal combinados con *Machine Learning*
- ▶ Usar técnicas de reducción de dimensionalidad (K-means, PCA)

Objetivos del curso

- ▶ Este es un curso de Advanced Analytics. El objetivo es proveerlos de las herramientas estadísticas y de programación desde el enfoque de economistas.
- ▶ Al final de curso espero que puedan:
- ▶ Construir modelos predictivos de Regresión o Clasificación (Random Forests, LASSO, XgBoosting)
- ▶ Construir modelos de inferencia causal combinados con *Machine Learning*
- ▶ Usar técnicas de reducción de dimensionalidad (K-means, PCA)
- ▶ Hacer uso de las herramientas más utilizadas en la industria de Data Science (R, Python, Github)

Objetivos del curso

- ▶ Este es un curso de Advanced Analytics. El objetivo es proveerlos de las herramientas estadísticas y de programación desde el enfoque de economistas.
- ▶ Al final de curso espero que puedan:
- ▶ Construir modelos predictivos de Regresión o Clasificación (Random Forests, LASSO, XgBoosting)
- ▶ Construir modelos de inferencia causal combinados con *Machine Learning*
- ▶ Usar técnicas de reducción de dimensionalidad (K-means, PCA)
- ▶ Hacer uso de las herramientas más utilizadas en la industria de Data Science (R, Python, Github)
- ▶ Entender como combinar teoría económica con Data Science.

Un poco sobre mi

- ▶ Soy Economista del ITAM y Master en Uchicago.
- ▶ Ahora trabajo como VP of Data Science en Bitso.
- ▶ Antes de eso fui Director de Analytics, Machine Learning and Experimentation en Banorte. Pasé por el camino tradicional de Banxico (Inveco), Research Assistant y Banco Mundial.
- ▶ En este tiempo conocí muchos data scientists que o: tienen las habilidades computacionales pero no saben construir relaciones entre variables (no son economistas...) o saben teoría económica pero no saben programar ni conocen algoritmos que involucren Big Data.
 - ▶ Los economistas que sepan combinar teoría (costo de oportunidad, marginalidad, etc) con herramientas estadísticas y de programación pueden conquistar el mundo de Data Science.

Diferencias entre BD, Econometría, Data Science, ML y AI

Big data, Econometría, Data Science, Machine Learning, Artificial Intelligence son varias de las palabras que escuchamos sobre cosas similares.

- ▶ La **econometría** se enfoca primordialmente en la inferencia de **parámetros**. A partir de ahí, se utilizan estos modelos para:
 - ▶ Inferir relaciones causales entre variables,
 - ▶ Predecir la variable endógena del modelo.
 - ▶ De cualquier manera, la econometría hace un énfasis fuerte en basarse en teoría para entender el fenómeno e construir intuición sobre él.
- ▶ **Big Data** se refiere a la extracción de valor (insight y/o predicciones) de bases de datos gigantes (i.e. muchas columnas y/o muchas filas) que no caben en una computadora. Es mucho más pragmático.

Diferencias entre BD, Econometría, Data Science, ML y AI

- ▶ **Data Science** se define como el uso de métodos científicos para extraer conocimiento y valor de los datos (estructurados y no estructurados).
 - ▶ Esto involucra los métodos para analizar los datos: Estadística, Econometría y Machine Learning; y
 - ▶ Computer Science: Los métodos para recibir, guardar y estructurar los datos (Data Engineering, MLE)
- ▶ **Machine Learning** son una colección de algoritmos ‘tontos’ que aprenden cosas solos. Esto es, algoritmos de estadística y econometría + esteroides (Re-muestreo y mucha capacidad de cómputo).
- ▶ Finalmente, **Artificial Intelligence** es una colección de algoritmos que aprenden con más autonomía que los algoritmos de machine learning.

De que es este curso entonces?

Los algoritmos que veremos cubren casi todo el espectro de Data Science menos la ingeniería de datos y MLE. Estos cubren:

- ▶ **Métodos Supervisados:** Tienes una variable éndogena y que gobierna el aprendizaje estadístico.
- ▶ **Métodos No Supervisados:** Sólo tienes X' s y no variable endógena.
- ▶ **Métodos Preescriptivos:** Queremos inferir relaciones causal. (Hint: Mucho más allá de ATE)

Finalmente, haré énfasis en el **Exploratory Data Analysis (EDA)** en todo el curso. El uso de gráficas y tablas es CRUCIAL para lograr comunicar efectivamente a audiencias no técnicas.

