EasyChair Preprint

№ 4389

OpenFaceR: Developing an R Package for the Convenient Analysis of OpenFace Facial Information.

Davide Cannata, Sam Redfern and Denis O'Hora

October 13, 2020

OpenFaceR: Developing an R Package for the convenient analysis of OpenFace facial information.

Davide Cannata[1[0000-0003-2254-2639]], Sam Redfern[1[0000-0002-4856-3756]], and Denis O'Hora[1[0000-0003-3776-6782]]

[1] National University of Galway, Ireland

d.cannata1@nuigalway.ie

Abstract. OpenFace is an open source tool designed to extract the most commonly used facial information from videos including facial points, head pose, gaze and Facial Action Units. OpenFaceR is a tool designed to help social scientists, and other researchers from less technical disciplines, who are interested in facial nonverbal behaviors (FNVBs), to easily use output from OpenFace 2.0. The output from OpenFace is one csv file for each video, with information on each feature for each frame of the analyzed video provided in rows. OpenFaceR constitutes a set of methods to convert information in this format into relevant summary statistics. In this paper, we focus on the set of methods in OpenFaceR to extract information from a series of videos and transform the output files into a single dataset in which each row reports the summary values of a feature for one video.

Keywords: OpenFace, R, Nonverbal Behaviors, Face, Computer Vision.

1 Introduction

Humans are social animals, capable of complex and variable behaviour. The face is a central element of human sociality[1], since it provides rich information for immediate social judgments through static cues (e.g. biometrics, skin colour, feminine/masculine features, regional traits etc..) and dynamic cues (e.g. smiles, blinks, gaze, emotion expression etc…). The latter, also called Facial Nonverbal Behaviors (FNVBs), have been widely studied in many fields such as display of emotions [2], lie detection [3], interpersonal relations [4], and personality recognition [5]. Research on FNVBs can be further divided into two streams which require different techniques of data collection and data analysis. The first one is concerned with how facial expressions change in time within subjects (e.g. studies on mimicry or on emotional reactions [6]) and therefore require FNVBs data per each temporal unit. The second stream is concerned about how FNVBs differ between individuals (e.g. nonverbal expression of personality [7]) or within the same individuals in different conditions (e.g. being ingenuous vs being deceitful [3]). In this second case, that is the focus of this paper, the analysis is performed on summary measures of FNVBs, such as their frequency.

FNVBs are traditionally annotated manually, through one of the many existing scales (ex: Riverside Q-Sort [8]; Münster Behavior coding system [9]). One of the most

popular is the Facial Actions Coding System (FACS) by Paul Ekman [10], which analyses the smallest independent movements of the facial muscles, called Action Units (AUs). The FACS provides with a detailed and objective approach to the classification of FNVBs, but manual annotation of AUs is a demanding job which requires considerable amount of time of well-trained observers.[11]. Recently, progresses in computer vision has allowed the development of software for automatic analysis and recognition of facial static and dynamic characteristics [12]. Amongst those OpenFace, an open-source software developed at Cambridge University by Baltrusaitis and colleagues [13], is one of the most used in the social sciences, with 753 citations by August 16, 2020. OpenFaceR, the GitHub repository presented in this paper, includes a set of R functions intended to facilitate the use of OpenFace 2.0 for social scientists.

2 OpenFace

The major goal of OpenFace is to provide a comprehensive, open source and free tool for describing facial behaviors [13]. OpenFace estimates the status of four different types of feature: facial landmarks; head pose; eye gaze; and facial expressions. The x, y and z position of 67 facial landmarks are identified using a Convolutional Experts Constrained Local Model [14]. Based on these values, head pose is estimated through the orthographic projection of an internal 3D representation of the facial landmarks [13]. To estimate the direction of eye gaze, OpenFace first uses a Constrained Local Neural Field to detect eyelids, pupils and iris. Then, an eyeball model and head pose information are incorporated in a complex process to estimate gaze direction [15]. Finally, OpenFace makes use of a linear kernel Support Vector approach to describe18 AUs [16] (e.g., movement of the lip corner puller, the muscle we use to smile). For each AU it estimates its intensity (e.g. a number between 0 and 1 describing how much the lip corner puller is contracted) and its presence (e.g,, if the movement of the lip corner puller is large enough for being observed as a smile). OpenFace 2.0 has been tested through two different datasets achieving state of the art performance, despite comparatively low computational demands [13]. The software can be run though the command prompt to analyse a single video or multiple videos stored in a folder. The output, for each video, is a Comma Separated Values (CSV) file including XX values for each frame:

