Package 'compost'

August 4, 2026

Version 0.2.0

Title Video Compositing via 'FFmpeg'

Description Wraps common 'FFmpeg' <https://ffmpeg.org/> filter_complex patterns (overlay, chromakey, concat, scale, vstack) into clean R functions for video compositing, and lowers an 'OpenTimelineIO' <https://github.com/AcademySoftwareFoundation/OpenTimelineIO> timeline (via the 'rotio' package) to a rendered video.

License MIT + file LICENSE

URL https://github.com/cornball-ai/compost

BugReports https://github.com/cornball-ai/compost/issues

Encoding UTF-8

SystemRequirements ffmpeg (>= 5.0)

Imports rotio

Suggests tinytest

NeedsCompilation no

Author Troy Hernandez [aut, cre] (ORCID: <https://orcid.org/0009-0005-4248-604X>), cornball.ai [cph]

Maintainer Troy Hernandez <troy@cornball.ai>

Repository CRAN

Date/Publication 2026-08-04 14:00:08 UTC

Contents

`align_audio` *Locate a clip's audio within a longer narration*

Description

Normalized cross-correlation of amplitude envelopes (10ms resolution by default): returns the offset in seconds at which `clip`'s audio best matches `narration`. Works on video files (their audio stream) and plain audio files alike.

Usage

```
align_audio(clip, narration, sr = 8000L, hop = 80L)
```

Arguments

`clip`	Media file whose audio to place.
`narration`	The continuous reference audio (e.g. the track's `audio.mp3`).
`sr`	Decode sample rate (default 8000).
`hop`	Envelope hop in samples (default 80 = 10ms at 8kHz).

Value

Offset in seconds (numeric), with attributes `correlation` (peak normalized correlation, 0-1ish) and `curve` (correlation per candidate lag).

Examples

```
## Not run:
align_audio("media/chunk02.mp4", "audio.mp3")

## End(Not run)
```

`align_overlaps` *Align a chunk sequence to its narration and derive per-join overlaps*

Description

Places each chunk on the narration clock via [align_audio], then derives each join's overlap from the positions: the seconds by which chunk `j`'s video runs past where chunk `j+1`'s content begins (clamped at 0 = butt join). This is layout truth by construction – a chained chunk's duplicated head and an undersized chunk's shortfall both fall out of where the words actually are.

Usage

```
align_overlaps(files, narration, fps = 24)
```

Arguments

`files`	Character vector of chunk media paths, in order.
`narration`	The continuous narration audio.
`fps`	Frame rate for the `overlap` frame counts (default 24).

Value

A data frame with one row per chunk: `chunk`, `file`, `frames`, `fps`, `offset` (narration seconds), `correlation`, and `overlap` (frames; 0 for the first chunk).

Examples

```
## Not run:
align_overlaps(c("media/chunk01.mp4", "media/chunk02.mp4"), "audio.mp3")

## End(Not run)
```

audio_concat *Concatenate Audio Files with Exact Silence Gaps*

Description

Joins `inputs` in order into one audio file, inserting `gap` seconds of silence between consecutive inputs and `tail` seconds after the last. Every input is resampled and remixed to a common format first, so inputs need not match. One encode, chosen by the output extension.

Usage

```
audio_concat(inputs, output, gap = 0, tail = 0, sample_rate = NULL,
             channels = NULL, bitrate = NULL, overwrite = TRUE, dry_run = FALSE)
```

Arguments

`inputs`	Character vector of audio file paths, in order.
`output`	Path for the output audio file; extension selects the format.
`gap`	Silence in seconds between consecutive inputs (default 0).
`tail`	Silence in seconds appended after the last input (default 0).
`sample_rate`	Output sample rate in Hz, or NULL (default) to use the first input's rate.
`channels`	Output channel count (1 = mono, 2 = stereo), or NULL (default) to use the first input's count.
`bitrate`	Output audio bitrate (e.g. "192k"), or NULL for the encoder default. Ignored by lossless formats such as WAV.
`overwrite`	If TRUE (default), overwrite the output file.
`dry_run`	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns the command string.

