

Creación de páginas web estáticas con el paquete blogdown

José Carlos Soage

10 de mayo de 2021

Contacto y otros enlaces

 jsoage@uvigo.es

 r-coder.com

 r-charts.com

 [@RCoderWeb](https://twitter.com/RCoderWeb)

 [@R-CoderDotCom](https://github.com/R-CoderDotCom)

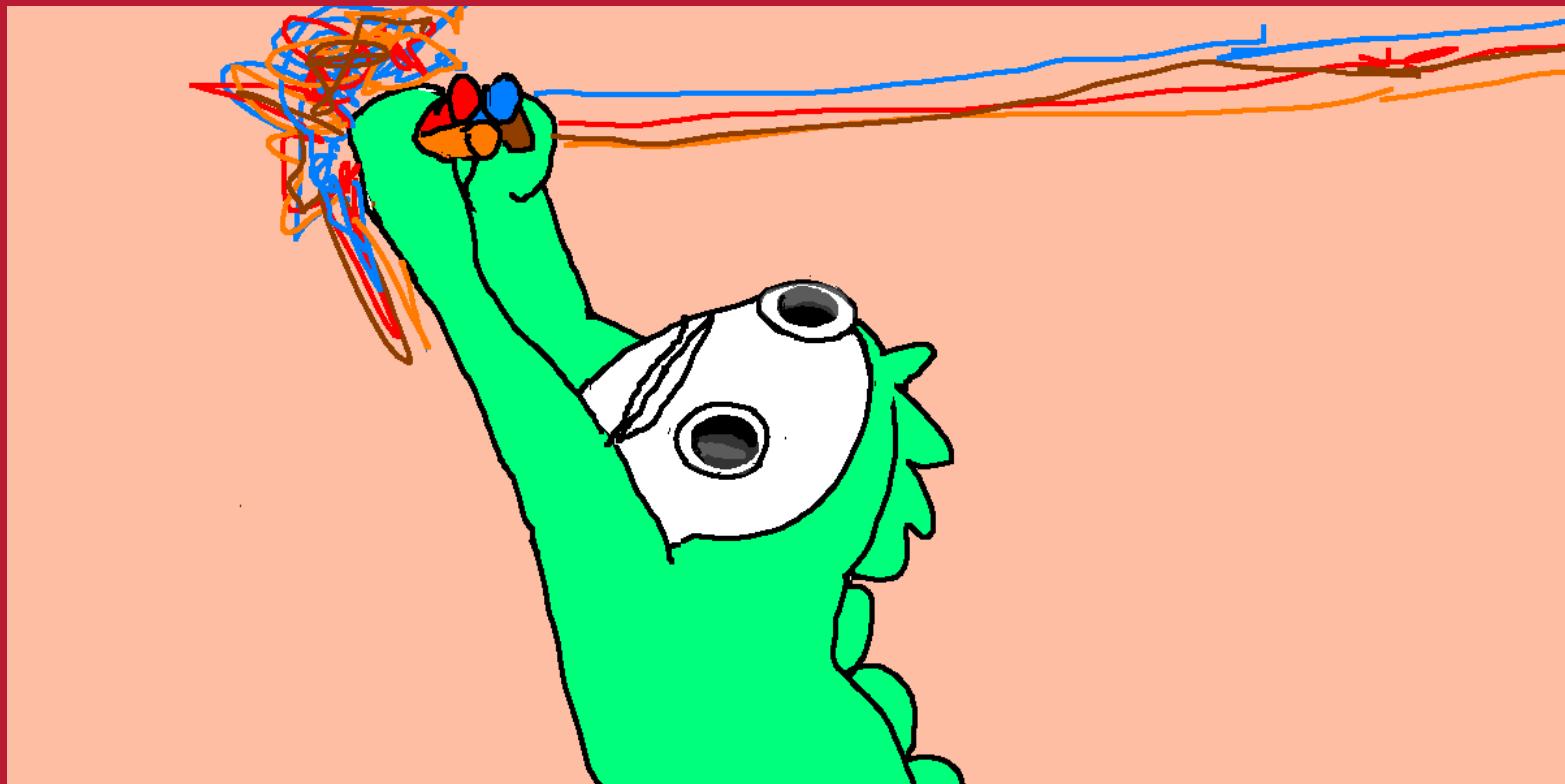
Objetivo del curso

En este curso aprenderemos a crear una **página web personal** a nuestro gusto y a publicarla en un servidor de manera gratuita.



¿Cómo seguir este curso?

Mira y aprende



¿Cómo seguir este curso?

Sigue mis pasos y pregunta si tienes dudas



Índice de contenidos

- 1. Introducción a los generadores de páginas web estáticas**
- 2. Creación de páginas web con blogdown**
- 3. Introducción al desarrollo web**
- 4. HUGO: aprendiendo a personalizar y crear plantillas**
- 5. Publicación de la web**

1. Introducción a los generadores de páginas webs estáticas

¿Qué es una página web estática? ¿Y dinámica?

Web estática

Una página web estática es aquella en la que **los archivos que la conforman son fijos** (estáticos). Estos archivos, por lo general, serán archivos HTML, hojas de estilos (CSS, SASS, ...), imágenes, tipografías, ...

Web dinámica

Una página dinámica es aquella en la que **los usuarios pueden crear nuevas páginas o contenido** (si se permite), tales como nuevas entradas o nuevos perfiles de usuarios. Este tipo de páginas dependen de una base de datos y necesitan de un *backend* escrito en PHP u otro lenguaje server-side para generar el contenido de manera dinámica.

Ejemplos: Wordpress, Joomla.

Webs estáticas VS dinámicas

Estáticas	Dinámicas
✓ Más rápidas	✗ Más lentas
✓ Más seguras / estables	✗ Son hakeables / inestables
✓ Se pueden publicar gratuitamente (GitHub Pages, Netlify, ...)	✗ Necesitan un servidor de pago
✗ No hay sistema de usuarios ni de comentarios ¹	✓ Se pueden crear usuarios y publicar comentarios
✗ Es complicado (pero posible) implementar búsquedas	✓ Se pueden realizar búsquedas

| No confundir páginas estáticas con páginas sin interactividad.

¹ Salvo que se utilice **Disqus** u otro sistema similar para los comentarios

Creando una página web estática

Si queremos crear un página web estática podemos crearla desde cero o utilizar una plantilla HTML hecha por otro desarrollador y cambiar el contenido y/o las imágenes.

Este proceso puede ser válido para crear una única página, pero si queremos escribir artículos o crear más contenido **tendremos varios problemas:**

- Estaremos repitiendo código una y otra vez.
- Escribir todo el contenido en HTML no es amigable.
- Si cometemos algún error al escribir en HTML será difícil encontrarlo.
- Si cometemos un error sistemático tendremos que corregirlo en todos los archivos.

En estas circunstancias **conviene utilizar un generador de páginas estáticas.**

¿Qué es y por qué utilizar un generador de webs estáticas?

Un generador permite **crear páginas basándose en plantillas** previamente desarrolladas **sin tener que tocar el código**¹.

De esta forma, si queremos crear una nueva página, en lugar de crear todo el código, podremos escribir el contenido y se le aplicará la plantilla que queramos.

Esto nos **ahorrará tiempo y evitará errores**, ya que el código de la plantilla estará solo en el archivo correspondiente.

[1] Salvo que queramos modificar la estructura de la plantilla.

¿Qué es HUGO?

HUGO es el **generador de webs estáticas** más rápido que existe.

- Está escrito en un lenguaje llamado GO (también conocido como Golang).
- Creado por Bjørn Erik Pedersen (@bep), Steve Francia (@spf13) y otros colaboradores.
- Es **open-source**, completamente gratuito.
- Permite agregar variables, condicionales y funciones a archivos HTML.
- Las variables que se pasan a los archivos HTML **vienen de archivos Markdown** (encabezado YAML + contenido) y **.yaml** o **.toml**.



HUGO utiliza Markdown, YAML y/o TOML

En el caso de HUGO, tendremos por una parte plantillas HTML y por otra archivos Markdown, YAML y/o TOML:

- **Markdown** es un lenguaje de marcado ligero creado por John Gruber que, a través de **Pandoc**, puede convertirse a HTML, PDF, Word, ...
- **YAML** (Yet Another Markup Language) es un formato de serialización de datos legible por humanos. El encabezado de un archivo Markdown es un encabezado YAML.
- **TOML** (Tom's Obvious Minimal Language) es un formato de archivo de configuración que es fácil de leer debido a la semántica obvia que pretende ser "mínima".

| YAML y TOML son prácticamente lo mismo. Cambia un poco el formato pero cumplen la misma función.

HUGO utiliza Markdown, YAML y/o TOML

Recordemos cómo es un archivo Markdown. Se compone de un **encabezado YAML** y del **contenido**:

```
1 - ---
2 title: "Ejemplo"
3 subtitle: "Subtítulo del ejemplo"
4 author: "José Carlos"
5 output: html_document
6 ---
7
8 ## Markdown
9
10 This is a Markdown document. Markdown is a simple formatting syntax for authoring HTML,
11 PDF, and MS Word documents.
```

ENCABEZADO YAML

CONTENIDO

Compilando un archivo Markdown como HTML

Cuando compilemos se generará un archivo HTML con las variables del encabezado YAML y el contenido del archivo:

Ejemplo

Subtítulo del ejemplo

José Carlos

Markdown

This is a Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents.

¿Por qué no aprovechar la conversión de Markdown a HTML para crear páginas web más complejas?

Compilando un archivo Markdown como HTML

Archivo Markdown

```
1+ ---  
2 title: 'Ridgeline en ggplot2 con ggridges'  
3 seotitle: ''  
4 description: "El paquete ggridges permite crear gráficos ridgeline (jo  
5 image: "es/distribucion/ggridges_files/figure-html/ggridges-tema.png"  
6  
7 date: '2021-04-24'  
8 lastmod: "'r Sys.time()'"  
9  
10 categories: ["Ridgeline"]  
11 tags: ["ggplot2"]  
12 keywords: ["distribution"]  
13  
14 package: "ggridges"  
15 link: "https://github.com/wilkelab/ggridges"  
16 author: "Claus O. Wilke"  
17  
18 # Search  
19 madetype: "Distribución"  
20 madewith: "ggplot2"  
21  
22 # Translation  
23 translationKey: "ggridges"  
24+ ---  
25  
26+ ```{r, echo = FALSE, eval = TRUE}  
27 knitr::opts_chunk$set(message = FALSE, dev='png', message=FALSE, warni  
28  
29 knitr::opts_chunk$set(  
30 fig.process = function(filename) {
```

Resultado

The diagram illustrates the compilation process from an R Markdown file to a final HTML output. On the left, the 'Archivo Markdown' section shows the source code with several sections highlighted by red boxes. Red arrows point from these highlighted sections to specific components in the resulting HTML page on the right.

Resultado:

- Header:** Ridgeline en ggplot2 con ggridges
- Librería:** ggridges
- Autor principal:** Claus O. Wilke
- Conjunto de datos de muestra:** A table showing a subset of the diamonds dataset from the ggplot2 package.

color	depth
E	61.5
E	59.8
E	56.9
I	62.4

Code Block:

```
# install.packages("ggplot2")
library(ggplot2)
```

Ejemplos de páginas webs creadas con HUGO

Veamos unos cuantos ejemplos de páginas webs hechas con el generador HUGO:

<https://gohugo.io/showcase/>



blogdown

Un paquete de R para crear páginas web usando
R Markdown y Hugo



blogdown no implica crear un blog

<https://yihui.name/en/2017/06/netlify-instead-of-github-pages/>

I F YOU DO NOT HAVE A PERSONAL WEBSITE YET, CONGRATULATIONS! I THINK NOW
is the best time to create a website.¹ I started blogging in 2005, and benefited a lot

1. However, I do think it is too late to start a blog, because the whole society has been distracted by social media, which favors short, quick, debatable, misleading, and even fake messages. I feel there are much fewer people reading or commenting on blog posts now. That said, writing blog posts is still helpful, even if nobody reads your posts. ↩

Alternativas

Existen múltiples **alternativas similares a HUGO**, tales como:

- Jekyll
- Gatsby
- Ghost

Otro tipo de alternativas basadas en R Markdown (librerías de R para otros usos):

- **distill**: "formato de publicación web optimizado para comunicación técnica y científica".
- **pkgdown**: páginas web para paquetes de R.
- **bookdown**: para escribir libros HTML, PDF, ePub y Kindle.
- **pagedown**: permite paginar la salida HTML de un R Markdown para generar PDFs listos para imprimir.

¿Todavía no has instalado R ni RStudio?

- Descargar R: <https://cran.r-project.org/bin/>
- Descargar RStudio: <https://rstudio.com/products/rstudio/download/>



¿Alguna duda? Consulta mi tutorial: <https://r-coder.com/instalar-r/>

Instalar blogdown

Desde CRAN

```
install.packages("blogdown")
```

Desde GitHub

```
if (!requireNamespace("devtools")) install.packages("devtools")
devtools::install_github("rstudio/blogdown")
```

Instalamos HUGO vía blogdown

Instalar HUGO es muy sencillo:

```
blogdown::install_hugo()  
  
# Equivalente:  
library(blogdown)  
install_hugo()
```

También podemos comprobar cuál es la **versión actual y actualizar HUGO** de manera sencilla:

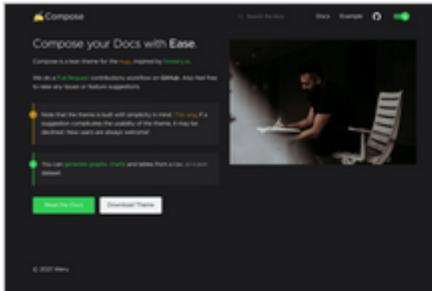
```
blogdown::hugo_version() # Versión actual  
blogdown::update_hugo() # Actualizar
```

2. Creación de páginas web con blogdown

Selección de un tema

Existen multitud de temas creados por otros usuarios. Los temas oficiales se pueden encontrar en el siguiente enlace:

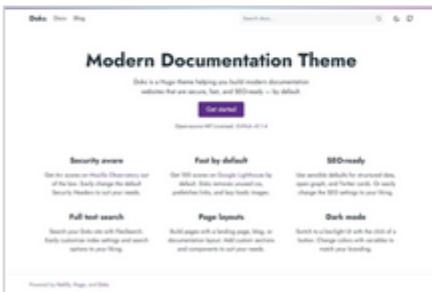
<https://themes.gohugo.io/>



Compose

Zen

Color Your World



Doks

Docsy

Geekdoc

Selección de un tema

Los temas que se encuentran en la diapositiva anterior pasan una serie de requisitos para entrar en la lista. De todas formas, **si buscamos en GitHub podemos encontrar más temas**, pero quizás no todos estén completos o sean estables.

Existen temas de:

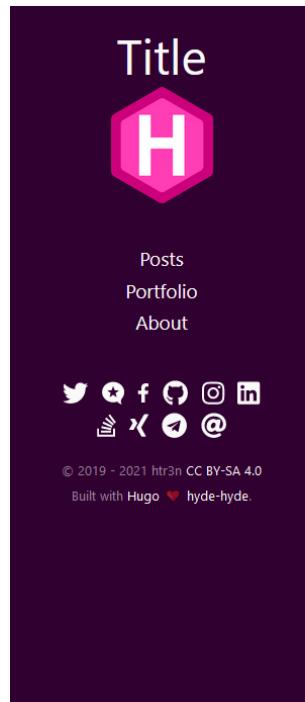
- Documentación
- Páginas personales
- Landing pages
- CV
- Cursos
- ...

Cada tema tendrá sus propias funcionalidades. Si te gusta un tema que no incorpora cierta funcionalidad que necesitas tendrás que buscar otro tema o implementarla tu mismo. Aprenderemos a modificar un tema en el **Tema 4** de este curso.