- frame number
- timestamp
- confidence (how accurate the analysis of the frame is likely to be) and success (whether confidence is high enough)
- x, y and z coordinates of the gaze for each eye
- z and y polar coordinates of the gaze angle
- 56 by 2 (x and y) 2D eye landmark positions
- 56 by 3 (x, y and z) 3D eye landmark positions
- x, y and z coordinates of the head position
- Roll, pitch and yaw of the head

- 68 by 3 (x, y and z) facial landmarks positions
- 18 by 2 (presence and intensity) AUs

3 Fitting OpenFace data to social scientists' needs

The output from OpenFace is rich and detailed, but, for just this reason, it is not ideal for data analysis by most social scientists. When OpenFace processes a video (usually depicting one person's participation), a long CSV document is produced, in which each row reports the 538 values noted previously for each frame. OpenFace typically analyses videos at a 30Hz frame rate, so the standard output is a csv with a number of rows equal to 30 times the duration of the video in seconds. Such data are perfectly suitable for time series analysis [17] but many social scientists are not trained in such techniques and wish to test hypotheses concerning summary statistics (e.g. frequency or mean and standard deviation) of FNVBs per each person (in a between-subjects design) or per each person in each condition (in a within-subjects design).

To provide data more suitable for the needs of social scientists, we employed the 'tidy' framework proposed by Hadley Wickham for easier data analysis and visualization [18]. Datasets are defined as tidy if each row corresponds to an observation, each column corresponds to a variable and each type of observational unit forms a table [18]. The challenge for social scientists using OpenFace, therefore, is how to transform frame-level output into a tidy dataset with output per person or condition. Figure 1 shows an example in which 60 second videos of three people have been analysed with OpenFace to annotate true smiles and blinks. The left of the figure represents the OpenFace output with one person per dataset, one frame per row and with each column representing the absence or presence of a facial action unit. On the right, there is a tidy dataset in which each row represents a person and each column is a summary of the person FNVBs, in this case the frequency of true smiles and blinks.

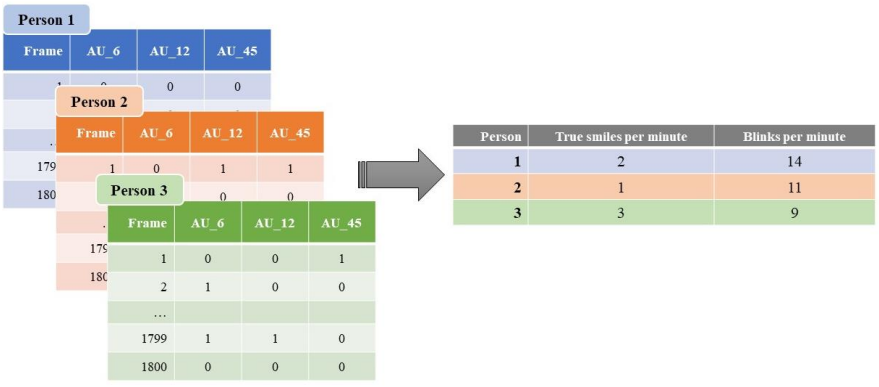

Fig. 1 Conceptual example of transformation from OpenFace output (on the left) to a tidy dataset (on the right) in which each row correspond to a person. AU_6 (cheek raiser) and AU_12 (lip corner puller) in combination signal a true smile. AU-45 represents a blink.

The main goal of OpenFaceR is to provide a set of tools and a workflow for the creation of such tidy datasets for social scientists.

4 OpenFaceR workflow

OpenFaceR assists social scientists through a workflow that leads from the analysis of videos to the consolidation of a tidy dataset with one person per row. Its functions make extensive use of the tidyverse package [19]. The tidyverse is a collection of packages aimed to "facilitate a conversation between a human and a computer about data" [19, pag.1]. It includes methods for data manipulation, data importing, data tidying, data manipulation and data visualisation. Notably, OpenFaceR uses and extends the functions "mutate"," filter", "select" and "summarise" from *dplyr* and makes extensive use of the pipe sign "%>%" from *magrittr*. Also, OpenFaceR import and returns datasets as *tibbles*[20], a tidyverse equivalent to R base dataframes offering better performance and visualisation methods.

