

# Package: audio.whisper (via r-universe)

October 3, 2024

**Type** Package

**Title** Transcribe Audio Files using the ``Whisper" Automatic Speech Recognition Model

**Version** 0.4.1

**Maintainer** Jan Wijffels <jwi jffels@bnosac.be>

**Description** The ``Whisper" models are trained for speech recognition and translation tasks, capable of transcribing speech audio into the text in the language it is spoken (Automatic Speech Recognition) as well as translated into English (speech translation). The package is an ``Rcpp" wrapper around the standalone C++ implementation provided at <<https://github.com/ggerganov/whisper.cpp>>. There are 10 pretrained models available of different sizes and language capabilities. ``Whisper" is explained in the paper: 'Robust Speech Recognition via Large-Scale Weak Supervision' by Radford et al. (2022), available at <[arXiv:2212.04356](https://arxiv.org/abs/2212.04356)>.

**License** MIT + file LICENSE

**URL** <https://github.com/bnosac/audio.whisper>

**Encoding** UTF-8

**Depends** R (>= 2.10)

**Imports** Rcpp (>= 0.11.5), utils

**Suggests** tinytest, audio, data.table (>= 1.12.4), audio.vadwebrtc (>= 0.2.0)

**LinkingTo** Rcpp

**SystemRequirements** GNU make

**RoxygenNote** 7.1.2

**Remotes** bnosac/audio.vadwebrtc

**Repository** <https://test.r-universe.dev>

**RemoteUrl** <https://github.com/bnosac/audio.whisper>

**RemoteRef** 0.4.1

**RemoteSha** 4b5c6a288c0f46a4cdc47f50d2d35395d3e32194

## Contents

predict.whisper . . . . .	2
predict.whisper_transcription . . . . .	4
whisper . . . . .	6
whisper_benchmark . . . . .	8
whisper_download_model . . . . .	9
whisper_languages . . . . .	11

<b>Index</b>	<b>12</b>
--------------	-----------

---

predict.whisper	<i>Transcribe audio files using a Whisper model</i>
-----------------	---

---

## Description

Automatic Speech Recognition using Whisper on 16-bit WAV files

## Usage

```
## S3 method for class 'whisper'
predict(
  object,
  newdata,
  type = c("transcribe", "translate"),
  language = "auto",
  sections = data.frame(start = integer(), duration = integer()),
  offset = 0L,
  duration = 0L,
  trim = FALSE,
  trace = TRUE,
  ...
)
```

## Arguments

object	a whisper object
newdata	the path to a 16-bit .wav file
type	character string with the type of prediction, can either be 'transcribe' or 'translate', where 'translate' will put the spoken text in English.
language	the language of the audio. Defaults to 'auto'. For a list of all languages the model can handle: see <a href="#">whisper_languages</a> .
sections	a data.frame with columns start and duration (measured in milliseconds) indicating voice segments to transcribe. This will make a new audio file with these sections, do the transcription and make sure the from/to timestamps are aligned to the original audio file. Defaults to transcribing the full audio file.

offset	an integer vector of offsets in milliseconds to start the transcription. Defaults to 0 - indicating to transcribe the full audio file.
duration	an integer vector of durations in milliseconds indicating how many milliseconds need to be transcribed from the corresponding offset onwards. Defaults to 0 - indicating to transcribe the full audio file.
trim	logical indicating to trim leading/trailing white space from the transcription using <code>trimws</code> . Defaults to FALSE.
trace	logical indicating to print the trace of the evolution of the transcription. Defaults to TRUE
...	further arguments, directly passed on to the C++ function, for expert usage only and subject to naming changes. See the details.

