# Colores, imágenes y presentaciones

Composición de textos científicos

27 de noviembre de 2020

## 1. Colores

El paquete más básico para incluir colores se llama `color` pero es muy habitual emplear en su lugar `xcolor` que añade algunas características más. En la cabecera de este documento se ha cargado este último con

```
\usepackage{xcolor}
```

Este paquete añade el comando `\color{...}` que cambia al color especificado por los puntos suspensivos. Este cambio se aplicará a partir del comando y hasta que se termine un bloque. Si esto suena muy críptico, seguro que estos ejemplos lo aclaran:

```
{El cielo es \color{cyan} azul celeste} cuando no está nublado

El cielo es {\color{cyan} azul celeste} cuando no está nublado
```

producen ambos "El cielo es azul celeste cuando no está nublado". Por supuesto, lo más normal es proceder de la segunda manera. Si no pusiéramos las llaves, todo el texto a partir de ese momento se volvería color cian. En realidad si haces una prueba con un texto muy extenso verás que se vuelve al color negro original cuando se cambia de página. Seguramente porque internamente hay bloques que incluyen cada página.

Si usamos el paquete sin parámetros adicionales y no definimos nada por nuestra cuenta, hay 19 nombres que podemos usar correspondientes a colores recogidos en esta tabla:

| | | | | |
|---|---|---|---|---|
| black | darkgray | lime | pink | violet |
| blue | gray | magenta | purple | |
| brown | green | olive | red | yellow |
| cyan | lightgray | orange | teal | ——— |

Antes de que pienses que te he timado porque solo hay 18 colores, si miras la fuente observarás que entre violet y yellow está `{\color{white}white}`, lo que pasa es que el blanco sobre blanco no se ve. Otra observación es que

si compilas con LaTeX, para producir un DVI, en lugar de con PDFLaTeX, para obtener un PDF, tu visor DVI hará de las suyas y no representará los colores de manera totalmente fidedigna. Incluso con PDFLaTeX, por muy buenas que sean tu pantalla y tu impresora, tampoco debes esperar una concordancia total entre ambas. En la impresión profesional hay todo un mundo alrededor de la determinación de los colores y la calibración de los dispositivos.

Si cargas el paquete con

```
\usepackage[usenames,dvipsnames]{xcolor}
```

tendrás además los 68 colores `dvips` que tienen a veces nombres tan poéticos como `CarnationPink` o `WildStrawberry`. Una lista completa la puedes encontrar en la documentación del paquete `xcolor`. Allí, se indican otras opciones que permiten tener nombres de más colores.

En la práctica casi nadie sabe de memoria grandes listas de nombres de colores ni siquiera las consulta en la documentación a la hora de componer textos en LaTeX, lo más habitual es crearlos uno mismo. Hay varias maneras de hacerlo. Aquí solo veremos dos. La primera, es combinar dos colores ya definidos (por nosotros o por defecto) siguiendo la estructura:

`\color{`*color1*`!`*porcentaje*`!`*color2*`}`

Por ejemplo, para oscurecer el cielo podemos usar `\color{cyan!60!black}` que combina un 60 % de cian y el resto, un 40 % de negro. Veamos el efecto de varios porcentajes poniendo el texto en negrita para que se aprecie mejor el color, es decir, usaremos como prueba

```
El cielo es {\bf\color{cyan!#!black} azul celeste}
```

donde `#` representa cierto número entre 0 y 100. Algunos resultados son:

| | | |
|---|---|---|
| 20 % | ⟶ | El cielo es **azul celeste** |
| 40 % | ⟶ | El cielo es **azul celeste** |
| 60 % | ⟶ | El cielo es **azul celeste** |
| 80 % | ⟶ | El cielo es **azul celeste** |

Otra forma de disponer de un nuevo color es definirlo en la cabecera. Hay varias posibilidades y una incluye la sintaxis que acabamos de ver. Aquí nos centraremos en

`\definecolor{`*nombre*`}{rgb}{`$R, G, B$`}`

donde *nombre* es la denominación convencional que queremos dar al color y $R, G, B \in [0, 1]$ indican la cantidad de color en los canales de color primarios rojo (**R**ed), verde (**G**reen) y azul (**B**lue). Seguro que en tu programa de retoque fotográfico favorito puedes elegir en un santiamén un color que te

guste y obtener sus coordenadas RGB. Normalmente en informática se dan estas coordenadas como enteros entre 0 y 255 por tanto es muy posible que debas dividir el resultado ofrecido por tu *software* entre 255.