The OpenFaceR workflow is designed to accomplish to the transformation from raw video material to a tidy dataset. To start the workflow, the user needs the following: video files of each person (or each person in each condition), the OpenFace software package, R [21] (we also recommend RStudio [22]), and OpenFaceR. It is easiest if the videos correspond to the unit of analysis. For example, in a within participants design, it is easiest if each condition is captured in a separate video file. However, it is possible to extract sections of videos by filtering which is described later. OpenFace is implemented in python [23] and *pyTorch* [24]. Detailed instructions for Linux and MacOS X installation are provided at https://cmusatyalab.github.io/openface/setup/. Instructions for the installation of the executable file for windows are provided here: https://github.com/TadasBaltrusaitis/OpenFace/wiki/Windows-Installation. R can be downloaded from the CRAN repository (https://cran.r-project.org/). At present, the OpenFaceR toolkit can be downloaded from GitHub at https://github.com/davidecannatanuig/, but installation using the *devtools* R package will be implemented in the near future. OpenFaceR requires the following R packages to be installed: *tidyverse* [19] and *pracma* [25] .Fig. 2, below, shows the six steps of the process, which are extensively discussed in the next paragraphs.

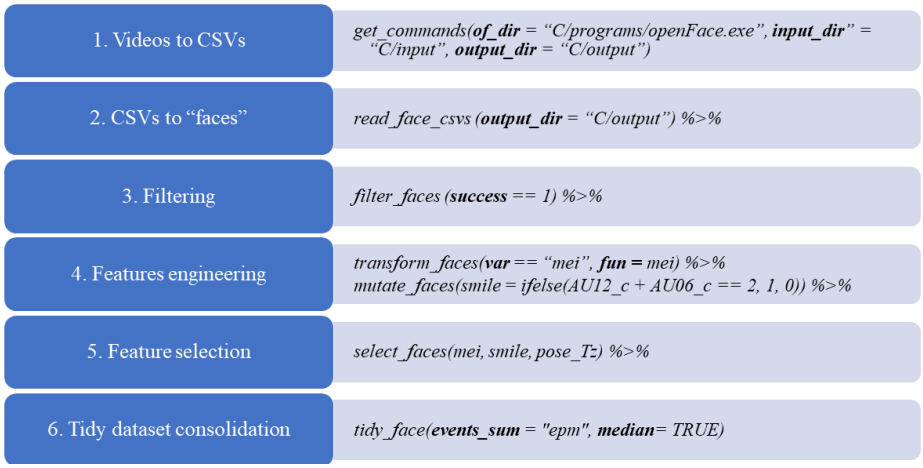

Fig. 2 OpenFaceR workflow and an operative example

To facilitate readers' comprehension of this six-step process, we employ an example of a simple psychological experiment investigating the effects of positive and negative memories on facial behaviours. In our example, the researcher has collected videos of 50 students telling one story about a personal success and one story about a personal failure in front a camera. The hypothesis is that students will smile more frequently and will display more intense facial activity in the success story condition.

4.1 Videos to CSVs using OpenFace

Prior to using the utility functions in OpenFaceR, users must process their videos using OpenFace. To help users produce the appropriate syntax for these commands in Windows, OpenFaceR provides the function *get_commands()* that outputs the commands and parameters for executing OpenFace on a single video or on a set of videos contained in a folder. After the user runs *get_commands()*, the user can copy and paste the output of *get_commands()* at the command line to initiate the analysis or analyses. In the example described above, the input_dir is the folder containing the 100 video files recording the students telling their stories. The output_dir will hold the 100 csvs produced by OpenFace. The duration of this process is dependent on the user's computer hardware, specifically the GPU, CPU and storage medium (e.g. SSD drive).

4.2 CSVs to *faces* objects

From Step 2, the remaining steps are completed in the R environment. The function *read_faces_csv()* allows the user to import all the csv files contained in a folder into an object of class "*faces*". *Faces* is a new bespoken C3 class that inherits from lists, and

in fact is a list of tibbles, with each tibble representing the output from one *video*. In our example, read_faces_csv will import the 100 csv files saved in the output_dir from the previous step and produce a faces object containing 100 tibbles. The time for the machine to perform this operation, although depending by the local machine specifications, can be significant.

4.3 Filtering

It is often necessary to filter out certain faces or conditions due to errors in the extraction of data, low confidence and so on. The verb *filter_faces()* allows the user to filter all the tibbles of a *faces* object, with a grammar that echoes the *dplyr* filter method. A typical filter is set up for "success", the variable indicating whether the extraction of data was reliably done for each frame of the video. It is also possible to standardize the extraction parameters of the videos by filtering. For example, to standardize the duration of videos that will be analyzed, one can filter the timestamps, restricting them to minimum and maximum values. In our example, the researcher wants to standardize the length of videos as time might also influence the production of smiles and face activity. They can therefore use *filter_face(timestamp < 180)* to take only the first 3 minutes of each video. The result can be stored in a new filtered faces object or by using the pipe %>% sign, steps 3 to 6 can be conducted in a series and outputted in a tidy dataframe.

