A pipeline for automated facial expression coding in dyads using OpenFace (feat. a replication study)

Julianna Calabrese
Nathaniel Haines, Ted Beauchaine
May 12, 2021

Introduction

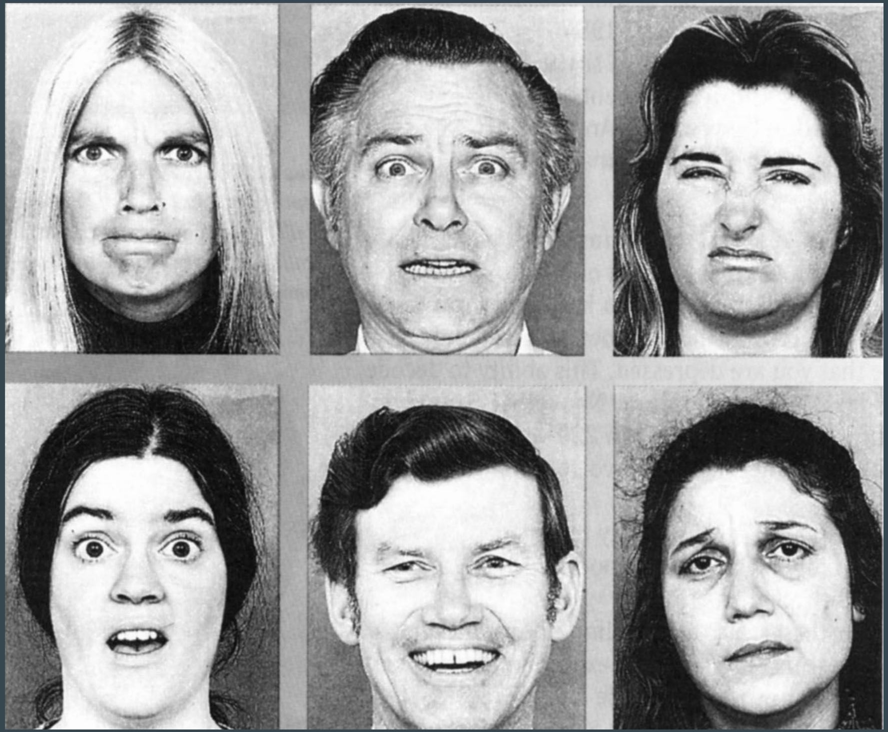
- Faces are important to human communication
- How do we measure facial expressions?

A history of facial expression research

- Darwin's *The Expression of Emotion of Man and Animals*
    - Argued that emotions are intrinsic and universal
    - Across cultures, people tend to generate similar spontaneous facial expressions
        - "The corners of the mouth are drawn downwards, which is so universally recognized as a sign of being out of spirits, that it is almost proverbial." (p. 177)
- Facial expressions facilitate social connectedness and interpersonal relationships (Schmidt & Cohn, 2001)
    - Theorized to be biological adaptations
    - If you see a scary animal, what's faster: trying to communicate in words or exhibiting a fearful facial expression? What's faster to recognize?

Ekman's basic emotions

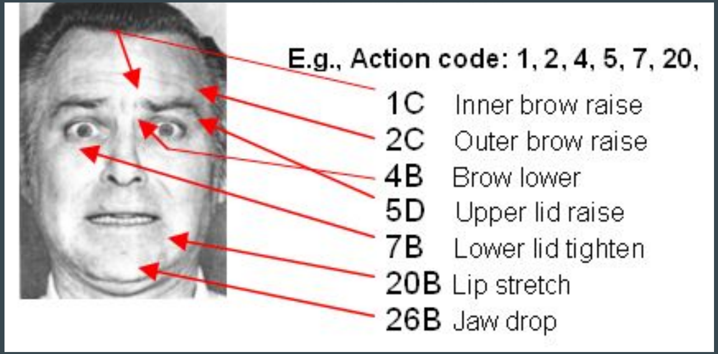
- Basic Emotions (Ekman & Rosenberg, 2005)
  - Anger, Fear, Disgust, Surprise, Happiness, Sadness (and sometimes Contempt)
- Situational reflexes
  - E.g., positive emotion in social relationships
- Link to subjective experience
  - "Mirror into the mind"
  - Social purpose



Two ways to measure facial expressions

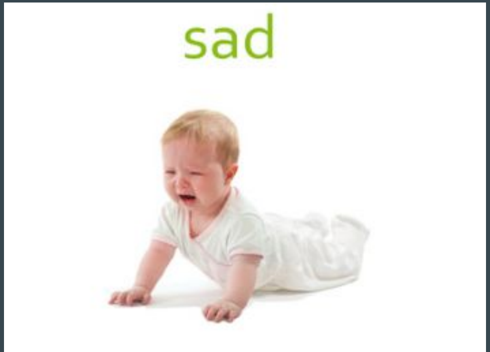
The Physical Features Approach

- Very carefully watch how muscles move in the face
  - E.g., Jaw drop, lip stretch
- Focuses on basic emotions



The Cultural Informants Approach

- Use your cultural familiarity to recognize how emotion is expressed in your culture
- Focuses on overt behavior



Physical Features: FACS Action Units

What’s the best way to measure facial expressions?

- Facial Action Coding System (FACS; Ekman & Friesen, 1975)
  - Physical features approach, discrete view of emotion
  - Pros: gold standard, very precise, focus on muscle movements
  - Cons: time-consuming to learn and conduct
- Facial Expression Coding System (FACES; Kring & Sloan, 2007)
  - Cultural informants approach, dimensional view of emotion
  - Pros: not time-consuming
  - Cons: not as precise as FACS, not as widely used
- Both systems emphasize slowing down video, taking your time, and not letting intuition get in the way with what’s on the screen
  - Both require multiple coders for reliability purposes
- Which one should we pick?

A solution: Automated facial expression coding (AFEC)

- Due to the time-consuming nature of facial coding, automated programs have been designed to code images and videos of faces
  - Eliminates the need to train multiple facial coders
  - Uses the physical features approach
- AFEC software
  - IntraFace (developed at CMU, purchased by Facebook)
  - Emotient (privately developed, purchased by Apple)
  - Noldus FaceReader ($10,000 with the academic discount)
  - iMotions ($2,000 with the academic discount)
  - OpenFace (free, open-source, and available on GitHub)

README.md

OpenFace 2.2.0: a facial behavior analysis toolkit

build passing build passing

Over the past few years, there has been an increased interest in automatic facial behavior analysis and understanding. We present OpenFace – a tool intended for computer vision and machine learning researchers, affective computing community and people interested in building interactive applications based on facial behavior analysis. OpenFace is the first toolkit capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation with available source code for both running and training the models. The computer vision algorithms which represent the core of OpenFace demonstrate state-of-the-art results in all of the above mentioned tasks. Furthermore, our tool is capable of real-time performance and is able to run from a simple webcam without any specialist hardware.

MultiComp Lab

OpenFace was originally developed by Tadas Baltrušaitis in collaboration with CMU MultiComp Lab led by Prof. Louis-Philippe Morency. Some of the original algorithms were created while at Rainbow Group, Cambridge University. The OpenFace library is still actively developed at the CMU MultiComp Lab in collaboration with Tadas Baltršaitis. Special thanks to researcher who helped developing, implementing and testing the algorithms present in OpenFace: Amir Zadeh and Yao Chong Lim on work on the CE-CLM model and Erroll Wood for the gaze estimation work.

OpenFace (Baltrušaitis et al., 2018)

- Developed by Tadas Baltrušaitis at CMU
- Provides data on:
  - Facial landmarks (where are the eyes?)
  - Head pose
  - Eye-gaze
  - FACS Action Units
- "OpenFace 2.0" released in 2018
  - Handles videos with poorly lit conditions, partially covered faces, and non-frontal/side profile faces better than OpenFace 1.0

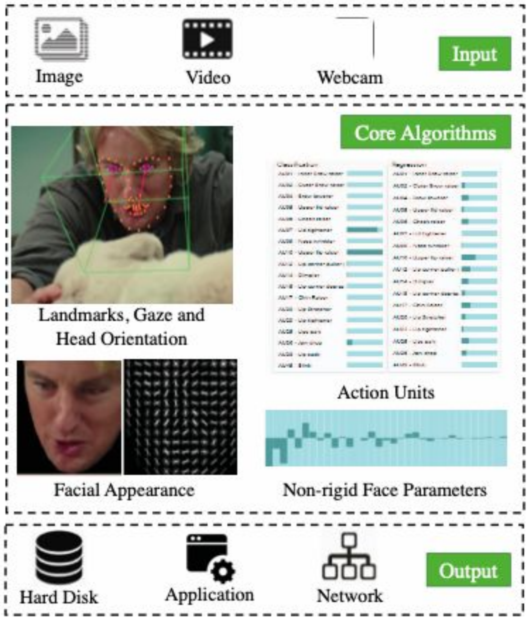


Fig. 1: OpenFace 2.0 is a framework that implements modern facial behavior analysis algorithms including: facial landmark detection, head pose tracking, eye gaze and facial action unit recognition.