Temario

Tendremos 16 sesiones. En ellas vamos a tener ~12 sesiones teóricas y 3 sesiones con invitados de la industria. Vamos al temario

Qué es Git

Es un sistema distribución open source para el control de versiones de código & trackeo del mismo durante el desarollo de software.

- ▶ Es como un Google Drive + Control de cambios (pero más cool).



Qué es Github

Github es una plataforma en línea para guardar repositorios (folders) e interactuar con Git de forma mas amigable.

Search or jump to... Pull requests Issues Marketplace Explore

isidorogu / RCT

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Go to file ...

3rd master - RCT / R/balance_regression.R

Latest commit checked out Jul 12 History

1 contributor

31 lines (36 select) 3,30 KB

```
#' #> balance_regression() Run a LPM of treatment status against all covariates (treatment=0/1).
#> #> given data a data.frame, tibble or data.table
#> #> return treatment a string with treatment status column
#> #> first argument is the outcome column, second argument is the treatment column
#> #> P <- p.adjust = value with the P values of each regression
#> #> pvalues
#> #> example
#> #> data <- carData::FleaTreatment <- mcmc(n = 100, seed = 123), xdn <- mcmc(n = 100, mean = 65),
#> #> x <- rnorm(100, 12, 4), y <- rnorm(100, 10, 2)
#> #> p <- balance_regression(x = mcmc, outcome = "treatment", treatment = "treat")
#> #> p <- p.adjust = 0.05, p <- 0.05
#> #> balance_regression(x = data, outcome = "treatment", treatment = "treat")
#> #> p <- p.adjust = 0.05, p <- 0.05
#> #> column ca data. For instance, if treatment column has values of {0,1,2}, balance_regression will run two
#> #> models: 1) LM(treatment=0|y=0) and 2) LM(treatment=0,2|y=1). The value are the regression tables and
#> #> pvalues are the P values of these models.
#> #> Descript
#> #> Usage
#> #> from magrittr
#> #> balance_regression <- function(data, treatment) {
#> #>   extracttreat<-NULL
#> #>   extractpvalue<-NULL
#> #>   extracttable<-NULL
#> #>   data <- data %>% dplyr::arrange(treatment)
#> #>   valancs_trat <- base::unique(dplyr::pull(data, c(treatment)))
#> #>   trat<-valancs_trat[2]:length(valancs_trat)
#> #>   trat
#> #>   trat_tables<-purrr::map(trat, function(x)
#> #>     data %>% filter((treatment == valancs_trat[1]) | (treatment == valancs_trat[2] == trat))
#> #>     )
#> #>   trat_tables
```

Vayan todos a abrir su cuenta en Github. Recuerden su mail personal y user.name

Instalación de Git

Pasos:

1. Ir a <https://git-scm.com>,
2. Descargar Git para tu sistema operativo (Windows, Mac, Linux),
3. Una vez instalado, Git va a mostrarte Git Bash en Windows. Para Mac, Git va a estar listo desde la Terminal.
 - ▶ Desde ambos puedes empezar a usar Git

Despues en Git Bash/Terminal:

```
git config --global user.name 'YOUR NAME' #El usuario  
git config --global user.email 'email@yemail.com'  
git config -l # Ver el resultado
```

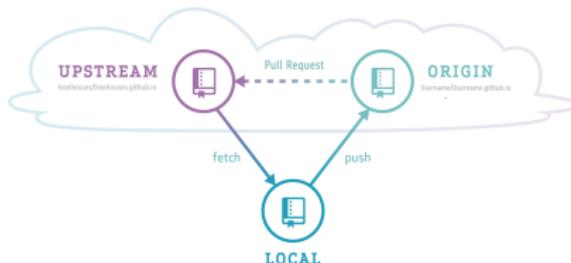
Repo local y remoto

Usar git te va a ayudar a controlar versiones en tus proyectos. Git se organiza en repositorios:

Repositorio: Es un “folder” donde vas a guardar todos los archivos de tu proyecto. Normalmente vas a tener un **Repositorio local (master)** y un **Repositorio Remoto (origin/upstream)** que se comunican entre sí.

Con esto tienes:

- ▶ Acceso al repositorio desde donde sea (como la nube en Google Drive)
- ▶ Acceso a cualquier versión del repositorio (versión por cambio)
- ▶ Una interfaz de control de cambios/versiones muy completa



SSH keys

Vamos a configurar la comunicacion entre local y remote. Para Windows, abran Git Bash. Para Mac abran Terminal. Corran lo siguiente:

```
cd ~/ #ve a home  
ssh-keygen -t rsa #Crea un par de keys
```

Esto genera un par de contraseñas (pública y privada) que se comunican entre sí de manera encriptada. La pública se guarda en Github y la privada se queda en tu computadora.