Examples

```
## Not run:
audio_concat(c("slide01.wav", "slide02.wav"), "audio.mp3",
             gap = 0.35, tail = 0.5)

## End(Not run)
```

audio_convert *Convert an Audio File*

Description

Re-encode an audio (or A/V) file's audio to a new container, sample rate, channel count, or codec via ffmpeg. The output format is inferred from the output extension. With no optional arguments it is a straight transcode.

Usage

```
audio_convert(input, output, sample_rate = NULL, channels = NULL,
              bitrate = NULL, codec = NULL, overwrite = TRUE, dry_run = FALSE)
```

Arguments

input	Path to input audio (or A/V) file.
output	Path for output file; extension selects the format.
sample_rate	Output sample rate in Hz, or NULL to keep the source rate.
channels	Output channel count (1 = mono, 2 = stereo), or NULL to keep.
bitrate	Output audio bitrate (e.g. "192k"), or NULL for the encoder default. Ignored by lossless formats such as WAV.
codec	Explicit audio codec (e.g. "libmp3lame", "pcm_s16le"), or NULL to let ffmpeg pick from the output extension.
overwrite	If TRUE (default), overwrite the output file.
dry_run	If TRUE, return the ffmpeg command without executing.

Details

Resampling to 16 kHz mono is the usual preparation for speech-to-text: `audio_convert("audio.mp3", "audio.wav", sample_rate = 16000)`.

Value

Invisibly returns the output path. If dry_run, returns the command string.

Examples

```
## Not run:
audio_convert("audio.mp3", "speech.wav", sample_rate = 16000)
audio_convert("take.wav", "take.mp3", bitrate = "192k")

## End(Not run)
```

broadcast_audio *Broadcast-Clean an Audio File*

Description

The standard broadcast chain: a high-pass to kill rumble, optional 60/120/180 Hz hum notches plus FFT denoise, a 4:1 compressor, loudness normalisation to `target_lufs`, and short in/out fades.

Usage

```
broadcast_audio(input, output, dehum = TRUE, target_lufs = -14, fade = 0.06,
                highpass = 80, overwrite = TRUE, dry_run = FALSE)
```

Arguments

`input`	Path to input audio (or A/V) file.
`output`	Path for output file.
`dehum`	Apply the 60/120/180 Hz hum notches + FFT denoise (default TRUE).
`target_lufs`	Integrated loudness target in LUFS (default -14).
`fade`	In/out fade length in seconds (default 0.06).
`highpass`	High-pass corner frequency in Hz (default 80).
`overwrite`	If TRUE (default), overwrite output file.
`dry_run`	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
broadcast_audio("raw.wav", "clean.wav")
broadcast_audio("take.wav", "clean.wav", dehum = FALSE, target_lufs = -16)

## End(Not run)
```

chromakey *Chromakey (Green/Blue Screen) Compositing*

Description

Remove a colored background from foreground footage and composite onto a background video or image using FFmpeg's chromakey filter.

Usage

```
chromakey(background, foreground, output, color = "0x00ff00", similarity = 0.1,
          blend = 0.075, overwrite = TRUE, dry_run = FALSE)
```

Arguments

background	Path to background video or image.
foreground	Path to foreground video with green/blue screen.
output	Path for output video file.
color	Hex color to key out (default "0x00ff00" for green).
similarity	Chromakey similarity threshold (default 0.1). Higher values key out more of the color.
blend	Chromakey blend amount (default 0.075). Higher values create softer edges.
overwrite	If TRUE (default), overwrite output file.
dry_run	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
chromakey("bg.mp4", "greenscreen.mp4", "out.mp4")
chromakey("bg.mp4", "bluescreen.mp4", "out.mp4", color = "0x0000ff")

## End(Not run)
```

colorkey *Colour-Key to a ProRes 4444 File with Alpha*

Description

Key out a flat background colour and write ProRes 4444 (`yuva444p10le`) so the alpha channel survives. Where `chromakey` composites the keyed foreground straight onto a background, this writes a standalone alpha clip you overlay later. By default the key colour is auto-sampled from a corner pixel (generated backgrounds are sometimes black, sometimes grey); pass `color` to override.