Por ejemplo, incluyendo en la cabecera

```
\definecolor{MiAzulOscuro}{rgb}{0,0.08,0.5}
```

tendremos un nuevo color llamado `MiAzulOscuro` que no tiene nada de rojo, una pizca de verde y un azul medio (1 sería el azul con más brillo). Escribiendo

```
El cielo es {\bf\color{MiAzulOscuro} azul celeste}
```

se obtiene: "El cielo es **azul celeste**".

## 2. Imágenes

Con ayuda de paquetes auxiliares en LaTeX se pueden incluir imágenes e incluso crearlas (por ejemplo gráficas de funciones o esquemas). Esto último es más difícil y no lo veremos aquí. A efectos prácticos, hay mucho *software* para crear imágenes y por muy fanáticos que seamos de LaTeX puede ser un pequeño dolor de cabeza crear con él diagramas medianamente sencillos si no tenemos unos conocimientos relativamente avanzados.

Hay dos paquetes principales para la inclusión de imágenes: `graphicx` y `graphics`. En principio, según la documentación, el primero extiende al segundo pero yo he notado algunos problemas con ficheros `.eps` sin el segundo, por tanto normalmente cargo ambos en la cabecera con

```
\usepackage{graphics}
\usepackage{graphicx}
```

La compilación con LaTeX da problemas con formatos diferentes de `.eps` porque no es capaz de detectar sus dimensiones[1], por ello en lo sucesivo daremos por hecho que la compilación se lleva a cabo con PDFLaTeX, lo que permite manejar imágenes en formatos más habituales como `.jpg` o `.png`. El comando básico es

`\includegraphics[`*parámetros*`]{`*fichero*`}`

El fichero incluye el *path*, esto es, podemos poner simplemente el nombre, como `imagen.jpg`, si el fichero está en el mismo directorio que el fichero fuente pero quizá tengamos que escribir cosas como `./imagenes/imagen.jpg` o `../imagen.jpg` si no lo está. Los parámetros no son estrictamente

[1] En realidad, en principio, esto se puede arreglar incluyendo entre los parámetros `natwidth=`$a$ y `natheight=`$h$ donde $a$ y $h$ son respectivamente la anchura y la altura de la imagen pero por razones que desconozco ha dejado de funcionar en versiones modernas.

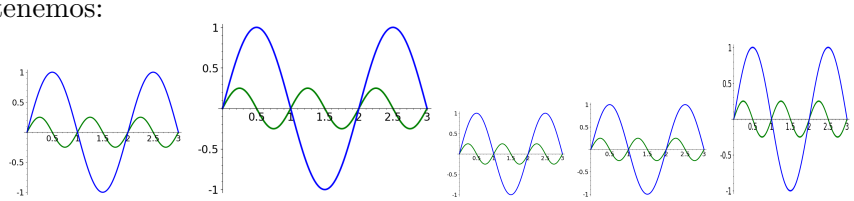
necesarios, si no los incluimos la imagen se muestra tal cual, con su tamaño original. Rara vez este será el que nosotros deseamos y por ello parámetros muy habituales son

`width=`$l$, `height=`$l$ y `scale=`$r$

donde $l$ son longitudes y $r$ es un número. Por supuesto, `scale` es incompatible con fijar un ancho o un alto. Por ejemplo, usando la imagen de prueba `fsin.png` que muestra las gráficas de $\text{sen}(\pi x)$ y $\text{sen}(2\pi x)$ en $[0, 3]$ y tiene unas dimensiones originales de $628 \times 468$ en píxeles, con

```
\includegraphics[height=50pt]{fsin.png}
\includegraphics[scale=0.2]{fsin.png}
\includegraphics[scale=0.1]{fsin.png}
\includegraphics[width=50pt]{fsin.png}
\includegraphics[width=50pt, height=60pt]{fsin.png}
```

obtenemos:



Hay que tener precaución al emplear simultáneamente `width` y `height` porque deformará la relación de aspecto. En una foto artística eso puede ser muy feo.