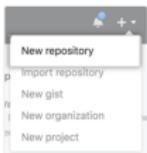
Despues ve a Github -> Settings -> SSH and GPG keys -> New SSH key. Sube la key publica. Nombrala algo como 'mi Mac'.

Con esto, puedes subir/bajar archivos entre tus repositorios de manera trivial

Crear remote desde un local existente

1. Creamos un repo 'bare' vacío en Github

1 Create a new repository on GitHub. To avoid errors, do not initialize the new repository with README, license, or .gitignore files. You can add these files after your project has been pushed to GitHub.



2. Copiamos la información SSH del repo remoto

A screenshot of a GitHub repository page for 'crossfit'. The page shows a single file, 'README.md', with the content 'crossfit library'. At the top right, there is a 'Code' dropdown menu. The 'SSH' option is selected, showing the URL 'git@github.com:isidoregu/crossfit.git'. Other options in the menu include 'Clone' (HTTP, SSH, GitHub CLI), 'Open with GitHub Desktop', and 'Download ZIP'.

3. En Terminal/Git Bash

```
cd './ITAM_Eco_Computacional/' # Cambias a donde estan los archivos
git init # Conviertes a git repository
git add . # Añadir todo '.' lo que esta a stage
git commit -m "Subo archivos"
#La opcion m es 'message'
git remote add origin git@github.com:username/repo.git
# Añade repo remoto al local
git remote -v # Verifica que esten conectados
git branch -M main # Renombra a main el local
git push -u origin main
# push 'empuja' los archivos al remoto '-u'
# 'u' es upstream
```

Listo! Tu repo local y remoto están sincronizados

Crear un repo local desde un remoto existente

```
cd './ITAM_Eco_Computacional/'  
# Cambias al directorio donde quieres que esten los archivos  
git clone git@github.com:username/repo.git  
# Clonas al repo remoto al local
```

Listo!

Como trabajar despues del set up

Ya tienes el repo local y remoto configurados. Ahora qué?

- ▶ Trabaja normal en tus archivos en R (Stata, Python, Word, ...)
- ▶ Ve a Terminal/Git Bash y cambia al directorio con cd
- ▶ Checa como se ve todo con git status
- ▶ Haz git add, git commit y git push

Lista de comandos básicos

Veamos la lista de comandos básicos

comando	descripcion
status	Status del repo
checkout	Moverme a alguna version de cambio o branch
add	Añadir archivos al stage
commit	Comprometer los cambios en add
push	Empujar los cambios al repo remoto
remote	Hacer algun cambio del repo remoto al que se conecta el local
fetch	Buscar los cambios hechos en el repo remoto
merge	Combinar tu repo local con los cambios que jalaste en fetch
pull	git fetch + git merge
rebase	Mueve el repo local a un punto particular. cambios secuenciales
branch	Crea copia del repo remoto para trabajo paralelo

git status

Te muestra el estado de tu repo local vs el remoto. En que branch estás, que archivos están stage y cuáles no.

Hay cambios que no hiciste git add

```
Marish KINGM64 /e/ToolsQA/First Project (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   ABC.txt

no changes added to commit (use "git add" and/or "git commit -a")
Marish KINGM64 /e/ToolsQA/First Project (master)
$ ]
```

git add

Guarda los cambios o los pone en stage. Esto significa que Git ahora sigue los cambios de los archivos add (stage).

```
Harish MINGW64 /e/ToolsQA/First Project (master)
$ git add ABC.txt

Harish MINGW64 /e/ToolsQA/First Project (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
    new file:   ABC.txt

Harish MINGW64 /e/ToolsQA/First Project (master)
$ |
```

git commit

Esto significa que te ‘comprometes’ a los cambios que añadiste con git add. Git guarda cada commit como una versión del repositorio.

```
(base) isidorogarcia@Isidoro-Garcias-MacBook-Pro ITAM_Eco_Computacional % git co  
main  
[main d8cfcfb] Cambios lec1  
 3 files changed, 1333 insertions(+)  
  rename Lectures/Lec1/Intro & High Dimension Inference.Rmd => Intro & Git.Rmd  
 (99%)  
  create mode 100644 Lectures/Lec1/Intro---Git.log  
  create mode 100644 Lectures/Lec1/Intro---Git.pdf
```

La opción -m es el mensaje del commit individual en cuestión. La idea es que este nombre sea corto y ayude a identificar esos cambios (i.e. ‘primera estimación’, ‘data cleaning’, etc).

git remote

Te deja crear, ver y borrar conexiones entre tu repositorio local y algún repositorio remoto.