Advantages and disadvantages of OpenFace

Advantages

- Free and open-source
  - Reproducibility
  - Transparency
- Cross-platform

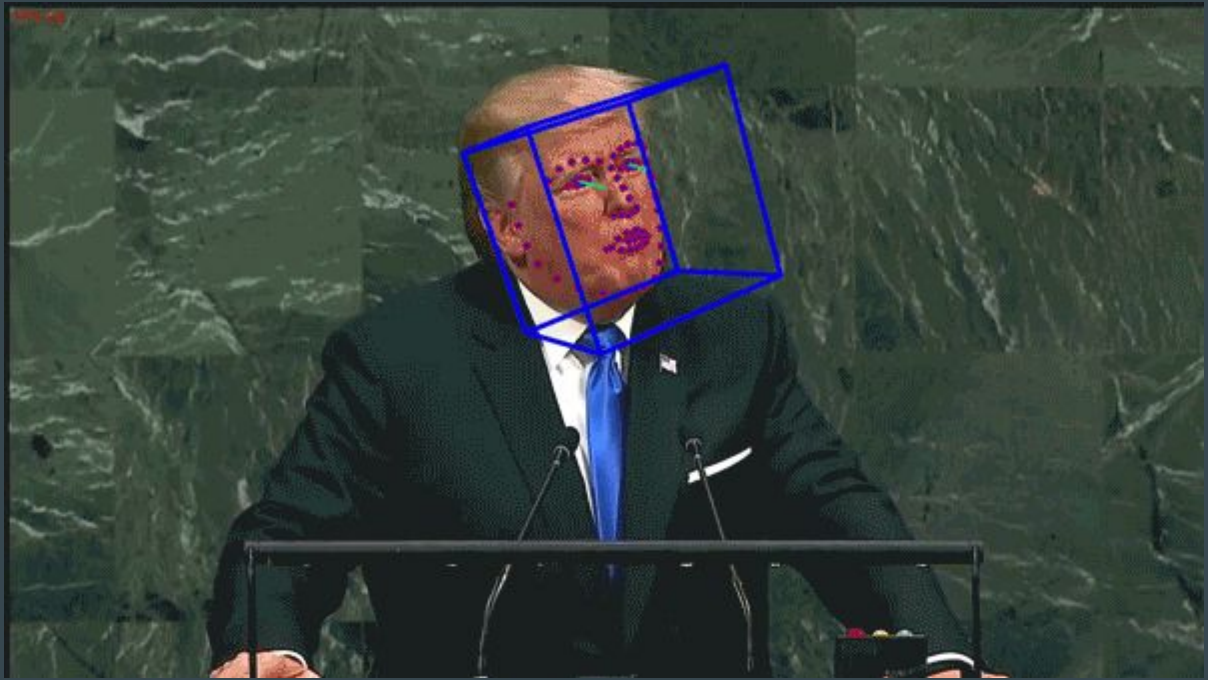
Disadvantages

- Not an “out-of-the-box” software
  - Doesn’t really have a GUI
- Downloading and using requires some hands-on work
- Doesn’t easily provide “composite” emotions*
  - Rather than providing “Happiness” or “Fear” codes, it provides Action Units 4, 5, 9, etc.

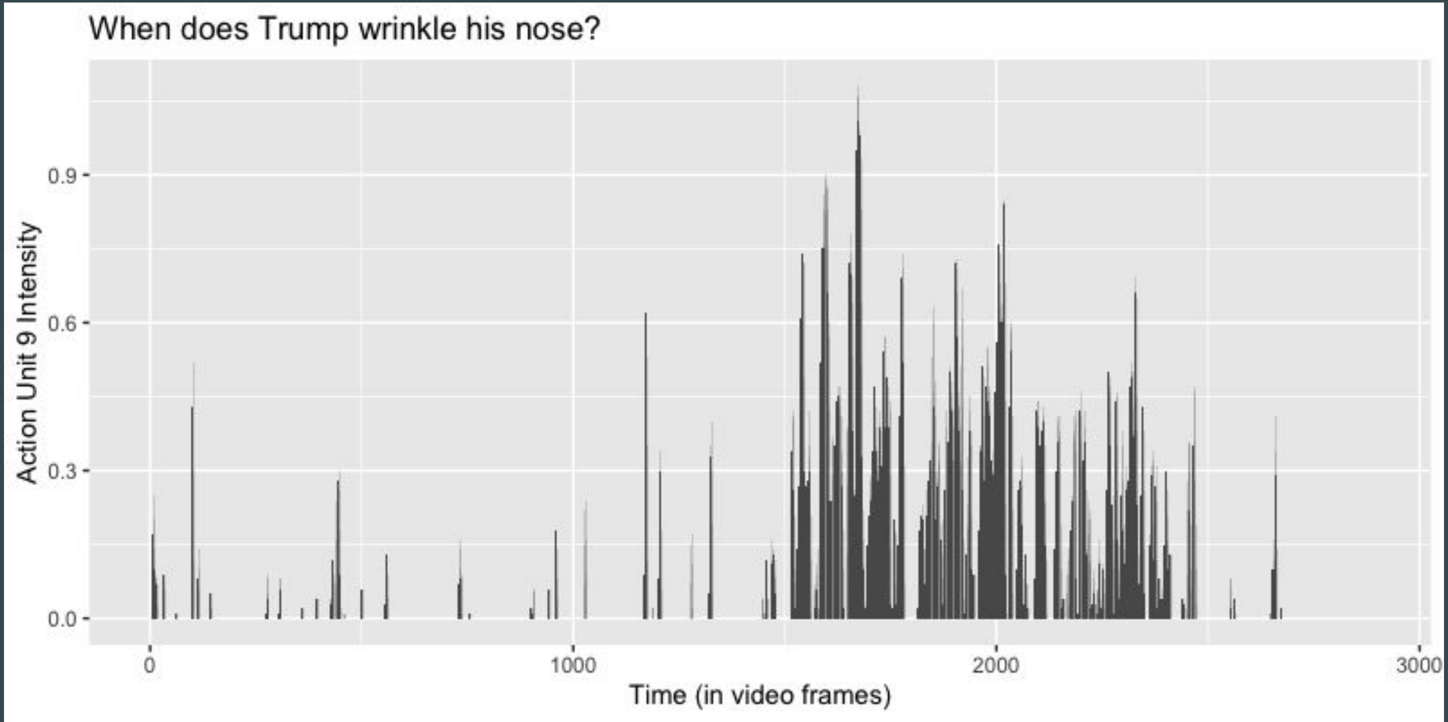
A demonstration of OpenFace

From a 1.5 minute clip where Trump talks about North Korea at the United Nations in September 2017

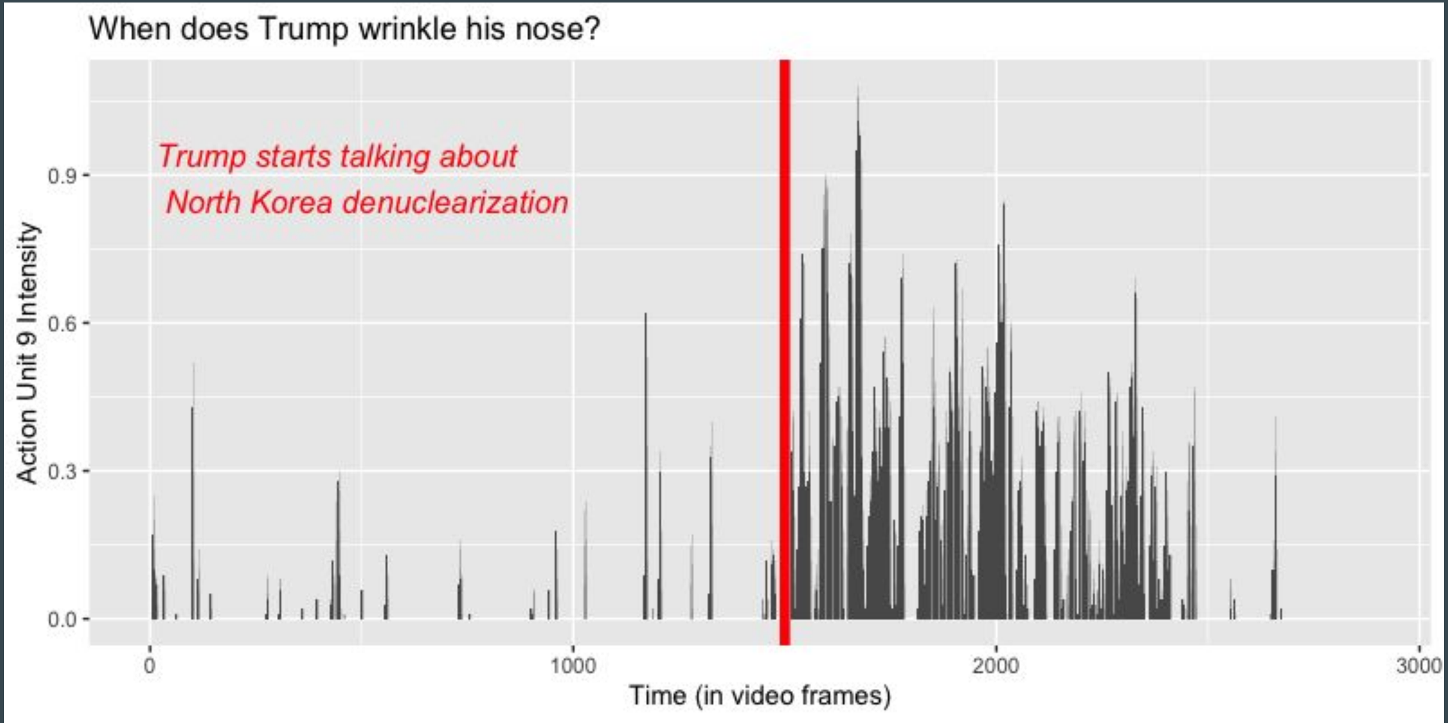
https://www.youtube.com/watch?v=uCngRM-QTmw



Action Unit 9 is “Nose Wrinkler,” associated with disgust



Action Unit 9 is “Nose Wrinkler,” associated with disgust



Project Goals

Project goals

1. Use OpenFace to conduct AFEC on a dataset of mother-daughter dyadic interactions
2. Replicate a study from our lab, Haines et al. (2019), that also used this dataset, but instead of using a commercial software, use OpenFace
   a. Can research that used commercial software be replicated using open-source software?
3. Attempt vector autoregression
4. Write really good R code
   a. The code written for this project can be applied to other videotaped dyadic interactions
5. Finish the presentation that's due on May 12th

Development and Psychopathology (2019), **31**, 871–886
doi:10.1017/S0954579419000312

CAMBRIDGE
UNIVERSITY PRESS

Special Issue Article

Using automated computer vision and machine learning to code facial expressions of affect and arousal: Implications for emotion dysregulation research

Nathaniel Haines[1], Ziv Bell[1], Sheila Crowell[2,3], Hunter Hahn[1], Dana Kamara[1], Heather McDonough-Caplan[1], Tiffany Shader[1] and Theodore P. Beauchaine[1]

[1]Department of Psychology, Ohio State University, Columbus, OH, USA; [2]Department of Psychology, University of Utah, Salt Lake City, UT, USA and [3]Department of Psychiatry, University of Utah, Salt Lake City, UT, USA

Abstract

As early as infancy, caregivers' facial expressions shape children's behaviors, help them regulate their emotions, and encourage or dissuade their interpersonal agency. In childhood and adolescence, proficiencies in producing and decoding facial expressions promote social competence, whereas deficiencies characterize several forms of psychopathology. To date, however, studying facial expressions has been hampered by the labor-intensive, time-consuming nature of human coding. We describe a partial solution: automated facial expression coding (AFEC), which combines computer vision and machine learning to code facial expressions in real time. Although AFEC cannot capture the full complexity of human emotion, it codes positive affect, negative affect, and arousal—core Research Domain Criteria constructs—as accurately as humans, and it characterizes emotion dysregulation with greater specificity than other objective measures such as autonomic responding. We provide an example in which we use AFEC to evaluate emotion dynamics in mother–daughter dyads engaged in conflict.

