

# Introducción a la transcodificación, edición y visualización de datos audiovisuales con FFmpeg

---

 [programminghistorian.org/es/lecciones/introduccion-a-ffmpeg](https://programminghistorian.org/es/lecciones/introduccion-a-ffmpeg)

## Programming Historian en español

---

### Dave Rodriguez

---

Esta lección introduce las funciones básicas de FFmpeg, una herramienta libre de línea de comandos utilizada para manipular y analizar materiales audiovisuales.

[Revisado por pares](#)

[CC-BY 4.0](#)

[Apoyar PH](#)

### editado por

---

Brandon Walsh

### revisado por

---

- Tesla Cariani
- Josh Romphf

### traducido por

---

- Dave Rodriguez
- Sebastian Fiori

### traducción editada por

---

Antonio Rojas Castro 

### traducción revisada por

---

- Jennifer Isasi
- José Antonio Motilla

### publicado

---

| 2018-12-20

## traducido

---

| 2020-12-11

## modificado

---

| 2022-08-04

## dificultad

---

| Media

 <https://doi.org/10.46430/pheso049>

## **¡Haz una donación!**[haz-una-donación](#)

---

Producir buenos tutoriales de acceso abierto cuesta dinero. Únete al creciente número de personas que apoya a Programming Historian para que podamos continuar compartiendo conocimientos de forma gratuita.

Disponible en: [EN](#) (original) | [ES](#)

## **Contenidos**[contenidos](#)

---

### Introducción

---

Historicamente, las Humanidades Digitales se han enfocado casi exclusivamente en el análisis de fuentes textuales a través de métodos computacionales (Hockey, 2004). Sin embargo, hay un interés creciente en el campo de la utilización de métodos computacionales para el análisis de materiales audiovisuales de patrimonio cultural, como refleja la creación de la Alianza de Organizaciones de Humanidades Digitales Grupo de Interés Especial: Materiales audiovisuales en Humanidades Digitales y el aumento de las presentaciones relacionadas con temas audiovisuales en la conferencia global de AOHD en los años anteriores. Investigaciones recientes, tal como Distant Viewing TV, indican un cambio en el campo hacia proyectos relacionados con el uso de técnicas computacionales para ampliar el alcance de los materiales que los y las humanistas digitales pueden explorar. Como afirma Erik Champion, “la audiencia de Humanidades Digitales no siempre está enfocada en la literatura o está interesada en las formas tradicionales de alfabetización” y la aplicación de metodologías digitales para estudiar cultura audiovisual es una faceta emergente y emocionante de las humanidades digitales (Champion, 2017, traducido por el autor). Hay muchas herramientas valiosas, gratuitas y de código abierto disponibles para aquellos interesados en trabajar con materiales audiovisuales (por ejemplo, el tutorial de Programming Historian Editar Audio con Audacity). Este tutorial presentará otra: FFmpeg.

FFmpeg es el *framework* multimedia de código abierto líder para transcodificar, editar, filtrar y reproducir casi cualquier tipo de formato audiovisual digital (sitio web de FFmpeg - “About”). Muchos programas comunes y sitios web usan FFmpeg para leer y escribir archivos audiovisuales, por ejemplo, VLC, Google Chrome, YouTube y muchos más. Además de ser una herramienta de programa y de desarrollo web, FFmpeg se puede usar en la interfaz de la línea de comandos para realizar muchas tareas comunes, complejas e importantes, relacionadas con la gestión, modificación y análisis de archivos audiovisuales. Estos tipos de procesos, tales como editar, transcodificar o extraer los metadatos de archivos, generalmente requieren acceso a otro programa (tal como editores de vídeo no lineal, como Adobe Premiere o Final Cut Pro); sin embargo, FFmpeg permite a un usuario operar directamente en archivos audiovisuales sin el uso de interfaces o programa de terceros. Como tal, el conocimiento del *framework* permite a los usuarios manipular materiales audiovisuales para satisfacer sus necesidades con una solución de código abierto y gratuita, que ofrece gran parte de la funcionalidad de un costoso programa de audio y vídeo. Este tutorial ofrece una introducción a la lectura y escritura de comandos de FFmpeg y una guía paso a paso a partir de un caso práctico para aprender a utilizar el *framework* en un trabajo específico para los humanistas digitales. Específicamente, se mostrará cómo FFmpeg puede ser utilizado para extraer y analizar datos de color en un video archivístico.

## **Objetivos de aprendizaje**objetivos-de-aprendizaje

---

- Instalar FFmpeg en tu computadora o usar una versión “demo” en el navegador web
- Comprender la estructura básica y la sintaxis de los comandos de FFmpeg
- Aprender varios comandos útiles, tales como:
  - “Re-wrap” (cambiar el contenedor) y transcodificar (recodificar archivos)
  - “Demux” de archivos (separar audio y vídeo)
  - Recortar/Editar archivos
  - Usar FFplay para reproducir archivos
  - Crear vectorscopios para visualizar los datos de color
  - Usar FFprobe para generar informes de los datos de color
- Introducir recursos para mayor exploración y experimentación

## **Requisitos previos**requisitos-previos

---

Antes de comenzar con este tutorial, es necesario que localices la Terminal de tu computadora u otra interfaz de línea de comandos, ya que ahí es donde ingresarás y ejecutarás los comandos de FFmpeg. Si necesitas instrucción para acceder y usar la interfaz de línea de comandos, te recomendamos la lección de *Programming Historian Introducción a la línea de comandos en Bash* para usuarios de Mac y Linux o, para usuarios de Windows, Introducción a la línea de comandos de Windows con PowerShell. Adicionalmente, será de utilidad tener conocimientos básicos de códecs y contenedores audiovisuales para entender

con mayor detalle el funcionamiento de FFmpeg. Proporcionaremos información adicional y revisaremos con mayor detalle sobre códecs y contenedores en la sección sobre ejemplos de comandos preliminares de este tutorial.

## Cómo instalar FFmpeg

---

La instalación de FFmpeg es posiblemente la parte más difícil de usar esta herramienta. Afortunadamente, existen algunas guías y recursos disponibles para instalar el *framework* para cada sistema operativo.

Nuevas versiones de FFmpeg son lanzadas aproximadamente cada seis meses. Para mantenerse al tanto de ellas, es recomendable seguir a FFmpeg en [Twitter](#) o en su sitio web. Las nuevas versiones de FFmpeg generalmente contienen características tales como filtros nuevos y actualizados, compatibilidades de códecs y corrección de errores. La sintaxis de FFmpeg no cambia con estas actualizaciones y las capacidades antiguas rara vez se eliminan. Puedes aprender más sobre estas actualizaciones consultando los anuncios de actualizaciones anteriores en la sección de [News](#) en el sitio web de FFmpeg.

## Para usuarios de Mac OS [para-usuarios-de-mac-os](#)

---

La opción más simple es usar un administrador de paquetes como [Homebrew](#) para instalar FFmpeg y asegurar que permanezca en la versión más reciente. Para completar este tipo de instalación, sigue estos pasos:

- Instala Homebrew de acuerdo a las instrucciones en el enlace de arriba
- Para comenzar con una instalación básica, ejecuta `brew install ffmpeg` en tu Terminal para comenzar una instalación básica **Nota:** generalmente se recomienda instalar FFmpeg con opciones adicionales a las incluidas en la instalación básica; esto proporcionará acceso a más herramientas y funciones. [La Guía de Instalación de Apple de Reto Kromer](#) proporciona un buen conjunto de opciones adicionales:

```
brew install ffmpeg --with-freetype --with-openjpeg --with-x265 --with-rubberband --with-tesseract
```

- Para una explicación de estas opciones adicionales, revisa [La Guía FFmpeg de Ashley Blewer](#).
- Además, puedes ejecutar `brew options ffmpeg` para ver qué características están o han estado disponibles en la versión actual de FFmpeg
- Para actualizar tu instalación a la versión más reciente, ejecuta:

```
brew update && brew upgrade ffmpeg
```

- Para más opciones de instalación para Mac OS, revisa [La Guía de Compilación de FFmpeg para Mac OS](#) (la guía solo está disponible en inglés).

## **Para usuarios de Windows**

---

Los usuarios de Windows pueden usar el administrador de paquetes [Chocolatey](#) para instalar y mantener FFmpeg. [La Guía de Instalación de Windows de Reto Kromer](#) proporciona toda la información necesaria para usar Chocolatey o construir el *framework* a partir del código fuente (la guía solo está disponible en inglés).