Usage

```
colorkey(input, output, color = "auto", similarity = 0.15, blend = 0.05,
         sample_xy = c(0, 0), overwrite = TRUE, dry_run = FALSE)
```

Arguments

`input`	Path to input video.
`output`	Path for output .mov file.
`color`	Hex colour (e.g. "0x000000"), a name ("green"), or "auto" to sample it from a corner pixel (default).
`similarity`	Key similarity threshold (default 0.15).
`blend`	Key edge blend (default 0.05).
`sample_xy`	Pixel c(x, y) to sample when color = "auto" (default c(0, 0)).
`overwrite`	If TRUE (default), overwrite output file.
`dry_run`	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
colorkey("ltx.mp4", "ltx_alpha.mov")
colorkey("scene.mp4", "scene_alpha.mov", color = "0x000000")

## End(Not run)
```

compose_layout *Compose Video Streams onto a Canvas per a Layout*

Description

Paints each named source into its slot's pixel rect on a black canvas, in slot order (first = bottom). `fit = "fill"` covers the rect (scale-up + center-crop, for heads); `fit = "fit"` contains (scale-down + centered pad, for content). Every chain is normalized (`fps`, square pixels, yuv420p), non-reference chains freeze their last frame if shorter, and the output is cut frame-exactly to the reference source's length. The output carries no audio.

Usage

```
compose_layout(sources, output, layout, reference = names(sources)[1],
               overwrite = TRUE, dry_run = FALSE)
```

Arguments

`sources`	Named character vector or list of video paths, keyed by slot name; must cover every slot in `layout$slots`.
`output`	Path for the output video file.
`layout`	A cornball layout: `list(schema, name, canvas,slots)`, with `slots` a named list of `list(rect = c(x, y, w, h), fit = "fill"|"fit")` in integer pixels (see `system.file("schema", "cornball-layout.md",package = "compost")`).
`reference`	Slot name whose source defines the output frame rate and length (default the first source).
`overwrite`	If TRUE (default), overwrite the output file.
`dry_run`	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns the command string.

Examples

```
## Not run:
layout <- list(schema = 1L, name = "vertical", canvas = c(1080L, 1920L),
               slots = list(
                   narrator = list(rect = c(0L, 0L, 1080L, 960L),
                                   fit = "fill"),
                   visual = list(rect = c(0L, 960L, 1080L, 960L),
                                 fit = "fit")))
compose_layout(c(narrator = "head.mp4", visual = "slides.mp4"),
               "composed.mp4", layout, reference = "visual")

## End(Not run)
```

concat *Concatenate Video Segments*

Description

Join multiple video files sequentially using FFmpeg's concat demuxer. All inputs should have compatible codecs and dimensions.

Usage

```
concat(inputs, output, overwrite = TRUE, dry_run = FALSE)
```

Arguments

inputs	Character vector of paths to input video files.
output	Path for output video file.
overwrite	If TRUE (default), overwrite output file.
dry_run	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
concat(c("part1.mp4", "part2.mp4", "part3.mp4"), "full.mp4")

## End(Not run)
```

crossfade_concat *Crossfade-concatenate clips, optionally overlaying one audio track*

Description

Joins `videos` in order, dissolving each transition over `fade` seconds (the duplicated conditioning overlap, so the blend is between near-identical frames and is invisible). When `audio` is given it becomes the sole audio track – the right move for chained chunks, whose per-chunk audio has a silent conditioning head; the continuous narration has no such gaps. Without `audio`, the first clip's audio is mapped through.