Un comentario a tener en cuenta, es que se supone que lo ortodoxo (según muchos manuales) es usar `\includegraphics` dentro del entorno `figure` pero yo rara vez lo hago. Este entorno facilita el poner títulos a las figuras y referencias a ellas y además automatiza su posición. Si no quieres ser tan cabezota como yo, deberías buscar información sobre `figure`. Para animarte a que lo hagas, comprueba el efecto de:

```
\begin{figure}[h]
\centering
\includegraphics[scale=0.2]{fsin.png}
\caption{Gráficas de seno}
\end{figure}
```

Incluyendo un `\label{mifigura}` el número de figura se mostraría con `\ref{mifigura}`. La `h` que aparece como parámetro del entorno indica *here*, es decir, que se olvide de la colocación automática y ponga la figura donde está el código, si es posible. Utilizando `figure` sin argumentos, la figura se reubicará, en general.

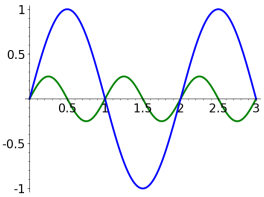
Mi consejo para conseguir una buena alineación horizontal de varias imágenes es incluir un único parámetro `height` igual para todas, porque

no aparecerán espacios sobrantes por arriba y por abajo aunque tengan diferentes dimensiones.

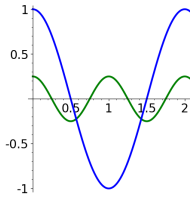
Por ejemplo, utilizando la imagen de prueba `fcos.png` que muestra las gráficas de $\cos(\pi x)$ y $\text{sen}(2\pi x)$ en $[0, 2]$ y tiene unas dimensiones originales de $440 \times 468$ píxeles, la fuente

```
\begin{center}
  \begin{tabular}{c}
  \includegraphics[height=75pt]{fsin.png}
  \\
  \textsf{Gráficas de seno}
 \end{tabular} \qquad
  \begin{tabular}{c}
  \includegraphics[height=75pt]{fcos.png}
  \\
  \textsf{Gráficas de coseno}
 \end{tabular}
\end{center}
```

produce

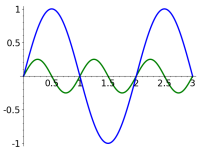


Gráficas de seno

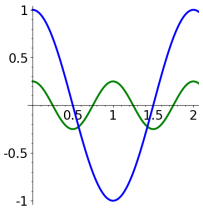


Gráficas de coseno

Si en vez de igualar las alturas lo hiciéramos con las anchuras cambiando `height` por `width`. El resultado sería:

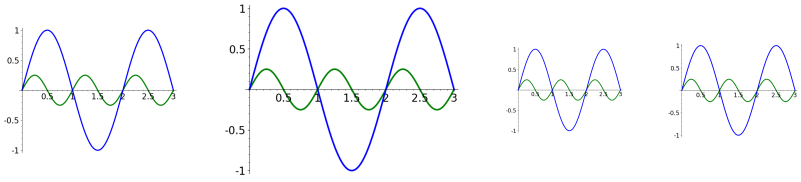


Gráficas de seno

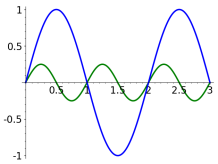
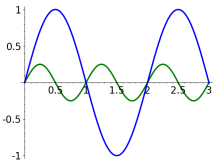


Gráficas de coseno

Hablando de alineamientos, si te fijas en el ejemplo de la sucesión de imágenes `fsin.png`, están alineadas por su base. Metiendo las cuatro primeras en un entorno `tabular` conseguimos centrarlas verticalmente:



Por si tienes curiosidad, LaTeX utiliza *cajas* para situar los elementos en cada línea y cada caja tiene una *línea base* que se continua a lo largo de la línea (por ejemplo, la caja de la letra "p" tiene su línea base justo bajo su bucle, por eso vemos "ap" en lugar de "aP". En las imágenes, la línea base es la de abajo mientras que en las tablas pasa por la mitad. Es útil tener esto en mente sobre todo al cambiar imágenes con fórmulas o textos. Por ejemplo,

$y=\sin(\pi nx)\to$  $y=\sin(\pi nx)\to$ 

responde (como *displayed formula*) a

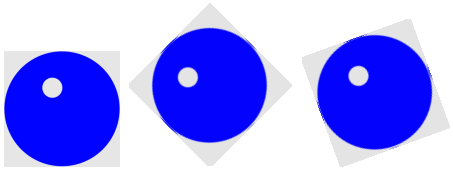
```
y=\sin(\pi nx)\to
\includegraphics[scale=0.1]{fsin.png} \qquad
y=\sin(\pi nx)\to
\begin{tabular}{c}
 \includegraphics[scale=0.1]{fsin.png}
\end{tabular}
```

En la mayor parte de los casos desearemos algo como lo segundo.