## **Para usuarios de Linux**

---

[Linuxbrew](#) es un programa similar a Homebrew que se puede utilizar para instalar y mantener FFmpeg en Linux. Reto Kromer también proporciona una guía útil, [la Guía de Instalación de Linux](#), que es similar a la instalación en Mac OS. Tu distribución de Linux puede tener su [propio administrador de paquetes](#) que incluye paquetes FFmpeg (la guía solo está disponible en inglés). Dependiendo de tu distribución de Linux (Ubuntu, Fedora, Arch Linux, etc.) estas versiones pueden variar, así que usar Linuxbrew podría ser útil para asegurar que la versión es la misma independientemente del tipo de Linux que utilices.

## **Otros recursos de instalación**

---

- [Descarga de paquetes](#)

FFmpeg permite el acceso a archivos binarios, código fuente y versiones estáticas para Mac, Windows y Linux directamente en su sitio web. Los usuarios pueden construir el *framework* sin un administrador de paquetes con estos recursos. Es probable que solo los usuarios avanzados quieran usar esta opción.

- [La Guía de Compilación de FFmpeg](#)

La página Wiki de FFmpeg también proporciona un compendio de guías y estrategias para instalar FFmpeg en tu computadora (la guía solo está disponible en inglés).

## **Probando la instalación**

---

- Para asegurarte de que FFmpeg se haya instalado correctamente, ejecuta:

```
ffmpeg -version
```

- Si ves una lista larga con información, ¡la instalación fue exitosa! Debe ser similar a lo siguiente:

```
ffmpeg version 4.0.1 Copyright (c) 2000-2018 the FFmpeg developers
built with Apple LLVM version 9.1.0 (clang-902.0.39.1)
configuration: --prefix=/usr/local/Cellar/ffmpeg/4.0.1 --enable-shared --enable-
pthreads --enable-version3 --enable-harded-tables --enable-avresample --cc=clang -
-host-cflags= --host-ldflags= --enable-gpl --enable-ffplay --enable-libfreetype --
enable-libmp3lame --enable-librubberband --enable-libtesseract --enable-libx264 --
enable-libx265 --enable-libxvid --enable-opencl --enable-videotoolbox --disable-lzma
--enable-libopenjpeg --disable-decoder=jpeg2000 --extra-cflags=-
I/usr/local/Cellar/openjpeg/2.3.0/include/openjpeg-2.3
libavcodec      58. 18.100 / 58. 18.100
libavformat     58. 12.100 / 58. 12.100
libavdevice     58.  3.100 / 58.  3.100
libavfilter      7. 16.100 /  7. 16.100
libavresample    4.  0. 0 /  4.  0. 0
libswscale       5.  1.100 /  5.  1.100
libswresample    3.  1.100 /  3.  1.100
libpostproc      55. 1.100 / 55. 1.100
```

Si el sistema arroja `-bash: ffmpeg: command not found`, algo ha ido mal.

Nota: Si estás usando un administrador de paquetes, es improbable que encuentres este mensaje de error. Sin embargo, si hay un problema después de instalar con un administrador de paquetes, es probable que haya un problema con el administrador de paquetes y no con FFmpeg. Consulta la solución de problemas en [Homebrew](#), [Chocolatey](#), o [Linuxbrew](#) para asegurar que el administrador de paquetes está funcionando correctamente en tu computadora (las guías solo están disponibles en inglés). Si estás intentando instalar sin un administrador de paquetes y ves este mensaje de error, haz una referencia cruzada de tu método con la Guía de Compilación de FFmpeg anterior.

## **Usando FFmpeg en el navegador usando-ffmpeg-en-el-navegador**

---

Si no quieras instalar FFmpeg en tu computadora pero te gustaría familiarizarte con el *framework* y usarlo en la interfaz de línea de comandos, [videoconverter.js](#) de Brian Grinstead proporciona un método para ejecutar los comandos FFmpeg en tu navegador (la interfaz está en inglés).

Esta interfaz del navegador no tiene las funcionalidades como para completar todo este tutorial, pero es útil para aprender los comandos esenciales de FFmpeg. Adicionalmente, este recurso opera en una versión anterior de FFmpeg y posiblemente no tenga todas las características de la versión más reciente.

## **Estructura básica y sintaxis de los comandos FFmpeg estructura-básica-y-sintaxis-de-los-comandos-ffmpeg**

---

El comando básico tiene cuatro partes:

[Símbolo del Sistema] [Archivo de Entrada] [Banderas/Acciones] [Archivo de Salida]

- Cada comando comenzará con un símbolo del sistema. Dependiendo del uso, este será `ffmpeg` (cambiar archivos), `ffprobe` (generar metadatos de archivos) o `ffplay` (reproducir archivos).
- Los archivos de entradas son los archivos que están siendo leídos, editados o examinados.
- Las banderas y acciones son las cosas que le estás diciendo a FFmpeg que haga con los archivos de entrada. La mayoría de los comandos contendrán múltiples banderas y acciones de complejidad variable.
- Los archivos de salida son los archivos creados por el comando o los informes creados por los comandos de `ffprobe`.

Escrito genéricamente, el comando básico es parecido a lo siguiente:

```
ffmpeg -i /ruta_de_archivo/archivo_de_entrada.ext -bandera alguna_acción
/ruta_de_archivo/archivo_de_salida.ext
```

Como con cualquier interfaz de línea de comandos, tendrás que escribir las rutas de los archivos de entrada y de salida dependiendo de las ubicaciones de tus directorios de trabajo. En los ejemplos proporcionados en este tutorial, las rutas de archivos no estarán escritas completamente y se supone que el usuario ha navegado al directorio de trabajo para ejecutar los comandos.

A continuación, examinaremos algunos ejemplos de varios comandos diferentes que usan esta estructura y sintaxis. Adicionalmente, estos comandos demostrarán algunas de las características más útiles de FFmpeg y nos permitirán familiarizarnos con la forma en que se construyen los archivos audiovisuales digitales.

## Para empezar

---

Para este tutorial, utilizaremos una película archivística que se llama *Destination Earth* como nuestro objeto de estudio. Esta película está publicada por los [Archivos Prelinger](#) y en el [Internet Archive](#). Esta película, estrenada en 1956 y producida por [El American Petroleum Institute](#) y [John Sutherland Productions](#), es un excelente ejemplo de la propaganda de la época de la Guerra Fría que exalta las virtudes del capitalismo y el estilo de vida estadounidense. Utilizando el proceso de [Technicolor](#), este corto animado de ciencia ficción cuenta la historia de una sociedad marciana que vive bajo un gobierno opresivo y sus esfuerzos para mejorar sus métodos industriales. Envían un emisario a la Tierra que descubre que la clave para esto es la refinación de petróleo y la libre empresa. Utilizaremos el vídeo para introducir algunas de las funcionalidades básicas de FFmpeg y analizar sus propiedades de color con relación a su retórica propagandística.



Destination Earth (1956)

En este tutorial se llevarán a cabo los siguientes pasos:

- Navegar a la página de *Destination Earth* en el Internet Archive
- Descargar dos archivos vídeos: las versiones “MPEG4” (extensión de archivo `.m4v`) y “OGG” (extensión de archivo `.ogv`) de la película
- Guardar estos archivos en la misma carpeta en algún lugar de tu computadora. Guárdalos con los nombres de archivos `destEarth`, seguido por su extensión.

Tómate unos minutos para ver el vídeo y tener una idea de su estructura, mensaje y motivos visuales antes de continuar con los siguientes comandos.