Haines et al. (2019)

- Original data collected around ~1999 (Crowell et al., 2013)
- 47 mother-daughter dyads
  - Daughters were between the ages of 13 and 18
  - Three groups:
    - Control (13)
    - Depressed (14)
    - Self-harm (20)
- Mother-daughter dyads participated in a 10-minute conflict discussion
  - Audio and video of the interaction was recorded
- Crowell et al. (2013) manually coded the interactions using the Family and Peer Process Code (FPPC), which uses verbal utterances and affective behavior

Haines et al. (2019): Goals

- Use iMotions (a commercial facial coding software) to automatically code the videos of mother-daughter interactions
- Compare automated facial expression coding (AFEC) to the Family Peer and Process Code (FPPC)
- Examine dyadic correspondence
    - Use mother AFEC codes to predict daughter AFEC codes
    - Does the mother “drive” emotion in the daughter?
- Examine group differences (control, depressed, self-harm) in AFEC codes
- Examine the effect of daughter age on dyadic correspondence

Haines et al. (2019): Results

- AFEC/FPPC correlations
  - Mostly medium effect sizes
- Overall, mothers and daughters demonstrated more intense positive affect than negative affect during the interaction
  - Strong across all groups
- Control mothers showed higher positive affect compared to depressed and self-harm mothers
- No group differences in positive affect dyadic correspondence
- Only self-harm dyads showed significant negative affect correspondence
  - Control and depressed dyads did not have significant negative affect correspondence
- Overall, older daughters had greater positive affect dyadic correspondence

iMotions versus OpenFace: a replication issue*

- OpenFace provides raw Action Unit scores
- iMotions provides discrete emotions ("Happiness", "Fear") and dimensional scores (positive, negative)
- How to best replicate?
    - We decided to select specific raw Action Unit scores that correspond well to familiar emotions
    - Making composite emotion scores from the OpenFace Action Units is a future research direction

A few Action Unit examples

→ Action Unit 6 (cheek raiser) – happiness

→ Action Unit 4 (brow lowerer) – sadness

→ Action Units 9 (nose wrinkler) and 10 (upper lip raiser) – anger

Data Wrangling

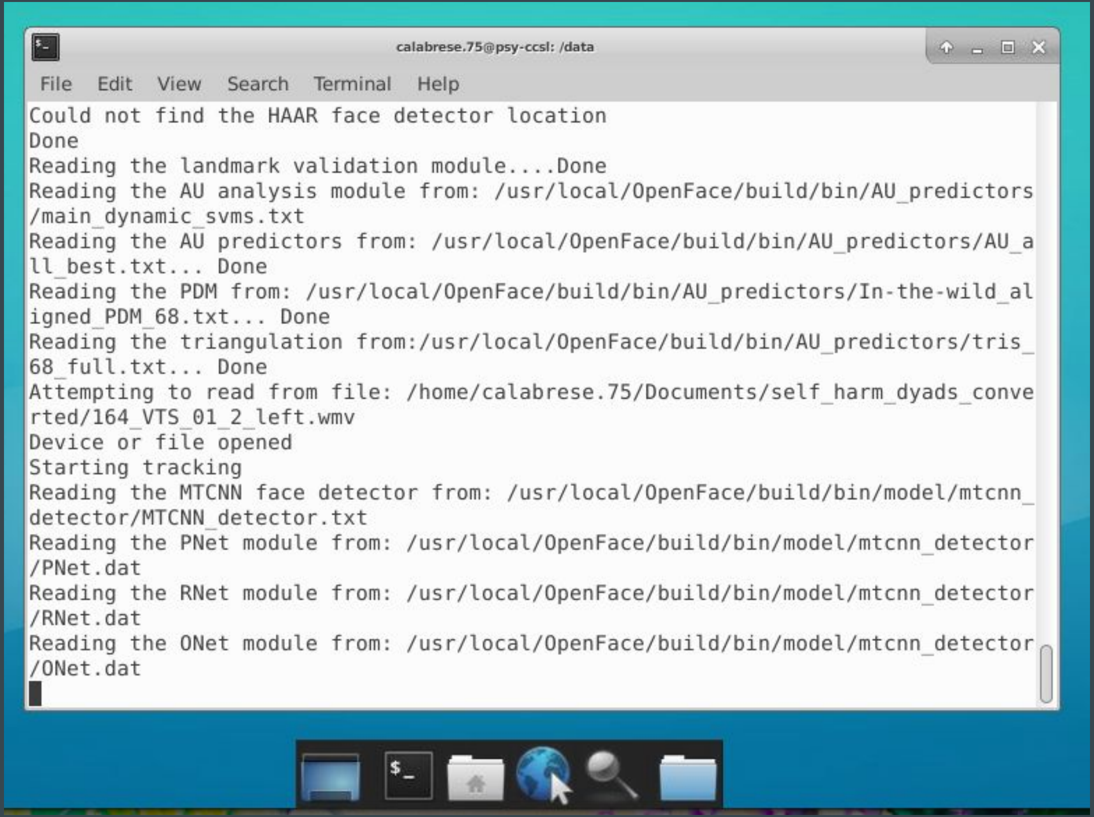
***I probably will have to cut this whole section

- Everything after data collection but before data analysis

What occurred before OpenFace video processing?

- A lot of processing occurred before/during Haines et al. (2019)
  - Videos only existed on DVDs, videos had to be digitized and then converted from .vob to .mp4 format
  - Videos originally had the mother's face on one side and the daughter's face on the other side, videos had to be "split in half" into two separate videos
  - These processes were conducted with ffmpeg
- Downloading OpenFace
  - We decided to download OpenFace on a Linux computing server
  - Other good places would be a desktop in a physical lab, just somewhere with good processing power that can be left alone to run for a few days
  - Downloading required a lot of help from ASCTech

Processing videos with OpenFace from the terminal



- OpenFace has no GUI – all commands are run from the terminal
- The whole process took about a weekend, running overnight
- When it's done, it saves the generated output to a specific folder

What does OpenFace give you when it's done running?

- For every video file you analyze, OpenFace gives you one .csv file that contains ~720 columns and has one row for every one frame of video
    - The columns include data on landmark detection, head position, eye gaze, and facial expressions
    - It also includes basic info like timestamp and tracker confidence
    - We really only care about the first 5 columns and the last ~40 columns
        - We can get rid of everything else
- OpenFace also generates a new video with an overlay of the blue tracker box and a folder of images with each video frame as a .bmp file, but those aren't useful in data analysis

What do OpenFace .csv files look like?

frame	face_id	timestamp	confidence	success	AU01_r	AU02_r	AU04_r
1	0	0	0.88	1	0	0	1.78
2	0	0.033	0.88	1	0	0	1.73
3	0	0.067	0.88	1	0	0	1.71
4	0	0.1	0.88	1	0	0	1.67
5	0	0.133	0.88	1	0	0	1.72
6	0	0.167	0.88	1	0	0	1.69
7	0	0.2	0.88	1	0	0	1.68
8	0	0.234	0.88	1	0	0	1.68
9	0	0.267	0.88	1	0	0	1.74
10	0	0.3	0.88	1	0	0	1.79
11	0	0.334	0.88	1	0	0	1.82
12	0	0.367	0.88	1	0	0	1.78
13	0	0.4	0.88	1	0	0	1.68
14	0	0.434	0.88	1	0	0	1.66
15	0	0.467	0.88	1	0	0	1.66
16	0	0.501	0.88	1	0	0	1.72
17	0	0.534	0.88	1	0	0	1.65
18	0	0.567	0.88	1	0	0	1.63
19	0	0.601	0.88	1	0	0	1.61
20	0	0.634	0.98	1	0	0	1.7
21	0	0.667	0.98	1	0	0	1.73
22	0	0.701	0.98	1	0	0	1.69

frame = frame of the video

face_id = whose face this is? (in case of multiple faces)

timestamp = recorded time

confidence = how confident is the tracker in the current landmark detection? (between 0 and 1)

success = is the track successful? (0 = unsuccessful, 1 = successful)

AUxx_r = AU X intensity
(0 min, 5 max)

AUxx_c = AU X presence
(0 = not there, 1 = there)

What’s the catch?

- Due to the way videos were recorded and stored, each entire dyadic interaction (~10 minutes) was split into video “chunks”, labelled by its by dyad ID, side (left or right), and “chunk” number
  - Each .csv file of each video “chunk” contained ~720 columns and ~10,000 rows
  - There were 5-10 “chunks” per person per dyad
  - Overall, there were 439 video “chunks”
- Information on dyad ID, sidedness, and “chunk” was only located in the .csv name, not in any columns or rows within the .csv file
- “Chunk” number was not sequential
  - Some videos started at 7 and ended at 10
- The Linux server that all these files are located on is very, very slow...

What’s the goal?

- Make something suitable for data analysis
  - Subset so we only have columns that we care about
  - Merge so everyone is in one dataframe
  - Make it long so timepoints are nested within dyad ID
  - Separate out by side so mothers and daughters have their own columns, but still make it so mother and daughter columns are included within the same dyad ID row
  - Scale the data so Action Units are zero-centered
  - Smooth the data into 10-second intervals
- Where to start?