4.4 Features engineering

OpenFaceR provides two verbs to manipulate the variables of each video and engineering new features. *mutate_faces()* echoes *dplyr::mutate()*. The function *transform_faces()* meets the need of using transformation functions that take as input a preset selection of variables, as opposed to the mutate method which is designed for working with user specified variables. The two verbs are accompanied by a growing number of functions specifically designed for analyzing faces. In our example the researcher uses *mutate_faces(smile = ifelse(AU06_c + AU12_c) == 2, 1, 0)* to calculate when the experimental subjects are displaying the two AUs characterizing smiles. Furthermore they will use the function *transform_faces("mei", mei)* for calculating the corrected average motion of the face region[25], a measure of facial activity.

4.5 Selection of features

We have implemented the verb *select_faces()* to select which features are eventually summarised, echoing the *select()* method from *dplyr*. As *frame, timestamp* and *success* are critical meta-variables, *select_faces()* always returns those in the output. In our example the researcher will use *select_faces(smile, mei)* to select the two variables they intend to summarise.

4.6 Tidy dataset consolidation

The function *tidy_face()* is designed to transform a preprocessed faces object into a tidy dataset with one person per row and all the most common statistics. Fig. 3 summarizes the function's architecture. Here, the arrows represent the logical steps, while the boxes represent the methods used, including the inputs they take from the main function. First, *tidy_face()* merges all the tibbles of the faces object into one single tibble through the *merge_faces()* method. Second, it calculates the length of each video. Third, it classifies the variables into continuous (e.g. distance of the face from the camera) and discrete (events, such as blinks and smiles). Fourth, if the events parameter is set as true (default) all the discrete variables are summarized. Events can be summarized by simply counting them ("count"), or as events per second ("eps"), events per minute ("epm") or events ratio ("ratio", the number of frames in which the event happen divided the total number of frames). Fifth, if the continuous parameter is set as True (default) all the continuous variables are summarized with a choice of methods including mean, median, standard deviation, minimum and maximum. Finally, all the summarized variables are merged into a tidy dataset.

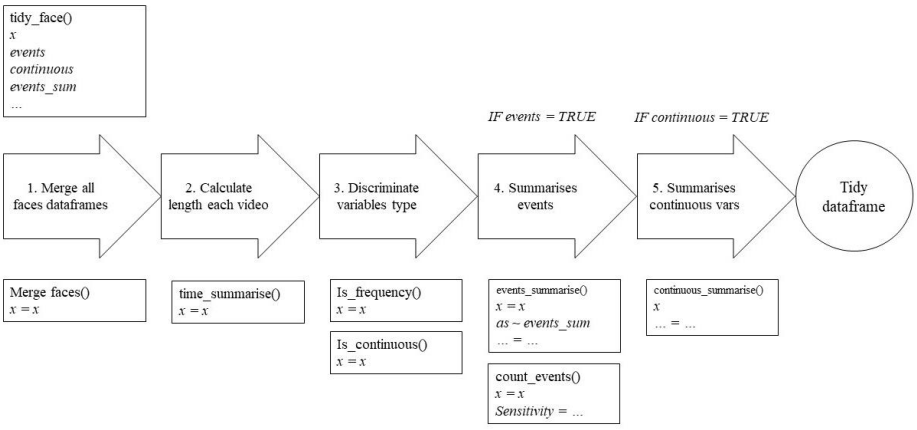

Fig. 3 *Tidy_face* architecture

In our example, calling *tidy_faces(events_sum = "epm", median = TRUE)* will return one data frame with 100 rows (one per video) with columns for video ID, video duration, mean, standard deviation and median of facial activity and the number of smiles per minutes. The researchers can then test their hypotheses by running t-tests, a repeated measures MANOVA or other statistical approaches available in R or other packages.

5 Conclusions

In this paper we have outlined and explained the goal of OpenFaceR and outlined the main characteristics of the workflow from raw video data to a dataset that can be used for typical statistical analysis in social sciences. OpenFaceR is still at its infancy and new functions are been built, providing the most common methods of summarizing FNVBs. The final goal of this enterprise is to compile an R package to publish on CRAN. Questions, feedback, collaborations, and ideas are welcome.

