Setup and Tutorial

Alexander P. Christensen, Hudson Golino, Aleksandar Tomašević

January 05, 2026

1. Python setup using `setup_miniconda()`

transforEmotion uses the *reticulate* package to automatically install a standalone miniconda version on your computer and download the necessary Python modules to run the transformer models.

After installing *transforEmotion* package, you need to run `setup_miniconda()` function. This function will performs the following tasks:

1. Install a standalone miniconda version on your computer.
2. Setup a virtual Python environment for *transforEmotion* package.
3. Install the necessary Python modules to run the transformer models.

By having a standalone miniconda installed through *transforEmotion*, you should not encounter any conflicts between miniconda and existing Python installations. It also ensures that the correct version of Python libraries are installed into the virtual environment, guaranteeing compatibility regardless of your operating system or whether you already have Python installed.

The miniconda installation process takes a few minutes to complete. At certain points, it may appear that the installer is stuck, but please allow a few minutes for the installer to finish its process. Remember, `setup_miniconda()` only needs to be run once after installing the package. Unless you run into issues with Python environment and libraries later on, there is no need to run this function again on the same system.

2. Using `transformer_scores`

You can use any number of Hugging Face text classification transformers. *transforEmotion* currently implements only the zero-shot classification models. Future updates to the package may include opportunities to train and fine-tune these models. For now, there are several options that work well for most classification tasks straight out-of-the-box. You can view different transformers that can be used in *transforEmotion* here.

As mentioned in section 1, before running the `transformer_scores` function, you need to execute `setup_miniconda()` to install the necessary modules. Running the `transformer_scores` function will also trigger the download of the Cross-Encoder's DistilRoBERTa transformer model. Therefore, the easiest way to get started is by using an example.

```
library(transforEmotion)

# Setup Python

setup_miniconda()

# Load data

data(neo_ipip_extraversion)

# Example text
```

```r
text <- neo_ipip_extraversion$friendliness[1:5] # positively worded items only

# Run transformer function

transformer_scores(
    text = text,
    classes = c(
      "friendly", "gregarious", "assertive",
      "active", "excitement", "cheerful"
    )
)
```

The downloads will take some time when you run a specific model for the first time. Assuming everything goes well with the code above, you should see output that looks like this:

```
$`make friends easily`
  friendly gregarious  assertive     active excitement   cheerful
     0.579      0.075      0.070      0.071      0.050      0.155

$`warm up quickly to others`
  friendly gregarious  assertive     active excitement   cheerful
     0.151      0.063      0.232      0.242      0.152      0.160

$`feel comfortable around people`
  friendly gregarious  assertive     active excitement   cheerful
     0.726      0.044      0.053      0.042      0.020      0.115

$`act comfortably around people`
  friendly gregarious  assertive     active excitement   cheerful
     0.524      0.062      0.109      0.183      0.019      0.103

$`cheer people up`
  friendly gregarious  assertive     active excitement   cheerful
     0.071      0.131      0.156      0.190      0.362      0.089
```

If you want to run `transformer_scores` on additional text, simply enter that text into the `text` argument of the function. The transformer models that you've used during your R session will remain in R's environment until you exit R or remove them from your environment.

That's it! You've successfully obtained sentiment analysis scores from the Cross-Encoder's DistilRoBERTa transformer model. Now, go forth and quantify the qualitative!

3. Using `image_scores`

transforEmotion also provides a Facial Expression Recognition (*FER*) function that uses the CLIP model to obtain sentiment analysis scores from images. The CLIP model is a transformer model that was trained on a dataset of 400 million image-text pairs. CLIP's input contains an image and text labels and the inference output is a score for each label. In case of emotion labels, the CLIP returns FER scores.

The `image_scores` function uses the CLIP model from Hugging Face. In a similar way to the `transformer_scores` function, the `image_scores` function also downloads the CLIP model the first time you run it, performs the inference and returns the output as an R dataframe.