## Ejemplos de comandos preliminares

---

**Ver metadatos básicos con FFprobever-metadatos-básicos-con-  
ffprobe**

---

Antes de comenzar a manipular nuestros archivos `destEarth`, usemos FFmpeg para examinar información básica sobre el archivo utilizando un simple comando de `ffprobe`. Esto ayudará a comprender cómo se construyen los archivos audiovisuales digitales y proporcionará una base para el resto del tutorial. Navega hasta el directorio del archivo y ejecuta:

```
ffprobe destEarth.ogv
```

Verás los metadatos técnicos básicos del archivo impresos en `stdout`:

```
wg-dhcp174d160d021:ph-video daverodriguez$ ffprobe destEarth.ogv ← Comando Original
ffprobe version 4.0.1 Copyright (c) 2007-2018 the FFmpeg developers
  built with Apple LLVM version 9.1.0 (clang-902.0.39.1)
configuration: --prefix=/usr/local/Cellar/ffmpeg/4.0.1 --enable-shared --enable-pthreads --enable-version3 --enable-hardcoded-tables --enable-avresample --cc=clang --host-cflags= --host-ldflags= --enable-gpl --enable-fplay --enable-libfreetype --enable-libmp3lame --enable-librubberband --enable-libtesseract --enable-libx264 --enable-libx265 --enable-libxvid --enable-opencp --enable-videotoolbox --disable-lzma --enable-libopenjpeg --disable-decoder-jpeg2000 --extra-cflags=-I/usr/local/Cellar/openjpeg/2.3.0/include/openjpeg-2.3
libavutil      56. 14.100 / 56. 14.100
libavcodec     58. 18.100 / 58. 18.100
libavformat    58. 12.100 / 58. 12.100
libavdevice    58.  3.100 / 58.  3.100
libavfilter     7. 16.100 /  7. 16.100
libavresample   4.   0.   0 /  4.   0.   0
libswscale      5.  1.100 /  5.  1.100
libswresample   3.  1.100 /  3.  1.100
libpostproc    55.  1.100 / 55.  1.100
Input #0, ogg, from 'destEarth.ogv': ← Metadatos del contenedor
  Duration: 00:13:39.62, start: 0.000000, bitrate: 595 kb/s
    Stream #0:0: Video: theora, yuv420p, 400x300 [SAR 1:1 DAR 4:3], 29.97 fps, 29.97 tbr, 29.97 tbn, 29.97 tbc ← Metadatos de la pista de video
      Metadata:
        TITLE       : Destination Earth
        DATE        : 1956
        LOCATION    : http://archive.org/details/4050_Destination_Earth_01_47_33_28
        ENCODER     : Lavf54.49.102
    Stream #0:1: Audio: vorbis, 44100 Hz, stereo, fltp, 128 kb/s ← Metadatos de la pista de audio
      Metadata:
        TITLE       : Destination Earth
        DATE        : 1956
        LOCATION    : http://archive.org/details/4050_Destination_Earth_01_47_33_28
        ENCODER     : Lavf54.49.102
```

El output de un comando básico `ffprobe` con destEarth.ogv

La línea `Input # 0` del informe identifica el **contenedor** como `ogg`. Los contenedores (también llamados “envoltorios” o “wrappers”, en inglés) proporcionan al archivo la estructura de sus diversas pistas. Los diferentes contenedores (otros más comunes incluyen `.mkv`, `.avi` y `.flv`) tienen diferentes características y compatibilidad con diversos programas. Examinaremos cómo y por qué es posible que deseas cambiar el contenedor de un archivo en el siguiente comando.

Las líneas `Stream #0:0` y `Stream #0:1` proporcionan información sobre las pistas del archivo (es decir, el contenido que ves en la pantalla y escuchas a través de sus altavoces) y también identifican el **códec** de cada pista. Los códecs especifican cómo se codifica/comprime (se escribe y almacena) y se decodifica (se reproduce) la información. La pista vídeo (`Stream #0:0`) de nuestro archivo `.ogv` usa el códec `theora` y la pista audio (`Stream #0:1`) usa el códec `vorbis`. Estas líneas también proporcionan información importante relacionada con el espacio de color de la pista de vídeo (`yuv420p`), resolución (`400x300`) y marcos por segundo (`29.97 fps`). Adicionalmente, proporcionan información de audio como la tasa de muestreo (`44100 Hz`) y la tasa de bits (`128 kb/s`).

Los códecs, en mayor medida que los contenedores, determinan la calidad y la compatibilidad de un archivo audiovisual con diferentes programas y plataformas (otros códecs comunes incluyen `DNXHD` y `ProRes` para vídeo y `mp3` y `FLAC` para audio). Examinaremos cómo y por qué es posible que también deseas cambiar el códec de un archivo en el siguiente comando.

Ejecuta otro comando de `ffprobe`, esta vez con el archivo `.m4v`:

```
ffprobe destEarth.m4v
```

Una vez más, verás los metadatos técnicos básicos impresos en el `stdout`:

```
wg-dhcp174d160d021:ph-video daverodriguez$ ffprobe destEarth.m4v ← Comando original
ffprobe version 4.0.1 Copyright (c) 2007-2018 the FFmpeg developers
  built with Apple LLVM version 9.1.0 (clang-902.0.39.1)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/4.0.1 --enable-shared --enable-pthreads --enable-version3 --enable-harcdcoded-tables --enable-avresample --cc=clang --host-cflags= --host-ldflags= --enable-gpl --enable-ffplay --enable-libfreetype --enable-libmp3lame --enable-librubberband --enable-libtesseract --enable-libx264 --enable-libx265 --enable-libxvid --enable-openccl --enable-videotoolbox --disable-lzma --enable-libopenjpeg --disable-decoder-jpeg2000 --extra-cflags=-I/usr/local/Cellar/openjpeg/2.3.0/include/openjpeg-2.3
libavutil      56. 14.100 / 56. 14.100
libavcodec     58. 18.100 / 58. 18.100
libavformat    58. 12.100 / 58. 12.100
libavdevice     58.  3.100 / 58.  3.100
libavfilter     7. 16.100 /  7. 16.100
libavresample   4.  0.  0 /  4.  0.  0
libswscale       5.  1.100 /  5.  1.100
libswresample   3.  1.100 /  3.  1.100
libpostproc     55.  1.100 / 55.  1.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'destEarth.m4v': ← Metadatos del contenedor
  Metadata:
    major_brand : mp42
    minor_version : 1
    compatible_brands: mp42mp41
    creation_time : 2013-08-03T13:58:40.000000Z
    Duration: 00:13:39.62, start: 0.000000, bitrate: 3162 kb/s
      Stream #0:0(eng): Video: h264 (Constrained Baseline) (avc1 / 0x31637661), yuv420p(tv, smpte170m/smpte170m/bt709), 640x480, 2999 kb/s, 29.97 fps,
29.97 tbr, 2997 tbn, 5994 tbc (default) ← Metadatos de la pista de video
      Metadata:
        creation_time : 2013-08-03T13:58:40.000000Z
        handler_name : Apple Video Media Handler
      Stream #0:1(eng): Audio: aac (LC) (mp4a / 0x6134706D), 48000 Hz, stereo, fltp, 159 kb/s (default) ← Metadatos de la pista de audio
      Metadata:
        creation_time : 2013-08-03T13:58:40.000000Z
        handler_name : Apple Sound Media Handler
```

El output de un comando básico `ffprobe` con `destEarth.m4v`

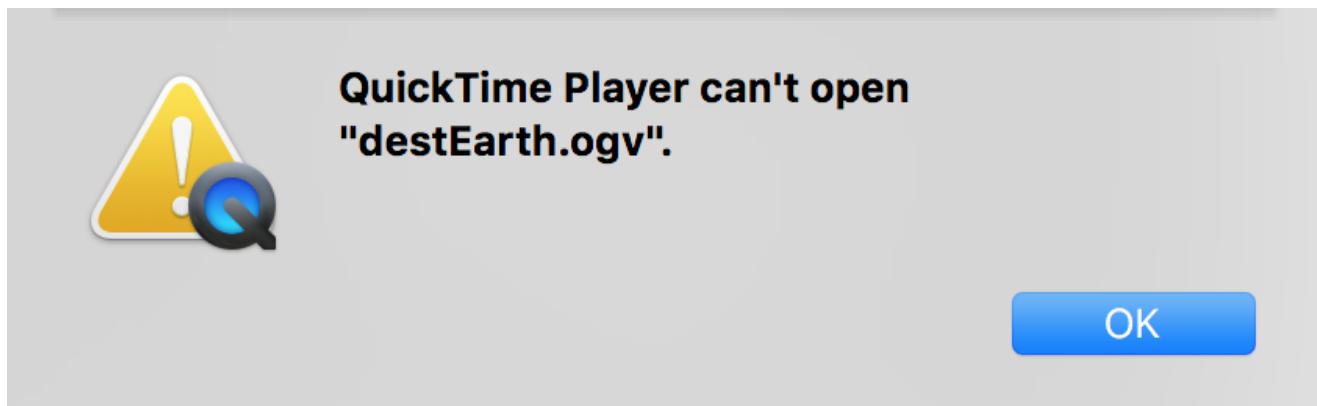
También notarás que el informe para el archivo `.m4v` contiene múltiples contenedores en la línea `Input # 0` como `mov` y `m4a`. No es necesario profundizar en los detalles para los fines de este tutorial, pero ten en cuenta que los contenedores `mp4` y `mov` se presentan en múltiples “sabores” y diferentes extensiones de archivo. Sin embargo, todos son muy similares en su construcción técnica y, como tal, pueden verse agrupados en metadatos técnicos. De manera similar, el archivo `ogg` tiene la extensión `.ogv`, un “sabor” o variante del formato `ogg`.