First, get rid of those ~675 columns

While on the Linux server:

- Read in the .csv files as a list
- Subset the dataframes within the list
- Read out the dataframes individually into a new folder
- And then move that to somewhere that isn't the Linux server

```
# set directory
setwd("/data/csv")

# this puts all the csv files in the csv folder
# in a big list
filelist <- list.files(pattern="*.csv", full.names=TRUE)
lst <- lapply(filelist, read.csv, header=TRUE, stringsAsFactors=FALSE)
# the line below this selects only the columns we're interested in
lst1 <- lapply(lst, "[", c(1:5,680:714))
rm(lst)

# fix names
names(lst1) <- filelist
names(lst1) <- substring(names(lst1), 3)

# set a new directory
setwd("/data/small")

# this writes out all the smaller csv files
# individually into the small folder
lapply(1:length(lst1), function(i) write.csv(lst1[[i]],
                                             file = paste0(names(lst1[i])),
                                             row.names = FALSE))
```

Now we have a list of smaller/manageable .csv files, but...

```
Console   Terminal ×   Markers ×   Jobs ×
~/OneDrive - The Ohio State University/firstyearproject/csv/
> # Read in file as a list
> filelist <- list.files(pattern="*.csv", full.names=TRUE)
> print(filelist)
 [1] "./121_VTS_01_7_left.csv"   "./121_VTS_01_7_right.csv"  "./121_VTS_01_8_left.csv"   "./121_VTS_01_8_right.csv"
 [5] "./121_VTS_01_9_left.csv"   "./121_VTS_01_9_right.csv"  "./128_VTS_01_6_left.csv"   "./128_VTS_01_6_right.csv"
 [9] "./128_VTS_01_7_left.csv"   "./128_VTS_01_7_right.csv"  "./130_VTS_01_2_left.csv"   "./130_VTS_01_2_right.csv"
[13] "./130_VTS_01_3_left.csv"   "./130_VTS_01_3_right.csv"  "./130_VTS_01_4_left.csv"   "./130_VTS_01_4_right.csv"
[17] "./130_VTS_01_5_left.csv"   "./130_VTS_01_5_right.csv"  "./130_VTS_01_6_left.csv"   "./130_VTS_01_6_right.csv"
[21] "./130_VTS_01_7_left.csv"   "./130_VTS_01_7_right.csv"  "./133_VTS_01_1_left.csv"   "./133_VTS_01_1_right.csv"
[25] "./133_VTS_01_2_left.csv"   "./133_VTS_01_2_right.csv"  "./133_VTS_01_3_left.csv"   "./133_VTS_01_3_right.csv"
[29] "./133_VTS_01_4_left.csv"   "./133_VTS_01_4_right.csv"  "./133_VTS_01_5_left.csv"   "./133_VTS_01_5_right.csv"
[33] "./133_VTS_01_6_left.csv"   "./133_VTS_01_6_right.csv"  "./133_VTS_01_7_left.csv"   "./133_VTS_01_7_right.csv"
[37] "./133_VTS_01_8_left.csv"   "./133_VTS_01_8_right.csv"  "./133_VTS_01_9_left.csv"   "./133_VTS_01_9_right.csv"
[41] "./134_VTS_01_1_left.csv"   "./134_VTS_01_1_right.csv"  "./134_VTS_01_2_left.csv"   "./134_VTS_01_2_right.csv"
[45] "./134_VTS_01_3_left.csv"   "./134_VTS_01_3_right.csv"  "./134_VTS_01_4_left.csv"   "./134_VTS_01_4_right.csv"
[49] "./134_VTS_01_5_left.csv"   "./134_VTS_01_5_right.csv"  "./134_VTS_01_6_left.csv"   "./134_VTS_01_6_right.csv"
[53] "./134_VTS_01_7_left.csv"   "./134_VTS_01_7_right.csv"  "./134_VTS_01_8_left.csv"   "./134_VTS_01_8_right.csv"
[57] "./134_VTS_01_9_left.csv"   "./134_VTS_01_9_right.csv"  "./135_VTS_01_1_right.csv"  "./135_VTS_01_2_right.csv"
[61] "./135_VTS_01_3_left.csv"   "./135_VTS_01_3_right.csv"  "./135_VTS_01_4_left.csv"   "./135_VTS_01_4_right.csv"
[65] "./135_VTS_01_5_left.csv"   "./135_VTS_01_5_right.csv"  "./135_VTS_01_6_left.csv"   "./135_VTS_01_6_right.csv"
[69] "./135_VTS_01_7_left.csv"   "./135_VTS_01_7_right.csv"  "./135_VTS_01_8_left.csv"   "./135_VTS_01_8_right.csv"
[73] "./135_VTS_01_9_left.csv"   "./135_VTS_01_9_right.csv"  "./136_VTS_01_1_left.csv"   "./136_VTS_01_1_right.csv"
```

There's no data on dyad ID, sidedness, or "chunk" in the dataframe itself, it's only in the dataframe title – how to get it out?

Dataframe title pattern: [dyad_ID]_VTS_01_[chunk_number]_[side].csv

Steal strings from titles, make variables

- Read in all the .csv files as a list
- Make another list that's a list of dataframe titles
- Steal words from titles, make new variables that indicate ID, sidedness, and "chunk" order
- The dataframes are still in a list, but now they have reference variables

````
### Create clipID

The clipID variable is for dyads -- a mother and daughter who are part of the same dyad will have the same variable.

```{r}
# Create clipID
namelist <- substr(filelist, 1, 3)
lst <- mapply(cbind, lst, "clipID"=namelist, SIMPLIFY=F)
```

### Create side

```{r}
# Create side
videolist <- substr(filelist, 14, 18)
lst <- mapply(cbind, lst, "side"=videolist, SIMPLIFY=F)
```

### Create order

This is needed because the DVD'ing process that occurred long ago split up videos into smaller bits. The order variables helps us keep track of the chronological order of the video bits within the whole interaction task.

```{r}
# Create order
orderlist <- substr(filelist, 12, 12)
lst <- mapply(cbind, lst, "order"=orderlist, SIMPLIFY=F)
```
````

Turn a list of things into one big thing

- Merge all the dataframes in the list into one (very large) long dataframe
  - This step alone took ~2 hours
- Reformat it into a wide-format, separated out by side
- Merge it with another reference dataframe on sidedness for later use
  - Sometimes mothers were on the right side, sometimes they were on the left
- The end result has about double the amount of columns, since there is now a value for both the left side and the right side
  - Now it has 1,374,913 rows and 83 columns
  - Still a lot, but it’s a big improvement
  - The variable “mother_side” exists for each dyad ID, which says whether the mother is on the left or right side

```
Console   Terminal   Markers   Jobs
~/Documents/OSU/LAP/Research/
> names(data)
 [1] "clipID"          "order"            "frame_left"      "frame_right"
 [5] "face_id_left"    "face_id_right"    "timestamp_left"  "timestamp_right"
 [9] "confidence_left" "confidence_right" "success_left"    "success_right"
[13] "AU01_r_left"     "AU01_r_right"     "AU02_r_left"     "AU02_r_right"
[17] "AU04_r_left"     "AU04_r_right"     "AU05_r_left"     "AU05_r_right"
[21] "AU06_r_left"     "AU06_r_right"     "AU07_r_left"     "AU07_r_right"
[25] "AU09_r_left"     "AU09_r_right"     "AU10_r_left"     "AU10_r_right"
[29] "AU12_r_left"     "AU12_r_right"     "AU14_r_left"     "AU14_r_right"
[33] "AU15_r_left"     "AU15_r_right"     "AU17_r_left"     "AU17_r_right"
[37] "AU20_r_left"     "AU20_r_right"     "AU23_r_left"     "AU23_r_right"
[41] "AU25_r_left"     "AU25_r_right"     "AU26_r_left"     "AU26_r_right"
[45] "AU45_r_left"     "AU45_r_right"     "AU01_c_left"     "AU01_c_right"
[49] "AU02_c_left"     "AU02_c_right"     "AU04_c_left"     "AU04_c_right"
[53] "AU05_c_left"     "AU05_c_right"     "AU06_c_left"     "AU06_c_right"
[57] "AU07_c_left"     "AU07_c_right"     "AU09_c_left"     "AU09_c_right"
[61] "AU10_c_left"     "AU10_c_right"     "AU12_c_left"     "AU12_c_right"
[65] "AU14_c_left"     "AU14_c_right"     "AU15_c_left"     "AU15_c_right"
[69] "AU17_c_left"     "AU17_c_right"     "AU20_c_left"     "AU20_c_right"
[73] "AU23_c_left"     "AU23_c_right"     "AU25_c_left"     "AU25_c_right"
[77] "AU26_c_left"     "AU26_c_right"     "AU28_c_left"     "AU28_c_right"
[81] "AU45_c_left"     "AU45_c_right"     "mother_side"
```

And then what?

- Loop through each dyad ID, rename each variable depending which side the mother is on

```
# Loop through each dyad and identify mothers/daughters

```{r}
# Loop through each dyad and identify mothers/daughters
#fit_dat$clipID <- as.factor(fit_dat$clipID)
num_dyads <- length(unique(data$clipID))

fit_dat <- foreach(i=1:num_dyads, .combine = "rbind") %do% {
  tmp_subj <- data %>%
    filter(clipID==unique(clipID)[i]) %>%
    mutate(frame_mother    = ifelse(mother_side=="right", frame_left, frame_right),
           frame_daughter   = ifelse(mother_side=="right", frame_right, frame_left),
           face_id_mother   = ifelse(mother_side=="right", face_id_left, face_id_right),
           face_id_daughter  = ifelse(mother_side=="right", face_id_right, face_id_left),
           timestamp_mother  = ifelse(mother_side=="right", timestamp_left, timestamp_right),
           timestamp_daughter  = ifelse(mother_side=="right", timestamp_right, timestamp_left),
           confidence_mother   = ifelse(mother_side=="right", confidence_left, confidence_right),
           confidence_daughter   = ifelse(mother_side=="right", confidence_right, confidence_left),
           success_mother   = ifelse(mother_side=="right", success_left, success_right),
           success_daughter   = ifelse(mother_side=="right", success_right, success_left),
           AU01_r_mother    = ifelse(mother_side=="right", AU01_r_left, AU01_r_right),
```

- Center each AUxx_r variable

```
   mutate(AU01_r_mother =  scale(AU01_r_mother),
          AU01_r_daughter =  scale(AU01_r_daughter),
          AU02_r_mother =  scale(AU02_r_mother),
          AU02_r_daughter =  scale(AU02_r_daughter),
          AU04_r_mother =  scale(AU04_r_mother),
          AU04_r_daughter =  scale(AU04_r_daughter),
          AU05_r_mother =  scale(AU05_r_mother),
          AU05_r_daughter =  scale(AU05_r_daughter),
          AU06_r_mother =  scale(AU06_r_mother),
          AU06_r_daughter =  scale(AU06_r_daughter),
```

- “Smooth” the data by averaging every ~10 second interval (about 30 rows)

```
fit_dat_smooth <- fit_dat %>%
  group_by(ID, GROUP, grp = as.integer(gl(n(), 30,
                                          n()))) %>%
  dplyr::summarise(across(starts_with(c('frame', 'AU')), mean), .groups = 'drop')
```
```

The end results looks like this, ready for analysis:

fit_dat_smooth

Filter   Cols: « ‹ 0 - 50 › »

	ID	GROUP	grp	frame_mother	frame_daughter	AU01_r_mother	AU01_r_daughter	AU02_r_mother	AU02_r_daughter
1	121	self-harm	1	15.5	15.5	-0.5202291685	-0.513425996	-0.367264280	-0.3620912
2	121	self-harm	2	45.5	45.5	-0.2978412244	-0.506254376	-0.367264280	-0.3557057
3	121	self-harm	3	75.5	75.5	-0.3498169508	-0.513425996	-0.367264280	-0.3620912
4	121	self-harm	4	105.5	105.5	-0.1223166401	-0.511035456	-0.360422617	-0.3620912
5	121	self-harm	5	135.5	135.5	-0.3038056520	-0.513425996	-0.367264280	-0.3620912
6	121	self-harm	6	165.5	165.5	0.1929459628	-0.513425996	-0.367264280	-0.3620912
7	121	self-harm	7	195.5	195.5	-0.0422228977	-0.513425996	-0.333055967	-0.3620912
8	121	self-harm	8	225.5	225.5	0.2645190942	-0.513425996	-0.367264280	-0.3450632
9	121	self-harm	9	255.5	255.5	-0.5151168020	-0.511035456	0.009027162	-0.3620912
10	121	self-harm	10	285.5	285.5	-0.0004719044	-0.513425996	-0.223589366	-0.3620912
11	121	self-harm	11	315.5	315.5	-0.1887774050	-0.513425996	-0.171136619	-0.3259068
12	121	self-harm	12	345.5	345.5	-0.4128694713	-0.513425996	-0.335336521	-0.3620912
13	121	self-harm	13	375.5	375.5	-0.1998541992	-0.513425996	-0.326214304	-0.3301638
14	121	self-harm	14	405.5	405.5	-0.0660806082	-0.513425996	-0.274901835	-0.3620912
15	121	self-harm	15	435.5	435.5	-0.0976068685	-0.491114290	-0.334196244	-0.3620912

Showing 1 to 15 of 45,877 entries, 75 total columns

Data Analysis & Results

- Will it replicate?