### Details

- `token_timestamps`: logical indicating to get the timepoints of each token
- `n_threads`: how many threads to use to make the prediction. Defaults to 1
- `prompt`: the initial prompt to pass on the model. Defaults to ""
- `entropy_thold`: entropy threshold for decoder fail. Defaults to 2.4
- `logprob_thold`: log probability threshold for decoder fail. Defaults to -1
- `beam_size`: beam size for beam search. Defaults to -1
- `best_of`: number of best candidates to keep. Defaults to 5
- `max_context`: maximum number of text context tokens to store. Defaults to -1
- `diarize`: logical indicating to perform speaker diarization for audio with more than 1 channel

If sections are provided If multiple offsets/durations are provided

### Value

an object of class `whisper_transcription` which is a list with the following elements:

- `n_segments`: the number of audio segments
- `data`: a data.frame with the transcription with columns `segment`, `segment_offset`, `text`, `from`, `to` and optionally `speaker` if `diarize=TRUE`
- `tokens`: a data.frame with the transcription tokens with columns `segment`, `token_id`, `token`, `token_prob` indicating the token probability given the context
- `params`: a list with parameters used for inference
- `timing`: a list with elements `start`, `end` and `duration` indicating how long it took to do the transcription

### See Also

[whisper](#), [whisper\\_languages](#)

**Examples**

```

model <- whisper("tiny")
audio <- system.file(package = "audio.whisper", "samples", "jfk.wav")
trans <- predict(model, newdata = audio)
trans <- predict(model, newdata = audio, language = "en")
trans <- predict(model, newdata = audio, language = "en", token_timestamps = TRUE)

audio <- system.file(package = "audio.whisper", "samples", "proficiat.wav")
model <- whisper("tiny")
trans <- predict(model, newdata = audio, language = "nl", type = "transcribe")
model <- whisper("tiny")
trans <- predict(model, newdata = audio, language = "nl", type = "translate")

## Predict using a quantised model
audio <- system.file(package = "audio.whisper", "samples", "jfk.wav")
path <- system.file(package = "audio.whisper", "repo", "ggml-tiny-q5_1.bin")
model <- whisper(path)
trans <- predict(model, newdata = audio, language = "en", trace = FALSE)
trans <- predict(model, newdata = audio, language = "en", token_timestamps = TRUE)
## Predict using a quantised model with the GPU
model <- whisper(path, use_gpu = TRUE)
trans <- predict(model, newdata = audio, language = "en")
trans <- predict(model, newdata = audio, language = "en", token_timestamps = TRUE)
## Example of providing further arguments to predict.whisper
audio <- system.file(package = "audio.whisper", "samples", "stereo.wav")
trans <- predict(model, newdata = audio, language = "auto", diarize = TRUE)

```

---

predict.whisper\_transcription

*Predict to which channel a transcription section belongs*

---

**Description**

Audio files containing 2 channels which were transcribed with [predict.whisper](#), you can use the results of a Voice Activity Detection by channel (either with R packages `audio.vadwebrtc` or `audio.vadsilero`) to assign the text segments to each of the channels.

This is done by looking for each text segment how many seconds overlap there is with the voiced sections which are identified by the Voice Activity Detection.

**Usage**

```

## S3 method for class 'whisper_transcription'
predict(object, vad, type = "channel", threshold = 0, ...)

```

**Arguments**

object	an object of class <code>whisper_transcription</code> as returned by <a href="#">predict.whisper</a>
vad	an object of class <code>webrtc-gmm-bychannel</code> as returned by function <code>VAD_channel</code> from R package <code>audio.vadwebrtc</code> with information of the detected voice in at least channels 1 and 2. <code>va</code> a list with element <code>vad_segments</code> containing a <code>data.frame</code> with columns <code>channel</code> , <code>start</code> , <code>end</code> and <code>has_voice</code> with information at which second there was a voice in the audio
type	character string with currently only possible value: <code>'channel'</code> which does a 2-speaker channel assignment
threshold	numeric in 0-1 range indicating if the difference between the probability that the segment was from the left channel 1 or the right channel 2 is smaller than this amount, the column <code>channel</code> will be set to <code>'both'</code> . Defaults to 0.
...	not used

**Value**

an object of class `whisper_transcription` as documented in [predict.whisper](#) where element `data` contains the following extra columns indicating which channel the transcription is probably from