Al igual que en nuestro comando anterior, las líneas `Stream # 0: 0` y `Stream # 0: 1` identifican el códec de cada pista. Podemos ver que nuestro archivo `.m4v` usa el códec vídeo H.264 y el códec audio aac. Ten en cuenta que se nos proporcionan metadatos similares a nuestro archivo `.ogv`, pero algunas características importantes relacionadas con el análisis visual (como la resolución) son significativamente diferentes. Nuestro `.m4v` tiene una resolución más alta (`640x480`) y, por lo tanto, utilizaremos esta versión de *Destination Earth* como nuestro vídeo de origen.

Ahora que sabemos más sobre la composición técnica de nuestro archivo, podemos comenzar a explorar las características y funcionalidades transformadoras de FFmpeg (volveremos a utilizar `ffprobe` más adelante en el tutorial para realizar una extracción de metadatos de color más avanzada).

## **Cambiar el contenedor (volver a envolver, “re-wrap”)cambiar-el-contenedor-volver-a-envolver-re-wrap**

Dependiendo de tu sistema operativo, puedes tener uno o más reproductores de medios instalados. Para efectos de demostración veamos qué sucede si intentas abrir `destEarth.ogv` usando el reproductor de medios QuickTime que viene con Mac OSX:



Los reproductores multimedia patentados como Quicktime a menudo están limitados en los tipos de archivos con los que pueden trabajar.

Una opción cuando te enfrentas a un mensaje de este tipo es simplemente usar otro reproductor de medios. VLC, que está construido con FFmpeg, es una excelente alternativa de código abierto, pero simplemente “usar otro programa” puede no ser siempre una solución viable (y es posible que no siempre tengas otra versión de archivo con la que trabajar). Muchos editores de vídeo populares, como Adobe Premiere, Final Cut Pro y DaVinci Resolve, tienen sus propias limitaciones en cuanto a los tipos de formatos con los que son compatibles. Además, las diferentes plataformas web y sitios de alojamiento/transmisión, como Vimeo, también tienen sus propios requisitos. Por lo tanto, es importante poder volver a envolver y transcodificar tus archivos para cumplir con las diversas especificaciones para la reproducción, edición, publicación digital y ajuste de archivos a los estándares requeridos por las plataformas de archivo o preservación digital.

Para obtener una lista completa de los códigos y contenedores compatibles con tu instalación de FFmpeg, ejecuta `ffmpeg -codecs` y `ffmpeg -formats`, respectivamente, para ver la lista impresa de tu `stdout`.

Como un ejercicio para aprender la sintaxis básica de FFmpeg y aprender a transcodificar entre formatos, comenzaremos con nuestro archivo `destEarth.ogv` y escribiremos un nuevo archivo con vídeo codificado en `H.264`, audio en `AAC` y envuelto en un contenedor

.mp4 , una combinación muy común y altamente portátil de códecs y contenedores que es prácticamente idéntico al archivo .m4v que originalmente descargamos. Aquí está el comando que ejecutarás, junto con una explicación de cada parte de la sintaxis:

```
ffmpeg -i destEarth.ogv -c:v libx264 -c:a aac destEarth_transcoded.mp4
```

- **ffmpeg** = comienza el comando
- **-i destEarth.ogv** = especifica el archivo de entrada
- **-c:v libx264** = transcodifica la pista de vídeo al codec H.264
- **-c:a aac** = transcodifica la pista de audio al codec AAC
- **destEarth\_transcoded.mp4** = especifica el archivo de salida. Ten en cuenta que aquí es donde se especifica el nuevo tipo de contenedor.

Si ejecutas como está escrito y en el mismo directorio que **destEarth.ogv** , verás un nuevo archivo llamado **destEarth\_transcoded.mp4** , que aparecerá en el directorio. Si estás operando en Mac OSX, también podrás reproducir este nuevo archivo con QuickTime. Una exploración completa de los convenios de códecs, contenedores, compatibilidad y extensión de archivos está más allá del alcance de este tutorial; sin embargo, este conjunto de ejemplos preliminares debería darles a aquellos que no estén familiarizados con la forma en que se construyen los archivos audiovisuales digitales un conjunto de conocimientos de referencia que les permitirá completar el resto del tutorial.

## **Creación de extractos y “demuxing” de audio y vídeo**

---

Ahora que tenemos un mejor entendimiento de las pistas, códecs, y contenedores, veamos formas en que FFmpeg puede trabajar con materiales de vídeo a un nivel más granular. Para este tutorial, examinaremos dos secciones separadas de *Destination Earth* para comparar cómo se usa el color en relación con la retórica propagandística de la película. Crearemos y prepararemos estos extractos para el análisis utilizando un comando que realiza dos funciones diferentes simultáneamente:

- Primero, el comando creará dos extractos de **destEarth.m4v** .
- Segundo, el comando eliminará (“demux”) los componentes de audio ( Stream # 0: 1 ) de estos extractos.

Estamos eliminando el audio para ahorrar espacio de almacenamiento (la información de audio no es necesaria para el análisis de color). Esto probablemente será útil si esperas utilizar este tipo de análisis a escalas más grandes. Cerca del final del tutorial, se discutirá más información sobre la ampliación del análisis de color.

El primer extracto que haremos contiene una secuencia correspondiente al comienzo de la película que describe las difíciles condiciones y la vida oprimida de la sociedad marciana. El siguiente comando especifica los puntos de inicio y finalización del extracto, le dice a FFmpeg

que retenga toda la información en la pista de vídeo sin transcodificar nada y le indica que escriba nuestro nuevo archivo sin la pista de audio:

```
ffmpeg -i destEarth.m4v -ss 00:01:00 -to 00:04:35 -c:v copy -an  
destEarth_Mars_video.mp4
```

- `ffmpeg` = comienza el comando
- `-i destEarth.m4v` = especifica el archivo de entrada
- `-ss 00:01:00` = establece el punto de inicio a 1 minuto del inicio del archivo
- `-to 00:04:45` = establece el punto final a 4 minutos y 45 segundos desde el inicio del archivo
- `-c:v copy` = copia la pista de vídeo directamente, sin transcodificar
- `-an` = le dice a FFmpeg que ignore la pista de audio al escribir el archivo de salida.
- `destEarth_Mars_video.mp4` = especifica el archivo de salida



Vida en Marte

Ahora, ejecutaremos un comando similar para crear un extracto de “Tierra”. Esta parte de la película tiene una secuencia similar que describe las maravillas de la vida en la Tierra y la riqueza de su sociedad gracias al capitalismo de libre empresa y al uso de petróleo y productos derivados de este:

```
ffmpeg -i destEarth.m4v -ss 00:07:30 -to 00:11:05 -c:v copy -an  
destEarth_Earth_video.mp4
```



La abundancia de la Tierra

Ahora deberías tener dos archivos nuevos en tu directorio llamados

`destEarth_Mars_video.mp4` y `destEarth_Earth_video.mp4`. Puedes probar uno o ambos archivos (o cualquiera de los otros archivos en el directorio) usando la función `ffplay` de FFmpeg. Simplemente ejecuta:

```
ffplay destEarth_Mars_video.mp4
```

y/o

```
ffplay destEarth_Earth_video.mp4
```

Verás una ventana abierta y el vídeo comenzará en el punto de inicio especificado. Se reproducirá una vez y luego la ventana se cerrará (además, notarás que no hay sonido en tu vídeo). También notarás que los comandos `ffplay` no requieren que se especifique una entrada (`-i`) o una salida porque la reproducción en sí misma es la salida.