Models from Haines et al. (2019)

```
```{r}
# Contrast matrix
mat1 <- rbind(c(1, -0.5, -0.5), # control vs. (depressed + self-harm) / 2
              c(0, -1, 1))      # depressed vs. self-harm
cMat1 <- MASS::ginv(mat1)
```

```{r echo=TRUE}
# Positive affect model
fit_posFR <- lmer(pos_mother ~ pos_daughter * GROUP + (pos_daughter | ID),
                  data = Nates_old_data,
                  contrasts = list(GROUP = cMat1))
```

```{echo=TRUE}
# Negative affect model
fit_negFR <- lmer(neg_mother ~ neg_daughter * GROUP + (neg_daughter | ID),
                  data = Nates_old_data,
                  contrasts = list(GROUP = cMat1))
```
```

Haines et al. (2019) used composite emotion scores generated by iMotions

Recreating Haines et al. (2019) with OpenFace

````
```{r echo=TRUE}
mod06 <- lmer(AU06_r_mother ~ AU06_r_daughter * GROUP + (AU06_r_daughter | ID),
                    data = fit_dat_smooth,
                    contrasts = list(GROUP = cMat1))
```

```{echo=TRUE}
mod12 <- lmer(AU12_r_mother ~ AU12_r_daughter * GROUP + (AU12_r_daughter | ID),
                    data = fit_dat_smooth,
                    contrasts = list(GROUP = cMat1))
```

```{echo=TRUE}
mod25 <- lmer(AU25_r_mother ~ AU25_r_daughter * GROUP + (AU25_r_daughter | ID),
                    data = fit_dat_smooth,
                    contrasts = list(GROUP = cMat1))
```
````

With OpenFace, we used raw Action Unit scores

The lmer() function from `lme4` package was used in both cases

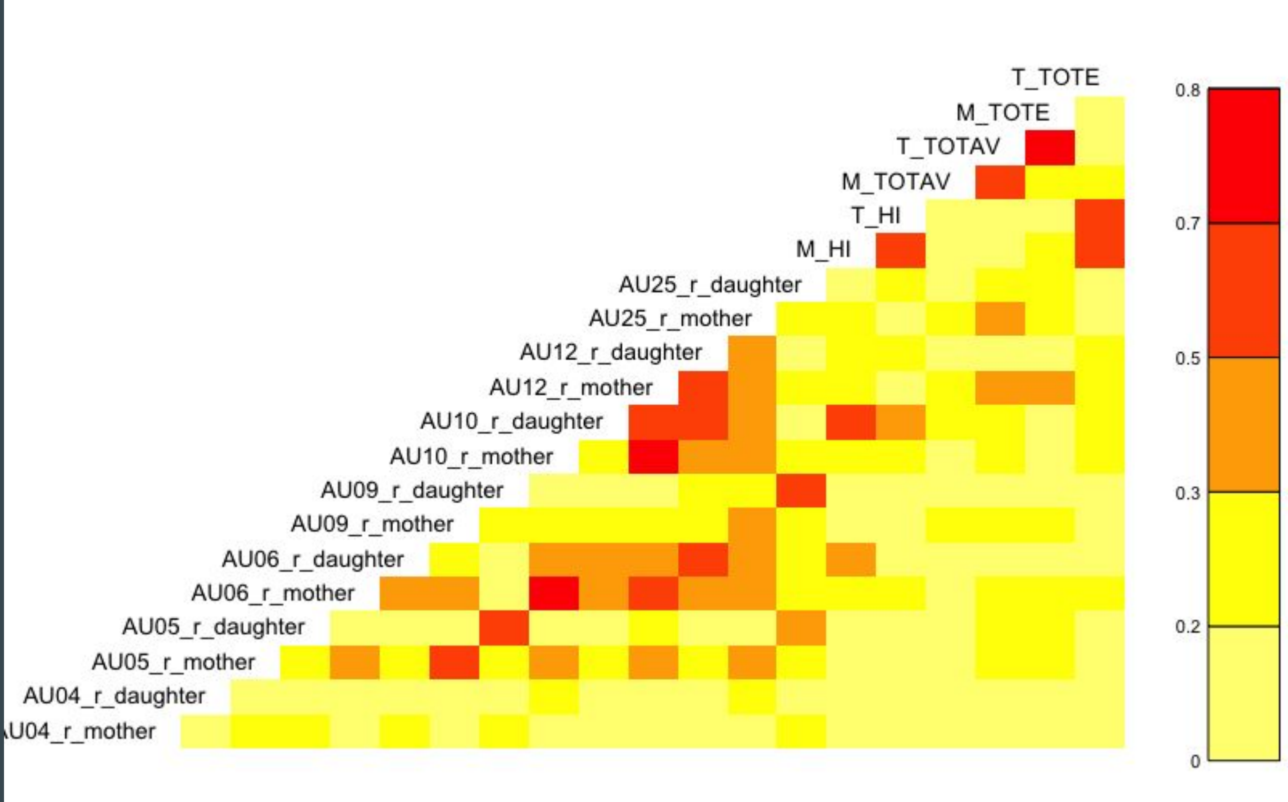
Results: Mixed models to examine dyadic correspondence

- There is a significant difference in control versus depressed+self-harm mothers in AU09 and AU25 dyadic correspondence
  - There is greater dyadic correspondence for AU09 and AU25 for control mothers compared to depressed+self-harm mothers
  - Control mothers tend to “drive” AU09 and AU25 in their daughters compared to depressed+self-harm mothers

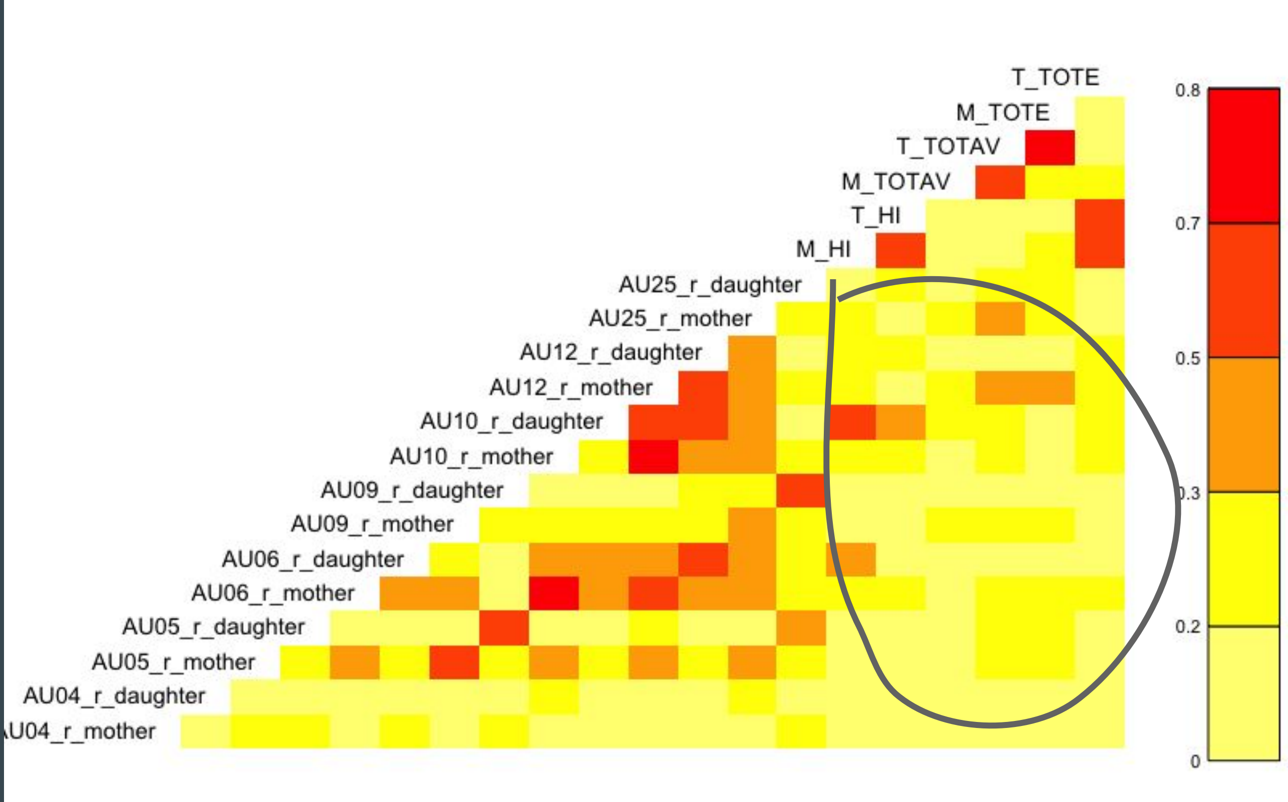
```
> round(coef(summary(mod09)), 3)
                     Estimate Std. Error     df t value Pr(>|t|)
(Intercept)             0.012      0.018 44.020   0.682    0.499
AU09_r_daughter         0.047      0.014 41.981   3.346    0.002
GROUP1                  0.006      0.041 43.961   0.154    0.878
GROUP2                  0.012      0.042 44.095   0.287    0.776
AU09_r_daughter:GROUP1  0.074      0.031 41.389   2.373    0.022
AU09_r_daughter:GROUP2  0.021      0.032 42.731   0.661    0.512
>
```

```
> round(coef(summary(mod25)), 3)
                       Estimate Std. Error     df t value Pr(>|t|)
(Intercept)               0.052      0.030 44.147   1.722    0.092
AU25_r_daughter           0.103      0.026 43.014   3.953    0.000
GROUP1                   -0.013      0.067 44.136  -0.191    0.850
GROUP2                   -0.091      0.069 44.162  -1.311    0.197
AU25_r_daughter:GROUP1    0.132      0.058 43.024   2.261    0.029
AU25_r_daughter:GROUP2    0.047      0.060 43.003   0.786    0.436
>
```