- `channel`: either `'left'`, `'right'` or `'both'` indicating the transcription segment was either from the left channel (1), the right channel (2) or probably from both as identified by the Voice Activity Detecion
- `channel_probability`: a number between 0 and 1 indicating for that specific segment the ratio of the amount of voiced seconds in the most probably channel to the sum of the amount of voiced seconds in the left + the right channel
- `duration`: how long (in seconds) the from-to segment is
- `duration_voiced_left`: how many seconds there was a voiced signal on the left channel (channel 1) as identified by `vad`
- `duration_voiced_right`: how many seconds there was a voiced signal on the right channel (channel 2) as identified by `vad`

**See Also**

[predict.whisper](#)

**Examples**

```
library(audio.whisper)
model <- whisper("tiny")
audio <- system.file(package = "audio.whisper", "samples", "stereo.wav")
trans <- predict(model, audio, language = "es")
## Not run:
library(audio.vadwebrtc)
vad <- VAD_channel(audio, channels = "all", mode = "veryaggressive", milliseconds = 30)

## End(Not run)
```

```

vad <- list(vad_segments = rbind(
  data.frame(channel = 1, start = c(0, 5, 15, 22), end = c(5, 9, 18, 23), has_voice = TRUE),
  data.frame(channel = 2, start = c(2, 9.5, 19, 22), end = c(2.5, 13.5, 21, 23), has_voice = TRUE)))
out <- predict(trans, vad, type = "channel", threshold = 0)
out$data

```

---

whisper

*Automatic Speech Recognition using Whisper*


---

### Description

Automatic Speech Recognition using Whisper on 16-bit WAV files. Load the speech recognition model.

### Usage

```

whisper(
  x,
  use_gpu = FALSE,
  overwrite = FALSE,
  model_dir = Sys.getenv("WHISPER_MODEL_DIR", unset = getwd()),
  ...
)

```

### Arguments

x	the path to a model, an object returned by <a href="#">whisper_download_model</a> or a character string with the name of the model which can be passed on to <a href="#">whisper_download_model</a>
use_gpu	logical indicating to use the GPU in case you have Metal or an NVIDIA GPU. Defaults to FALSE.
overwrite	logical indicating to overwrite the model file if the model file was already downloaded, passed on to <a href="#">whisper_download_model</a> . Defaults to FALSE.
model_dir	a path where the model will be downloaded to, passed on to <a href="#">whisper_download_model</a> . Defaults to the environment variable WHISPER_MODEL_DIR and if this is not set, the current working directory
...	further arguments, passed on to the internal C++ function <code>whisper_load_model</code>

### Value

an object of class `whisper` which is list with the following elements:

- file: path to the model
- model: an Rcpp pointer to the loaded Whisper model

### See Also

[predict.whisper](#)



---

whisper_benchmark	<i>Benchmark a Whisper model</i>
-------------------	----------------------------------

---

## Description

Benchmark a Whisper model to see how good it runs on your architecture by printing it's performance on fake data. <https://github.com/ggerganov/whisper.cpp/issues/89>

## Usage

```
whisper_benchmark(  
  object = whisper(system.file(package = "audio.whisper", "models",  
    "for-tests-ggml-tiny.bin")),  
  threads = 1  
)
```

## Arguments

object	a whisper object
threads	the number of threads to use, defaults to 1

## Value

invisible()

## See Also

[whisper](#)

## Examples

```
## Not run:  
model <- whisper("tiny", overwrite = FALSE)  
whisper_benchmark(model)  
  
## End(Not run)
```



---

whisper\_download\_model

*Download a pretrained Whisper model*


---

## Description

Download a pretrained Whisper model. The list of available models are

- tiny & tiny.en: 75 MB, RAM required: ~390 MB. Multilingual and English only version.
- base & base.en: 142 MB, RAM required: ~500 MB. Multilingual and English only version.
- small & small.en: 466 MB, RAM required: ~1.0 GB. Multilingual and English only version.
- medium & medium.en: 1.5 GB, RAM required: ~2.6 GB. Multilingual and English only version.
- large-v1, large-v2, large-v3: 2.9 GB, RAM required: ~4.7 GB. Multilingual
- quantised models: tiny-q5\_1, tiny.en-q5\_1, base-q5\_1, base.en-q5\_1, small-q5\_1, small.en-q5\_1, medium-q5\_0, medium.en-q5\_0, large-v2-q5\_0 and large-v3-q5\_0 (only - from version 1.5.4 onwards)