Hay otros parámetros posibles en `\includegraphics`, aquí solo veremos los relativos a los giros. Con `angle=`$n$ la figura se girará en sentido antihorario el número de grados especificados por $n$. Por ejemplo,

```
\includegraphics[scale=0.2]{circua.png}
\includegraphics[scale=0.2, angle=45]{circua.png}
\includegraphics[scale=0.2, angle=20]{circua.png}
```

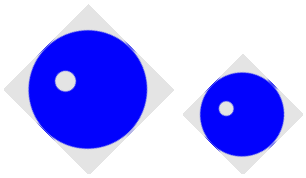
muestra la imagen de prueba `circua.png` reducida a un un quinto de su tamaño original y esta misma girada 45° y 20°.



Hay que tener en cuenta que fijar cierta altura y girar después no es lo mismo que girar y fijar cierta altura a continuación. Por ello con

```
\includegraphics[height=45pt, angle=45]{circua.png}
\includegraphics[angle=45, height=45pt]{circua.png}
```

no obtenemos dos figuras iguales:



La altura de la segunda figura es la misma que el lado del cuadrado de la primera: `45 pt` porque los parámetros de leen de izquierda a derecha.

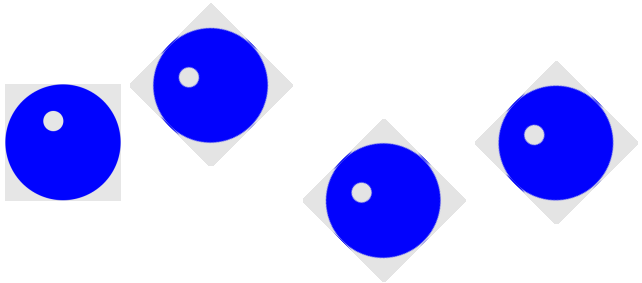
Si nos fijamos en los ejemplos anteriores veremos que por defecto el giro se produce por la esquina inferior izquierda (estrictamente con respecto al llamado *punto de referencia*, situado a la izquierda de la línea base). La manera de modificar este comportamiento es por medio del parámetro `origin=`*c* donde *c* es una cadena de uno o dos caracteres con los siguientes significados:

| | | | |
|---|---|---|---|
| `l` → izquierda, | `r` → derecha, | `c` → centro, | |
| `t` → arriba, | `b` → abajo, | `c` → centro, | `B` → línea de base. |

Las de la primera línea dan la alineación horizontal y las de la segunda la vertical, por eso se ha repetido `c`. Si uno no hace nada especial, como ya se ha mencionado, la línea base es la parte de abajo de la figura y `B` se convierte en un parámetro innecesario porque equivale a `b`. Se puede abreviar `cc` por `c`. Por ejemplo,

```
\includegraphics[scale=0.2]{circua.png}
\includegraphics[scale=0.2, origin=tl, angle=45]{circua.png}
\includegraphics[scale=0.2, origin=br, angle=45]{circua.png}
\includegraphics[scale=0.2, origin=c, angle=45]{circua.png}
```

aparte de la figura original, muestra los giros de 45° por la esquina superior izquierda, por la inferior derecha y por el centro de la imagen:



Además de `origin=c`, también se admiten expresiones con cualquiera de los otros caracteres en solitario, como `origin=l`. En ese caso se interpreta que el carácter ausente es `c`.