Results: OpenFace/FPPC Correlations



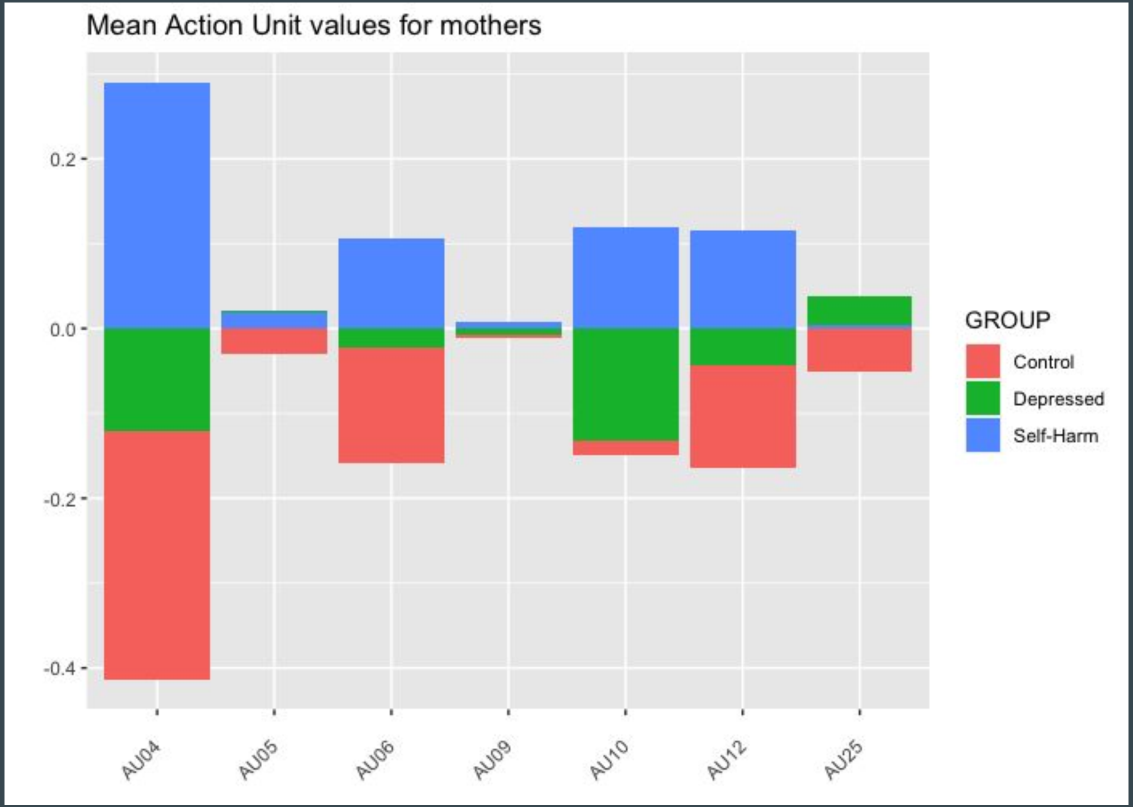
Results: OpenFace/FPPC Correlations



Results: OpenFace/FPPC Correlations

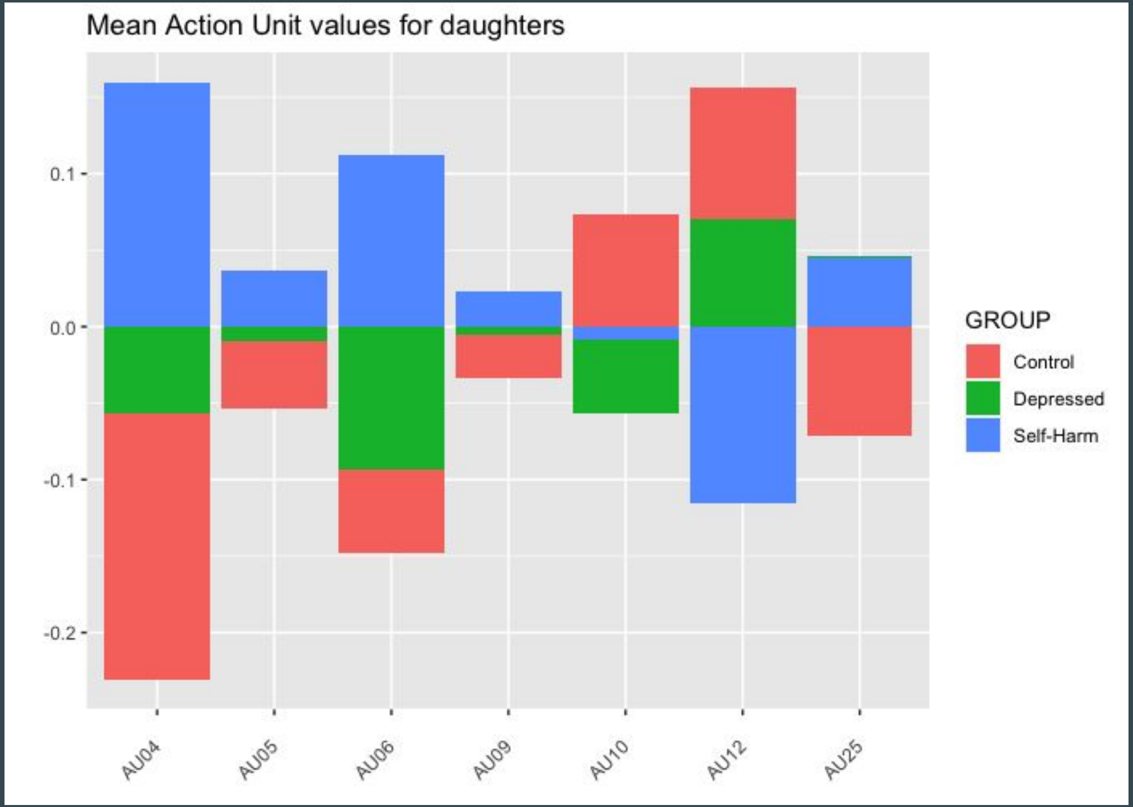
- Daughter AUs
  - +0.50 r between daughter AU10 and mother high aversiveness
  - +0.37 r between daughter AU10 and daughter high aversiveness
- Mother AUs
  - -0.41 r between mother AU12 and daughter total aversiveness
  - +0.37 r between mother AU12 and mother total escalations
  - -0.39 r between mother AU25 and daughter high aversiveness
- These findings make sense – negative AUs were positively associated with aversiveness, AU25 (positive emotion) was negatively associated with aversiveness

Results: Group differences in Mother Action Units



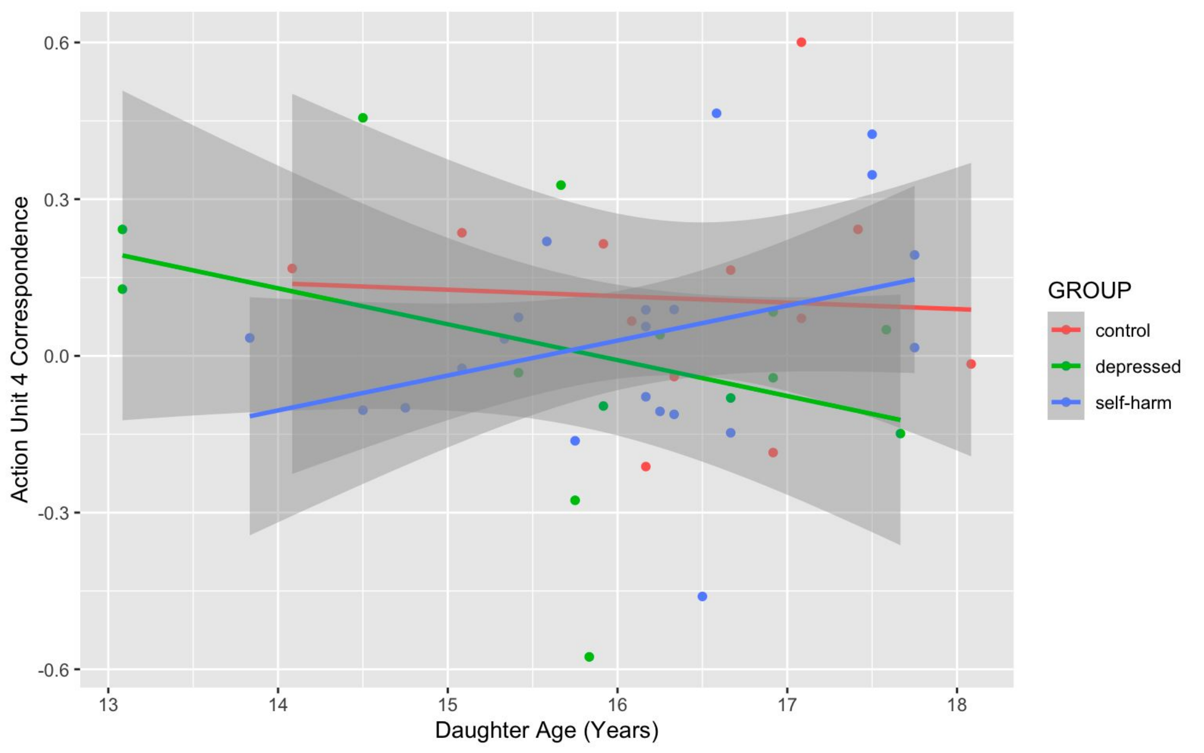
- Mothers in self-harm dyads tended to show greater intensity of all AUs