**FFplay** es un reproductor multimedia muy versátil que viene con una serie de opciones para personalizar la reproducción. Por ejemplo, si agregas ` -loop 0` al comando se reproducirá en bucle indefinidamente.

Ahora hemos creado nuestros dos extractos para el análisis. Si vemos estos clips por separado, parece haber diferencias significativas en la forma en que se utilizan el color y la variedad de colores. En la siguiente parte del tutorial examinaremos y extraeremos datos de los archivos de vídeo para cuantificar y apoyar esta hipótesis.

## **Análisis de datos de color**

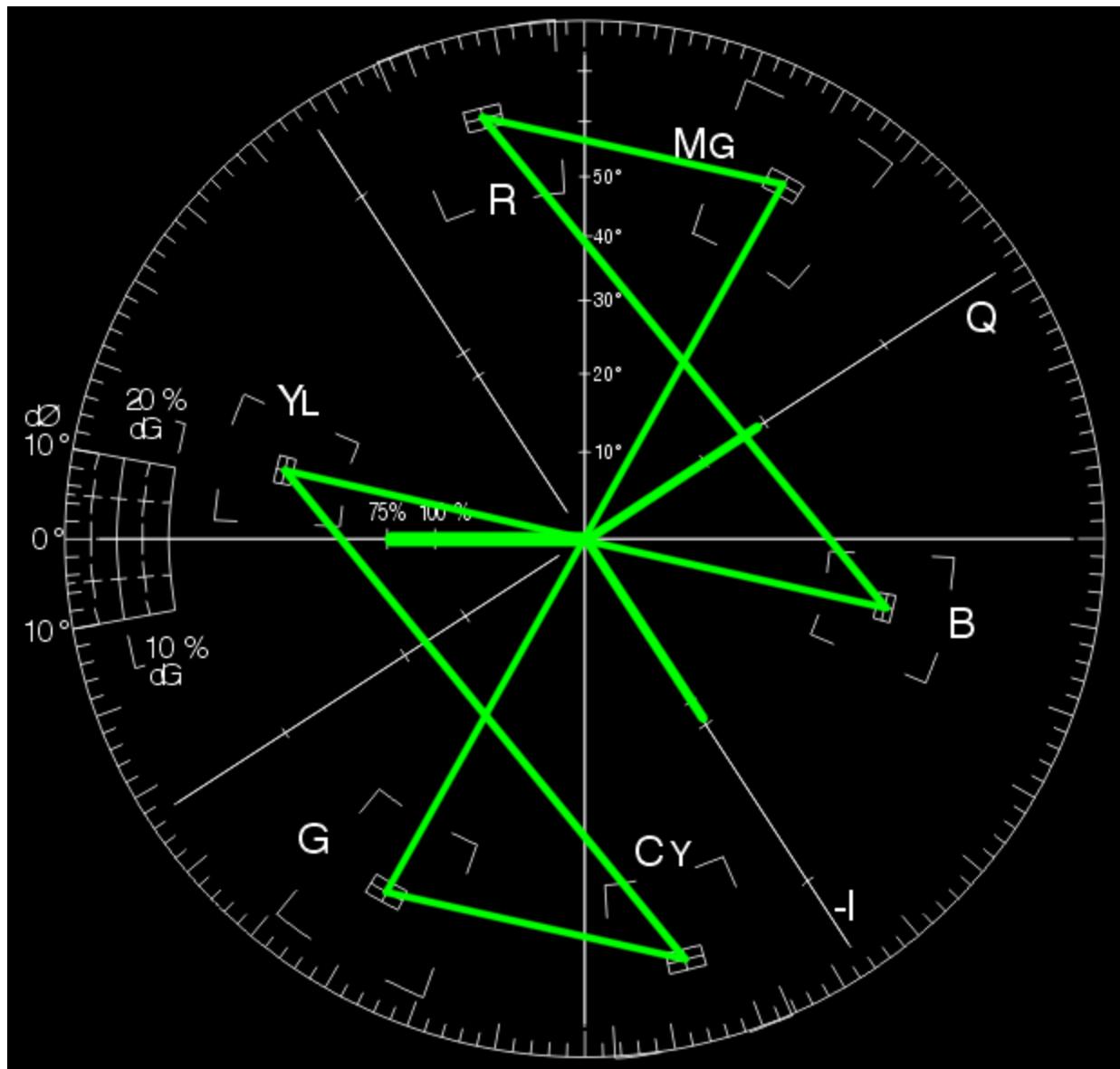
---

El uso de herramientas digitales para analizar la información de color en películas es otra faceta emergente de las Humanidades Digitales que se superpone con los estudios cinematográficos tradicionales. En particular, el proyecto FilmColors de la Universidad de Zurich cuestiona la intersección crítica de las “características estéticas formales de los aspectos semánticos, históricos y tecnológicos” de su producción, recepción y difusión a través del uso de herramientas de análisis y anotación digital (Flueckiger, 2017, traducido por el autor). Aunque no hay un método estandarizado para este tipo de investigación, en el momento de escribir esta lección el comando **ffprobe** que se describe a continuación es una una herramienta útil para extraer información de color que se puede usar en el análisis computacional. Primero, veamos otra manera estandarizada de representar la información de color que informa este enfoque cuantitativo, basado en datos, para el análisis de color: los vectorscopios.

## **Vectorscopios**

---

Durante años, profesionales del vídeo han confiado en los vectorscopios para ver la información del color de una manera estandarizada y fácilmente legible. Un vectorscopio grafica información de color en una gráfica circular. La posición del gráfico corresponde a los tonos particulares encontrados en una señal de vídeo. Otros factores, como la saturación, determinan también el tamaño de un gráfico. A continuación se presenta un ejemplo de un vectorscopio que muestra los valores de color de las barras SMPTE.



Una lectura de vectorescopio que representa las barras SMPTE NTSC estándar. Fuente: Wikimedia Commons



Las barras SMPTE. Fuente: Wikimedia Commons

FFmpeg se puede utilizar para reproducir y crear archivos de vídeo con vectorscopios integrados en ellos para proporcionar una referencia en tiempo real para la información de color del vídeo. Los siguientes comandos `ffplay` incorporarán un vectorscopio en la esquina inferior derecha del marco. A medida que se reproduce el vídeo, notarás el cambio en el gráfico del vectorscopio a medida que cambia el color en pantalla:

```
ffplay destEarth_Mars_video.mp4 -vf "split=2[m][v], [v]vectorscope=b=0.7:m=color3:g=green[v], [m][v]overlay=x=w-w:y=h-h"
```

- `ffplay` = comienza el comando
- `-i entrada_archivo.ext` = la ruta y el nombre del archivo de entrada
- `-vf` = crea un *filter-graph* para usar con las pistas
- `"` = una comilla para comenzar el *filter-graph*. La información entre las comillas especifica los parámetros de la apariencia y posición del vectorscopio
- `split=2[m][v]` = divide la entrada en dos salidas idénticas llamadas `[m]` y `[v]`
- `,` = la coma indica que viene otro parámetro
- `[v]vectorscope=b=0.7:m=color3:g=green[v]` = asigna la salida `[v]` al filtro del vectorscopio

- `[m][v]overlay=x=W-w:y=H-h` = superpone el vectorscopio encima de la imagen de vídeo en una cierta ubicación (en este caso, en la esquina inferior derecha de la pantalla)
- `"` = termina el *filter-graph*

Para obtener más información sobre las diversas opciones para crear vectorscopios, consulta [la documentación oficial](#) y [la página Wiki FFmpeg Vectorscope](#). Además, puedes encontrar más información sobre cómo colocar las superposiciones en [la documentación del filtro de superposición FFmpeg](#).