# 3. La clase beamer

La clase más usada para hacer presentaciones es `beamer`. El hecho de que sea una *clase* significa que es un posible argumento de `\documentclass`. El resto de la cabera la podemos conservar como siempre, salvo que en ejemplos anteriores con `article` no hemos aprovechado las opciones para poner título y autor y es muy raro que no las necesitemos en una presentación. Un ejemplo bastante escueto de la cabecera de una presentación que vaya a contener símbolos matemáticos es:

```
\documentclass{beamer}

\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}

\usepackage[spanish]{babel}
\usepackage[latin1]{inputenc}


\title{Ejemplo de presentación}
\author{Fernando Chamizo}
\institute{Universidad Autónoma de Madrid}
\date{27 de noviembre de 2020}
```

La idea básica es que las diferentes diapositivas de la presentación se separan encerrando los contenidos en un entorno llamado `frame`. Por ejemplo:

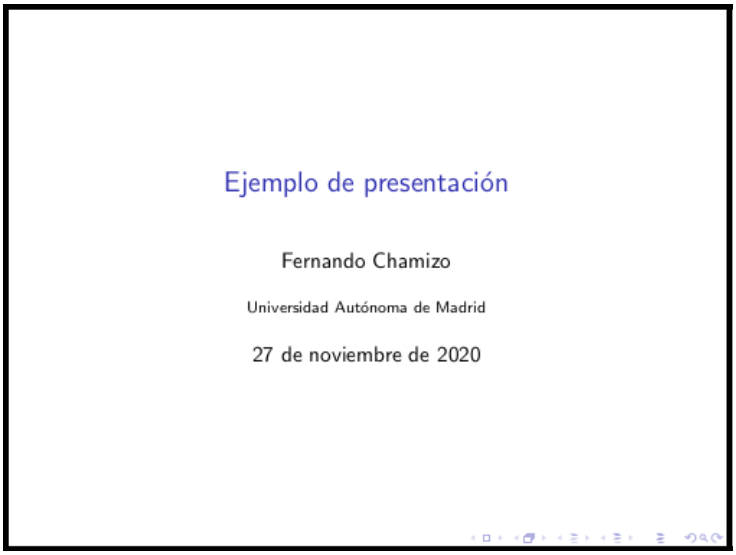
```
\begin{frame}
 Mi primera diapositiva
\end{frame}
```

El tipo de letra deja de ser la habitual no solo por la familia empleada (palo seco) sino por el tamaño. También vemos que el formato de cada página ha cambiado. Ahora está apaisada para que la proyectemos adecuadamente con un cañón. Además abajo a la derecha aparecen unos pequeños símbolos que responden al ratón para una navegación básica por las diapositivas (la verdad es que no he encontrado esto muy útil, en mi experiencia).

Lo normal es comenzar una presentación no por una diapositiva como la anterior sino con el título y autor. El comando `\titlepage` genera esta información a partir de lo escrito en la cabecera anterior. Entonces sustituimos la diapositiva anterior comenzando con:

```
\begin{frame}
 \titlepage
\end{frame}
```

Tampoco es que el resultado sea demasiado espectacular:




El recuadro negro no aparece, es solo para mostrar aquí los límites de la página.

En cada diapositiva se puede poner un título con `\frametitle{...}`. Además varias diapositivas pueden estar dentro de una sección que se indica con `\section{...}`, como ya sabemos. Muchas veces en las presentaciones el texto está en recuadros con un título. Esto se consigue con un entorno `block` cuyo único argumento es el título, que puede dejarse vacío.

Tomemos como continuación de nuestra diapositiva de título una sección llamada "Primera" con dos diapositivas, la segunda con bloques:

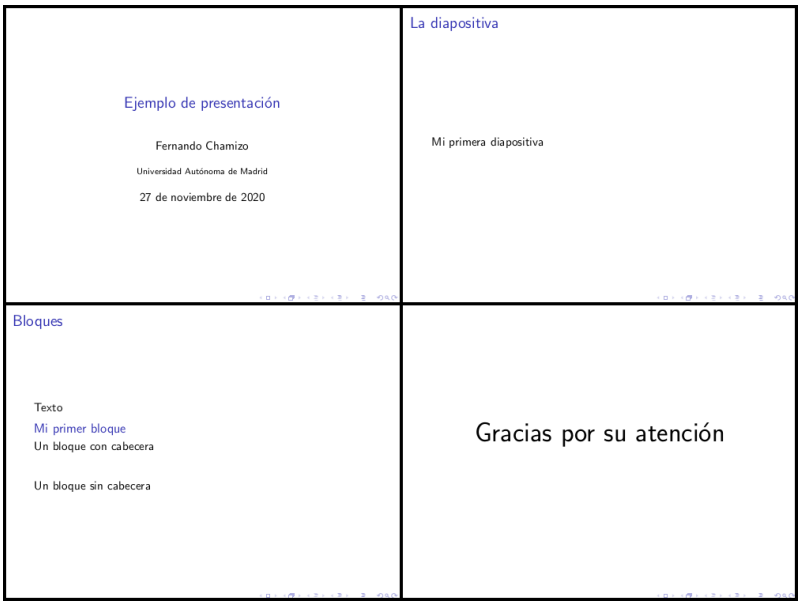
```
\section{Primera}
\begin{frame}
  \frametitle{La diapositiva}
  Mi primera diapositiva
\end{frame}

\begin{frame}
  \frametitle{Bloques}
  Texto
  \begin{block}{Mi primer bloque}
  Un bloque con cabecera
  \end{block}
  \begin{block}{}
  Un bloque sin cabecera
  \end{block}
\end{frame}
```