Usage

```
crossfade_concat(videos, output, fade = 0.375, audio = NULL,
                 transition = "dissolve", cuts = NULL, windows = NULL,
                 overwrite = TRUE, dry_run = FALSE)
```

Arguments

videos	Character vector of video paths, in order. Must share resolution / fps / pixel format (xfade requires it).
output	Output video path.
fade	Crossfade duration in seconds: a scalar applied to every join, or a vector with one duration per join (length length(videos) - 1). A join with fade 0 is a plain butt join – no overlap consumed, no dissolve – for boundaries where the next clip has no duplicated conditioning head (see align_overlaps). Default 0.375 (= 9 frames @ 24fps, the default conditioning overlap).
audio	Optional path to a continuous audio track to overlay as the sole audio (e.g. the track's full narration mp3).
transition	xfade transition name (default "dissolve").
cuts	Optional logical vector, one per join (length length(videos) - 1). TRUE makes that join a hard cut (trim the next clip's head by that join's fade and butt-join) instead of a crossfade. NULL (default) = all crossfades. Output length is the same either way; a cut at a zero-fade join is just a butt join.
windows	Optional list, one element per video: c(start, end) in seconds selecting the part of the file to feed in (NA end = to the end of file), or NULL for the whole file. For a crossfaded join the incoming window should *include* the head handle the fade consumes (the fade eats fade seconds off the window's front against the previous clip's tail). Used by render_timeline to honor OTIO source ranges.
overwrite	Overwrite output (default TRUE).
dry_run	If TRUE, return the ffmpeg command string without running.

Details

Each join consumes its fade seconds of overlap, so the result runs sum(durations) - sum(fades) long – slightly shorter than a hard concat. One ffmpeg pass via xfade.

Value

Invisibly, output (or the command string when dry_run).

Examples

```
## Not run:
crossfade_concat(c("chunk01.mp4", "chunk02_chained.mp4", "chunk03_chained.mp4"),
                 "video.mp4", fade = 0.375, audio = "audio.mp3")

## End(Not run)
```

`frames_clip` *Encode a Numbered Frame Sequence as a Video Clip*

Description

Encodes `dir/pattern` (a C-style numbered pattern such as `"frame_%04d.png"`) into a video at `fps`. The sequence's frame count sets the clip length. Encode settings match `still_clip`, so stills and frame sequences concatenate cleanly.

Usage

```
frames_clip(dir, output, fps = 30, pattern = "frame_%04d.png", start = 1L,
            size = NULL, overwrite = TRUE, dry_run = FALSE)
```

Arguments

`dir`	Directory holding the frames.
`output`	Path for the output video file.
`fps`	Frame rate the sequence was drawn at (default 30).
`pattern`	C-style filename pattern of the frames (default `"frame_%04d.png"`).
`start`	Number of the first frame (default 1).
`size`	Output dimensions as `c(width, height)`, or NULL (default) to keep the source dimensions. Scaling ignores aspect; pass a size that preserves it (see `pad` for letterboxing instead).
`overwrite`	If TRUE (default), overwrite the output file.
`dry_run`	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns the command string.

Examples

```
## Not run:
frames_clip("scene01/", "scene01.mp4", fps = 30)

## End(Not run)
```

frame_export *Export a Single Frame at a Given Time*

Description

Pull one frame from a video at `time` seconds and write it as an image. A call-by-call primitive: no timeline, no asset lookup. The caller (an editor such as kerNLE) decides which file and which time; this just extracts the frame. Seeking is placed before `-i` for a fast keyframe seek.