El tema Hyde

Un primer ejemplo

Vamos a crear una web estática muy sencilla basándonos en una plantilla llamada Hyde, que por defecto es como se muestra en la siguiente imagen:



Markdown Syntax Guide

Mar 11, 2019

Sample article showcasing basic Markdown syntax and formatting for HTML elements. ... ➔

Rich Content

Mar 10, 2019

A brief description of Hugo Shortcodes ... ➔

Placeholder Text

Mar 9, 2019

Lorem Ipsum Dolor Si Amet ... ➔

Math Typesetting

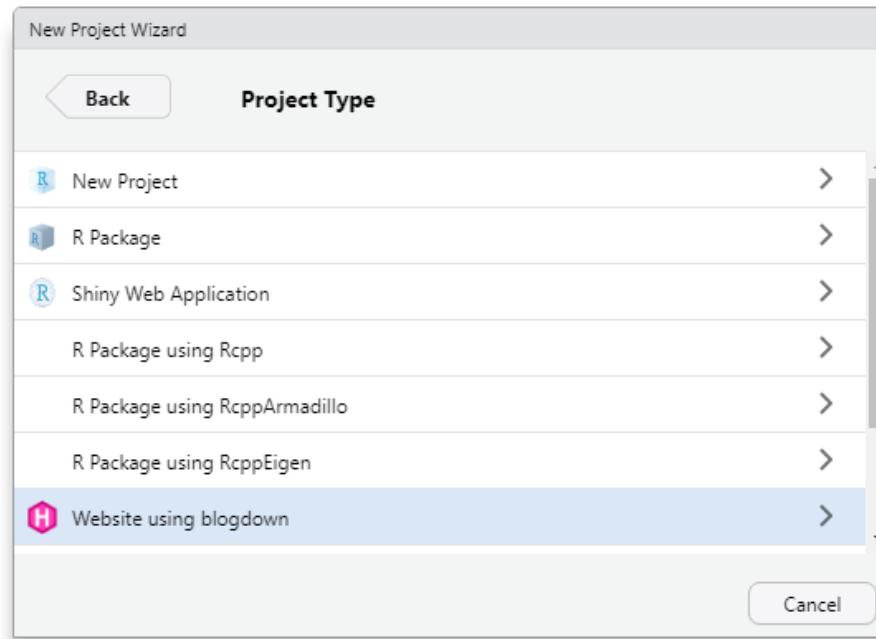
Mar 8, 2019

A brief guide to setup KaTeX ... ➔

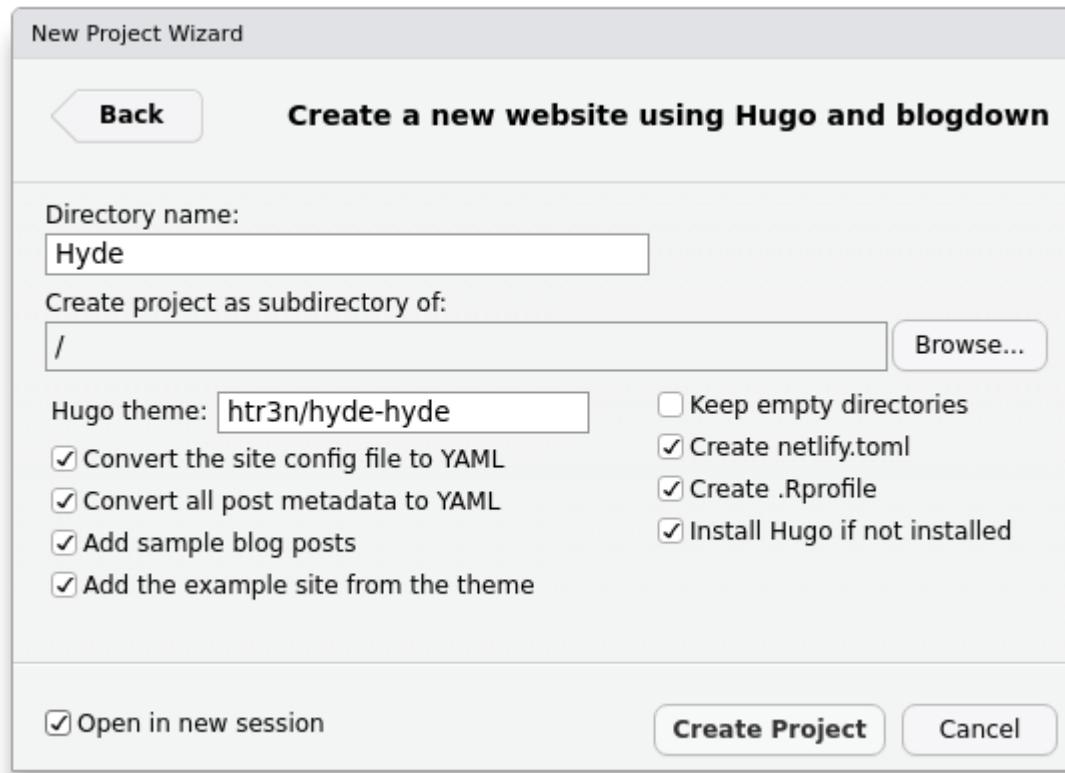
Creamos un nuevo proyecto en RStudio

Se recomienda utilizar el **RStudio IDE**, pero no es un requisito indispensable.

- Crea un nuevo proyecto desde `File` → `New Project` → `New Directory` → `Website using blogdown`



Instala el tema Hyde



Ten en cuenta que si no has instalado HUGO se instalará automáticamente.

Alternativa

Creamos un proyecto desde `File` → `New Project` → `New Directory` → `New Project` e instalamos un tema:

- Tema por defecto:

```
blogdown::new_site()
```

- Otro tema:

```
blogdown::new_site(theme = "htr3n/hyde-hyde",
                    theme_example = TRUE) # Con todos los ejemplos
```

En el argumento `theme` escribiremos el nombre de la cuenta de GitHub y del repositorio con la plantilla.

Antes de continuar actualiza las herramientas de compilación del proyecto

Vea `Tools` → `Project Options` y desmarca las opciones. Esto es una recomendación del autor de `blogdown`:

<https://bookdown.org/yihui/blogdown/rstudio-ide.html#fig:project-options>

The screenshot shows the 'Project Options' dialog box in RStudio. The left sidebar has icons for General, Code Editing, Sweave, Build Tools (which is selected and highlighted in blue), Git/SVN, and Packrat. The main area shows 'Project build tools: Website' and 'Site directory: (Project Root)'. Below these are two unchecked checkboxes: 'Preview site after building' and 'Re-knit current preview when supporting files change'. A note at the bottom states: 'Supporting files include Rmd partials, R scripts, YAML config files, etc.'

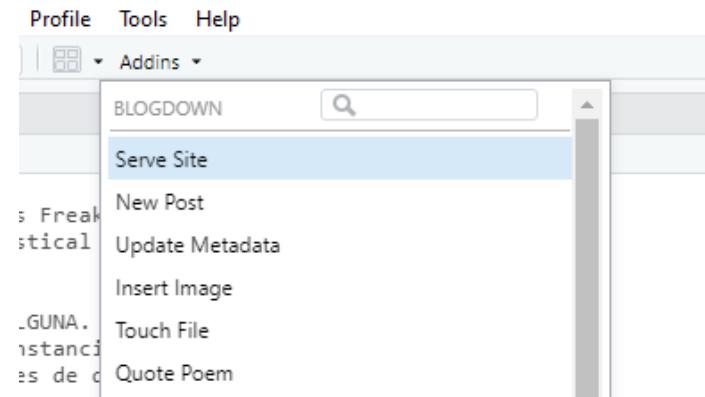
Servir el sitio web

Dentro del proyecto se habrán creado los archivos correspondientes a la web. Para visualizarla puedes:

- Ejecutar la función `serve_site` desde la consola.

```
blogdown::serve_site()
```

- Usar el Addin de RStudio.



No ejecutes varias veces `blogdown::serve_site` a la vez.

Parar el servidor

Cuando quieras parar el servidor puedes:

- Ejecutar la función `stop_server` desde la consola.

```
blogdown::stop_server()
```

- Cerrar el proyecto o reiniciar RStudio con `.rs.restartR`.

```
.rs.restartR()
```

LiveReload

La creación de páginas webs con **blogdown** soporta **LiveReload**, es decir, los cambios que hagas en los archivos se actualizarán en vivo.

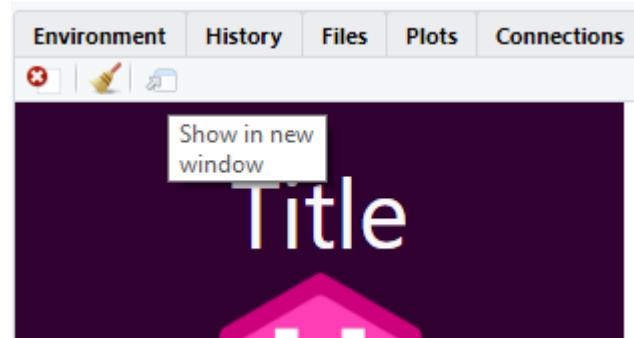
1. Modifica un archivo y guárdalo.
2. Mientras los cambios se compilan la consola se bloqueará.

Comentario: esta característica, aunque útil, puede ocasionar problemas si se comete algún error al modificar los archivos base de la plantilla. Es aconsejable ejecutar `blogdown::build_site(local = TRUE)` cada cierto tiempo.

Abrir la web en el navegador

Por defecto, al ejecutar `serve_site` nuestro sitio web se abrirá en el *Viewer pane* de RStudio. Es recomendable abrirlo con un navegador, para ello puedes:

- Pulsar el botón de abrir en nueva pestaña.



- Abrir el navegador y (si has cerrado o parado el servidor del proyecto anterior) escribir:

```
localhost:4321
```

Estructura de la plantilla

Podrás ver que se han creado una serie de carpetas junto con el proyecto.

Esta estructura puede variar ligeramente dependiendo del tema instalado.

Por ejemplo, hay proyectos que incluirán una carpeta llamada **data**.

Nombre	Tipo
content	Carpeta de archivos
layouts	Carpeta de archivos
public	Carpeta de archivos
R	Carpeta de archivos
resources	Carpeta de archivos
static	Carpeta de archivos
themes	Carpeta de archivos
.Rprofile	Archivo RPROFILE
config.yaml	Archivo YAML
Hyde	R Project
index	Archivo RMD
netlify.toml	Archivo TOML



Directorio y archivos principales

Dentro de cada carpeta podemos encontrar otras carpetas con más carpetas o archivos. De momento vamos a centrarnos en las carpetas `content`, `static` y en el archivo `config.yaml`.

```
.
├── content
│   ├── portfolio
│   ├── posts
│   └── about.md
├── layouts
├── public
├── R
├── resources
└── static
    ├── img
    └── themes
└── config.yaml
├── index.Rmd
└── netlify.toml
```

El archivo config.yaml: configuración global de la web

Buscamos lo siguiente y lo editamos:

```
languageCode: en
title: Title
```

Ahora podemos ir a `static` → `img` y pegamos una imagen.

```
params:
  author: Author
  authorimage: /img/hugo.png
```

Podemos ir modificando otros parámetros, como nuestras redes sociales o los textos ancla (enlaces) del menú.

El archivo config.yaml: configuración global de la web

José
Carlos



[Artículos](#)
[Portafolio](#)
[Acerca](#)



© 2019 - 2021 htr3n CC BY-SA 4.0
Built with Hugo ❤️ hyde-hyde.

Creating a New Theme

Sep 28, 2014

Introduction This tutorial will show you how to create a simple theme in Hugo. I assume that you are familiar with HTML, the bash command line, and that you are comfortable using Markdown to format content. I'll explain how Hugo uses templates and how you can organize your templates to create a theme. I won't cover using CSS to style your theme. We'll start with creating a new site with a very basic template. ... ➔

Migrate to Hugo from Jekyll

Mar 10, 2014

Move static content to static Jekyll has a rule that any directory not starting with _ will be copied as-is to the _site output. Hugo keeps all static content under static. You should therefore move it all there. With Jekyll, something that looked like ▶ <root>/ ▶ images/ logo.png should become ▶ <root>/ ▶

La carpeta **content**: organización del contenido

La carpeta **content** permite **organizar el contenido** de nuestra página web. HUGO asume (por defecto) que la estructura de carpetas utilizada es la misma que quieras que tenga la web.

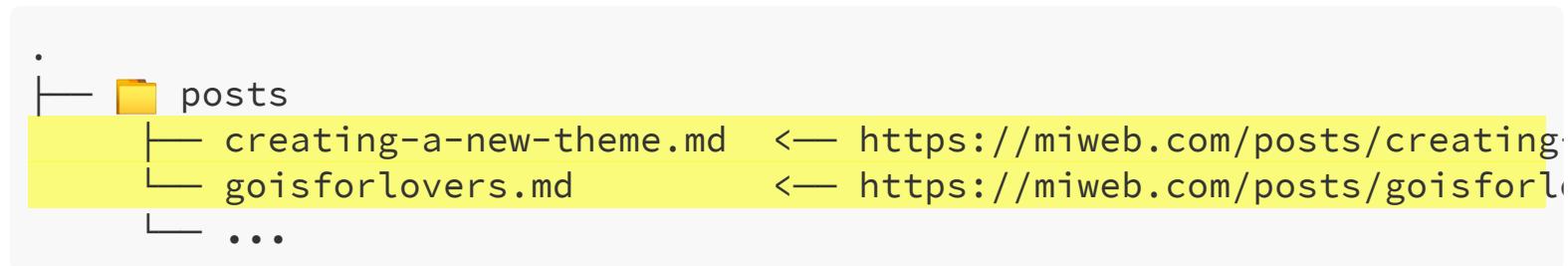
```
.
├── content
    ├── portfolio  ← https://miweb.com/portfolio/
    ├── posts      ← https://miweb.com/posts/
    └── about.md   ← https://miweb.com/about/
```

portfolio y **posts** son carpetas porque tendremos varios archivos dentro de ellas. Sin embargo, solo tendremos un **about**, por lo que podemos incluir directamente el archivo (R)Markdown, o crear una carpeta llamada **about** y dentro de ella crear el archivo pero llamado **index.md**.

La carpeta **content**: organización del contenido

Dentro de cada carpeta podemos tener más carpetas o los archivos que componen el contenido, formando, por defecto, las URL de forma anidada.

Si creamos un archivo llamado `_index.md` dentro de esta carpeta (lo haremos después) definirá el contenido o las variables de la página de la carpeta (por lo general una lista de artículos):



La carpeta **content**: creando contenido

Vamos a `content` → `about.md` y lo editamos.

Si ya conoces R Markdown la sintaxis es la misma. De hecho, puedes borrar el archivo y sustituirlo por un R Markdown (.Rmd) del mismo nombre y ejecutar código R.



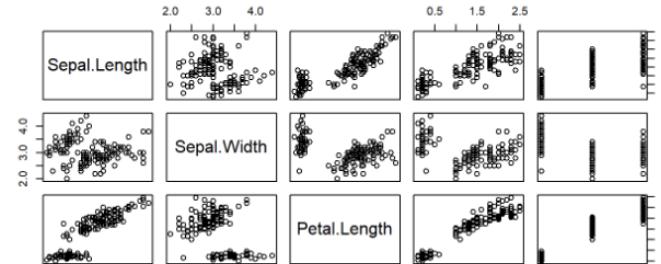
Sobre mi

Apr 9, 2021

1 min read

Hugo is a static site engine written in Go.

```
plot(iris)
```



La carpeta **content**: creando contenido

Vamos a `content` → `posts`. Creamos un nuevo archivo R Markdown y lo guardamos.