References

1. Jack, R. E. & Schyns, P. G. Toward a Social Psychophysics of Face Communication. (2017). doi:10.1146/annurev-psych-010416-044242
2. Hall, J. A., Gunnery, S. D. & Andrzejewski, S. A. Nonverbal emotion displays , communication modality , and the judgment of personality. J. Res. Pers. 45, 77–83 (2011).
3. Cohen, D., Beattie, G. & Shovelton, H. Nonverbal indicators of deception: How iconic gestures reveal thoughts that cannot be suppressed. Semiotica 2010, 133–174 (2010).
4. Grahe, J. E. & Bernieri, F. J. The importance of nonverbal cues in judging rapport. J. Nonverbal Behav. 23, 253–269 (1999).
5. Breil, S. M., Osterholz, S., Nestler, S. & Back, M. D. Contributions of Nonverbal Cues to the Accurate Judgment of Personality Traits. in The Oxford handbook of accurate personality judgment. (eds. Letzring, T. D. & Spain, J.) 1–54 (Oxford University Press, 2019).
6. Arnold, A. J. & Winkielman, P. The Mimicry Among Us: Intra- and Inter-Personal Mechanisms of Spontaneous Mimicry. J. Nonverbal Behav. 44, 195–212 (2020).
7. Back, M. D. & Nestler, S. Accuracy of Judging Personality. in The Social Psychology of Perceiving Others Accurately (eds. Hall, J. A., Schmid Mast, M. & West, T. V) 98–124 (Cambridge University Press, 2016).
8. Funder, D. C., Furr, R. M. & Colvin, C. R. The Riverside Behavioral Q-sort: A Tool for the Description of Social Behavior. J. Pers. 68, 451–489 (2000).
9. Grünberg, M., Mattern, J., Geukes, K., Küfner, A. C. P. & Back, M. D. Assessing Group Interactions in Personality Psychology. in The Cambridge Handbook of Group Interaction Analysis 53, 602–611 (Cambridge University Press, 2019).
10. Ekman, P. & Friesen, W. V. Facial action coding system: A technique for the measurement of facial movement. (Consulting Psychologist Press, 1978).
11. Furr, R. M. & Funder, D. C. Behavioral observation. in Handbook of research methods in personality psychology (eds. Robins, R. W., Fraley, C. & Krueger, R. F.) 273–291 (Guilford Press, 2009).
12. Cannata, D., Simon, B., Lepri, B., Back, M. D. & O'Hora, D. (In press) Toward an Integrative Approach to Nonverbal Personality Detection.
13. Baltrušaitis, T., Zadeh, A., Lim, Y. C. & Morency, L. P. OpenFace 2.0: Facial Behavior Analysis Toolkit. in 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018) 59–66 (2018).
14. Zadeh, A., Lim, Y. C., Baltrušaitis, T. & Morency, L. P. Convolutional experts constrained local model for 3D facial landmark detection. Proc. - 2017 IEEE Int. Conf. Comput. Vis. Work. ICCVW 2017 2018-Janua, 2519–2528 (2017).

15. Wood, E. et al. Rendering of Eyes for Eye-Shape Registration and Gaze Estimation Erroll. in Proceedings of the IEEE International Conference on Computer Vision 3756–3764 (2015).
16. Baltrušaitis, T., Mahmoud, M. & Robinson, P. Cross-dataset learning and person-specific normalisation for automatic Action Unit detection. in 2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG) 1–6 (2015).
17. Paxton, A., & Dale, R. (2013). Frame-differencing methods for measuring bodily synchrony in conversation. Behav. Res. Meth., 45(2), 329-343.Wickham, H. Tiday Data. J. Stat. Softw. 59, 1–23 (2014).
18. Wickham, H. et al. Welcome to the Tidyverse. J. Open Source Softw. 4, 1686 (2019).
19. Müller, K. & Wickham, H. tibble: Simple Data Frames. (2019).
20. Team, R. C. R: A Language and Environment for Statistical Computing. (2019).
21. RStudio Team. RStudio: Integrated Development Environment for R. (2020).
22. Van Rossum, G. & Drake, F. L. Python 3 Reference Manual. (CreateSpace, 2009).
23. Pazske, A. et al. Automatic differentiation in PyTorch. in 31st Conference on Neural Information Processing Systems (NIPS 2017) (2017).
24. Brochers, H. W. pracma: Practical numerical math functions. (2017). R package 2.0.7.
25. Ramseyer, F. T. Motion energy analysis (MEA): A primer on the assessment of motion from video. J. Couns. Psychol. 67, 536–549 (2020).