Usage

```
frame_export(input, time, output, overwrite = TRUE, dry_run = FALSE)
```

Arguments

`input`	Path to input video.
`time`	Time in seconds to grab the frame at.
`output`	Path for output image (e.g. .png, .jpg).
`overwrite`	If TRUE (default), overwrite output file.
`dry_run`	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
frame_export("scene.mp4", 4.0, "frame.png")

## End(Not run)
```

hstack *Horizontally Stack Two Videos*

Description

Stack two videos side by side using FFmpeg's hstack filter.

Usage

```
hstack(left, right, output, width_left = NULL, width_right = NULL,
       height = 1080, overwrite = TRUE, dry_run = FALSE)
```

Arguments

left	Path to left video.
right	Path to right video.
output	Path for output video file.
width_left	Width in pixels for left video (default: auto from height).
width_right	Width in pixels for right video (default: auto from height).
height	Output height in pixels (default 1080).
overwrite	If TRUE (default), overwrite output file.
dry_run	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
hstack("left.mp4", "right.mp4", "wide.mp4")

## End(Not run)
```

---

normalize_audio *Normalize Audio Loudness*

Description

A single loudnorm pass to an EBU R128 / ITU-R BS.1770 target – the light-touch cousin of broadcast_audio, which wraps loudnorm in a full mastering chain (high-pass, de-hum notches, compression, fades). Use this to level takes that are already clean, e.g. TTS output across a batch of tracks.

Usage

```
normalize_audio(input, output = input, integrated = -16, lra = 11, tp = -1.5,
                bitrate = "192k", overwrite = TRUE, dry_run = FALSE)
```

Arguments

input	Path to input audio (or A/V) file.
output	Path for output file (default: overwrite input in place).
integrated	Target integrated loudness in LUFS (default -16).
lra	Loudness range target in LU (default 11).
tp	True-peak ceiling in dBTP (default -1.5).

bitrate	Output audio bitrate for lossy formats (default "192k"), or NULL for the encoder default.
overwrite	If TRUE (default), overwrite the output file.
dry_run	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns the command string.

Examples

```
## Not run:
normalize_audio("audio.mp3")
normalize_audio("take.wav", "leveled.wav", integrated = -14)

## End(Not run)
```

---

overlay *Overlay Foreground onto Background*

Description

Composite a PNG (with alpha) or video over a background video or image using FFmpeg's overlay filter.

Usage

```
overlay(background, foreground, output, x = 0, y = 0, scale = NULL,
        shortest = TRUE, overwrite = TRUE, dry_run = FALSE)
```

Arguments

background	Path to background video or image.
foreground	Path to foreground video or PNG (alpha supported).
output	Path for output video file.
x	Horizontal offset for overlay placement (default 0).
y	Vertical offset for overlay placement (default 0).
scale	Optional scale string for the foreground (e.g. "320:240" or "iw/2:ih/2").
shortest	If TRUE (default), end output when the shortest input ends.
overwrite	If TRUE (default), overwrite output file.
dry_run	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
overlay("bg.mp4", "character.png", "out.mp4", x = 312, y = 580)
overlay("bg.mp4", "logo.png", "out.mp4", x = 10, y = 10, scale = "100:100")

## End(Not run)
```

---

pad *Scale to Fit and Pad onto a Canvas*

Description

Scales `input` to fit within `fit_width` x `fit_height` (preserving aspect ratio, never upscaling past the box), then pads the result onto a `width` x `height` canvas at `x`,`y` filled with `color`. This is the ffmpeg `scale=...:force_original_aspect_ratio=decrease,pad=...` idiom for letterboxing media into a fixed frame (e.g. a square clip into 1080x1920 vertical shorts with the video at the top and a black caption strip below).

Usage

```
pad(input, output, width, height, fit_width = width, fit_height = height,
    x = "(ow-iw)/2", y = "(oh-ih)/2", color = "black", overwrite = TRUE,
    dry_run = FALSE)
```

Arguments

`color`	Pad color (default `"black"`).
`overwrite`	Overwrite the output if it exists (default TRUE).
`dry_run`	If TRUE, return the ffmpeg command string instead of running.
`input`, `output`	Input and output media paths.
`width`, `height`	Final canvas dimensions (pixels).
`fit_width`, `fit_height`	Box to scale the input into before padding (default the canvas dimensions).
`x`, `y`	Position of the scaled input on the canvas as ffmpeg `pad` expressions. Default centers it (`"(ow-iw)/2"`, `"(oh-ih)/2"`); pass `0`, `0` to top-left align (e.g. video at the top of a taller canvas).

Value

The output path, invisibly (or the command string on `dry_run`).

Examples