Post de prueba

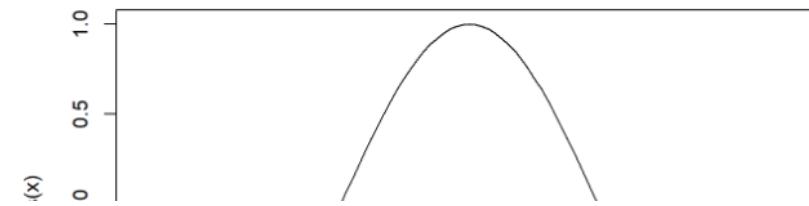
Dec 1, 2020 in `R`

→ `r markdown`

1 min read

R Markdown

```
curve(cos, -4, 4)
```



Lo mismo se aplica para los contenidos dentro de `content` → `portfolio`.

La carpeta **content**: creando contenido

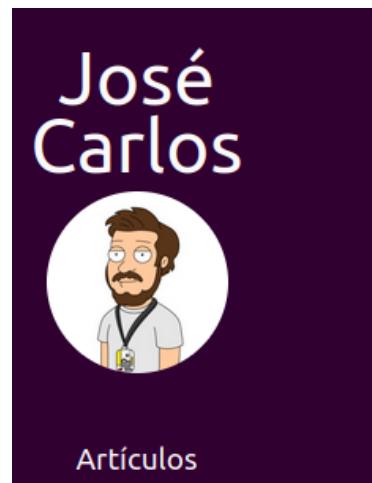
Ahora creamos un archivo dentro de **posts** llamado **_index.md** y dentro de él ponemos:

```
---
```

```
title: Artículos
```

```
--
```

Esto cambiará el título de la sección.



Artículos



- [Creating a New Theme](#)
- [Migrate to Hugo from Jekyll](#)
- [Hello R Markdown](#)
- [Post de prueba](#)
- [\(Hu\)go Template Primer](#)
- [Getting Started with Hugo](#)

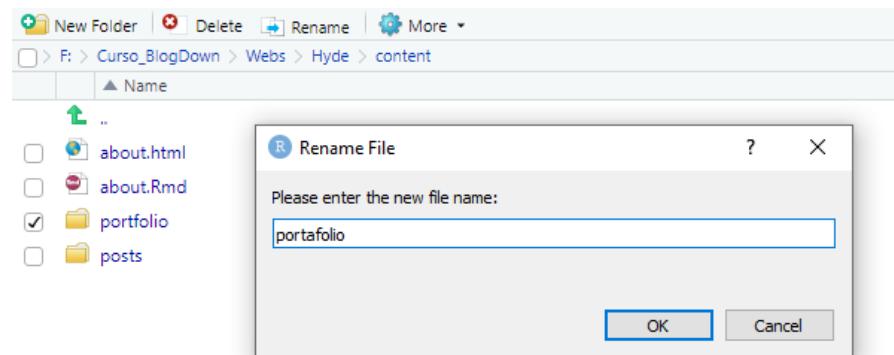
Sep 28, 2014
Mar 10, 2014
Dec 1, 2020
Dec 1, 2020
Apr 2, 2014
Apr 2, 2014

Traducción de las URLs

Si te fijas en la dirección URL los contenidos están dentro de `/post/`, el portafolio dentro de `/portfolio/` y la página de información en `/about/`. Estas URL se corresponden con el nombre de las carpetas y del archivo `about.md`.

Si las cambiamos cambiará la dirección URL, por lo que también tendremos que cambiar los enlaces del menú dentro de `config.yaml`.

También hay que cambiar los nombres de otras carpetas para que todo siga igual (dependiendo del tema). [Ir a la sección](#).



Comprobando que todo está correcto

```
> blogdown::check_site()
– Running a series of automated checks for your blogdown website project
-----
○ A successful check looks like this.
● [TODO] A check that needs your attention looks like this.
| Let's check out your blogdown site!
-----
– Checking config.yaml
| Checking "baseURL" setting for Hugo...
● [TODO] Set "baseURL" to "/" if you do not yet have a domain.
| Checking "ignoreFiles" setting for Hugo...
● [TODO] Set "ignoreFiles" to ["\\.Rmd$", "\\.Rmarkdown$", "_cache$",
| Checking setting for Hugo's Markdown renderer...
| You are using the Markdown renderer 'goldmark'.
● [TODO] Allow goldmark to render raw HTML by adding this setting to config.yaml
markup:
  goldmark:
    renderer:
      unsafe: true
```

Comprobando que todo está correcto

```
> blogdown::check_config()
– Checking config.yaml
| Checking "baseURL" setting for Hugo...
● [TODO] Update "baseURL" to your actual URL when ready to publish.
| Checking "ignoreFiles" setting for Hugo...
○ "ignoreFiles" looks good – nothing to do here!
| Checking setting for Hugo's Markdown renderer...
○ All set! Found the "unsafe" setting for goldmark.
– Check complete: config.yaml
```

Montando el sitio para la publicación

La función `build_site` se utiliza para construir los archivos definitivos. Estos archivos aparecerán dentro de la carpeta `public`. El contenido de esta carpeta podremos subirlo a nuestro servidor (Tema 5) para visualizar nuestra web. **Es recomendable borrar todas las carpetas dentro de `public` antes de construir la web.**

```
> blogdown::build_site(local = FALSE)
Start building sites ...
```

	EN
Pages	56
Paginator pages	0
Non-page files	5
Static files	24
Processed images	0
Aliases	1
Sitemaps	1
Cleaned	0

Total in 8174 ms

El tema Academic

¿Qué es el tema Academic?

El tema Academic forma parte de los temas de HUGO de la empresa **Wowchemy** y permite crear una web con HUGO bastante personalizable y completa sin tener que tocar ni una sola línea de HTML.

- Diseñada para investigadores
- No es necesario saber código
- Tiene un buscador incorporado
- Permite filtrar publicaciones
- Permite crear cursos
- Permite crear diapositivas
- Multilenguaje
- Modo noche

- Recientemente se ha convertido en un tema freemium
- La plantilla es muy compleja si se quiere editar en profundidad
- La enorme cantidad de opciones que trae puede ser abrumadora al principio

Documentación oficial: <https://wowchemy.com/docs/>

Precaución

El tema es muy completo pero a la vez internamente es muy complejo. Esto ha provocado que sea inestable. **En este curso utilizaremos una versión estable que he preparado**, pero si instalas la nueva versión encontrarás una estructura diferente. Además, Yihui Xie, creador de **blogdown**, recomienda no actualizar nunca el tema y comenta que en RStudio están trabajando para crear un tema estable similar a Academic.



Yihui Xie @xieyihui · 14h

Replies to @xieyihui @RCoderWeb and @delaBJL

Before that, my advice is to pin both the Hugo version (run `blogdown::check_site()`) and the theme. When you have a working combination, don't upgrade. Our tears on the academic theme are probably no less than yours, so we are highly motivated to develop our own themes.



...



Yihui Xie @xieyihui · 14h

Replies to @RCoderWeb and @delaBJL

Rest assured that we will provide simple, beautiful, and stable (!) themes created and maintained by ourselves. I started developing one last year but have been constantly drowned in numerous other things. Another R Markdown team member also developed one. Stay tuned.



...

¿Qué vamos a hacer?

Vamos a crear una página web como la siguiente, con biografía, experiencia laboral, artículos, publicaciones científicas, cursos y contacto. En inglés y castellano.

Todo ello lo haremos a partir de archivos **.md**, **.Rmd** y **.toml**.

Mi nombre



Inicio Posts Publicaciones Cursos Contacto 🔍 🌙 🌎

Mi nombre

Profesor de Estadística
Universidad de Vigo

[✉](#) [🐦](#) [🎓](#) [🐙](#) [LinkedIn](#)

Sobre mi

Soy profesor de Estadística en la Universidad de Vigo. Mis intereses son..... Mis investigaciones tratan sobre..... Soy jefe de..... También desarrollo.....

Además, doy clase de..... y en mi tiempo libre.....

[⬇ Descarga mi CV.](#)

Intereses

- Estadística no paramétrica
- Procesos estocásticos
- Series temporales

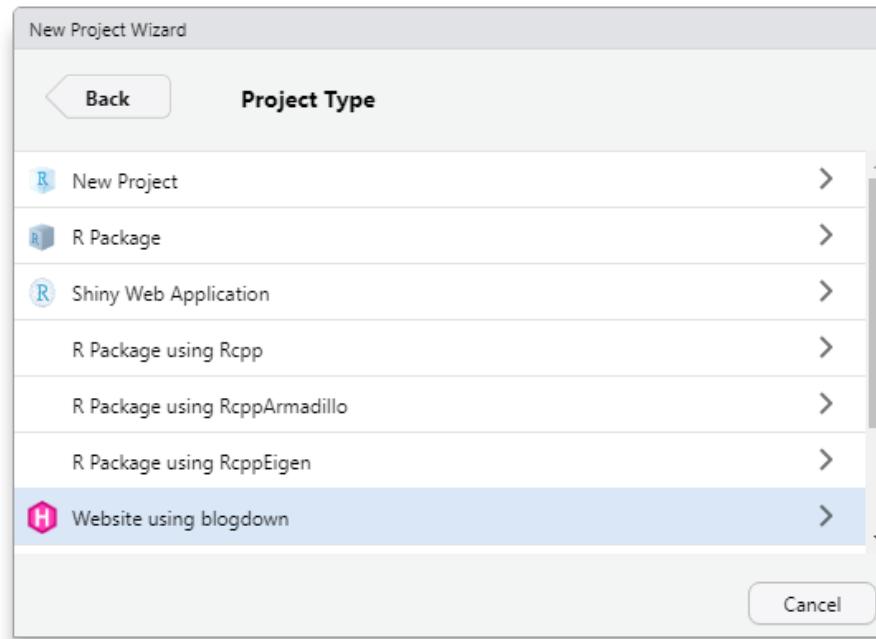
Educación

- 🎓 Doctorado en Estadística, 2012
Universidad de Vigo
- 🎓 Máster en Estadística, 2009
Universidad de Vigo
- 🎓 Grado en Matemáticas, 2008
Universidad de Santiago de Compostela

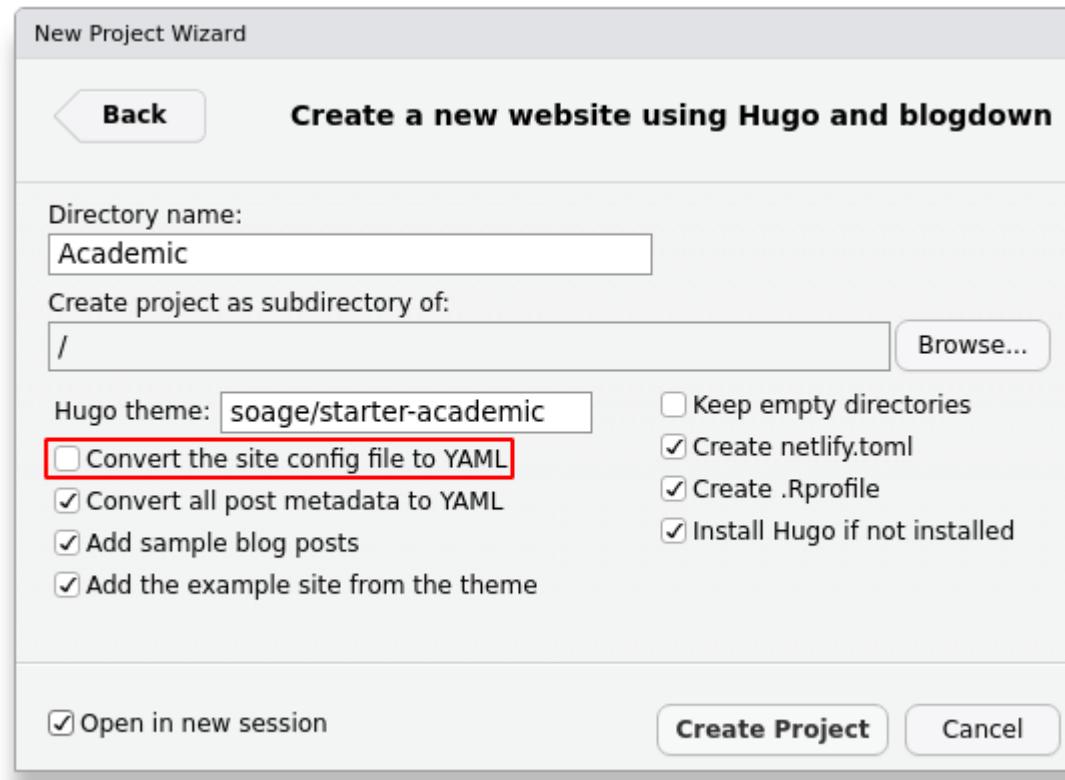
Creamos un nuevo proyecto en RStudio

Se recomienda utilizar el **RStudio IDE**, pero no es un requisito indispensable.

- Crea un nuevo proyecto desde `File` → `New Project` → `New Directory` → `Website using blogdown`



Instala el tema Academic



Ten en cuenta que si no has instalado HUGO se instalará automáticamente.

Alternativa

Creamos un proyecto desde `File` → `New Project` → `New Directory` → `New Project` e instalamos el tema:

- En este caso vamos a instalar el tema con el archivo `config.toml` en lugar de `config.yaml`:

```
blogdown::new_site(theme = "soage/starter-academic",
                    format = "toml",
                    theme_example = TRUE) # Con todos los ejemplos
```

En el argumento `theme` escribiremos el nombre de la cuenta de GitHub y del repositorio con la plantilla.

Antes de continuar actualiza las herramientas de compilación del proyecto

Vea **Tools** → **Project Options** y desmarca las opciones.

Project Options

Project build tools: Website

Site directory:
(Project Root) [Browse...](#)

Preview site after building

Re-knit current preview when supporting files change

Supporting files include Rmd partials, R scripts, YAML config files, etc.