The main input of this function is the path to image. It can be a local filepath or an URL pointin to an image. The second input is an array of emotion labels.

```
image <- ""https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/Mona_Lisa%2C_by_Leonardo_da_Vinci%

emotions <- c("excitement", "happiness", "pride", "anger", "fear", "sadness", "neutral")

image_scores(image, emotions)
```

This code runs FER on the Mona Lisa painting and returns the following output, a dataframe with 7 columns and 1 row.

```
  excitement  happiness     pride      anger       fear   sadness   neutral
1 0.02142187 0.02024468 0.0604699 0.04037686 0.03273294 0.1061871 0.7185667
```

In case there is no face recognized on the image, the function will return an empty dataframe, preceeded by a warning message. We can test this with an image of three penguins from the South Shetland Islands.

```
image <- "https://upload.wikimedia.org/wikipedia/commons/thumb/0/0e/Adelie_penguins_in_the_South_Shetla

image_scores(image, emotions)
```

This gives the following output:

```
No face found in the image
data frame with 0 columns and 0 rows
```

4. Using 'video_scores'

We can use the workflow described in section 3 to run FER on videos. The `video_scores` function takes a video as input and returns a dataframe with FER scores for each frame of the video.

The video can be a local filepath or an URL pointing to a YouTube video. In case of YouTUbe, the function will download the video and save it in the temporary directory. The function also takes an array of emotion labels as input. After downloading video, this function will extract frames from the video and the run the same workflow as in case of images.

Here's an example of running FER on a YouTube video of Boris Johnson.

```
video_url <- "https://www.youtube.com/watch?v=hdYNcv-chgY&ab_channel=Conservatives"

emotions <- c("excitement", "happiness", "pride", "anger", "fear", "sadness", "neutral")

result <- video_scores(video_url, classes = emotions,
                       nframes = 10, save_video = TRUE,
                       save_frames = TRUE, video_name = 'boris-johnson',
                       start = 10, end = 120)

head(result)
```

Working with videos is more computationally complex. This example extracts only 10 frames from the video and I shouldn't take longer than few minutes on an average laptop without GPU (depending on your internet connection needed to download the entire video and CLIP model). In research applicatons, we will usually extract 100-300 frames from the video. This can take much longer, so pantience is advised while waiting for the results.

Working with videos is more complex, so this function takes several additional arguments. The `nframes` argument specifies the number of frames to extract from the video. The `save_video` argument specifies whether to save the video in the temporary directory. The `save_frames` argument specifies whether to save the extracted frames in the temporary directory. The `video_name` argument specifies the name of the video.

Saving the video and frames is useful for purposes of quality control and potential evalution of the result by human coders.

The `start` and `end` arguments specify the start and end time of the video. The `start` and `end` arguments are in seconds and point to the timestamps of the video where the analysis should start and end. If they are not provided, the function will try to process the entire video.

The result is a dataframe with 7 columns and 10 rows, one row for each frame. The dataframe contains FER scores for each frame of the video. The ouput of `head(result)` looks like this:

```
  excitement  happiness     pride      anger       fear   sadness   neutral
1 0.08960483 0.006041054 0.05632496 0.2259102 0.2781007 0.1757137 0.1683045
2 0.11524552 0.011083936 0.08131301 0.1672127 0.3321840 0.1652457 0.1277151
3 0.09541881 0.007240616 0.05629114 0.1665660 0.3410282 0.1952039 0.1382514
4 0.09860725 0.011296707 0.07909032 0.1693194 0.3010349 0.1759851 0.1646665
5 0.08856109 0.007197607 0.07237346 0.2261922 0.3237688 0.1515539 0.1303529
6 0.10022306 0.011431777 0.09256416 0.1467394 0.3202718 0.1574203 0.1713494
```

These 3 functions are the core of *transforEmotion* package. They allow you to run sentiment analysis on text, images and videos. If you run into any issues, please report them on the GitHub issues page. If you have any suggestions for improvements, please let us know. We are always looking for ways to improve the package and make it more useful for researchers.