Arruiné mi Blog ¡¡¡¡Ayuda!!!!

Miguel Equihua

Xalapa, Ver., 21 de febrero, 2025

Descompuse archivos clave

Como ya nos pasó, puede ser que eches a perder tu archivo *index.qmd* en la raíz de tu proyecto. Ese documento se encarga de construir la estructura general de tu *blog*. Es la que define la llamada **landing page** de tu proyecto, la cara del primer contacto de tu blog. Ojalá hayas echo un **commit** inicial justo acabando de crear tu proyecto. Si fuiste así de sabia o sabio, todo lo que tienes que hacer es buscar el archivo en cuestión y recuperarlo. **Github desktop** será tu amigo para eso o en la ventana de comandos de *RStudio*, la instrucción `git restore`, ve la sección siguiente.

Por si ocurriera un accidente de este tipo y no hubiera nada más que hacer pongo aquí las líneas generales que suele tener este archivo, cópialas en caso de emergencia y substituye todo lo que tengas en el archivo dañado (si pusiste cosas muy importantes que quiere conservar, haz una copia y haz la reparación en una de ellas. ¡Suerte!

```
---
title: "Título de inicio en tu blog "
listing:
  contents: posts
  sort: "date desc"
  type: default
  categories: cloud
  sort-ui: false
  filter-ui: false
page-layout: full
title-block-banner: true
---
```

Recuerda sólo hacer lo escencial en los documentos en la raíz de tu proyecto: el nombre de tu blog, título inicial y cosas así. No hagas más de lo indispensable en los archivos **index.qmd** y **_quarto.yml** que están en esta ubicación. Toda tu creatividad debes expresarla dentro de subcarpetas contenidas en la carpeta **posts**.

!Tengo una versión buena en git!

Lo primero es !serenidad y paciencia! Toda la idea de usar *git* es para estos casos. Claro, podrías volverte experta o experto en su uso más profesional mediante comandos, ojalá te animes. Pero por lo pronto hagámoslo con lo que ya tenemos a la mano. Veamos algunos escenarios posibles:

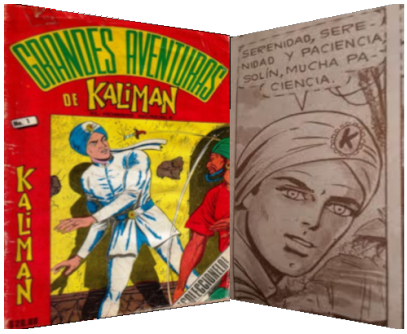

Figura 1: Kaliman personaje creado por Rafael Cutberto Navarro y Modesto Vázquez González en 1965. Se publicó en forma de *comic* semanalmente por 26 años. Vendió más de mil millones de copias.

1. Un escenario muy sencillo es cuando estas actualizando un documento que ya pusiste en el registro de *git*. Intentas *renderizarlo* y para tu sorpresa, ¡falla de manera inexplicable! Para colmo, hiciste un montón de cambios que ya ni te acuerdas. Eso sí, estas seguro o segura de que tu última versión, a la que le hiciste un *commit*, sí funcionaba, no a tu gusto completo, pero no se trababa como el que ahora te tiene de malas. En este caso, busca en la pestaña de *git* en *RStudio* el arhivo que estás trabajando, imagino un **index.qmd** en un folder dentro de la carpeta **posts**. Seleccionalo y aprieta el botón derecho del mouse. En las ventanita emergente encontrarás la palabra *revert...*. ¡Eso es!, dale click y recuperará el arhivo de la últia versión que resguardaste con un *commit*. ¡Listo!, estás de regreso y ahora puedes volver a empezar, con más cuidado y mejores ideas.

2. Otro caso es cuando has sido muy diligente y has echo todo bien. Haces cambios a tu gusto y cada vez que lo sientes apropiado has echo tus *commits*. Te vas por un tesito, cambias de humor, trabajas en otra parte de tu **blog** y haces commit a tus nuevos cambios. De pronto, un rayo de inspiración, te hace comprender que lo que hiciste al principio del día no era buena ideas, te arrepientes y quieres regresar a una versión que está seis o diez *commits* atras, sabes a donde te gustaría regresar pues anotaste un mensaje alusivo que te recuerda claramente el estado del documento que quieres retomar como inicio. ¿Cómo puedes viajar hacia atras en el tiempo a ese preciso momento? Bueno, hacer esto es relativamente sencillo con el comando `git restore` que puede rescatarte desde la *terminal* de *RStudio*. Las indicaciones *–source* y *main~x* van de cajón (la **x** toma el valor de cuantos *commits* atras hay que ir en la rama **main**), y terminas con la ruta al archivo de tu interés, en este ejemplo *posts/primer-ejemplo/index.qmd*.

```
git restore --source main~5 posts/primer-ejemplo/index.qmd
```

Otra opción para identificar el *commit* preciso de mi interés es poner enseguida de *–source* una parte reconocible en forma única de la firma hash *SHA* de 40 caracteres, que identifica en forma exacta a cada *commit*, y que puedes encontrar con la opción *History* en la pestaña *git*.