Results: Group differences in Daughter Action Units



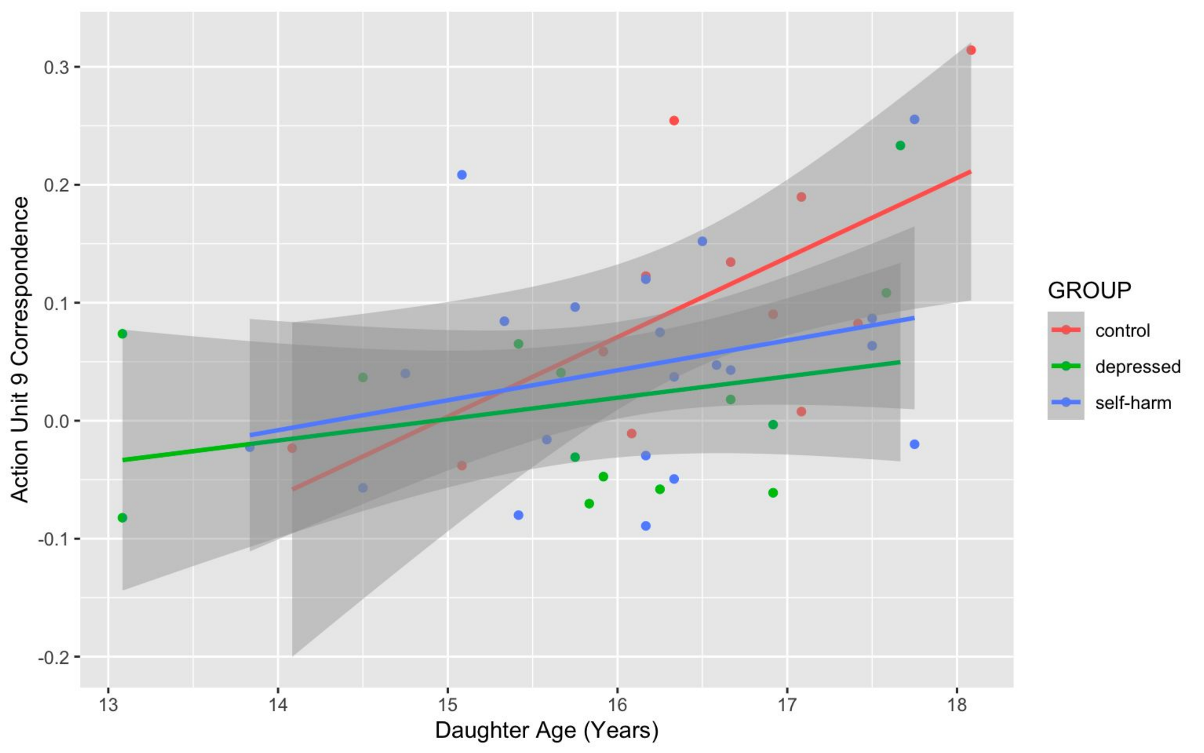
- Daughters in control dyads showed less negative AUs, more positive AUs, compared to daughters from other dyads

Results: The effects of age (Action Unit 4)



- Not much of an effect of age on Action Unit 4

Results: The effects of age (Action Unit 9)



- A pretty strong effect of age on Action Unit 9, especially in control dyads

Did it replicate?

- OpenFace AUs correlated okay with the FPPC codes
  - FPPC was coded with the cultural informants approach, while OpenFace takes a particularly fine-grained physical features approach
- Found some group effects in dyadic correspondence
  - Control mothers “drove” AU09 (negative emotion) and AU25 (positive emotion) in their daughters compared to depressed+self-harm mothers
  - In Haines et al. (2019), only self-harm dyads showed negative affect correspondence
- Replicated effect of age on AUs
  - E.g., strong effect of age on AU09 dyadic correspondence
- Conclusion – not bad

Vector Autoregression

***?

Vector autoregression

- What is vector autoregression?
    - Models bidirectional effects across time (Eason et al., 2020)
    - In vector autoregression, a current value is explained using its own lagged value (Stock & Watson, 2001)
    - Originally developed by econometricians

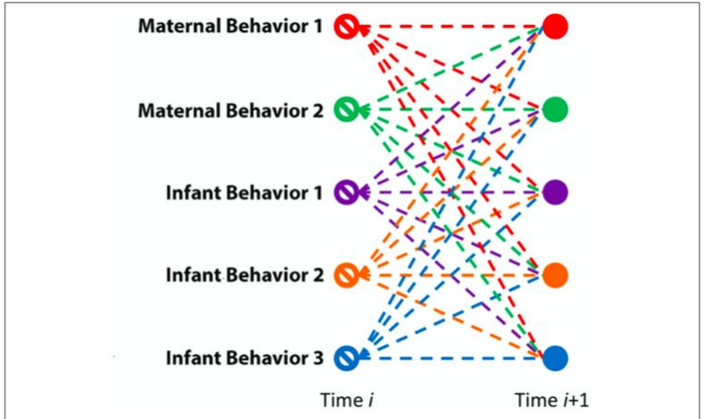


FIGURE 1 | VAR(1). Schematic of relationships modeled by vector autoregression at a lag of one time step.

Eason et al. (2020)

frontiers in Psychology

ORIGINAL RESEARCH
published: 05 August 2020
doi: 10.3389/fpsyg.2020.01507

Check for updates

Using Vector Autoregression Modeling to Reveal Bidirectional Relationships in Gender/Sex-Related Interactions in Mother–Infant Dyads

Elizabeth G. Eason[1], Nicole S. Carver[2], Damian G. Kelty-Stephen[3] and Anne Fausto-Sterling[4]*

[1] Department of Mathematics and Statistics, Grinnell College, Grinnell, IA, United States, [2] Department of Psychology, University of Cincinnati, Cincinnati, OH, United States, [3] Department of Psychology, Grinnell College, Grinnell, IA, United States, [4] Department of Molecular Biology, Cell Biology and Biochemistry, Brown University, Providence, RI, United States

OPEN ACCESS

Edited by:
Klaus Libertus,
University of Pittsburgh, United States

Vector autoregression (VAR) modeling allows probing bidirectional relationships in gender/sex development and may support hypothesis testing following multi-modal data collection. We show VAR in three lights: supporting a hypothesis, rejecting a hypothesis, and opening up new questions. To illustrate these capacities of VAR, we reanalyzed longitudinal data that recorded dyadic mother–infant interactions for 15 boys

frontiers

Supplementary Material

Using vector-autoregressive modeling to reveal bidirectional relationships in sex-related interactions in mother infant dyads

Elizabeth G. Eason, Nicole S. Carver, Damian G. Kelty-Stephen*, and Anne Fausto-Sterling

*** Correspondence:** Damian G. Kelty-Stephen, foovian@gmail.com

Supplementary Data 2. Example R script for running VAR and IRF procedures

```
library(vars) #begin by invoking the library "vars"

data<-read.csv("inputfilename.csv") #read in the data file.

varout<-VAR(data,p=1) #where data is a data.frame, and p = the number of lags

#see help(VAR) for more details of other input arguments

arch.test(varout) #autoregressive conditional heteroscedasticity test. This is a test for
#heteroscedasticity for multivariate series, and it applies here to test whether the residuals of the
#VAR model are heteroscedastic.

serial.test(varout) #test for serial correlations

#Both of the above tests should be nonsignificant, as we do not want to reject the null hypotheses of
#homoscedasticity and independence across time

irfout<-irf(varout) #generates the IRFs for vars.
```

An attempt at vector autoregression

- To start off, we only looked at individual dyads, not the whole dataset
- Smooth the dataset so videoframes are averaged every 90 rows (~30 seconds) rather than 30 rows (~10 seconds)
- Use mother and daughter AUs to predict other mother and daughter AUs
  - Use the `vars` package and code from Eason et al. (2020)
  - Look at AUs 6, 9, 10, and 12
- Look at only three random dyads:
  - 147 (Control), 146 (Depressed), and 175 (Self-harm)

````
```{r}
# 90 is about 30 seconds
smoother_data <- fit_dat_unsmooth %>%
  group_by(ID, GROUP, grp = as.integer(gl(n(), 90, # <- change to smooth more or less
                                          n()))) %>%
  dplyr::summarise(across(starts_with(c('frame', 'AU')), mean), .groups = 'drop')
```
````

How to do vector autoregression...

```{r}
## Dyad 147 (Control)
# Daughter is 17.4 years old

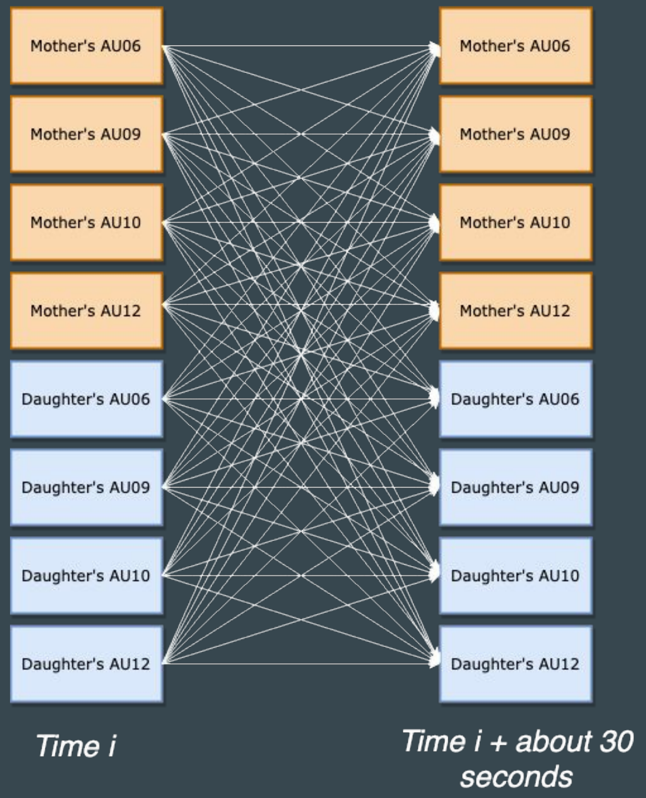
# Subset data to only include relevant mother/daughter variables
dyad_147 <- smoother_data %>%
  filter(ID == c("147")) %>%
  dplyr::select(contains(c("06_r", "12_r", # Positive emotion
                           "09_r", "10_r" # Negative emotion
                           )))

# Make sure it's a dataframe
dyad_147 <- data.frame(dyad_147)

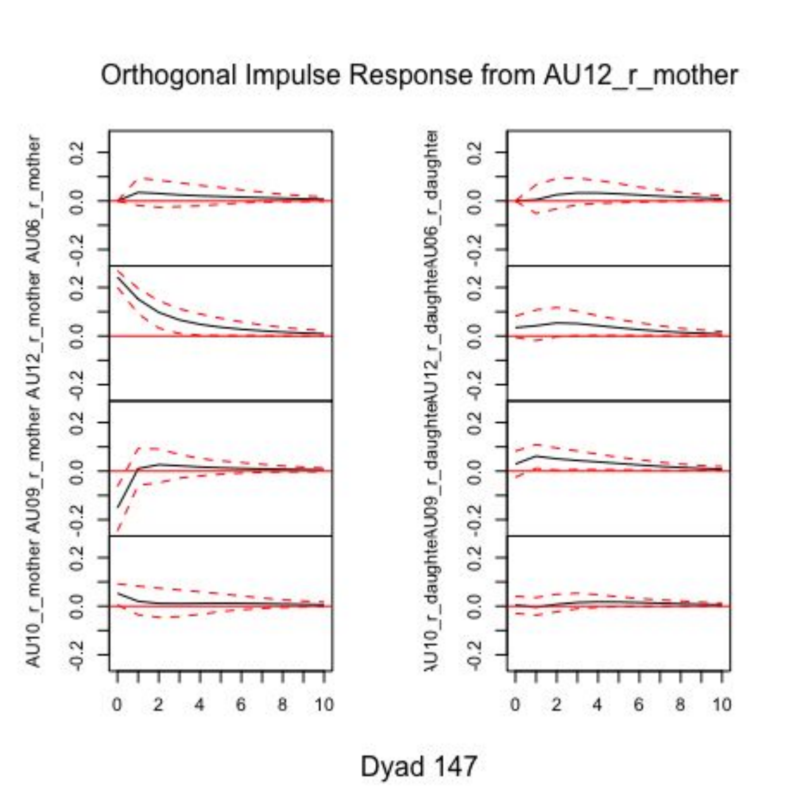
# Create a var object for the dyad
varout <- VAR(dyad_147, p=1)

# Look at it
summary(varout)

# Plot it
plot(irf(varout), sub = "Dyad 147")
```