Captura de pantalla de la ventana de FFplay con vectorscopio incorporado

Y para el extracto de “Tierra”:

```
ffplay destEarth_Earth_video.mp4 -vf "split=2[m][v], [v]vectorscope=b=0.7:m=color3:g=green[v], [m][v]overlay=x=W-w:y=H-h"
```



Captura de pantalla de la ventana de FFplay con vectorscopio incorporado

También podemos ajustar este comando para escribir nuevos archivos de vídeo con vectorscopios:

```
ffmpeg -i destEarth_Mars_video.mp4 -vf "split=2[m][v],
[v]vectorscope=b=0.7:m=color3:g=green[v],[m][v]overlay=x=w-w:y=H-h" -c:v libx264
destEarth_Mars_vectorscope.mp4
```

```
ffmpeg -i destEarth_Earth_video.mp4 -vf "split=2[m][v],
[v]vectorscope=b=0.7:m=color3:g=green[v],[m][v]overlay=x=w-w:y=H-h" -c:v libx264
destEarth_Earth_vectorscope.mp4
```

Nota los pequeños pero importantes cambios en sintaxis:

- Hemos agregado una bandera de `-i` porque es un comando de `ffmpeg`
- Hemos especificado el códec del vídeo del archivo de salida como H.264 con la bandera `-c:v libx264` y no estamos recodificando el códec de audio (`-c:a copy`), aunque puedes especificar otro códec de audio si lo necesitas.
- Hemos definido el nombre del archivo de salida

Tómate unos minutos para ver estos videos con los vectorscopios integrados en ellos. Observa cuán dinámicos (o no) son los cambios entre los extractos de “Marte” y “Tierra”. Compara lo que ves en el vectorscopio con tus propias impresiones del video mismo. Podríamos usar las observaciones de estos vectorscopios para hacer determinaciones sobre qué tonos de color aparecen de manera más regular o intensa en el video, o podemos comparar diferentes formatos uno al lado del otro para ver cómo el color se codifica o representa de manera diferente en función de diferentes códecs, resoluciones, etc.

Aunque los vectorscopios proporcionan una representación útil y en tiempo real de la información del color, es posible que también deseemos acceder a los datos sin procesar que se encuentran debajo de ellos. Luego, podemos usar estos datos para desarrollar visualizaciones más flexibles que no dependan de ver el archivo de video simultáneamente y que ofrezcan un enfoque más cuantitativo para el análisis de color. En nuestros próximos comandos, utilizaremos `ffprobe` para producir un conjunto tabular de datos que pueda usarse para crear un gráfico de datos de color.

## Extracción de datos de color con FFprobe [extracción-de-datos-de-color-con-ffprobe](#)

---

Al comienzo de este tutorial, utilizamos un comando `ffprobe` para ver los metadatos básicos de nuestro archivo impresos en el `stdout`. En los siguientes ejemplos, utilizaremos `ffprobe` para extraer datos de color de nuestros extractos de video y enviar esta información a archivos `.csv`. Dentro de nuestro comando `ffprobe`, vamos a utilizar el filtro `signalstats` para crear reportes `.csv` de información de tono de color medio para cada marco en la secuencia de video de `destEarth_Mars_video.mp4` y `destEarth_Earth_video.mp4`, respectivamente.

```
ffprobe -f lavfi -i movie=destEarth_Mars_video.mp4,signalstats -show_entries frame=pkt_pts_time:frame_tags=lavfi.signalstats.HUEMED -print_format csv > destEarth_Mars_hue.csv
```

- `ffprobe` = comienza el comando
- `-f lavfi` = especifica el dispositivo de entrada virtual `libavfilter` como el formato elegido. Esto es necesario cuando se usa `signalstats` y muchos filtros en comandos FFmpeg más complejos.
- `-i movie=destEarth_Mars_video.mp4` = nombre del archivo de entrada
- `,signalstats` = especifica el uso del filtro `signalstats` con el archivo de entrada
- `-show_entries` = establece una lista de entradas que se mostrarán en el informe. Estos se especifican en las siguientes opciones.
- `frame=pkt_pts_time` = especifica mostrar cada marco con su correspondiente `pkt_pts_time`, creando una entrada única para cada marco de video
- `:frame_tags=lavfi.signalstats.HUEMED` = crea una etiqueta para cada marco que contiene el valor de tono medio
- `-print_format csv` = especifica el formato del informe de metadatos

- > destEarth\_Mars\_hue.csv = escribe un nuevo archivo .csv que contiene el informe de metadatos usando > , un operador de redireccionamiento de Bash. Este operador toma el comando que lo precede y “redirige” la salida a otra ubicación. En este caso, está escribiendo la salida en un nuevo archivo .csv . La extensión de archivo proporcionada aquí también debe coincidir con el formato especificado por el indicador print\_format .

A continuación, ejecuta el mismo comando para el extracto de “Tierra”:

```
ffprobe -f lavfi -i movie=destEarth_Earth_video.mp4,signalstats -show_entries
frame=pkt_pts_time:frame_tags=lavfi.signalstats.HUEMED -print_format csv >
destEarth_Earth_hue.csv
```

Para obtener más información sobre el filtro de signalstats y las diversas métricas que se pueden extraer de las transmisiones de vídeo, consulta [la documentación del filtro FFmpeg](#). Ahora deberías tener dos archivos .csv en tu directorio. Si los abres en un editor de texto o en un programa de hoja de cálculo, verás tres columnas de datos:

### destEarth\_Earth\_hue.csv

```
1 frame,0.450033,116
2 frame,0.483400,117
3 frame,0.516767,117
4 frame,0.550133,115
5 frame,0.583500,113
6 frame,0.616867,113
7 frame,0.650234,114
8 frame,0.683600,112
9 frame,0.716967,116
10 frame,0.750334,116
11 frame,0.783700,112
12 frame,0.817067,111
13 frame,0.850434,110
14 frame,0.883800,110
15 frame,0.917167,110
16 frame,0.950534,111
17 frame,0.983901,111
18 frame,1.017267,110
19 frame,1.050634,110
```

```
19 frame,1.050007,110
20 frame,1.084001,110
21 frame,1.117367,110
22 frame,1.150734,110
23 frame,1.184101,110
24 frame,1.217467,111
25 frame,1.250834,111
26 frame,1.284201,111
27 frame,1.317568,111
28 frame,1.350934,111
29 frame,1.384301,111
30 frame,1.417668,111
31 frame,1.451034,110
32 frame,1.484401,111
33 frame,1.517768,111
34 frame,1.551134,112
35 frame,1.584501,112
36 frame,1.617868,113
37 frame,1.651235,114
```

Las primeras filas de nuestro informe de color en formato .csv

Comenzando a la izquierda y moviéndose a la derecha, las dos primeras columnas nos dan información sobre dónde estamos en el vídeo. Los números decimales representan fracciones de segundo que también corresponden aproximadamente a la base de tiempo de vídeo de 30 marcos por segundo. Cada fila en nuestro `.csv` corresponde a un marco de vídeo. La tercera columna lleva un número entero entre 0-360, valor que representa el tono medio para ese marco de vídeo. Estos números son los datos cuantitativos subyacentes del diagrama de vectorescopio y corresponden a su posición (en radianes) en la gráfica circular. Haciendo referencia a nuestra imagen de vectorescopio de antes, puedes ver que comenzando en la parte inferior del círculo (0 grados) y moviéndose a la izquierda, los “verdes” comienzan alrededor de los 38 grados, los “amarillos” en los 99 grados, los “rojos” en los 161 grados, los “magentas” en los 218 grados, los “azules” en los 279 grados y los “cianes” en los 341 grados. Una vez que comprendas estos “rangos” de tono, puedes hacerte una idea de cuál es el valor de tono medio para un marco de vídeo con solo mirar este valor numérico.

Además, ten en cuenta que este valor extraído por el filtro `signalstats` no es una medida absoluta o completa de las cualidades de color de una imagen, sino simplemente un punto de referencia significativo desde el cual podemos explorar una estrategia basada en datos para el análisis de color. La percepción del color y la teoría del color son [áreas complejas y en evolución de la investigación académica](#) que incorporan muchas estrategias diferentes de las humanidades, las ciencias sociales y las ciencias cognitivas. Es por eso que debemos tener en cuenta que cualquier estrategia analítica debe tomarse dentro del contexto de estos discursos más amplios y con un espíritu colaborativo y generativo.

## **Visualizando datos de color**

---

Los dos archivos `.csv` que creamos con los comandos anteriores ahora se pueden usar para crear gráficos que visualicen los datos. Hay una serie de plataformas (tanto propietarias como de código abierto) que se pueden usar para lograr esto, como [Microsoft Excel](#), [RawGraphs](#) y/o [plotly](#). Una discusión en profundidad sobre cómo usar cualquiera de estas plataformas está fuera del alcance de este tutorial; sin embargo, a continuación se muestra la visualización final de los comandos anteriores, que se creó con los archivos `.csv` y `plotly`.