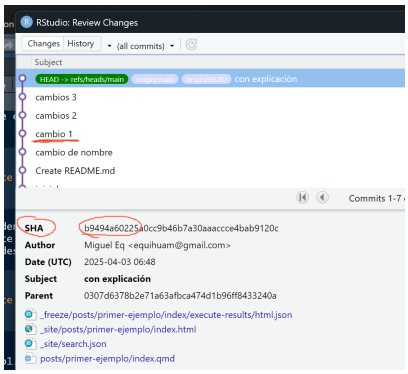

Figura 2: Ir hacia atras en la historia de mi proyecto

```
git restore --source cc12d23841a  posts/primer-ejemplo/index.qmd
```

Como por arte de magia, estás de regreso en el documento solicitado. Quizás *RStudio* te advierta que hubo cambios y te pida le des indicaciones, lee con cuidado lo que te diga y responde adecuadamente. ¡Listo!

¿Como Evitar arriesgar mi blog a cada commit?

Una manera de trabajar con más tranquilidad en tu **blog** es otra vez recurriendo a la ayuda que ofrece *git*. Te sugiero evitar trabajar el desarrollo de tus ideas directamente sobre la rama *main* o la que sea tu rama **principal/publicación/producción**. La idea es que cuando estes planeado algo nuevo crees una rama nueva o si ya tienes una y está en la misma cosa, seleccionala como la rama activa.

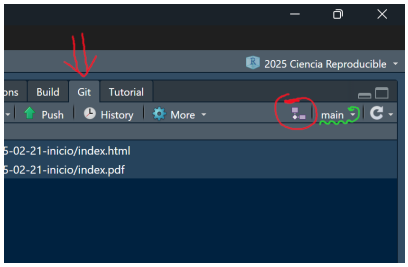

Figura 3: Crear y usar una nueva rama de trabajo

Asegurate de que la rama del nuevo desarrollo sea la activa. Como recordarás, la publicación de tu **blog** ocurre a través de *Netlify* **exclusivamenre con lo que pongas en la rama**

main.. Así que todo lo que hagas en la nueva rama no pasará a publicación en tu **blog**, sólo lo podrás ver localmente cuando hagas *render*, será una especie de borrador. Cuando todo esté a tu gusto y ya lo quieras publicar tienes que combinar el contenido de la rama en la que has estado trabajando con la rama *main*. Para hacerlo hay que seguir estos pasos:

1. Cambia el espacio de trabajo en *git* a la rama *main*.
2. En la pestaña de **terminal** de *RStudio*. Indicale a *git* que jale para combinar *la_rama_alterna_con_mi_trabajo*. Escribe para esto los siguientes comandos de *git* en la pestaña de **terminal**:

```
git merge la_rama_alterna_con_mi_trabajo
```

Como podrás imaginar, el proceso requiere verificar muchas cosas, lo cual hace *git* por ti. Normalmente y si te portaste bien, la fusión de las ramas debe ocurrir sin tropiezos. Si hubiera dificultades, *git* te lo hará saber. Basicamente lo que hará es indicarte que hay cambios contradictorios. Es decir, hiciste cambios tú, al mismo archivo en las dos ramas, así que no se puede decidir cuál es el cambio que hay que conservar sin preguntarte a ti que es lo que prefieres. En tal caso, veras los archivos inconsistentes marcados con una etiqueta naranja y tendrás que revisarlos para que tú resuelvas directamente lo conducente. Puede ser que haya archivos que en realidad no cambiaste tú directamente, como serían los que se producen automáticamente en **__site** y que sólo arrastrán las consecuencias de tus cambios. De esos no te preocupes, se renovarán cuando hayas resuelto los importantes, que seguramente son los que tienes en **posts**, o algún detallito que arreglaste en los archivos general en la raíz de tu proyecto.