La sintaxis es `git remote [add| rm | rename | -v]`

```
git remote add <name> <url>
```

Create a new connection to a remote repository. After adding a remote, you'll be able to use as a convenient shortcut for in other Git commands.

```
git remote rm <name>
```

Remove the connection to the remote repository called .

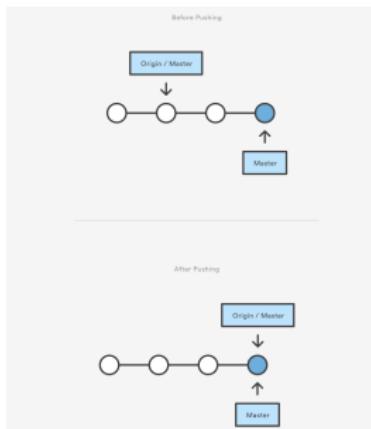
```
git remote rename <old-name> <new-name>
```

Rename a remote connection from to .

git push

Manda los cambios de git commit al repositorio remoto configurado.

La sintaxis es `git push -u <remoto> <local branch>`. La `-u` sólo se usa la primera vez cuando defines que el repo remoto esta conectado al local.

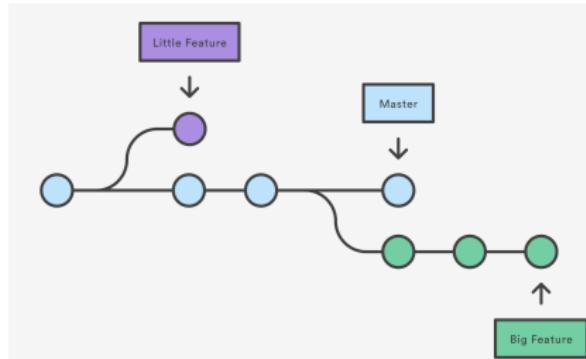


La opcion `<local branch>` es opcional. Si estas en el branch adecuado no necesitas poner la parte de `<local branch>`.

git branch

Cuando tienes un proyecto grande, quieres guardar una versión del código que funcione. Si quieres desarrollar una mejora o cambio, no quieres alterar ese código “maestro” (`master!`) y arruinar todo. Por eso creas un branch.

La branch (le pones un nombre autocontenido) de deja trabajar libremente sin alterar el código maestro.



git branch (2)

La sintaxis es:

`git branch nombre_branch`: Crea una nueva branch

`git branch`: Lista todas las branches.

`git branch -d`: Soft delete de la branch. La borra si es que no hay cambios unmerge.

`git branch -D`: Hard delete de la branch.

`git branch -m <>`: Renombra la branch <>,

`git branch -a`: Lista todas las branches en el remoto.

git checkout

Te mueve a la branch local existente (master, branch_x, branch_y) donde quieras trabajar.

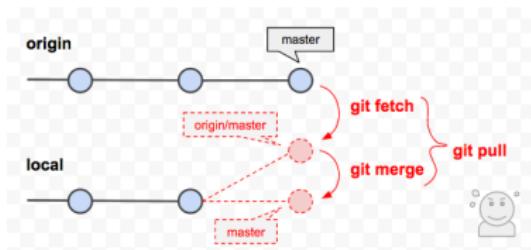
La sintaxis es `git checkout branch_x` para ir a la `branch_x`.

Adicionalmente, puedes hacer checkout a una branch remota (i.e. una donde ha estado trabajando tu compañerx de equipo) con:

`git fetch --all` y despues `git checkout branch_remota`

git fetch

Descarga los commits y archivos que se hicieron al repo remoto al repo local y lo guarda en `remote-repo(origin)/main`.



La sintaxis es:

`git fetch` : Descarga todo del remoto

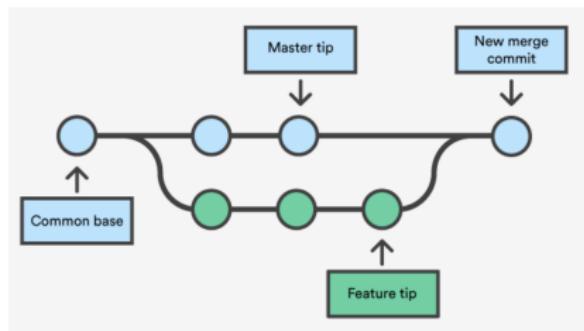
`git fetch <remote> <branch>`: Sólo descarga una branch en específico

Nota: Esto sólo baja los commits remotos. No los junta con tu repo local (porque puedes cambios propios). Para **juntar** ambos (cambios remotos y locales) hay dos opciones: `git fetch + git merge` o `git fetch + git rebase`.

git merge

Junta dos secuencias de commits en uno sólo y crea un commit de la unión de ambas secuencias. git merge es **ideal** para la colaboración en equipo.