```
## Not run:
# Letterbox a square clip into vertical shorts, video at the top:
pad("clip.mp4", "shorts.mp4", 1080, 1920,
    fit_width = 1080, fit_height = 1080, x = 0, y = 0)

## End(Not run)
```

---

pip *Picture-in-Picture Corner Overlay*

Description

Composite a foreground clip (a character, a webcam, a logo) into a corner of the background, scaled and inset by a pixel margin. A convenience wrapper over the overlay filter: it computes corner placement with `main_w`/`overlay_w` expressions, which the fixed numeric offsets of `overlay` cannot express. Coordinates are top-left, +Y down.

Usage

```
pip(background, foreground, output, scale = 0.604, margin = 230,
    corner = c("upper-right", "upper-left", "lower-right", "lower-left"),
    overwrite = TRUE, dry_run = FALSE)
```

Arguments

`background`	Path to background video or image.
`foreground`	Path to foreground video or image (the PiP).
`output`	Path for output file.
`scale`	Foreground scale factor (default 0.604).
`margin`	Pixel margin from the corner (default 230).
`corner`	One of "upper-right", "upper-left", "lower-right", "lower-left".
`overwrite`	If TRUE (default), overwrite output file.
`dry_run`	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
pip("scene.mp4", "casey.mov", "out.mp4", corner = "upper-right")

## End(Not run)
```

probe *Probe Video Metadata*

Description

Query video file properties via ffprobe.

Usage

```
probe(file, field = "duration")
```

Arguments

file	Path to media file.
field	Field to query. One of "duration", "width", "height", "codec_name", "r_frame_rate", or any valid ffprobe stream entry. Use "all" to return duration, width, height, and codec as a named list.

Value

For single fields, returns a numeric value (duration, width, height) or character string (codec_name, etc.). For field = "all", returns a named list.

Examples

```
## Not run:
probe("video.mp4")                    # duration (default)
probe("video.mp4", "width")           # video width
probe("video.mp4", "all")             # list of duration, width, height, codec

## End(Not run)
```

render_timeline *Render an OTIO Timeline to a Video File*

Description

Lowers a rotio (OpenTimelineIO) Timeline to a single ffmpeg invocation and renders it. The video track's clips are concatenated in order; an optional separate audio track is mapped over them; a framing transform (scale + pad, from `metadata$cornball$framing`) and an optional caption burn (from a caption track referencing an .ass/.srt file) are applied.

Usage

```
render_timeline(timeline, output, media_dir = NULL, overwrite = TRUE,
                dry_run = FALSE)
```

Arguments

timeline	A rotio Timeline, or a path to a .otio file.
output	Path for the output video file.
media_dir	Base directory for resolving relative media urls. Defaults to the timeline file's directory when timeline is a path, otherwise NULL (urls used as-is).
overwrite	If TRUE (default), overwrite the output file.
dry_run	If TRUE, return the ffmpeg command string without executing. Note: concatenation of multiple video clips, still/sequence pre-renders, and layout composition still run, since they produce intermediates the final command depends on.

Details

The video track is lowered honoring per-clip source_range trims and Transitions: a transition between two clips becomes a dissolve over exactly its duration, fed by the incoming clip's head handle (the media before its source range – for chained generation, the conditioning-head replay). Only transitions with out_offset == 0 are lowered (the cornductor bundle shape: the outgoing clip has no tail handle). Gaps and overlapping layers are not lowered yet.