Note that the larger models may take longer than 60 seconds to download, so consider increasing the timeout option in R via `options(timeout = 120)`

## Usage

```
whisper_download_model(
  x = c("tiny", "tiny.en", "base", "base.en", "small", "small.en", "medium",
        "medium.en", "large-v1", "large-v2", "large-v3", "large", "tiny-q5_1",
        "tiny.en-q5_1", "base-q5_1", "base.en-q5_1", "small-q5_1", "small.en-q5_1",
        "medium-q5_0", "medium.en-q5_0", "large-v2-q5_0", "large-v3-q5_0"),
  model_dir = Sys.getenv("WHISPER_MODEL_DIR", unset = getwd()),
  repos = c("huggingface", "ggerganov"),
  version = c("1.5.4", "1.2.1"),
  overwrite = TRUE,
  ...
)
```

## Arguments

x	the name of the model
model_dir	a path where the model will be downloaded to. Defaults to the environment variable WHISPER_MODEL_DIR and if this is not set, the current working directory
repos	character string with the repository to download the model from. Either <ul style="list-style-type: none"> <li>• 'huggingface': <a href="https://huggingface.co/ggerganov/whisper.cpp">https://huggingface.co/ggerganov/whisper.cpp</a> - the default</li> <li>• 'ggerganov': <a href="https://ggml.ggerganov.com/">https://ggml.ggerganov.com/</a> - no longer supported as the resource by ggerganov can become unavailable</li> </ul>
version	character string with the version of the model. Defaults to "1.5.4".

<code>overwrite</code>	logical indicating to overwrite the file if the file was already downloaded. Defaults to TRUE indicating it will download the model and overwrite the file if the file already existed. If set to FALSE, the model will only be downloaded if it does not exist on disk yet in the <code>model_dir</code> folder.
<code>...</code>	currently not used

**Value**

A data.frame with 1 row and the following columns:

- `model`: The model as provided by the input parameter `x`
- `file_model`: The path to the file on disk where the model was downloaded to
- `url`: The URL where the model was downloaded from
- `download_success`: A logical indicating if the download has succeeded or not due to internet connectivity issues
- `download_message`: A character string with the error message in case the downloading of the model failed

**See Also**

[whisper](#), [predict.whisper](#), [whisper\\_languages](#)

**Examples**

```
path <- whisper_download_model("tiny")
path <- whisper_download_model("tiny", overwrite = FALSE)
## Not run:
whisper_download_model("tiny.en")
whisper_download_model("base")
whisper_download_model("base.en")
whisper_download_model("small")
whisper_download_model("small.en")
whisper_download_model("medium")
whisper_download_model("medium.en")
whisper_download_model("large-v1")
whisper_download_model("large-v2")
whisper_download_model("large-v3")
whisper_download_model("tiny-q5_1")
whisper_download_model("base-q5_1")
whisper_download_model("small-q5_1")
whisper_download_model("medium-q5_0")
whisper_download_model("large-v2-q5_0")
whisper_download_model("large-v3-q5_0")

## End(Not run)
```

---

`whisper_languages`      *Get the language capabilities of Whisper*

---

**Description**

Extract the list of languages a multilingual whisper model is able to handle

**Usage**

```
whisper_languages()
```

**Value**

a data.frame with columns `id`, `language` and `language_label` showing the languages

**Examples**

```
x <- whisper_languages()
x
```

# Index

`predict.whisper`, [2](#), [4-6](#), [10](#)  
`predict.whisper_transcription`, [4](#)

`trimws`, [3](#)

`whisper`, [3](#), [6](#), [8](#), [10](#)  
`whisper_benchmark`, [8](#)  
`whisper_download_model`, [6](#), [9](#)  
`whisper_languages`, [2](#), [3](#), [10](#), [11](#)