y acabemos con una diapositiva de agradecimiento dentro de una sección llamada “Fin”:

```
\section{Fin}
\begin{frame}
  \begin{center}
    \Huge Gracias por su atención
  \end{center}
\end{frame}
```

Si lo compilamos obtendremos cuatro páginas del tipo




Esto es bastante soso y por ejemplo las secciones no se muestran por ningún lado. Para decorar las presentaciones se pueden usar “temas” y “colores de tema”. Lo primero se consigue con `\usetheme{...}` y lo segundo con `\usecolortheme{...}`. Aquí hay una brevísima tabla que recoge tres posibilidades para cada uno:

| `\usetheme` | `Frankfurt` | `Madrid` | `Warsaw` |
|---|---|---|---|
| `\usecolortheme` | `beaver` | `seahorse` | `rose` |

Hay que elegir a lo más un tema y un color de tema. Si cargamos dos temas es posible que haya incompatibilidades y obtengamos errores al compilar.

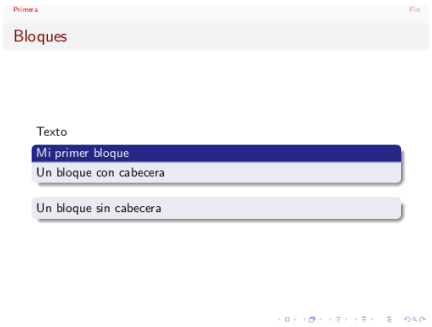
Si en el ejemplo anterior cargamos `Frankfurt` con

```
\usetheme{Frankfurt}
```

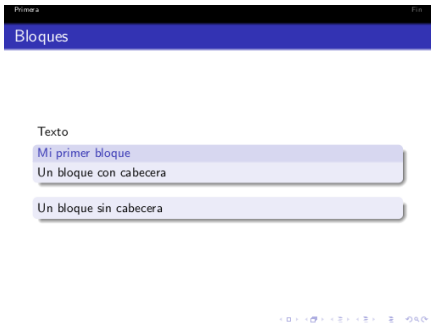
sin especificar ningún color de tema, ya veremos cambios sustanciales. En la primera diapositiva el título aparecerá dentro de un cuadrado azul sombreado y en la parte superior de todas las páginas aparecerá el nombre de sección

en pequeño de forma que al pinchar con el ratón pasamos automáticamente a ellas. Por otro lado los bloques aparecerán destacados.

Hay tantos temas que es imposible dar una idea de todos los posibles resultados. Solo como ilustración se muestra una miniatura de la página que contiene los bloques para todas las combinaciones de los temas y colores de tema antes mencionados. Además se muestra al final la diapositiva del título con `Madrid` y `rose`.