La sintaxis es: `git merge <branch_a_merge>`. Es importante hacer `git checkout` antes a la branch que quieras que reciba el merge.

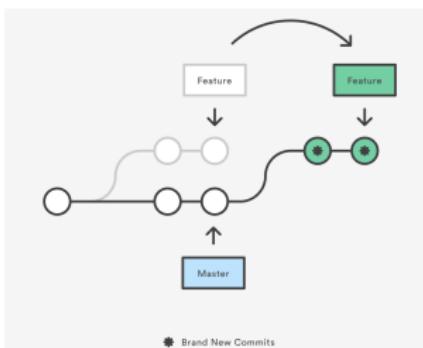


git merge es muy poderoso. Une 'magicamente' dos historias de cambios de manera automática. Git se fija si ambas historias son compatibles o divergentes. En este último, Git te indica exactamente en donde hay divergencia para que decidas cuál historia quedarse ahí.

git rebase

Te ayuda a mantener una historia lineal de tu branch respecto a master.

La sintaxis es: `git rebase <branch_a_rebase>`. Es importante hacer `git checkout` antes a la branch que quieras que reciba el rebase.



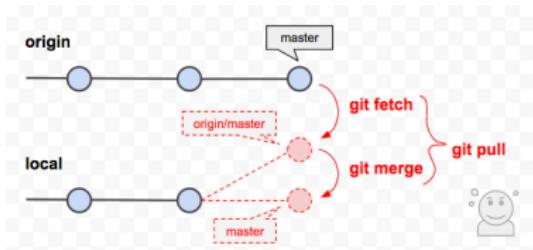
Imaginen que están trabajando en un proyecto. Uno trabaja en la estimación y otro en la validación de un modelo. Se encuentra un bug en la estimación mientras ya estaban trabajando en la validación. `git rebase` te ayuda a incorporar ese cambio para que parezca que la validación empezó con la última versión de la estimación.

git pull

Es como `git pull` = `git fetch + git merge` en un comando.

También existe `git pull` = `git fetch + git rebase`.

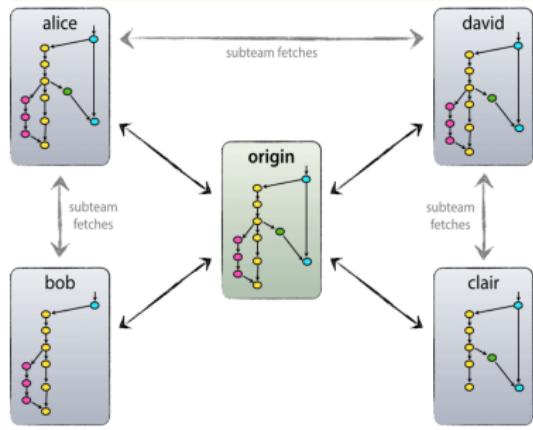
La sintaxis es: `git pull` para la opción `merge` o `git pull --rebase` para `fetch`.



Puedes checar si tu git tiene como default merge o rebase con `git config --list`. Para cambiar la configuración corre: `git config --global pull.rebase true`.

Pull requests

Los pull requests son peticiones para incorporar cambios al código maestro por algún miembro del equipo. Los otros miembros revisan el código y aprueban o piden cambios del autor. Una vez aprobado, se hace un merge a master. Además, uno puede configurar cuántas revisiones se necesitan para que un código se apruebe.



Pull requests 2

The screenshot shows a GitHub pull request page with two open pull requests:

- biderango's pull request #99:** A pull request titled "Data for revenue model (from financials from Argentinean users)" with 21 files changed. It has 300 reviews and is currently in progress. The code changes involve moving data from Argentina to the main branch. It was created 17 days ago and has 10 commits.
- biderango's pull request #100:** A pull request titled "requested review from bisees@data-science and bisees@data-science-models as code owners" with 17 days ago. It has 10 changes and 10 reviews. It was created 17 days ago and has 10 commits.
- karimkhalil's pull request #101:** A pull request titled "requested changes on behalf of bisees@data-science and bisees@data-science-models" with 17 days ago. It has 10 changes and 10 reviews. It was created 17 days ago and has 10 commits. The code changes involve merging data from multiple tables into a single one.

Pull requests 3

