# Examen reto 4

* Entregables del examen:
  + Deberá compartirse el video grabado con OBS y el código hash del mismo (como en exámenes anteriores).
  + Deberá entregarse (En el Word) el link del repositorio de GitHub (el cual deberá tener acceso el usuario: jcabau.
  + Puede entregarse el proyecto de R comprimido en el entregable de MUdle, esta entrega es voluntaria y solo si no se ha conseguido subir a GitHub el proyecto. En caso de entregarse un archivo comprimido del proyecto y exista un repositorio de github se evaluará el repositorio.
  + El Word completado. Las respuestas dadas deberán estar **en Azul**
* Todo fallo o errata en la entrega que no sea derivada de fallos externos (mudle caído, problemas de wifi…) serán responsabilidad del alumnado.
* Consideraciones:
  + Puede consultarse las hojas de Git/GitHub como apoyo
  + Puede autogenerarse un .gitignore en GitHub
  + Mientras el ejercicio no especifique lo contrario puede resolverse tanto por consola de comandos como por las opciones gráficas de RStudio o el propio repositorio de GitHub.
  + No se puede utilizar internet, apuntes, o cualquier tipo de IA generativa.
  + Se puede hacer uso de la ayuda de RStudio, en cuanto a documentación y generación de archivos.

## Git

* Las preguntas relacionadas con el control de versiones serán evaluadas analizando el historial (o log). Por ello es importante y está contemplado dentro de la evaluación **cuando** realizar los diferentes, commits y push.
* Es importante seguir el orden de los commits y push indicados en los ejercicios. Commits o push extras que se realicen para corregir y reorientar el trabajo penalizarán (con -0,1/acción) a partir de **10 acciones correctoras.**

## Parte práctica

**GIT INICIALIZACIÓN (2,5 puntos)**

1- Crea un repositorio de GitHub llamado Nombre\_Apellido\_examen\_reto4 y compártelo con jcabau. Añade el link: **(0,1 puntos)**

[danielsoponaru/daniel\_a\_soponaru\_examen\_reto4](https://github.com/danielsoponaru/daniel_a_soponaru_examen_reto4)

2- ‘Enlaza’ un proyecto de RStudio creado en local de modo que podamos realizar un control de versiones sobre los scripts que trabajemos a lo largo del examen. **(1,5 punto)**

3- Introduce dentro del control de versiones los scripts de RStudio proporcionados **(0,1 puntos)**

4- crea un .gitignore que ignore todos los .csv y asegúrate que no hay un control sobre el dataset con el que vamos a trabajar. **(0,5 puntos)**

5- Realiza el primer commit (en la rama main) **(0,2 puntos)**

6- Haz push al repositorio de GitHub **(0,1 puntos)**

**EXPLORACIÓN Y LIMPIEZA DE DATOS (3 puntos)**

1- Crea una Rama (Branch) para la exploración y limpieza de datos. **(0,2 puntos)**

2- Realiza la exploración de los datos en el script correspondiente:

A) Completa la siguiente tabla: **(0,5 puntos)**

|  |  |  |
| --- | --- | --- |
|  | Minutes | Seasons |
| Max |  |  |
| 3. cuartil |  |  |
| Mediana |  |  |
| Media |  |  |
| 1. cuartil |  |  |
| Min |  |  |
| NAs |  |  |
| Desviación standard |  |  |

\* Para calcular las estadísticas descriptivas es importante omitir nos NAs

B) ¿Cuántos elementos únicos hay en cada una de las categorías? **(0,5 puntos)**

* **Type:**
* **Country:**
* **Director:**

3- Realiza un commit de la exploración de datos. **(0,1 puntos)**

4- Realiza la limpieza de los datos en el script correspondiente:

1. ¿Analiza el número de NAs por variable? ¿Cuál es la variable con mayor número de NAs? ¿Qué porcentaje representa? **(0,4 puntos)**
2. Visualiza la distribución de NAs. En función de la distribución (MACR, MAR, MNAR) y las variables afectadas ¿Qué tratamiento de NAs podríamos aplicar? **(0,5 puntos)**
3. Aplica el tratamiento de NAs de modo que podamos trabajar más adelante con las variables ‘Minutes’ y ‘Seasons’. **(0,5 puntos)**

5- Realiza un commit de la limpieza de datos **(0,1 puntos)**

6- Realiza un push al repositorio de GitHub **(0,2 puntos)**

**API (3 puntos)**

1- En la rama que has creado añade un script para generar una API con las siguientes características (Nota: puedes programar directamente la API de plumber o generar una función para luego implementarla en la API) **(1,5 puntos)**:

1. Debe trabajar sobre la variable ‘Minutes del dataframe de películas
2. Debe realizar una transformación de los datos (logarítmica, normalización,… elección libre).
3. Debe devolver un histograma que muestre la distribución de la variable ‘duration’
4. El input que proporcione el usuario debe indicar si vamos a hacer o no la transformación de los datos

2- Haz un commit y un push con la API funcional. **(0,2 puntos)**

3- Haz merge de la Rama creada con la rama principal (Main) y resuelve los conflictos si fuese necesario. **(1 puntos)**

4- haz un pull a la Rama principal de modo que el historial quede actualizado con el merge. **(0,3 puntos)**

## Parte teórica

**Escoge y explica tres áreas de la Gestión de Datos (1,5 puntos):**