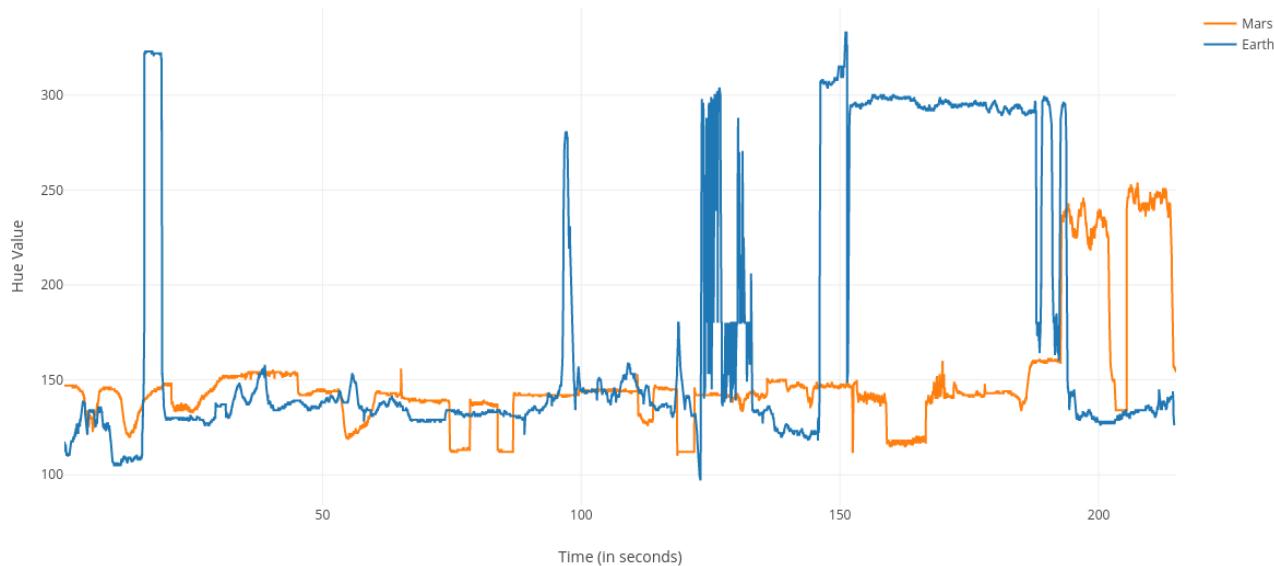


Gráfico que incluye datos de tono medio de ambos extractos de vídeo

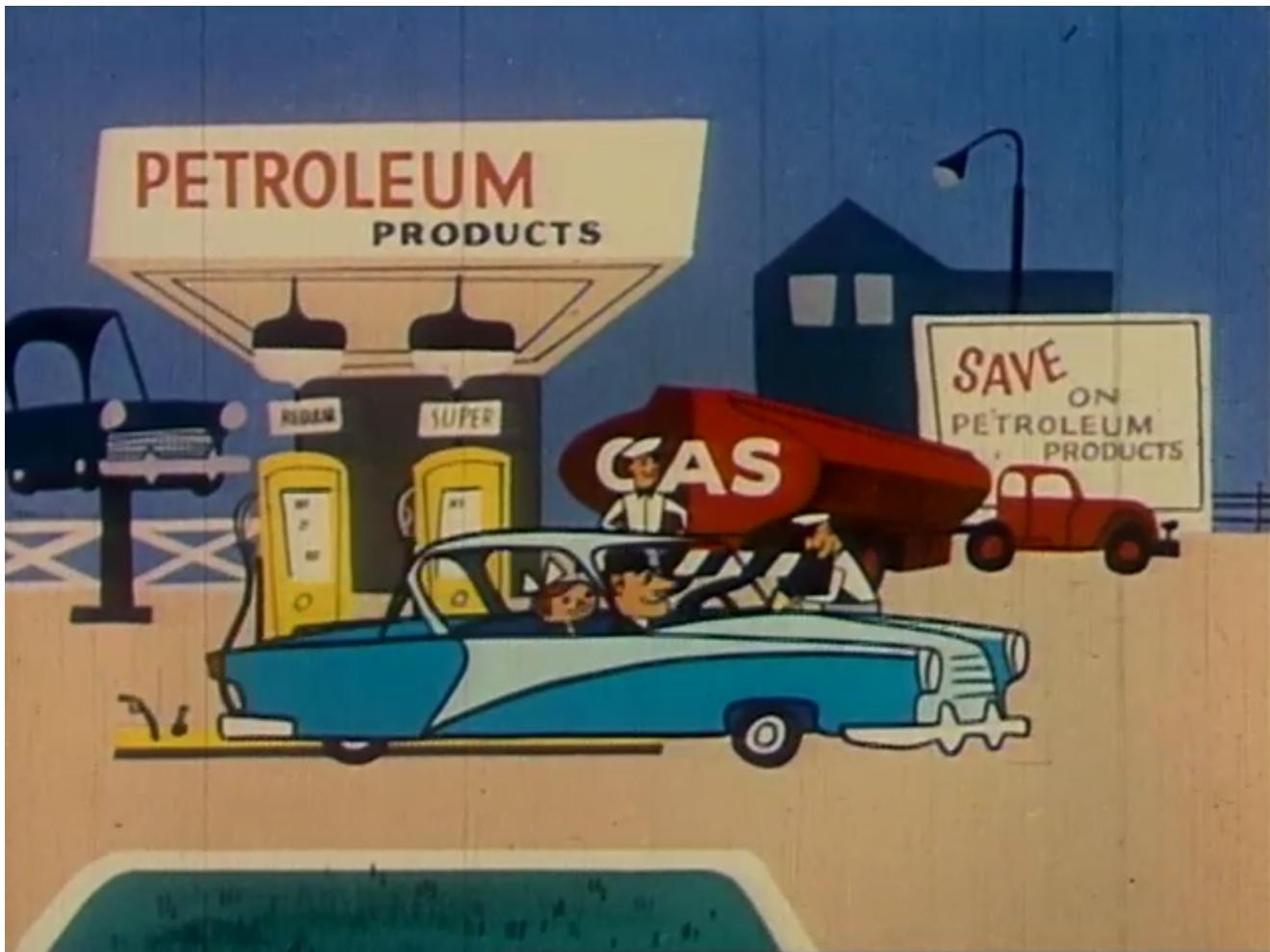
## Conclusiones

Al observar el gráfico, podemos ver que las trazas de Marte y la Tierra tienen rangos dinámicos muy diferentes en sus valores de tono medio. La traza de Marte es muy limitada y se mantiene dentro de los rangos rojo y amarillo (aproximadamente entre 100 y 160) en la mayoría del extracto. Esto sugiere algo sobre el uso del color en la película como un dispositivo retórico que sirve como mensaje propagandístico. Recuerda que esta sección presenta una visión antipática de la forma de vida y el sistema político marcianos: una población uniforme e infeliz, que depende de tecnología y transporte inefficientes mientras se les exige que observen la obediencia total a un gobernante supremo totalitario. La película conecta esta experiencia negativa con una paleta de tonos relativamente opacos de rojo y amarillo. También deberíamos considerar el público objetivo original de esta película, los jóvenes ciudadanos de los Estados Unidos en la década de 1950, y cómo probablemente habrían interpretado estas imágenes y usos del color en ese momento histórico. En particular, podemos considerar este uso del color en el contexto de las crecientes tensiones geopolíticas entre la Unión Soviética y los Estados Unidos y sus aliados en Europa occidental. El color rojo, específicamente, se usaba comúnmente en los medios impresos y de difusión para describir la "amenaza" del comunismo global durante esta era de la historia mundial. Además, la elección de presentar al líder totalitario marciano con una apariencia muy similar al icónico líder soviético Joseph Stalin puede leerse como una señal visual y cultural explícita para la audiencia. Así, esta representación de Marte parece ser una caricatura alegórica de la vida bajo el velo del comunismo, tal como la percibe un observador externo y un oponente político/ideológico. Esta caricatura emplea no solo una paleta de colores limitada, sino una que está cargada con otras referencias culturales. El uso del color aprovecha los prejuicios y

asociaciones que están presentes en el imaginario de la audiencia y, por lo tanto, está ligado estrechamente al argumento central de la película, que sostiene que el comunismo no es un sistema político viable.