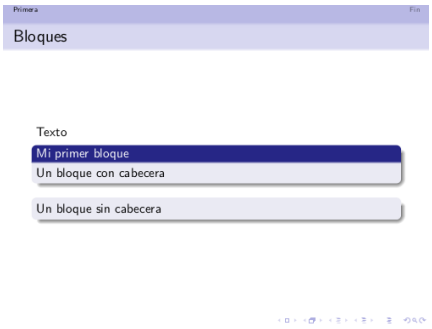



`Frankfurt, beaver`




`Frankfurt, rose`

---




`Frankfurt, seahorse`

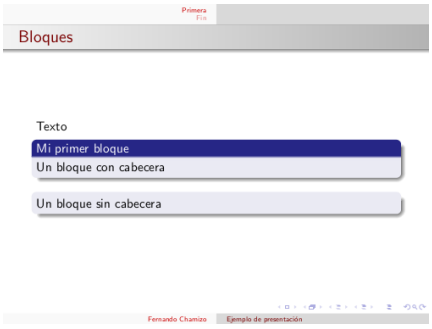

`Madrid, beaver`

---


`Madrid, rose`


`Madrid, seahorse`

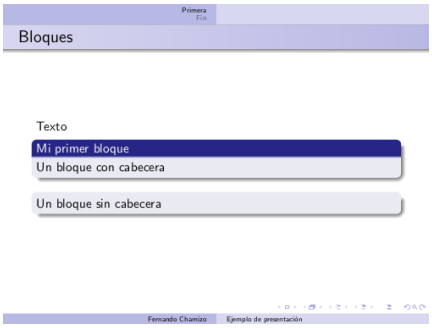
---



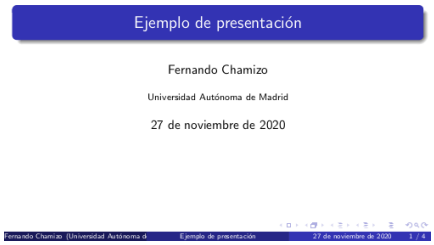

`Warsaw, beaver`


`Warsaw, rose`

---




`Warsaw, seahorse`




`Madrid, rose` (título)

---

En algunos casos, como `Madrid` con `rose` obtenemos *overfulls* porque el texto que resume los datos de la cabecera no cabe bien en la línea inferior. Una manera de evitarlo es poner un parámetro entre corchetes en estos datos que indique qué debe mostrar como versión abreviada. Por ejemplo, en el caso mencionado lo normal sería cambiar

```
\institute{Universidad Autónoma de Madrid}
```

por

```
\institute[UAM]{Universidad Autónoma de Madrid}
```

para abreviar por "UAM", o incluso por

```
\institute[UAM]{Universidad Autónoma de Madrid}
```

si queremos que no aparezca la institución al pie de cada página distinta del título.

Las diapositivas de las presentaciones a menudo se muestran haciendo aparecer sus contenidos poco a poco. Una manera muy básica de conseguirlo es con el comando `\pause` que espera una pulsación para seguir avanzando. Más versátil es el comando `\uncover<n>{...}` que descubre la parte

contenida en los puntos suspensivos en la pulsación $n$-ésima dentro de la diapositiva. Si en vez de `<n>` escribimos `<n->`, será visible a partir de la pulsación `n` y si escribimos `<-n>`, hasta antes de ella.

Por ejemplo, si cambiamos el código de la diapositiva que contiene los bloques a

```
\begin{frame}
\frametitle{Bloques}
\pause
Texto
\uncover<3>{\begin{block}{Mi primer bloque}
 Un bloque con cabecera
\end{block}}
\begin{block}{}
 Un bloque sin cabecera
\end{block}
\end{frame}
```

entonces en la primera pulsación que nos lleva a esa diapositiva la veremos vacía salvo el título "Bloques", en la siguiente aparecerán "Texto" y el bloque sin cabecera, dejando hueco para el bloque titulado "Mi primer bloque" que aparecerá en la tercera y última pulsación.

Con `beamer` podemos emplear todo lo que hemos aprendido[2] pero no hay que perder de vista, que el propósito de una presentación es totalmente distinto del de un artículo o un trabajo. Los principiantes que solo tienen experiencia con estos últimos, tienden a hacer presentaciones muy densas y con muchos símbolos. Este es un error a evitar. Los asistentes a una presentación no pueden pasar páginas y por tanto solo tendrán acceso a lo que vean en la diapositiva en curso y a lo, posiblemente poco, que recuerden de las anteriores. Cada diapositiva debe tener pocas líneas y ser esquemática. Además, es conveniente que sacrifiquemos las definiciones y notaciones que no sean absolutamente necesarias. A lo que debemos aspirar en la mayoría de los casos es a transmitir ideas, no detalles.

[2]Para decir toda la verdad, hay algún tipo de incompatibilidad ocasional, por ejemplo con el entorno `verbatim`.