ANOTACIONES EXAMEN 1 LPE

1. Terminal 2ª Clase 23/09/22

1.1. Comandos de terminal utilizados:

`pwd` = saber en que carpeta nos encontramos
`ls` = listado de carpetas del lugar donde donde nos encontramos
`cd nombre_carpeta` = dirigirnos a una carpeta
`cd ..` = volver hacia atrás
`cd\` ?
`touch cp =` crear archivo
`nano cp.txt` = invocar un notepad
`less` = ver el archivo
`cat` = lo mismo que less sin mover el ratón
`were, which =` encontrar archivo

1.2. Comandos git

```
git branch –M main
git remote add origin “https_enlacegithub_repo”
```

2. Terminal 30/09/22 Kaggle

`pip install kaggle` = instalamos kaggle
`mkdir .kaggle` ? = creamos la carpeta (es invisible)
`cd .kaggle` = nos dirigimos a la carpeta
`ls kaggle.json`
`less kaggle.json`
`–h` = para saber lo que podemos hacer (`usage: kaggle datasets [–h] {list,files,download,create,version,init,metadata,status}`)
`kaggle datasets list` = listado de datasets
`kaggle d list –s ‘crime’` = filtrar listado de dataset por crímenes

3. Terminal 05/10/2022 SSH

Sirve para transmitir información del ordenador hasta el repositorio. (Instalar Homebrew)

`brew install gh`
`gh auth login` = aparece lo de abajo

```
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Generate a new SSH key to add to your GitHub account? Yes
? Enter a passphrase for your new SSH key (Optional)
? Title for your SSH key: MacPro
? How would you like to authenticate GitHub CLI? Login with a web browser
! First copy your one-time code: A908-D03F
Press Enter to open github.com in your browser...
```

dan un código y lo abrimos en git hub para autentificar la cuenta.

4. Anotaciones del “Curso de Git hub”

4.1. Que es GIT?

1. Version control
2. Time Machine, volver a cualquier punto atrás.
3. Check points (commits), genera commits mediante funciones.
4. Multiverse (branch) permite crear diferentes versiones de un mismo código.
5. Sync (merging) sincronizar el código del repo local vs el repo remoto.

4.2. Instalar git

1. GIT website
2. Terminal
3. Credenciales (token, usuario y contraseña y ssh)
4. Homebrew

4.3. Setting up a project local

1. User and mail
2. Create a project
3. Install GH
4. Ssh

Abrimos terminal:

```
git config –l #ver credenciales
Cd LPE
```

Git init
Nano lee.txt crea archivo .txt

4.4. Git enviroments

1. Working, metemos archivo
2. Staging, esta comprometido
3. Commit, subido genera un check point con un hush

Git add .
Git status
Git commit -m “Added all files”

4.5. File states

4.5.1. Tracked

1. Modificado
2. No modificado
3. Stateg listo para hacer un commit

4.5.2. Untracked

4.6. Ignore files

Ej: Archivos que no queremos trackear
1. Personal info
2. Credential
3. System files

Terminal:

```
touch .gitignore
ls –alh
```

4.7. Clear cache

1. git rm -r --cached (borrar la caché)
2. git restore --S (recupera el archivo)

4.8. Differences

1. git diff
2. git log --online

4.9. Reset

1. git reset (volvemos a un punto anterior
git log #y salen todos los puntos

4.10. Rebase

No lo usaremos
1. Git switch
2. Git branch -D < >

Terminal
git branch
Nano pgr.txt

```
      def fib(n):
        if n in {1,0}:
                return
        else:
                return (fib(n-1)+fib(n-2))
ls
Git status
Git add .
Git commit -m “fib function”

Y subido

git switch -c cristina
Git branch
Nano prg.txt # añade lo de cris
Git status
Git switch main # volvemos al principal nuestro
Git branch
```

4.11. Git flo (Pregunta examen)

```
1. Features / fix
2. Make changes
3. Merge
4. Delete old branch
```

4.12. Git clean

```
git clean -n
Git clean -dnf
```

4.13. Working w / GitHub

1. Set up remote
2. Push
3.

```
1. Git push -u origin main ‘https://github.com/Aleemc11/LPE22013334'
```

4.14. Clona el siguiente repositorio usando ssh

Nos da un repo y una dirección

Hacemos:
Vamos GitHub al repo, donde code utilizamos ssh y lo copiamos y pegamos.

```
#Prueba de subida de archivo en GITHUB
git clone "https..  .git delrepo”.  #si no va bien la carpeta
git init
git add .     #si funciona la carpeta
git commit -m "message"
git push origin main 'https de la pg' / git push -u origin main
```

5. R Studio

5.1. Cargar librerías

```
install.packages (c("tidyverse", "httr","janitor")) #no me suele funcionar
install.packages ("tidyverse")
install.packages ("httr")
install.packages("janitor")
library(tidyverse) # para usarlas
library(janitor)
library(httr)
```

5.2. Comandos git

```
git status
git init
git add
git commit –m "message"
git push
git push –u origin main
git branch –M
git merge
git remote add origin urldelproyecto
git clone urldeloquequeremosclonarjiji
git fetch
```

5.3. Operaciones básicas

5.3.1. Guardar varios elementos de un mismo tipo: `clase_lep <- c("marta","emilia","pablo")`
5.3.2. Guardar varios elementeos de ≠ tipo con listas: `lista_prueba <- list("marta", 42)`

5.4. Obtener datos de internet

```
url <– (“url”)
df <– GET("url")
View (df) # visualizamos los datos
```

Crear columnas, etc. en archivo R