Still-image clips (a media reference targeting a .png/.jpg/ ...) are pre-rendered into video via still_clip at the source_range's rate, honoring cornball.* motion effects on the clip (Ken Burns pan/zoom; the schema lives in system.file("schema", "cornball-effects.md", package = "compost")). A still-image clip must carry a source_range. ImageSequenceReference clips are pre-rendered via frames_clip at the reference's rate; motion effects are not applied to sequences. Effects the renderer cannot honor degrade to a static clip with a warning; effects outside the cornball. namespace are ignored silently.

When the timeline carries metadata$cornball$layout (schema in system.file("schema", "cornball-layout.md", package = "compost")), tracks whose cornball$role matches a layout slot are each assembled like the content track and composed onto the canvas via compose_layout – muted, painted in slot order, with the content track bound to the visual slot as the timing reference. The framing transform is suppressed when composing (the layout defines the canvas). A layout the renderer cannot honor – schema too new, a slot's track missing, or a roled track without a slot – degrades to the plain single-stream render with one warning.

Caption tracks are identified by metadata$cornball$role == "caption" or a name containing "caption". When the primary video track has a single clip that already carries its own audio (the talking-head case), no separate audio track is needed.

Value

Invisibly returns the output path. If dry_run, returns the command string for the final render pass.

Examples

```
## Not run:
render_timeline("AAA/20260131/t41_n22_intro/timeline.otio",
                "AAA/20260131/t41_n22_intro/video.mp4")

## End(Not run)
```

rms_curve *Per-window RMS curve of an audio file*

Description

Runs one ffmpeg `astats` pass over `file` and returns the RMS level of each fixed-size window. Quieter windows (more-negative dB) mark pauses, so the curve is a cheap way to find good places to cut. About 140x realtime.

Usage

```
rms_curve(file, window = 1024L)
```

Arguments

`file`	Path to an audio (or audio-bearing video) file.
`window`	Window size in samples (default 1024; ~64ms at 16kHz). Smaller windows give finer time resolution at more rows.

Value

A list with numeric `time` (window centre, seconds) and `rms` (RMS level in dB; digital silence is -120). Empty vectors if the pass ran but produced no metadata. Errors if ffmpeg itself fails.

Examples

```
## Not run:
cur <- rms_curve("speech.mp3")
plot(cur$time, cur$rms, type = "l")   # dips are pauses

## End(Not run)
```

still_clip *Render a Still Image as a Video Clip*

Description

Turns one image into a video of `duration` seconds, optionally animated by a deterministic Ken Burns pan/zoom described by `motion`. Both the static and the animated case go through ffmpeg's `zoompan`, so the outputs are byte-compatible for `concat` and `crossfade_concat` regardless of motion.

Usage

```
still_clip(image, output, duration, fps = 30, size = NULL, motion = NULL,
           overwrite = TRUE, dry_run = FALSE)
```

Arguments

image	Path to the input image (png, jpg, ...).
output	Path for the output video file.
duration	Clip length in seconds.
fps	Output frame rate (default 30).
size	Output dimensions as c(width, height), or NULL (default) to use the source dimensions rounded to even.
motion	NULL (default) for a static clip, or a Ken Burns spec: a list with any of zoom_from, zoom_to (numeric >= 1; values below 1 are clamped), from, to (crop-window anchor as c(x, y) fractions of the source, top-left origin; the window is centered on the anchor and clamped to the frame), and ease ("linear" or "smooth", default smooth).
overwrite	If TRUE (default), overwrite the output file.
dry_run	If TRUE, return the FFmpeg command without executing.

Details

zoompan stretches its crop to size without preserving the source aspect ratio, so size should match the source's aspect (or a deliberate crop of it). Renderers letterbox afterwards; see pad.

Value

Invisibly returns the output path. If dry_run, returns the command string.