R General
Code Editing
Sweave
Build Tools
Git/SVN
Packrat

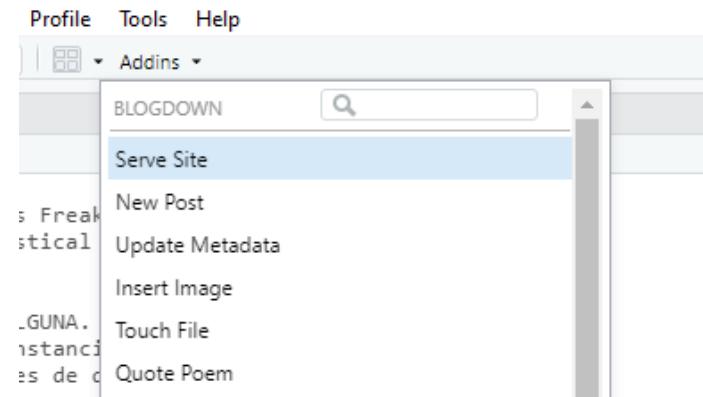
Servir el sitio web

Como vimos antes, dentro del proyecto se habrán creado los archivos correspondientes a la web y para visualizarla puedes:

- Ejecutar `serve_site` desde la consola:

```
blogdown::serve_site()
```

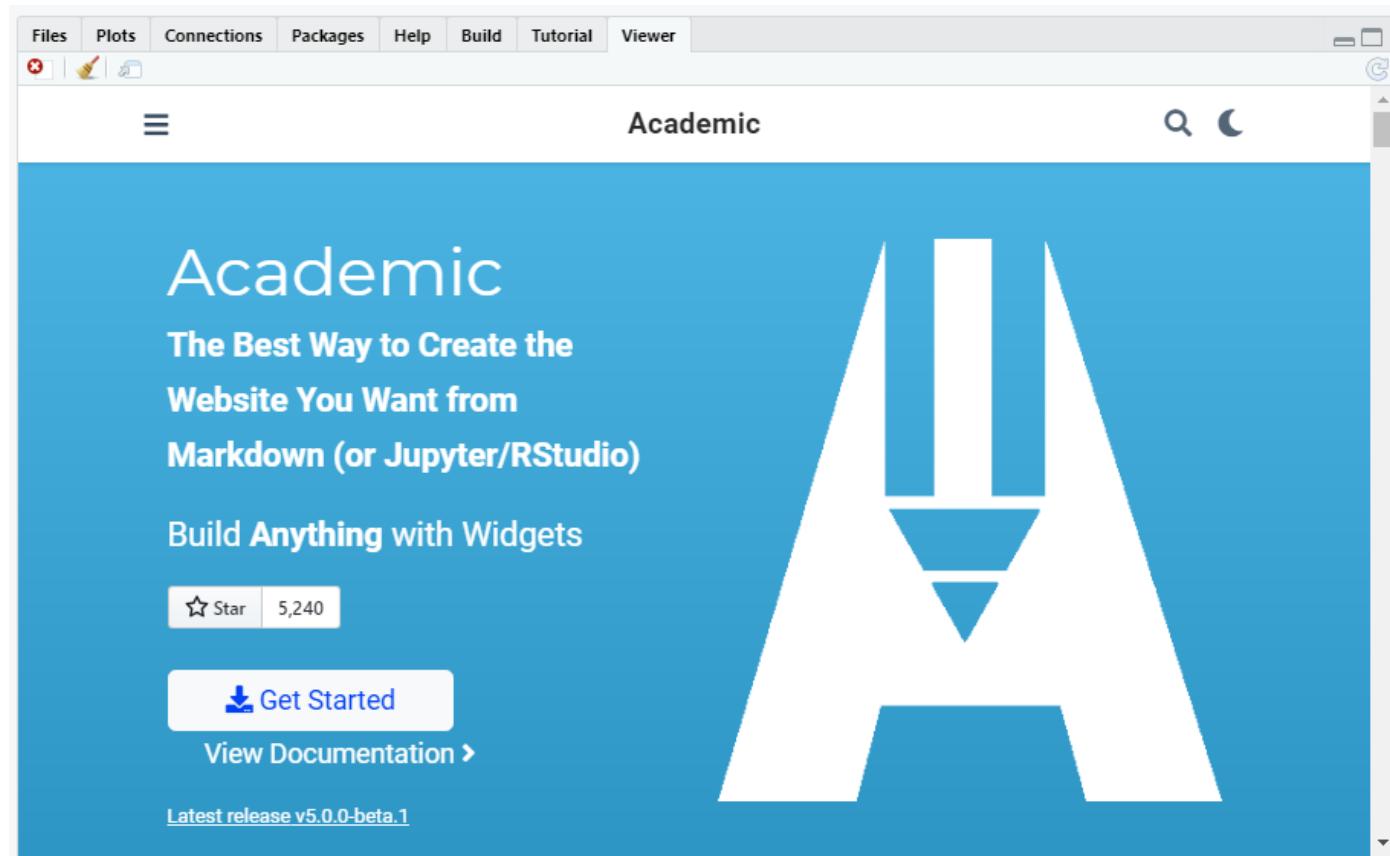
- Usar el Addin de RStudio:



No ejecutes varias veces `blogdown::serve_site` a la vez.

¡Ya tenemos nuestra plantilla lista!

Ahora podrás ver lo siguiente en el *Viewer Pane* de RStudio



Parar el servidor

Para parar el servidor:

- Ejecuta lo siguiente desde la consola:

```
blogdown::stop_server()
```

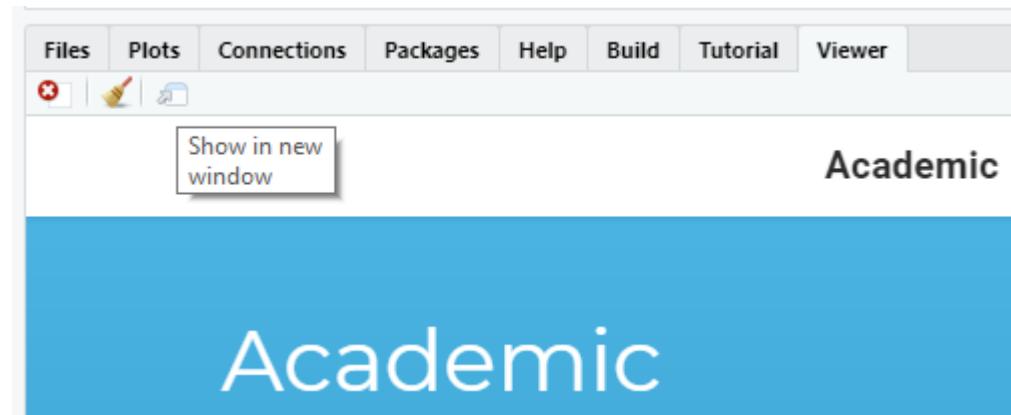
- Cierra el proyecto o reinicia RStudio con:

```
.rs.restartR()
```

Abrir la web en el navegador

Por defecto, al ejecutar `serve_site` nuestro sitio web se abrirá en el *Viewer Pane* de RStudio. Es recomendable abrirla con un navegador, para ello puedes:

- Pulsar el botón de abrir en nueva pestaña.



- Abrir el navegador y escribir:

```
localhost:4321
```

Estructura de la plantilla

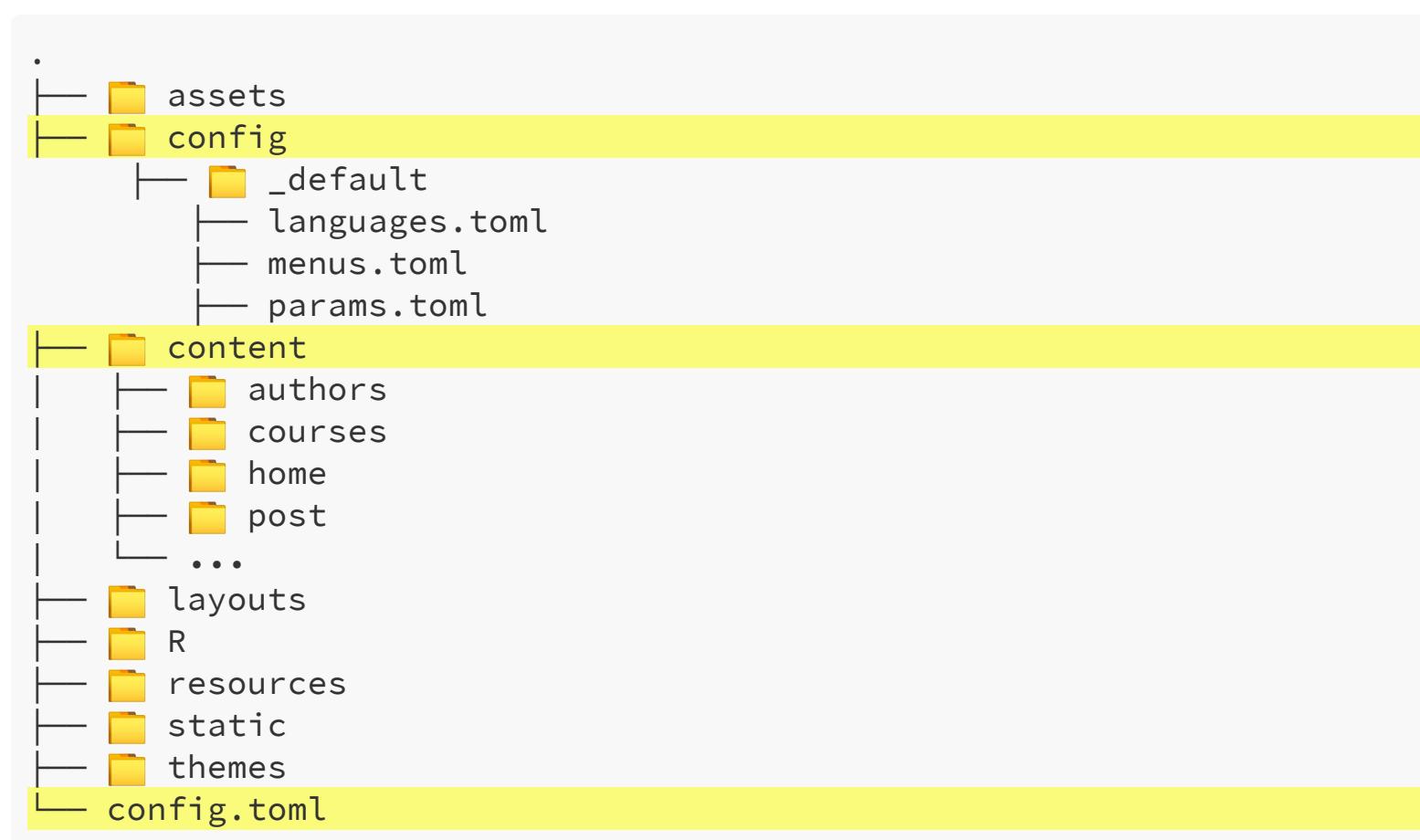
Junto al proyecto se han creado una serie de carpetas. **Como puedes ver, esta estructura es diferente a la del proyecto anterior.**

Nombre	Tipo
assets	Carpeta de archivos
config	Carpeta de archivos
content	Carpeta de archivos
layouts	Carpeta de archivos
R	Carpeta de archivos
resources	Carpeta de archivos
static	Carpeta de archivos
themes	Carpeta de archivos
.Rprofile	Archivo RPROFILE
Academic_curso	R Project
config.toml	Archivo TOML
index	Archivo RMD
netlify.toml	Archivo TOML

- assets
- config
- content
- layouts
- R
- resources
- static
- themes
- config.toml
- index.Rmd
- netlify.toml

El tema Academic es un tema especialmente complejo.

Directorio y archivos principales



El archivo config.toml: Antes de empezar

Antes de nada vamos a hacer una comprobación del sitio ejecutando:
`blogdown::check_site()`. Un aviso será el siguiente:

- [TODO] Add these items to the "ignoreFiles" setting: "\\.knit*.md\$"
- [TODO] Remove "_files\$" from "ignoreFiles"

Para solucionarlo abrimos el archivo `config.toml`:



El archivo config.toml: Antes de empezar

Buscamos lo siguiente dentro del archivo:

```
ignoreFiles = ["\\.ipynb$", ".ipynb_checkpoints$", "\\*.Rmd$", "\\*.Rmar$"]
```

Y lo sustituimos por:

```
ignoreFiles = ["\\.ipynb$", ".ipynb_checkpoints$", "\\*.Rmd$", "\\*.Rmar$"]
```

La próxima vez que ejecutemos la comprobación ya no nos saldrá el aviso.

¡Ahora todo está listo para empezar a editar nuestro sitio web!

El archivo config.toml: Cambiar el nombre a la web y el copyright

Buscamos lo siguiente y lo editamos:

```
# Title of your site  
title = "Academic"
```

```
# Enter a copyright notice to display in the site footer.  
# To display a copyright symbol, type `&copy;`. For current year, type  
copyright = ""
```

El archivo `languages.toml`: Cambiar el idioma

Vamos a `config` → `_default` → `languages.toml` y cambiamos `[en]` por `[es]` y `en-us` por `es-ES`.

```
# Languages
#   Create a `'[X]` block for each language you want, where X is the la
#   Refer to https://sourcethemes.com/academic/docs/language/

# Configure the English version of the site.
[en]
  languageCode = "en-us"
```

Si quieras crear un sitio multilenguaje vamos a archivo `config.toml` y modificamos:

```
# Default language to use (if you setup multilingual support)
defaultContentLanguage = "en"
```

Ahora veremos varios textos traducidos al español.

El archivo `menus.toml`: Cambiar el menú

El menú sigue apareciendo en inglés porque se configura desde el archivo `menus.toml`, que se encuentra en `config` → `_default` → `menus.toml` y tiene la siguiente apariencia:

```
[[main]]
  name = "Demo"
  url = "#hero"
  weight = 10

[[main]]
  name = "Posts"
  url = "#posts"
  weight = 20

...
```

`name` es el nombre que aparecerá en el menú, `url` el enlace o sección a la que lleva dentro de la misma página (#) y `weight` determinará el orden.

El archivo `params.toml`: Cambiar otros ajustes

Vea `config` → `_default` → `params.toml`. Este archivo permite cambiar varios ajustes como el tema, el tamaño de la fuente, el modo noche, entre otros ajustes.

Cambiando el tema por defecto

```
theme = "minimal"
```

Temas disponibles: "1950s", "apogee", "cofee", "cyberpunk", "dark",
"forest", "minimal", "mr_robot", "ocean", "rose" y "strawberry"

Ver: <https://wowchemy.com/docs/customization/>

Crear un nuevo tema

Ve a `themes` → `github.com` → `wowchemy` → `wowchemy-hugo-modules` → `wowchemy` → `data` → `themes`, donde se encuentran los archivos `.toml` asociados a los temas por defecto.

Crea un nuevo archivo toml

Copia y pega uno de los archivos que ya existen y cámbiale el nombre. Lo llamaremos `custom.toml`.

Crear un nuevo tema

custom.toml

```
# Theme metadata
name = "custom"

# Is theme light or dark?
light = true

# Primary
primary = "#2962ff"

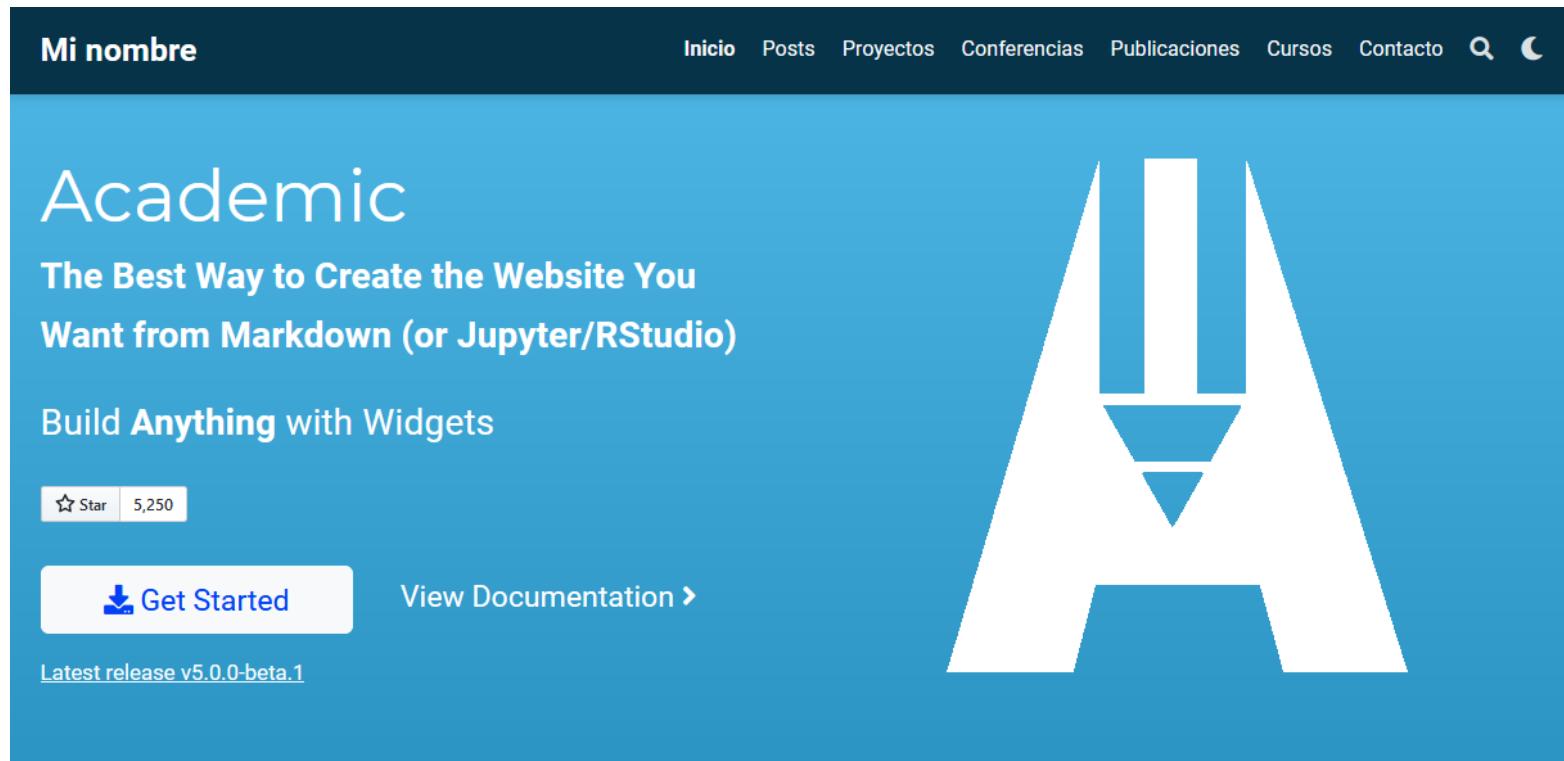
# Menu
menu_primary = "#073349"
menu_text = "#fff"
menu_text_active = "#f7f7f7"
menu_title = "#fff"

# Home sections
home_section_odd = "rgb(255, 255, 255)"
home_section_even = "rgb(247, 247, 247)"
```