En contraste con el uso limitado del color en nuestro extracto de Marte, la traza de la Tierra cubre un rango dinámico mucho más amplio de valores de tono. En este pasaje, el emisario marciano está aprendiendo sobre el maravilloso y rico estilo de vida de los terrícolas gracias a un sistema capitalista y a la explotación de petróleo y de productos derivados de este. La secuencia enfatiza la riqueza material y la libertad empresarial ofrecida bajo un sistema capitalista usando una variedad y vivacidad de color mucho mayor que en el extracto de Marte. Los productos comerciales y las personas se representan utilizando el espectro completo del proceso Technicolor, creando asociaciones positivas entre los resultados de la industria petrolera y el estilo de vida acomodado de quienes se benefician de él. Al igual que el extracto de Marte, a la audiencia se le ofrece una caricatura unilateral de un sistema político y una forma de vida, pero en esta sección la representación reduccionista es laudable y próspera en lugar de desoladora y opresiva.

Como una pieza de propaganda, *Destination Earth* se basa en estas distinciones poderosas pero demasiado simplistas entre dos sistemas políticos para influir en la opinión pública y promover el consumo de productos derivados del petróleo. La manera en que se usa (o no se usa) el color es una herramienta importante para elaborar y enfatizar este mensaje. Además, una vez que podemos extraer datos de color y visualizarlos utiliza técnicas gráficas simples, podemos ver que la disparidad en el rango dinámico proporciona una medida cuantitativa para vincular el uso técnico y estético del color en esta película animada con la retórica propagandística presentada por sus productores.



El petróleo y los ideales estadounidenses de riqueza y prosperidad se expresan en esplendor colorido

## **Escalando el análisis de color con FFprobe**[\*\*escalando-el-análisis-de-color-con-ffprobe\*\*](#)

Uno de los límites de esta metodología es que estamos generando manualmente informes de color en un solo archivo a la vez. Si quisiéramos adoptar un enfoque de visión distante más en línea con las metodologías tradicionales de Humanidades Digitales, podríamos emplear un script de Bash para ejecutar nuestro comando `ffprobe` en todos los archivos en un determinado directorio. Esto es útil si, por ejemplo, un(a) investigador(a) está interesado en realizar un análisis similar en todas las películas animadas de John Sutherland encontradas en la colección de Archivos Prelinger u otro conjunto de material de vídeo de archivo.

Una vez que tengas un conjunto de material para trabajar guardado en un solo lugar, puedes guardar el siguiente bucle for de Bash o “for loop” dentro del directorio y ejecutarlo para generar archivos `.csv` que contengan los mismos datos de tono medio a nivel de fotograma que extrajimos de nuestros extractos de *Destination Earth*.

```

for file in *.m4v; do
ffprobe -f lavfi -i movie="$file",signalstats -show_entries
frame=pkt_pts_time:frame_tags=lavfi.signalstats.HUEMED -print_format csv >
"${file%.m4v}.csv";
done

```

- `for file in *.m4v; do` = inicia el bucle *for*. Esta primera línea le dice a FFmpeg “para todos los archivos en este directorio con la extensión `.m4v` , ejecuta el siguiente comando.”
- El `*` es un comodín de Bash adjunto a un tipo de archivo dado para especificarlos como archivos de entrada.
- La palabra `file` es una variable arbitraria que representará cada archivo a medida que se ejecuta a través del bucle.
- `ffprobe -f lavfi -i movie="$file",signalstats -show_entries frame=pkt_pts_time:frame_tags=lavfi.signalstats.HUEMED -print_format csv > "${file%.m4v}.csv"; done` = el mismo comando de extracción de metadatos de color que ejecutamos en nuestros dos extractos de *Destination Earth*, con algunas pequeñas modificaciones en la sintaxis para explicar su uso en varios archivos en un directorio:
  - `"$file"` = recuerda cada variable. Las comillas aseguran que se conserva el nombre de archivo original.
  - `> "${file%.m4v}.csv";` = conserva el nombre de archivo original al escribir los archivos de salida `.csv` . Esto asegurará que los nombres de los archivos de vídeo originales coincidan con sus correspondientes reportes en `.csv` .
  - `done` = termina el script una vez que se hayan completado todos los archivos del directorio.

También puedes usar `signalstats` para obtener otra información valiosa relacionada con el color. Consulta la [documentación del filtro](#) para obtener una lista completa de las métricas visuales disponibles.

Una vez que ejecutas este script, verás que cada archivo de vídeo en el directorio ahora tiene un archivo `.csv` correspondiente que contiene el conjunto de datos especificado.

## En resumen

---

En este tutorial, hemos aprendido:

- cómo instalar FFmpeg en diferentes sistemas operativos y cómo acceder al *framework* en el navegador web
- cuál es la sintaxis básica y la estructura de los comandos FFmpeg
- cómo visualizar metadatos técnicos básicos de un archivo audiovisual
- cómo transformar un archivo audiovisual a través de la transcodificación y el “re-wrapping”

- cómo analizar y editar ese archivo audiovisual separando sus componentes (“demux”) y crear extractos
- cómo reproducir archivos audiovisuales usando `ffplay`
- cómo crear nuevos archivos de vídeo con vectorscopios integrados
- cómo exportar datos tabulares relacionados con el color de una pista de vídeo usando `ffprobe`
- cómo crear un bucle `for` de Bash para extraer información de datos de color de múltiples archivos de vídeo con un solo comando

A un nivel más amplio, este tutorial aspira a proporcionar una introducción informada y atractiva sobre cómo se pueden incorporar las herramientas y metodologías audiovisuales en los proyectos y las prácticas de Humanidades Digitales. Con herramientas abiertas y potentes como FFmpeg, existe un gran potencial para expandir el alcance del campo para incluir tipos de medios y análisis más ricos y complejos que nunca.

## Más recursos

---

FFmpeg tiene una comunidad grande y bien apoyada de usuarios a través de todo el mundo. Como tal, hay muchos recursos gratuitos y de código abierto para descubrir nuevos comandos y técnicas para trabajar con materiales audiovisuales. Por favor, contacta al autor con cualquier adición a esta lista, especialmente si se trata de recursos educativos en español para aprender FFmpeg.

- [La documentación oficial de FFmpeg](#)
- [FFmpeg Wiki](#)
- [ffmprovisr](#) de [La Asociación de Archivistas de Imágenes en Movimiento](#)
- [Entrenamiento de preservación audiovisual de Ashley Blewer](#)
- [La presentación de Andrew Weaver: “Demystifying FFmpeg”](#)
- [FFmpeg: Presentación de Ben Turkus](#)
- [FFmpeg Cookbook for Archivists](#) de Reto Kromer

## Programas de código abierto de análisis audiovisual que usan FFmpeg FFmpegprogramas-de-código-aberto-de-análisis-audiovisual-que-usan-ffmpeg

---

- [MediaInfo](#)
- [QC Tools](#)

## Referencias

---

- Champion, E. (2017) “Digital Humanities is text heavy, visualization light, and simulation poor,” *Digital Scholarship in the Humanities* 32(S1), i25-i32

- Hockey, S. (2004) "The History of Humanities Computing," A Companion to Digital Humanities, ed. Susan Schreibman, Ray Siemens, John Unsworth. Oxford: Blackwell

Este tutorial fue posible gracias al apoyo de la Academia Británica y fue escrito durante el Taller de *Programming Historian* desarrollado en la Universidad de Los Andes en Bogotá, Colombia, entre el 31 de julio y 3 de agosto de 2018.

## Acerca del autor

---

Dave Rodriguez es archivista audiovisual, curador y cineasta. Actualmente es el bibliotecario residente de la Office of Digital Research and Scholarship de Florida State University.

## Cita sugerida

---

Dave Rodriguez, "Introducción a la transcodificación, edición y visualización de datos audiovisuales con FFmpeg", traducido por Dave Rodriguez y Sebastian Fiori, *Programming Historian en español* 4 (2020), <https://doi.org/10.46430/pheso049>.

## ¡Haz una donación![haz-una-donación](#)

---

Producir buenos tutoriales de acceso abierto cuesta dinero. Únete al creciente número de personas que apoya a *Programming Historian* para que podamos continuar compartiendo conocimientos de forma gratuita.