Examples

```
## Not run:
still_clip("slide01.png", "slide01.mp4", duration = 4.2)
still_clip("slide01.png", "slide01.mp4", duration = 4.2,
           motion = list(zoom_from = 1, zoom_to = 1.15,
                         from = c(0.5, 0.5), to = c(0.5, 0.4)))

## End(Not run)
```

---

subclip *Extract a Time Range from a Media File*

Description

Cut [start, start + duration] (or [start, end]) out of an audio or video file via ffmpeg. Re-encodes by default for frame-accurate cuts; set reencode = FALSE for a fast stream copy (cuts land on keyframes).

Usage

```
subclip(input, output, start = 0, duration = NULL, end = NULL, reencode = TRUE,
        overwrite = TRUE, dry_run = FALSE)
```

Arguments

input	Path to input media.
output	Path for the extracted output; extension selects the format.
start	Start time in seconds (default 0).
duration	Length in seconds, or NULL to use end or run to the end of the file.
end	End time in seconds (ignored when duration is set).
reencode	If TRUE (default), re-encode for accurate cuts; if FALSE, stream-copy (fast, keyframe-aligned).
overwrite	If TRUE (default), overwrite the output file.
dry_run	If TRUE, return the ffmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns the command string.

Examples

```
## Not run:
subclip("audio.mp3", "chunk.mp3", start = 2.14, duration = 5.0)
subclip("clip.mp4", "head.mp4", start = 0, end = 3, reencode = FALSE)

## End(Not run)
```

---

vstack *Vertically Stack Two Videos*

Description

Stack two videos vertically (top over bottom) using FFmpeg's vstack filter. Useful for YouTube Shorts layout (e.g. SadTalker top + slideshow bottom).

Usage

```
vstack(top, bottom, output, height_top = NULL, height_bottom = NULL,
       width = 1080, overwrite = TRUE, dry_run = FALSE)
```

Arguments

top	Path to top video.
bottom	Path to bottom video.
output	Path for output video file.
height_top	Height in pixels for top video (default: auto from width).
height_bottom	Height in pixels for bottom video (default: auto from width).
width	Output width in pixels (default 1080).
overwrite	If TRUE (default), overwrite output file.
dry_run	If TRUE, return the FFmpeg command without executing.

Details

Both inputs are scaled to the target width. Heights can be specified individually; by default each gets half the frame.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
vstack("talking_head.mp4", "slideshow.mp4", "shorts.mp4")
vstack("top.mp4", "bottom.mp4", "out.mp4", height_top = 960, height_bottom = 960)

## End(Not run)
```

---

zoom *Zoom Video with Stepped Levels*

Description

Apply a stepped zoom effect to a video, changing zoom level at STT line boundaries. Uses 3 levels (no zoom, single, double) with smooth transitions between them. The pattern randomly alternates between zoom-in, hold, and zoom-out phases.

Usage

```
zoom(input, output, stt, amount = 0.05, levels = 3, transition = 0.3,
     seed = NULL, width = NULL, height = NULL, overwrite = TRUE, dry_run = FALSE)
```

Arguments

input	Path to input video.
output	Path for output video file.
stt	A whisper_transcription object or data frame with from and to timestamp columns (HH:MM:SS.mmm format).
amount	Zoom per level (default 0.05). Level 1 = 1+amount, level 2 = 1+2*amount.
levels	Number of zoom levels (default 3: none, single, double).
transition	Transition duration in seconds between zoom levels (default 0.3).
seed	Random seed for reproducible zoom patterns.
width	Output width in pixels. If NULL, detected from input.
height	Output height in pixels. If NULL, detected from input.
overwrite	If TRUE (default), overwrite output file.
dry_run	If TRUE, return the FFmpeg command without executing.

Value

Invisibly returns the output path. If dry_run, returns command string.

Examples

```
## Not run:
stt <- readRDS("srt_text.RDS")
zoom("talking_head.mp4", "zoomed.mp4", stt = stt)
zoom("head.mp4", "zoomed.mp4", stt = stt, amount = 0.05, seed = 42)

## End(Not run)
```

Index