Crear un nuevo tema

Volvemos al archivo **params.toml** y escribimos el nombre del nuevo tema.

```
theme = "custom"
```



The screenshot shows the homepage of the Academic website. The header features a dark blue navigation bar with the title "Mi nombre" on the left and links for Inicio, Posts, Proyectos, Conferencias, Publicaciones, Cursos, Contacto, a search icon, and a moon icon for dark mode. The main content area has a light blue background. On the left, there's a large white letter "A" logo. The text "Academic" is displayed above the logo. Below it, the tagline "The Best Way to Create the Website You Want from Markdown (or Jupyter/RStudio)" is written in bold blue text. Further down, the text "Build Anything with Widgets" is shown. At the bottom left, there's a button with a star icon labeled "Star" and the number "5,250". Below the main content, there are two buttons: "Get Started" with a download icon and "View Documentation >". At the very bottom, a link to "Latest release v5.0.0-beta.1" is provided.

El archivo `params.toml`: Cambiar otros ajustes

Modo noche, tipo y tamaño de fuente

```
# Enable users to switch between day and night mode?  
day_night = true  
  
# Override the theme's font set (optional).  
# Latest font sets (may require updating): https://sourcethemes.com/  
# Browse built-in font sets in `themes/academic/data/fonts/`  
# Browse user installed font sets in `data/fonts/`  
font = ""  
  
# Choose a font size.  
# Sizes: XS (extra small), S (small), M (medium), L (large - DEFAULT),  
font_size = "L"
```

Fuentes disponibles: **"Minimal"** (modern), **"Classic"**, **"Rose"** (traditional serif), **"Mr Robot"** (futuristic) y **"Native"**. Si se deja en blanco usa la fuente del tema.

El archivo `params.toml`: Cambiar otros ajustes

Características del sitio

```
highlight = true
highlight_languages = ["r", "latex"]  # Add support for highlighting a
# highlight_style = "github"  # For supported styles, see https://cdnj

math = false

diagram = false

privacy_pack = false

edit_page = {repo_url = "https://github.com/gcushen/hugo-academic", co

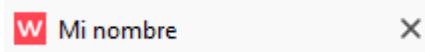
show_related = {docs = true, page = false, post = true, project = true}
```

Ejemplos `highlight`, `math` y `diagram`: <https://academic-demo.netlify.app/post/writing-technical-content/>

La carpeta **assets**. Cambiando el favicon

La carpeta **assets** contiene otra carpeta llamada **images**. En ella podemos guardar una imagen de tipo PNG llamada **icon.png** que reemplazará el favicon por defecto.

Como ejemplo, podemos agregar el logo de la Universidad de Vigo:



La carpeta content

La carpeta content es la que contiene los archivos que permiten editar la estructura y el contenido, así como crear nuevos artículos, cursos, diapositivas,

...

content → home

La carpeta `home` dentro de `content` nos permite cambiar la página de inicio. Podemos activar, desactivar y cambiar los contenidos, así como editar el orden de cada sección (widget).

Vamos a desactivar las secciones de los archivos `hero.md`, `demo.md`, `gallery` → `index.md`, `featured.md`¹, `talks.md`, `projects.md` y `tags.md`.

[1] Como desactivamos `featured.md` y `hero.md`, tenemos que volver a `menus.toml`.

La carpeta **content**

content → home → about.md

Ve a **about .md** dentro de **home** y cambia o borra el título (elemento **title**).
Como ejemplo puedes poner **title: Sobre mi**



Mi nombre

[Inicio](#) [Posts](#) [Proyectos](#) [Conferencias](#) [Publicaciones](#) [Cursos](#) [Contacto](#)  



Nelson Bighetti

Professor of Artificial Intelligence

Stanford University

Sobre mi

Nelson Bighetti is a professor of artificial intelligence at the Stanford AI Lab. His research interests include distributed robotics, mobile computing and programmable matter. He leads the Robotic Neurobiology group, which develops self-reconfiguring robots, systems of self-organizing robots, and mobile sensor networks.

Placeholder text: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed neque elit, tristique placerat feugiat ac, facilisis vitae arcu. Proin eget egestas augue. Praesent ut sem nec arcu pellentesque aliquet. Duis dapibus diam vel metus tempus vulputate.

 [Download my resumé.](#)

La carpeta **content**

content → home

Vamos a hacer unos pequeños cambios en varios archivos:

- `about.md`
- `experience.md`
- `accomplishments.md`
- `posts.md`
- `publications.md`
- `contact.md`

La carpeta **content**

content → **authors**

Dentro de la carpeta **authors** ve a **admin** y agrega una foto, por ejemplo yo he agregado una foto llamada **avatar-2.JPG**.

También puedes crear una nueva carpeta que en vez de **admin** se llame como quieras (o crear carpetas para otros autores), pero recuerda que dentro de **content** → **home** → **about.md** debes cambiar **author: admin** por el nuevo nombre.

Dentro de la carpeta **authors** ve a **admin**, abre **_index.md** y escribe tu datos.

Podemos agregar nuestro CV en **static** → **media**.

La carpeta **content**

content → authors (personalización avanzada)

También vamos a agregar un borde a la imagen que agregamos antes.

Para ello tenemos que ir a `themes` → `github.com` → `wowchemy` → `wowchemy-hugo-modules` → `wowchemy` → `assets` → `scss` → `wowchemy` → `_widgets.css` y agregamos `border: 2px solid black;` a la clase `.avatar`

La carpeta content

content → authors

Mi nombre

Inicio Posts Publicaciones Cursos Contacto



Mi nombre

Profesor de Estadística

Universidad de Vigo



Sobre mi

Soy profesor de Estadística en la Universidad de Vigo. Mis intereses son..... Mis investigaciones tratan sobre..... Soy jefe de..... También desarrollo.....

Además, doy clase de..... y en mi tiempo libre.....

Descarga mi [CV](#).

Intereses

- Estadística no paramétrica
- Procesos estocásticos
- Series temporales

Educación

- Doctorado en Estadística, 2012
Universidad de Vigo
- Máster en Estadística, 2009
Universidad de Vigo
- Grado en Matemáticas, 2008
Universidad de Santiago de Compostela

La carpeta **content**

content → **authors**

Lo que escribamos en **bio** aparecerá como descripción cuando escribamos un artículo en la web.



Mi nombre

Profesor de Estadística

Breve biografía de lo que hago, lo que me gusta o lo que quiera. Un par de líneas.



La carpeta **content**

content → publications

Dentro de **publications** tenemos 1 archivo y 3 carpetas.

- Dentro de **_index.md** podemos cambiar **title: Publications** por **title: Publicaciones**.
- Si queréis que al darle al menú la página nos lleve directamente a **/publications** podemos volver al archivo **menus.toml**.

```
[[main]]  
  name = "Publicaciones"  
  url = "#publications" # Cambiaríamos "#publications" por "publicatio  
  weight = 50
```

La carpeta content

content → publications

Si dentro de `/publications` aparecen 4 publicaciones en lugar de 3 es por un bug en el tema. Vamos a `themes` → `starter-academic` → `content` → `publication` y borramos la carpeta `example`. Paramos el servidor, construimos el sitio y volvemos a ejecutar el servidor.

The screenshot shows a dark-themed website interface. At the top, there is a navigation bar with the text "Mi nombre" on the left and links for "Inicio", "Posts", "Publicaciones", "Cursos", "Contacto", a search icon, and a moon icon for dark mode. Below the navigation bar, the page title is "Publicaciones". There are three search/filter input fields: "Buscar...", "Tipo", and "Fecha". The main content area displays four publication entries, each with a small thumbnail icon, the author's name, the year, the title, and a journal name in parentheses. Each entry has a set of blue-colored download or view buttons below it. The publications listed are:

- ✉ Mi nombre (2020). *Estimación tipo núcleo de densidades*. *Journal of Source Themes*, 1(1).
[PDF] [Citar] [Diapositivas] [DOI]
- ✉ Mi nombre (2019). *An example preprint / working paper*.
[PDF] [Código fuente] [Datos] [Proyecto] [Poster] [Diapositivas] [Video] [Documento fuente] [Custom Link]
- ✉ Mi nombre i, Robert Ford (2015). *An example journal article*. *Journal of Source Themes*, 1(1).
[PDF] [Citar] [Diapositivas]
- ✉ Mi nombre, Robert Ford (2013). *An example conference paper*. In STC.
[PDF] [Citar] [Código fuente] [Datos] [Proyecto] [Poster] [Diapositivas] [Video] [Documento fuente] [Custom Link]

La carpeta **content**

content → publications

Vamos a crear un nuevo artículo de revista.

- Creamos una nueva carpeta llamada **nuevo-articulo**.
- Copiamos y pegamos los contenidos de **journal-article** dentro de la nueva carpeta.
- Editamos el archivo.

Consejo: podemos guardar las carpetas de ejemplo en otro lugar y borrarlas. Así, si necesitamos crear un nuevo artículo tendremos las plantillas de ejemplo preparadas.

La carpeta **content**

content → **courses**

- Dentro de `_index.md` podemos cambiar `title: Courses` por `title: Cursos`.
- Copiamos y pegamos la carpeta `example` y la llamamos `curso-1`.
- Abrimos `_index.md` dentro de la carpeta `curso-1` y lo editamos.
- Renombramos `example1` por `capítulo1` y `example2` por `capítulo1`.
- Podemos crear **tantos cursos y capítulos dentro de cada curso como queramos**.

La carpeta **content**

Lo que hemos visto hasta ahora se aplica al resto de carpetas dentro de **content**.

Puedes crear, cambiar o modificar lo que quieras con facilidad, manteniendo la estructura indicada en los comentarios de cada archivo Markdown.

A continuación veremos cómo tener la página web que hemos creado en varios idiomas.

Traducir los contenidos

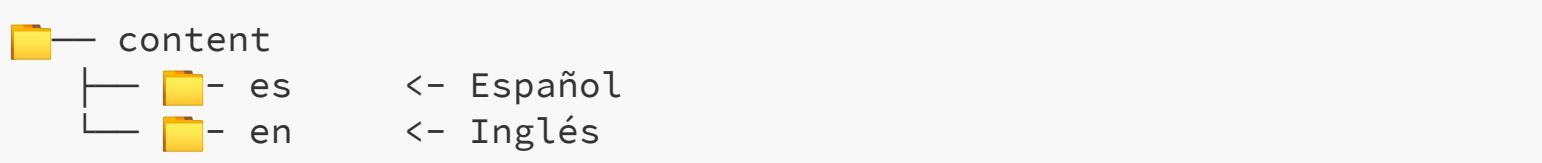
Para traducir los contenidos volvemos a `languages.toml` y ponemos lo siguiente:

```
[es]
languageCode = "es-ES"
contentDir = "content/es"

[en]
languageCode = "en-US"
contentDir = "content/en"
```

Traducir los contenidos

- Paramos el servidor con `blogdown::stop_server()`, creamos la carpeta `es` dentro de content y copiamos todos los archivos a ella.
- Una vez creada la duplicamos y la llamamos `en`.



- Volvemos a ejecutar el servidor con `blogdown::serve_site()`.

Traducir los contenidos

Mi nombre

Inicio Posts Publicaciones Cursos Contacto

English
Español



Mi nombre

Profesor de Estadística

Universidad de Vigo



Sobre mi

Soy profesor de Estadística en la Universidad de Vigo. Mis intereses son..... Mis investigaciones tratan sobre..... Soy jefe de..... También desarrollo.....

Además, doy clase de..... y en mi tiempo libre.....

Descarga mi [CV](#).

Interests

- Estadística no paramétrica
- Procesos estocásticos
- Series temporales

Education

- Doctorado en Estadística, 2012
Universidad de Vigo
- Máster en Estadística, 2009
Universidad de Vigo
- Grado en Matemáticas, 2008
Universidad de Santiago de Compostela

Traducir los contenidos

Ahora podemos traducir todos los contenidos dentro de la carpeta **en**.

Para traducir el menú:

- Copiamos el archivo **menus.toml** y lo renombramos como **menus.en.toml**. En este nuevo archivo podemos traducir los elementos del menú.



Ya tenemos nuestra página web lista

Ahora, para **comprobar** que todo está correcto, ejecuta:

```
blogdown::check_site()
```

Por último, **construye** los archivos con:

```
blogdown::build_site()
```

Dentro de la carpeta **public** están todos los archivos generados **para publicar** la página web.

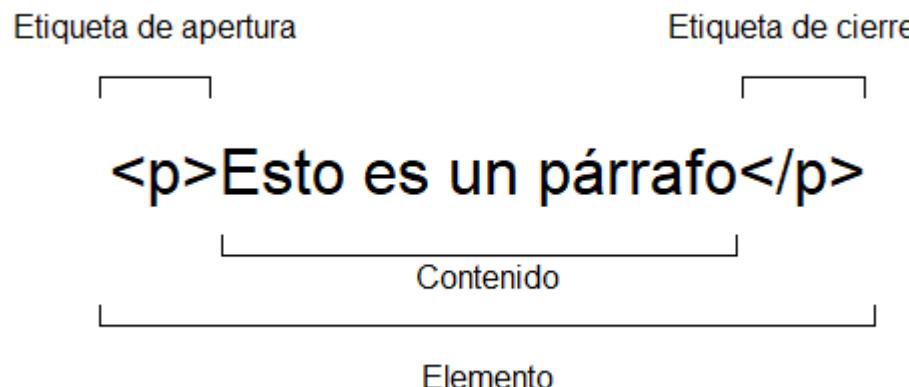
3. Introducción al desarrollo web

Conceptos básicos de HTML

HTML (HyperText Markup Language) es un lenguaje de marcado que se utiliza para crear la parte visual de las páginas web. Sus extensiones de archivo son **.html** y **.htm**.

El HTML define el *frontend* (lo que vemos) de una página web, pero no puede realizar funciones por sí mismo. Es completamente estático.

HTML **consiste en una serie de elementos creados con etiquetas**, que enmarcan ciertos contenidos. Como ejemplo, un párrafo se compone de:



Conceptos básicos de HTML: etiquetas

Las etiquetas más habituales son:

Etiqueta	Descripción
<html> </html>	Definen el documento
<head> </head>	Encabezado
<meta />	Metaetiquetas
<body> </body>	Cuerpo de la página
<header> </header>	Cabezera
<h1> </h1>, <h2> </h2>, ... <h6> </h6>	Encabezados (Títulos)

Etiqueta	Descripción
<p> </p>	Párrafos
 	Negrita
<div> </div>	División
<a> 	Enlaces
<style> </style>	Hojas de estilos
<script> </script>	Archivos JavaScript
<footer> </footer>	Pie de página

Conceptos básicos de **HTML**: creando una página muy simple

Vamos a crear una carpeta y dentro de esa carpeta vamos a crear un bloc de notas (o similar) y lo guardamos como **web.htm**. Dentro vamos a pegar lo siguiente:

```
<!DOCTYPE html>

<html lang="es">

<head>
</head>

<body>
</body>

</html>
```

Verifica que la extensión del artchivo sea **.htm** y no **.txt**

Conceptos básicos de **HTML**: creando una página muy simple

Haz **Click derecho** → **Abrir con** y selecciona un navegador con el que abrir el archivo.

Verás que se abre una página en blanco.

Dentro de la etiqueta **<head>** vamos a pegar lo siguiente:

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, sh
<title>Este es el título de la página</title>
```

Volvemos al navegador y actualizamos la página y veremos los cambios en la pestaña del navegador. En Windows podemos pulsar **F5** o **ctrl** + **F5** para recargar la página evitando que hayan quedado recursos en caché.

Conceptos básicos de **HTML**: creando una página muy simple

Ahora, dentro de la etiqueta **<body>** vamos a crear una cabezera con un título principal

```
<header>
  <h1> Título principal </h1>
</header>
```

Si actualizamos veremos el título en la web:

Título principal

Nota: **<h1>** es equivalente a **#** en R Markdown, **<h2>** es equivalente a **##** en R Markdown, ...

Conceptos básicos de **HTML**: creando una página muy simple

Podemos agregar más elementos dentro de **<body>**, como párrafos, listas, o incluso código.

```
<p>Párrafo 1</p>

<!-- Lista -->
<ul>
  <li> Elemento 1 </li>
  <li> Elemento 2 </li>
</ul>
<!-- Código -->
<pre>
<code>
# Creamos un gráfico
plot(mtcars$cyl, col = 3)
</code>
</pre>
```

<!-- Texto --> es un comentario en HTML.

Conceptos básicos de **HTML**: creando una página muy simple

También podemos **anidar etiquetas**. Por ejemplo, podemos agregar negrita a una parte del párrafo que creamos antes:

```
<p> <strong>Párrafo</strong> 1</p>
```

Título principal

Párrafo 1

- Elemento 1
- Elemento 2

```
# Creamos un gráfico  
plot(mtcars$cyl, col = 3)
```

Conceptos básicos de **HTML**: creando una página muy simple

También vamos a agregar un par de imágenes. Dentro del directorio creamos una carpeta llamada **índice** y ponemos los archivos dentro de ella. Una vez hecho podemos agregarlos con la etiqueta ****.

```
<img src = "índice/calendR-logo.png"></img>
<img src = "índice/econocharts-logo.png"></img>
```

Ten en cuenta que las imágenes se verán a tamaño real.

Conceptos básicos de **HTML**: creando una página muy simple

Atributos **HTML**

A las etiquetas **HTML** se les pueden agregar una serie de atributos, tales como **id** o **class**, que veremos en la sección sobre **conceptos básicos de CSS**.

La etiqueta **** tiene los atributos **width** y **height**, que permiten cambiar el alto y el ancho de las imágenes:

```
<img src = "imagenes/calendR-logo.png" width = 100px></img>
<img src = "imagenes/econocharts-logo.png" width = 100px></img>
```

Conceptos básicos de **HTML**: creando una página muy simple

Por último, dentro de la carpeta donde tenemos nuestro archivo **web.htm** vamos a crear otra carpeta llamada **blog** y dentro de ella vamos a crear otro archivo **.htm** llamado **blog1.htm**.

Vamos a copiar el contenido de **web.htm** a nuestro nuevo archivo, pero vamos a cambiar el título para que podamos distinguirlos:

```
<h1> Título blog </h1>
```

Dentro de **web.htm** vamos a crear un enlace hacia esta nueva página:

```
<a href = "blog/blog1.htm"> Enlace 1 </a>
```

Conceptos básicos de CSS

Ya tenemos nuestro código HTML, pero le falta un poco de estilo. Para ello existen las hojas de estilo CSS (Cascading Style Sheets).

CSS es un lenguaje de diseño gráfico para ser utilizado junto con lenguajes de marcado. Permite cambiar márgenes, bordes, colores, fuentes, tamaños, ... e incluso crear animaciones.

Cada estilo se cambia con una serie de palabras clave que definen ciertas propiedades. Por ejemplo:

Etiqueta	Descripción
color	Color del elemento
background-color	Color de fondo del elemento

Hay varias formas de agregar estilos a las etiquetas: agregándolas como un atributo (*inline*) de la etiqueta o con clases.

Conceptos básicos de CSS: el atributo style

Volvamos a `web.htm`. Vamos a cambiar el color del título principal.

- Para ello podemos cambiar:

```
<h1> Título principal </h1>
```

- por:

```
<h1 style = "color: red;"> Título principal </h1>
```

- Resultado:

Título principal

Párrafo 1

Conceptos básicos de CSS: el atributo style

También podemos agregar, por ejemplo, márgenes a las imágenes y un borde:

- Sustituye

```
<img src = "imagenes/calendR-logo.png" width = 100px></img>  
<img src = "imagenes/econocharts-logo.png" width = 100px></img>
```

- Por:

```
<img style = "margin: 20px; border: 2px solid black" src = "imagenes/c  
<img style = "margin: 20px; border: 2px solid black" src = "imagenes/e
```

Conceptos básicos de CSS: asignar un estilo a los elementos del mismo tipo

Si queremos aplicar el mismo estilo a todos los párrafos de la web resultaría poco eficiente escribir siempre el estilo de manera *inline* en todos párrafos de la web.

Vamos a cambiar el estilo de todos los elementos de las listas. Para ello podemos escribir lo siguiente dentro de la etiqueta <head>:

```
<style>
  li {
    font-size: 20px;      /*Tamaño de la fuente*/
    color: blue;         /*Color*/
    margin-top: 10px;     /*Margen superior*/
  }
</style>
```

/*Texto*/ es un comentario en CSS.

Conceptos básicos de CSS: asignar un estilo a los elementos mediante **clases**

Mediante el atributo **class** podemos asignar clases a los elementos HTML, de esta forma podemos crear un estilo por cada clase que se aplicará a los elementos que contengan esa clase. Ten en cuenta que se pueden asignar varias clases al mismo elemento.

```
<!-- Lista -->
<ul>
  <li class = "clase1"> Elemento 1 </li>
  <li> Elemento 2 </li>
</ul>
```

```
<style>
.clase1 {
  font-size: 30px;          /*Tamaño de la fuente*/
  color: red;              /*Color*/
  background-color: gray;  /*Color de fondo*/
}
</style>
```

Conceptos básicos de CSS: asignar un estilo a los elementos mediante id

También podemos usar el atributo **id**. La diferencia entre **id** y **class** es que los **id** tienen que ser únicos ya que su objetivo es identificar un elemento.

```
<!-- Lista -->
<ul>
  <li id = "id-1"> Elemento 1 </li>
  <li> Elemento 2 </li>
</ul>
```

```
<style>
#id-1 {
  font-size: 30px;           /*Tamaño de la fuente*/
  color: red;                /*Color*/
  background-color: lightgray; /*Color de fondo*/
}
</style>
```

Conceptos básicos de CSS: ¡sepáralo del HTML!

Lo ideal es separar el CSS en uno o varios archivos. Crea un archivo CSS en la raíz del directorio (de la misma manera y en el mismo lugar en donde creaste el `.htm`) y llámalo `estilos.css`.

Dentro de este archivo pega los siguientes estilos que creamos antes, pero **sin la etiqueta `<style>`**:

```
li {  
    font-size: 20px; /*Tamaño de  
    color: blue; /*Color*/  
    margin-top: 10px; /*Margen su  
}
```

```
.clase1 {  
    font-size: 30px; /*Tan  
    color: red; /*Col  
    background-color: gray; /*Col  
}
```

Ahora borra los estilos que agregamos en la etiqueta `<head>` de `web.htm` en la diapositiva anterior y sustitúyelo por lo siguiente:

```
<link rel = "stylesheet" href = "estilos.css">
```

Conceptos básicos de CSS: librerías

Twitter Bootstrap

Bootstrap es un framework CSS que incorpora una gran cantidad de clases CSS predefinidas que nos permiten crear páginas web responsive sin tener que escribir las clases CSS.

Podemos agregarlo a nuestra web añadiendo lo siguiente dentro de <head>:

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCeMQqXZGnqJLWVzDkQHdZPfLWZuNjCwvKUxqZB" crossorigin="anonymous">
```

Las hojas de estilos van en la etiqueta <head>.

Conceptos básicos de CSS: librerías

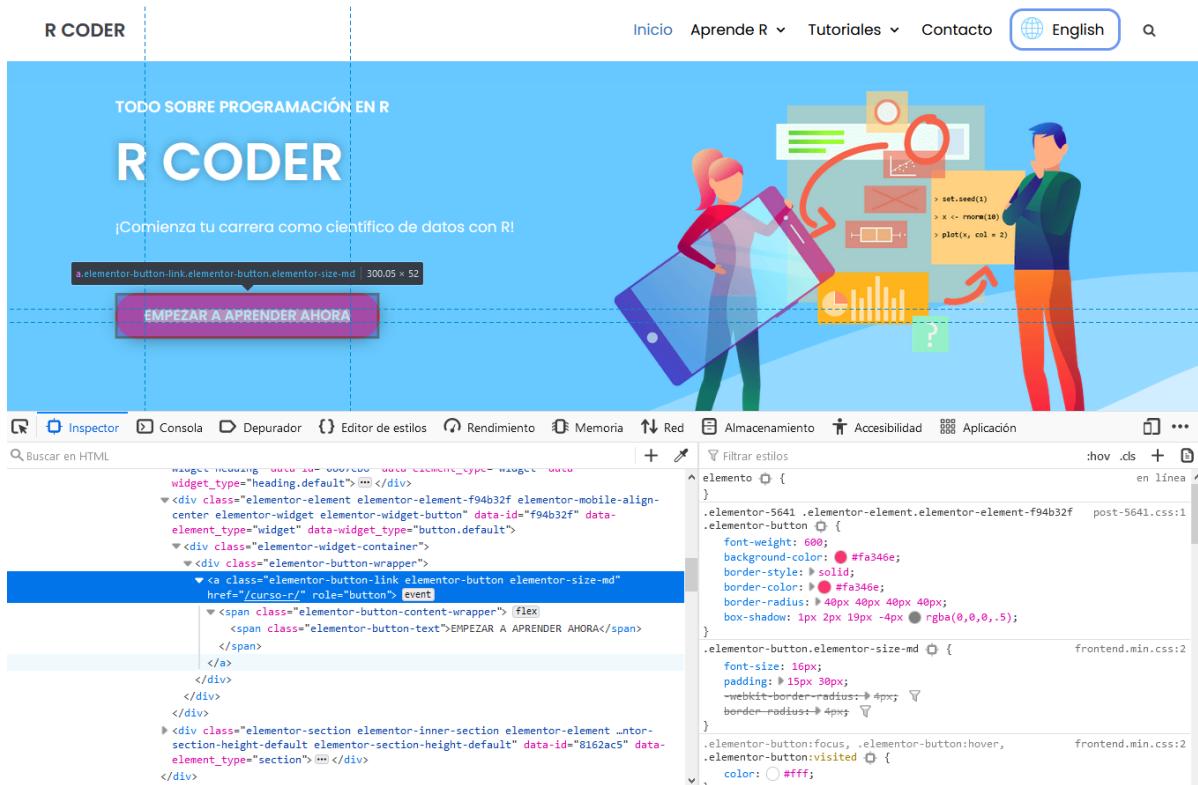
Twitter Bootstrap

Vamos a agregar una barra de navegación hecha con Bootstrap del siguiente ejemplo:

| <https://getbootstrap.com/docs/4.0/components/navbar/>

Pegamos el código dentro del <header>, guardamos y vemos los resultados.

Conceptos básicos de CSS: aprende CSS viendo ejemplos con el inspector de elementos



Conceptos básicos de CSS: aprende CSS viendo ejemplos de otras webs

Existen diferentes fuentes donde puedes encontrar ejemplos. Recomiendo las siguientes:

CodePen

| <https://codepen.io/>

Free Frontend

| <https://freefrontend.com/>

Conceptos básicos de CSS: aprende CSS viendo ejemplos de otras webs

Vamos a agregar otro ejemplo de Boostrap que podemos encontrar en la página Free Frontend.

| <https://freefrontend.com/bootstrap-footers/>

En concreto:

| <https://bbbbootstrap.com/snippets/simple-footer-social-media-icons-37929510>

Agregando JavaScript

JavaScript es un lenguaje de programación web que permite crear aplicaciones web o realizar ciertas funciones dinámicas.

Aunque no vamos a aprender JavaScript, **vamos a ver cómo agregar librerías de JavaScript a nuestra web.**

Como ejemplo vamos a agregar la librería **prism.js**, que agrega colores de manera automática a los elementos de código de una web.

En concreto, vamos a descargar los archivos necesarios para resaltar la sintaxis de códigos de R.

| <https://prismjs.com/download.html#themes=prism&languages=r>

Agregando JavaScript: prism.js

En el <head>

```
<link rel = "stylesheet" href = "prism.css">
```

En el <footer>

```
<script src = "prism.js"></script>
```

Agregamos la clase language-r al código

```
<pre class = "language-r">  
<code class = "language-r">
```

4. HUGO: aprendiendo a personalizar y crear plantillas

¿Cómo funcionan las plantillas?

Dentro del directorio principal encontrarás una carpeta llamada **theme**.

Dentro de esta carpeta podemos encontrar una serie de archivos y carpetas.

Los más relevantes son:

- **assets**, donde se encuentran las hojas de estilos y los archivos JavaScript (si los hay).
- **layouts**, donde se encuentran las plantillas HTML.

Las plantillas dentro de **layouts** no son HTML puro, sino que mezclan HTML con HUGO. Las variables del archivo **config**, así como las variables del encabezado YAML de los Markdown y su contenido se pasan a estas plantillas para crear el HTML definitivo.

El código HUGO utiliza llaves dobles: {{ }}

La carpeta `layouts`

Dentro de la carpeta `layouts` se encuentran las plantillas en las que se basa cada web. Por ejemplo, en el tema `Hyde` tendremos:

```
.  
├── _default  
│   ├── baseof.html  
│   ├── list.html  
│   └── single.html  
├── 404.html  
└── index.html  
└── partials  
    └── portfolio  
    └── shortcodes
```

En todas las plantillas nos encontraremos una carpeta llamada `_default`, donde se definen las plantillas por defecto y otra llamada `partials`, donde se crean plantillas parciales para usarse en otras plantillas HTML. A su vez, puedes crear carpetas (plantillas) para cada sección, **usando el mismo nombre** (cambia `portfolio` por `portafolio` y `about` por `acerca`).

Creando variables

Dentro de la primera web que hicimos en el Tema 2 abre el artículo que creamos con R Markdown llamado **post-prueba.Rmd**.

En el encabezado YAML del archivo vamos a agregar lo siguiente:

```
texto: "Este es un texto"
```

Esta es una **variable del encabezado YAML** y podremos utilizarla donde queramos. Para ello, vamos a la plantilla donde queramos hacer el cambio y escribimos:

```
{{ .Params.texto }}
```

Creando variables

Primero tenemos que encontrar qué archivo queremos modificar. Como queremos cambiar la plantilla de los artículos en primer lugar tenemos que ir a `layouts` → `_default` → `single.html`.

```
 {{ define "header" }}
    {{ partial "page-single/variables-init.html" . }}
    {{ partial "header.html" . }}          # <-- Encabezado
{{ end }}

{{ define "content" }}
    {{ partial "page-single/content.html" . }}  # <-- Aquí se define el contenido
{{ end }}

{{ define "footer" }}
    {{ partial "page-single/footer.html" . }}   # <-- Pie de página
    {{ partial "page-single/variables-deinit.html" . }}
{{ end }}
```

En este caso: `partials` → `page-single` → `content.html`.

Creando variables

Abrimos `content.html` y (por ejemplo) debajo de:

```
 {{ partial "page-single/post-meta.html" . }}
```

escribimos:

```
<p>{{.Params.texto}}</p>
```

Sin embargo, si la variable no se define en todos los artículos estaremos creando párrafos HTML vacíos. Para evitarlo podemos crear un condicional:

```
 {{with .Params.texto}}<p>{{.}}</p>{{ end }}
```

También podemos crear un else:

```
 {{with .Params.texto}}<p>{{.}}</p>{{else}}<p>No hay texto</p>{{ end }}
```

Creando variables

Variables lógicas

Dentro de los encabezados YAML también podemos crear variables booleanas o lógicas:

```
variable: true
```

En este caso, si queremos que se muestre un párrafo cuando la variable toma el valor **true** podemos escribir:

```
{{if .Params.variable}}<p>Texto si true</p>{{else}}<p>Texto si false</p>
```

También podemos concatenar condiciones entre variables de tipo cadena y variables lógicas:

```
{{if .Params.variable}}{{with .Params.texto}}<p>{{.}}</p>{{else}}<p>No
```

Creando variables

Variables globales

También podemos establecer una variable global definida dentro de `config.yaml`. Vamos a agregar la siguiente línea al archivo dentro de `params`:

```
info: "Esta es una información sobre mi"
```

Este texto lo pondremos debajo de nuestra imagen de perfil. Es una variable global porque afecta a todas las páginas de la web.

Creando variables

Variables globales

Vamos a `layouts` → `partials` → `sidebar.html` y buscamos dónde queremos agregar el texto.

```
{{ with .Site.Params.info}} <p>{{.}}</p> {{end}}
```



Traducción de contenidos

En el capítulo anterior ya vimos como podíamos traducir los contenidos. Pero si nuestra plantilla no tiene soporte para traducción tendremos que hacer unos pasos adicionales.

En primer lugar vamos a `config.yaml` y ponemos lo siguiente:

```
defaultContentLanguage: es
languages:
  es:
    contentDir: content/es
    languageName: Español
    weight: 1
  en:
    contentDir: content/en
    languageName: English
    weight: 2
```

Traducción de contenidos

Ahora podemos parar el servidor con `blogdown::stop_server()`, ir a la carpeta `content` y crear dos nuevas carpetas llamadas `es` y `en`. Podemos meter todo el contenido que ya teníamos en el idioma que queramos, por ejemplo en `es`.

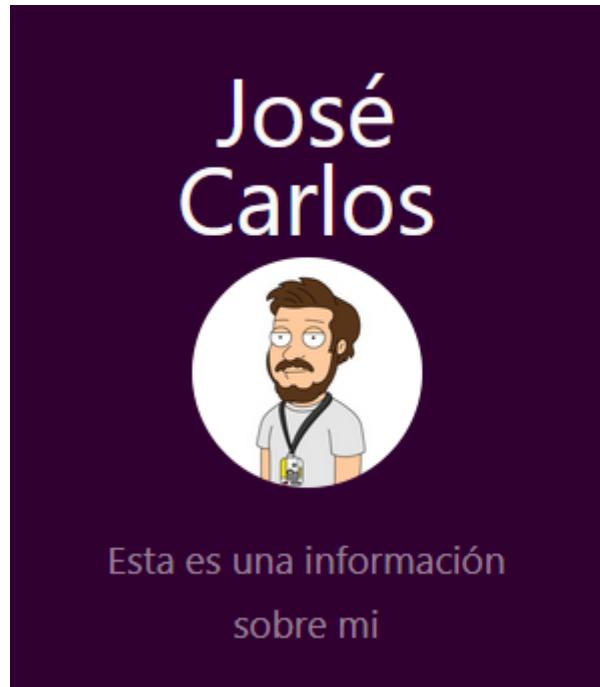
- Si volvemos a ejecutar el servidor y entramos en `/en` podremos ver los artículos en inglés.
- Dentro de `en` podemos crear las mismas carpetas que dentro de `es`. Vamos a crear una llamada `articulos`.

Ahora vamos a crear un artículo en inglés llamado `post-prueba.Rmd` dentro de `en → articulos` que será la traducción del artículo que creamos como ejemplo en el tema anterior correspondiente.

Podemos repetir el proceso para todos los archivos y carpetas.

Traducción de contenidos

Ve ahora a </en/articulos>:



Artículos

- [Sample post](#)

Problema: el título de la sección es "Artículos" en lugar de "Posts". Para solucionarlo podemos crear un archivo llamado `_index.md` equivalente al que existe dentro de `es` → `articulos` y cambiar el título.

Traducción de contenidos

¿Cómo hacemos para referenciar las páginas en castellano con el inglés (y viceversa)?

Si vamos a la página de inicio veremos que no hay ningún botón para traducir el contenido. En el artículo que tradujimos tampoco.

El primer paso es referenciar las páginas entre si con mediante links *rel alternate*. Vamos a `layouts` → `partials` → `header` → `meta.html` y escribimos:

```
 {{ if .IsTranslated }}  
   {{ range .AllTranslations }}  
     <link rel="alternate" hreflang="{{ .Language.Lang }}" href="{{ .Permalink }}>  
   {{ end }}  
 {{ end }}
```

Traducción de contenidos

¿Cómo hacemos para referenciar las páginas en castellano con el inglés (y viceversa)?

Si vemos el código fuente de los artículos traducidos veremos algo como lo siguiente:

```
<link rel="alternate" hreflang="es" href="//localhost:4321/articulos/p  
<link rel="alternate" hreflang="en" href="//localhost:4321/en/articulo
```

Ahora las páginas están referenciadas entre si, de modo que una se considera la traducción de la otra, pero todavía necesitamos un botón para cambiar entre un idioma y otro.

Traducción de contenidos

¿Cómo hacemos para referenciar las páginas en castellano con el inglés (y viceversa)?

Vamos a agregar un botón en la barra lateral para poder cambiar de idioma en caso de que la traducción esté disponible.

Nos dirigimos a `layouts` → `partials` → `sidebar.html` y agregamos lo siguiente antes del penúltimo `div`.

```
 {{ if .IsTranslated }}  
   <div style = "padding: 7px; margin-top:10px; border: 1px solid  
     {{ range .Translations }}  
       <a href="{{ .Permalink }}">{{ .Language.LanguageName }}  
     {{ end }}  
   </div>  
 {{ end }}
```

Traducción de contenidos

¿Cómo hacemos para referenciar las páginas en castellano con el inglés (y viceversa)?

Artículos

Portafolio

Acerca



© 2019 - 2021 htr3n CC BY-SA 4.0

Built with Hugo ❤️ hyde-hyde.

Español

Traducción de contenidos

¿Y si queremos que los nombres de las carpetas y los documentos sean distintos?

Cuando creamos la carpeta `en` dentro de ella creamos una carpeta llamada de la misma manera que en la carpeta `es`, donde a su vez creamos un artículo en inglés pero con el mismo nombre. Esto se debe a que HUGO identifica los archivos de ambos idiomas según los nombres de carpeta y de documento.

Si cambiamos el nombre de la carpeta `articulos` dentro de `en` por `posts` veremos que la traducción de la barra lateral deja de estar disponible (se consideran páginas distintas no relacionadas).

Traducción de contenidos

¿Y si queremos que los nombres de las carpetas y los documentos sean distintos?

Cambia el nombre de la carpeta `articulos` dentro de `en` por `posts`. Dentro del `_index.md` de las carpeta (`post` y `articulos`) agregamos la siguiente línea:

```
translationKey: "articulos"
```

Esta línea relaciona mediante un identificador una serie de páginas. Podemos hacer lo mismo para los documentos R Markdown.

Traducción de otros textos: el enlace a Inicio

Si nos dirigimos a una página en inglés y hacemos click en el Título de la página (texto superior de la barra lateral) nos llevará a la página de inicio en español. Podemos hacer unos cambios para que ese enlace sea distinto dependiendo del idioma en el que estemos navegando.

Vamos a `layouts` → `partials` → `sidebar.html` e identificamos ese enlace:

```
<a href="{{ .Site.BaseURL }}>
```

Y lo sustituimos por:

```
<a href="{{ "" | absLangURL }}>
```

Traducción del menú

Ya hemos aprendido a traducir los contenidos, pero todavía nos queda por traducir los textos de la barra lateral (el menú). Tenemos que ir al `config.yaml` y agregar el menú dentro de `en:` como se muestra (¡cuidado con la identación!):

```
menu:  
  main:  
    - identifier: posts  
      URL: /en/posts  
      name: Posts  
      weight: 100  
    - identifier: portafolio  
      name: Portafolio  
      url: /portafolio/  
      weight: 200  
    - identifier: acerca  
      name: Acerca  
      url: /acerca/  
      weight: 300
```

Traducción de textos: i18n

En cada artículo aparece el tiempo estimado de lectura, pero el texto "mins read" siempre aparece en inglés porque forma parte del HTML. En este caso necesitamos utilizar el sistema de internacionalización i18n.

- Creamos una carpeta llamada **i18n** dentro de la raíz del proyecto.
- Dentro de la nueva carpeta creamos un archivo llamado **en.yaml** y otro llamado **es.yaml**.

en.yaml

```
- id: mins  
  translation: "min read"
```

es.yaml

```
- id: mins  
  translation: "minutos de lectur
```

Traducción de textos: i18n

- Podemos ir a `partials` → `page-single` → `post-meta.html`, que es donde se encuentra el texto que queremos traducir y sustituimos "min read" por:

```
 {{i18n "mins"}}
```

Ahora el texto estará traducido según el idioma. Esto es particularmente útil si no queremos agregar variables en el `config.yaml`, si tenemos textos largos o si queremos agregar textos con componentes de HTML anidados (e.g. textos con palabras en negrita).

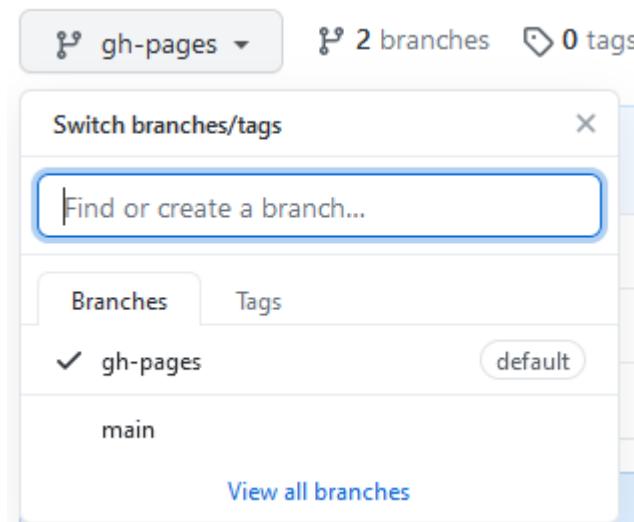
5. Publicación de la web

Publicación

- El directorio de publicación es por defecto `public`.
- Si ejecutas `blogdown::build_site()` los archivos generados se escribirán en `public`.
- Puedes subir los contenidos de la carpeta `public` a cualquier servidor que pueda servir páginas web estáticas (como GitHub Pages). Un vez subidos la página funcionará con normalidad.
- También es posible utilizar un `servicio de despliegue continuo`, como Netlify. Cuando se enlaza con un repositorio git, cada git actualizará el sitio.
- Si utilizas despliegue continuo, tendrás que agregar un archivo `.gitignore` y tendrás que añadir `public` a la lista de archivos para ignorar por parte de git.

GitHub pages

- Crea un nuevo repositorio.
- Sube **el contenido** de la carpeta **public** al repositorio. Si la web es muy grande (más de 100 archivos) tendrás que utilizar **GitKraken** o **GitHub Desktop** y sincronizar tu cuenta.
- Creamos una nueva rama llamada **gh-pages**.



GitHub pages

- Vamos a **Settings** y en la sección **GitHub Pages** elegimos la rama **gh-pages**.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

✓ Your site is published at <https://soage.github.io/academic-site/>

Source
Your GitHub Pages site is currently being built from the `gh-pages` branch. [Learn more](#).

Theme Chooser
Select a theme to publish your site with a Jekyll theme. [Learn more](#).

Custom domain
Custom domains allow you to serve your site from a domain other than `soage.github.io`. [Learn more](#).

Enforce HTTPS
— Required for your site because you are using the default domain (`soage.github.io`)

HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site.
When HTTPS is enforced, your site will only be served over HTTPS. [Learn more](#).

Netlify

Netlify es una empresa de computación en la nube con sede en San Francisco que **ofrece servicios de alojamiento y backend** sin servidor para aplicaciones web y sitios web estáticos.

De manera gratuita ofrece:

- Subir una cantidad ilimitada de sitios.
- Servicio de despliegue continuo.
- 300 minutos al mes para compilar los sitios.
- 100 Gb al mes de bandwidth.
- Redirecciones 301.
- Previsualización al subir los archivos.

Inconveniente: si superásemos los límites mensuales gratuitos (algo bastante complicado) los precios son muy elevados, por lo que sería recomendable migrar a un servicio de hosting normal.

Netlify

Para subir los archivos a Netlify necesitaremos que nuestra web incluya un archivo llamado **netlify.toml**. Si no está creado ejecuta:

```
blogdown::config_netlify()
```

Si haces algún cambio en el archivo puedes comprobar que todo está correcto con:

```
blogdown::check_netlify()
```

Consider Netlify instead of GitHub Pages for
Your Static Websites

GitHub Pages is Dead, Long Live Netlify!

Netlify

Team overview Sites Builds Plugins Domains Members Audit log Billing Team settings

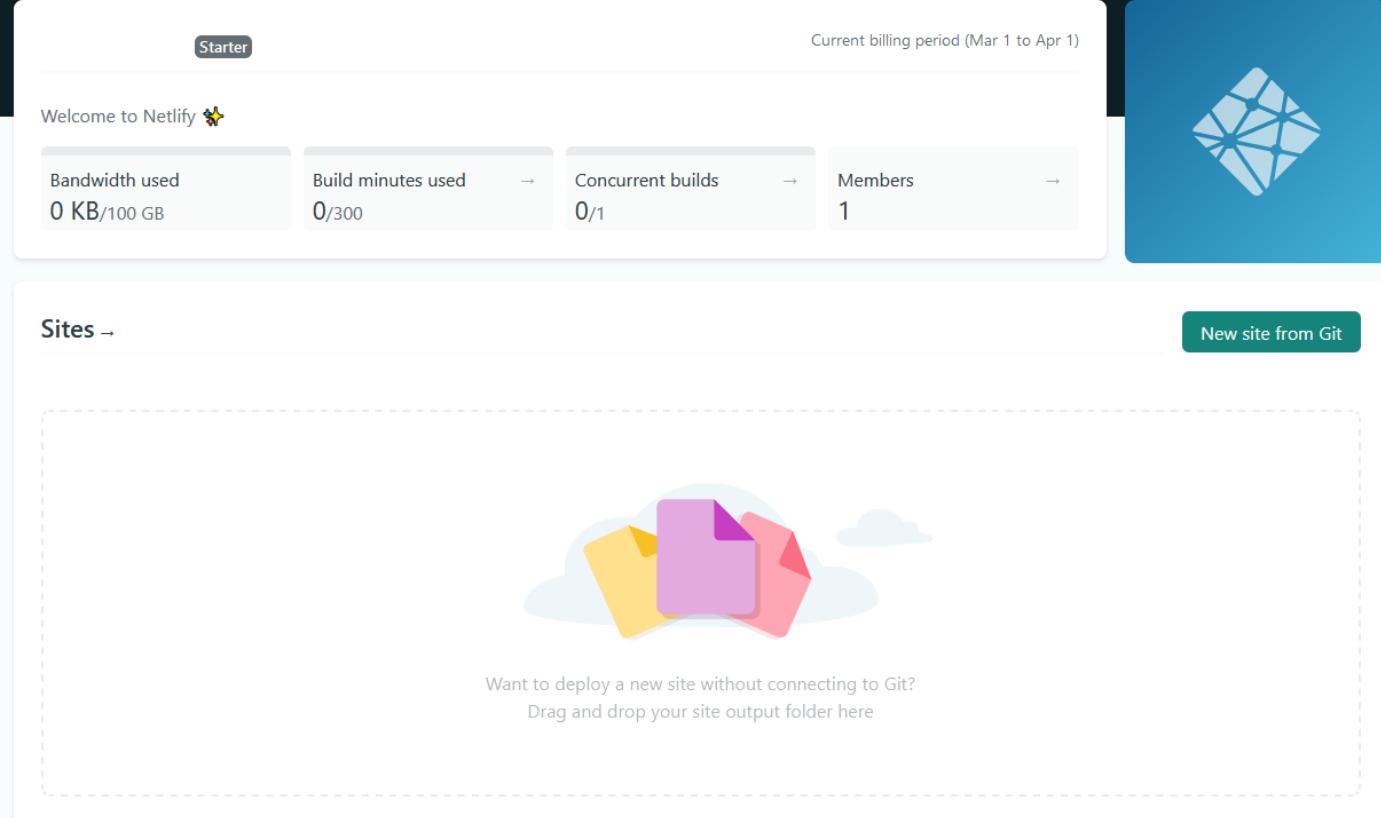
Starter Current billing period (Mar 1 to Apr 1)

Welcome to Netlify 🎉

Bandwidth used 0 KB/100 GB	Build minutes used 0/300	Concurrent builds 0/1	Members 1
-------------------------------	-----------------------------	--------------------------	--------------

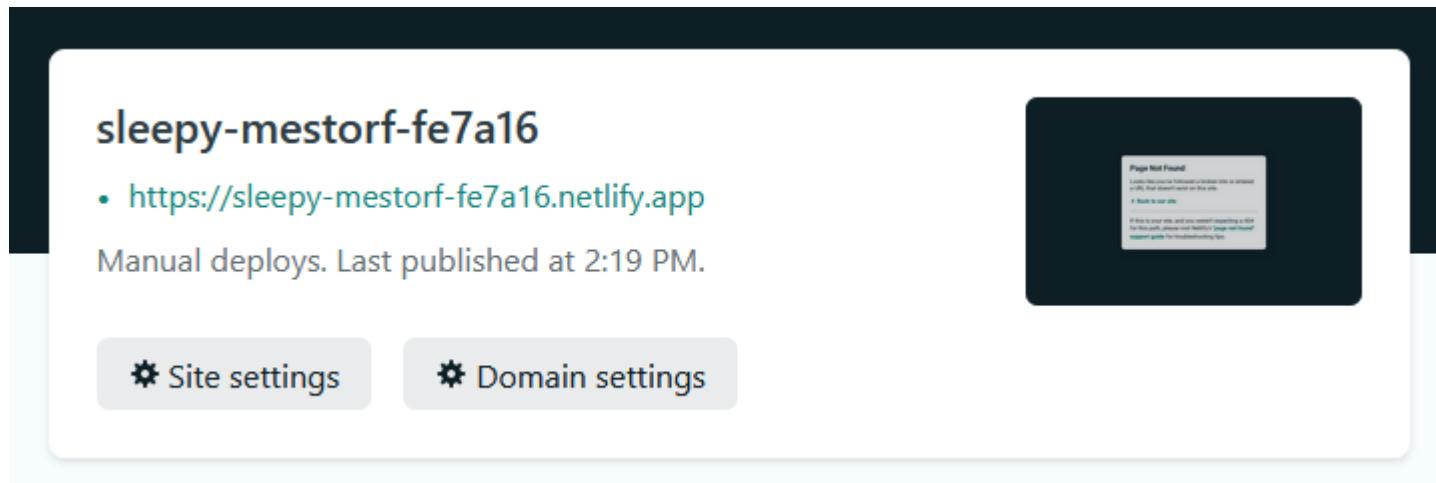
Sites → **New site from Git**

Want to deploy a new site without connecting to Git?
Drag and drop your site output folder here



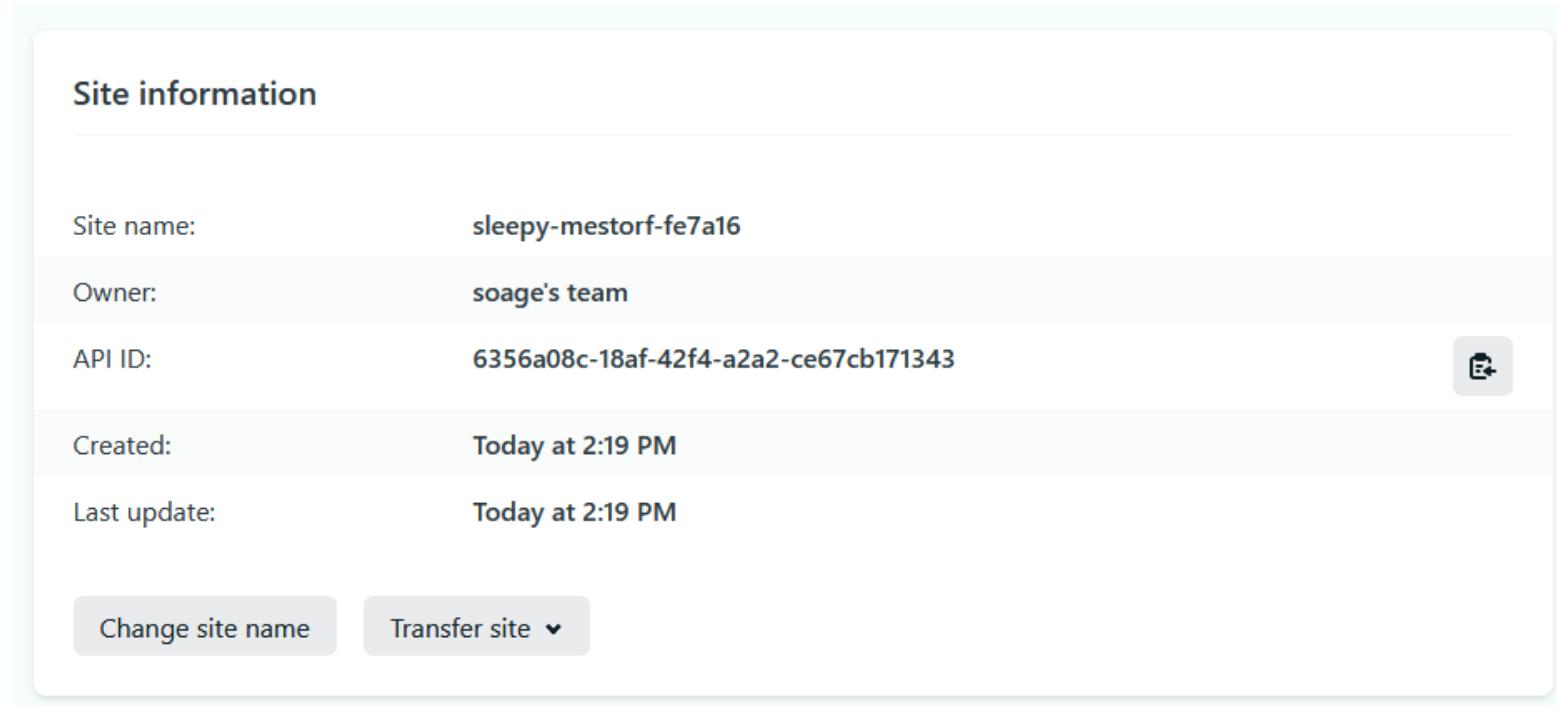
Netlify: cambiando el nombre por defecto

Por defecto Netlify asigna un nombre aleatorio a nuestra nueva página, en mi caso el nombre es el siguiente:



Netlify: cambiando el nombre por defecto

Si hacemos click en **Site settings** nos aparecerá lo siguiente:



The screenshot shows the 'Site information' section of the Netlify site settings. It displays the following details:

Site information	
Site name:	sleepy-mestorf-fe7a16
Owner:	soage's team
API ID:	6356a08c-18af-42f4-a2a2-ce67cb171343
Created:	Today at 2:19 PM
Last update:	Today at 2:19 PM

At the bottom, there are two buttons: 'Change site name' and 'Transfer site ▾'. To the right of the API ID, there is a small icon with a gear and a minus sign.

Haciendo click en **Change site name** podremos elegir un nombre siempre y cuando ningún usuario lo haya elegido antes.

Netlify: cambiando el nombre por defecto

Si tenemos algún nombre de dominio propio (de pago, de nuestra organización, ...) también podemos utilizarlo en lugar del `.netlify.com`.

2

Set up a custom domain →

Buy a new domain or setup a domain you already
own.

Ten en cuenta que si no registras el dominio con Netlify deberás agregar registros DNS en tu proveedor para apuntar a el dominio o subdominio del sitio en Netlify.

Subdominios rbind

Rbind regala dominios [.rbind.io](#). Para solicitar uno tendrás que crear un Issue en su [repositorio de GitHub](#) y escribirles algo como lo siguiente:

The screenshot shows a GitHub repository interface for 'rbind / support'. The 'Issues' tab is selected, showing 9 issues. A specific issue titled 'subdomain request for rladies-pdx #76' is open. The issue was created by 'apreshill' an hour ago and has 0 comments. The comment section contains the following text:

apreshill commented an hour ago Member +
Hello,
I'd like to request an rbind.io subdomain for rladies-pdx. The site is deployed through Netlify and currently lives at: <http://rladies-pdx.netlify.com>.
Can we have rladies-pdx.rbind.io?
Thanks!
Alison

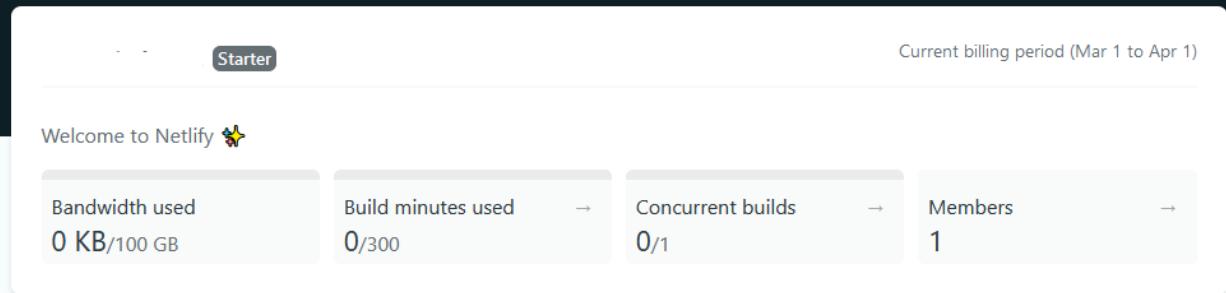
Actualizar el dominio

Si vas a utilizar un dominio personalizado dentro de `config.toml` o `config.yaml` tendrás que cambiar el `baseURL`:

```
baseURL = "https://midominio.rbind.io/"
```

En [Netlify](#) tendrás que especificar el nuevo dominio como se indicó antes.

Netlify: actualizar la web



Welcome to Netlify 🎉

Starter

Current billing period (Mar 1 to Apr 1)

Bandwidth used: 0 KB/100 GB

Build minutes used: 0/300

Concurrent builds: 0/1

Members: 1

Sites →

New site from Git

hyde-theme

Manual deploys

A screenshot of the Netlify dashboard. At the top, it shows a 'Starter' plan with usage statistics: 0 KB bandwidth used, 0 build minutes used, 0 concurrent builds, and 1 member. A large blue button on the right says 'New site from Git'. Below this, there's a section titled 'Sites' with a link '→' and a 'New site from Git' button. Under 'Sites', there's a card for 'hyde-theme' which has a small preview image, the name 'hyde-theme', and the text 'Manual deploys'.

Netlify: actualizar la web

The screenshot shows the Netlify Deploy section. At the top, there's a header with navigation links: Site overview, Deployes (which is highlighted in red), Plugins, Functions, Identity, Forms, Large Media, Split Testing, Analytics, and Site settings. Below the header, a box titled "Deploys for hyde-theme" contains a single deployment entry:

- https://hyde-theme.netlify.app

Below this, there are two buttons: "Deploy settings" and "Notifications". A note says "Manual deploys." and "Auto publishing is on. Deploys are published automatically." At the bottom of this box are three buttons: "Deploy settings", "Notifications", and "Stop auto publishing".

Below this box is another section titled "Deploys" which lists three recent production deployments:

Environment	Status	Date	Action
Production	Published	Today at 2:52 PM	→
Production	Published	Today at 2:36 PM	→
Production	Published	Today at 2:19 PM	→

Each deployment row has a "No deploy message" link. At the bottom of the page, there's a dashed box with the text "Need to update your site? Drag and drop your site output folder here".

Otras alternativas de alojamiento

GitLab Pages

GitLab es un servicio de control de versiones similar a GitHub. También ofrecen un servicio para alojar páginas web estáticas, con más recursos que GitHub Pages. El proceso para subir una página es similar en ambos casos.

Documentación: <https://docs.gitlab.com/ee/user/project/pages/>

CloudFlare Pages

Lanzado recientemente (abril de 2021), ofrece de manera gratuita más servicios y recursos que Netlify, pero publicar y actualizar la web no es tan trivial como con Netlify, ya que debemos tener alojado el sitio en GitHub.

Documentación: <https://developers.cloudflare.com/pages/>

Siguientes pasos

Si quieres que tu página aparezca en los resultados de Google regístrala en la [Google Search Console](#). Esto nos permitirá:

- Ver **qué páginas están en Google y cuáles no** (y cómo solucionar posibles problemas).
- Estadísticas sobre el **número de clicks** que recibes, el número de veces que tus páginas aparecen en los resultados de Google, el porcentaje de clicks y la posición media.
- **Métricas** sobre velocidad de carga de las páginas, usabilidad móvil y experiencia de la página.
- **Estadísticas de rastreo**, posibles mejoras y páginas que enlazan a la nuestra.

Siguientes pasos

Podemos comprobar la velocidad de carga de nuestras páginas con [PageSpeed Insights](#), tanto en móvil como en ordenador, y las posibles acciones a tomar.

También podemos probar [web.dev](#), para **medir la velocidad, encontrar malas prácticas, mejorar la accesibilidad y evaluar el SEO**.

Por último, si queremos mejorar la velocidad de nuestro sitio web y reducir el ancho de banda consumido podemos usar un CDN (Content Delivery Network), como el que proporciona CloudFlare de manera gratuita (aunque Netlify también proporciona uno).

¡Gracias!