Dyad 147 (Control) for Mother's AU12

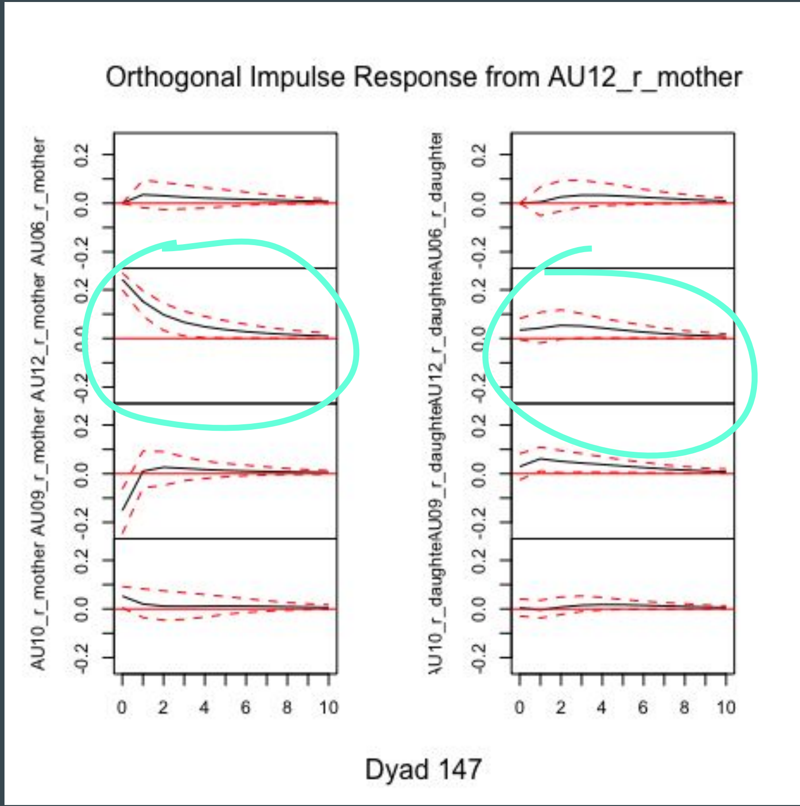



```
Estimation results for equation AU12_r_mother:
==============================================
AU12_r_mother = AU06_r_mother.l1 + AU06_r_daughter.l1 + AU12_r_mother.l1 + AU12_r_daughter.l1
 + AU09_r_mother.l1 + AU09_r_daughter.l1 + AU10_r_mother.l1 + AU10_r_daughter.l1 + const

                    Estimate Std. Error t value    Pr(>|t|)
AU06_r_mother.l1    0.006283   0.144405   0.044      0.9653
AU06_r_daughter.l1 -0.098606   0.082592  -1.194      0.2335
AU12_r_mother.l1    0.543117   0.104372   5.204 0.000000376
AU12_r_daughter.l1  0.206293   0.101884   2.025      0.0438
AU09_r_mother.l1   -0.055698   0.040112  -1.389      0.1661
AU09_r_daughter.l1  0.109692   0.057374   1.912      0.0569
AU10_r_mother.l1    0.018599   0.101550   0.183      0.8548
AU10_r_daughter.l1  0.054606   0.098801   0.553      0.5809
const              -0.018202   0.051950  -0.350      0.7263


Residual standard error: 0.4868 on 283 degrees of freedom
Multiple R-Squared: 0.4831,     Adjusted R-squared: 0.4685
F-statistic: 33.06 on 8 and 283 DF,  p-value: < 0.00000000000000022
```

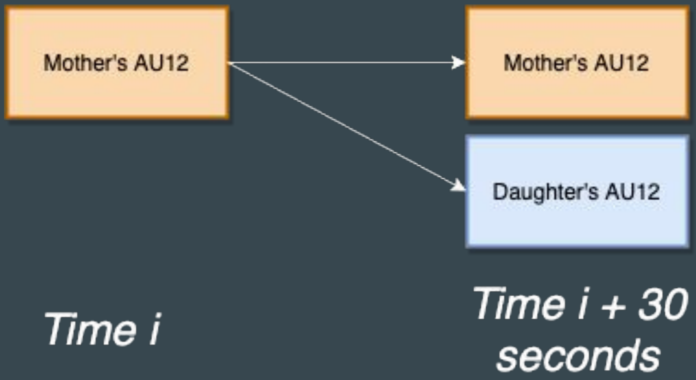
Dyad 147 (Control) for Mother’s AU12



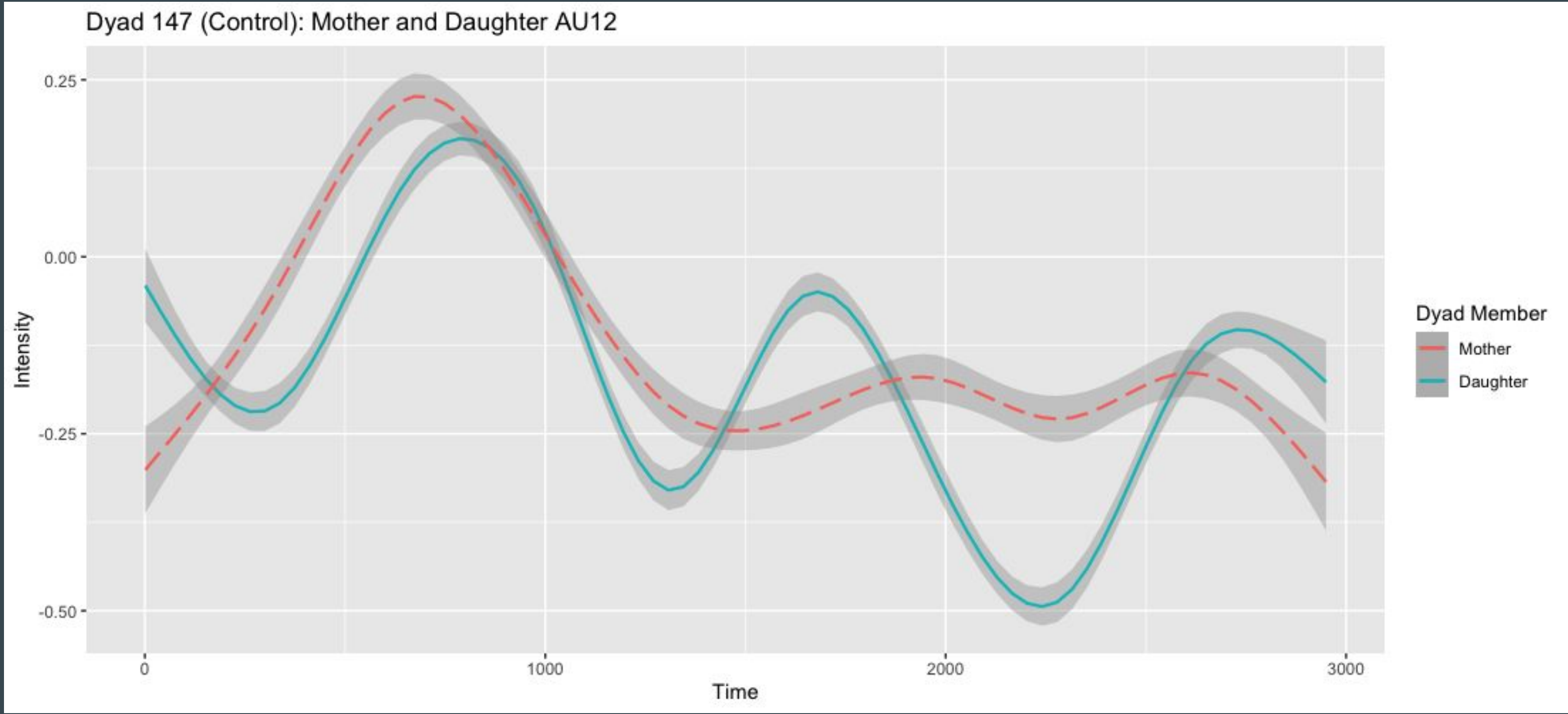

```
Estimation results for equation AU12_r_mother:
==============================================
AU12_r_mother = AU06_r_mother.l1 + AU06_r_daughter.l1 + AU12_r_mother.l1 + AU12_r_daughter.l1
 + AU09_r_mother.l1 + AU09_r_daughter.l1 + AU10_r_mother.l1 + AU10_r_daughter.l1 + const

                    Estimate Std. Error t value    Pr(>|t|)
AU06_r_mother.l1    0.006283   0.144405   0.044      0.9653
AU06_r_daughter.l1 -0.098606   0.082592  -1.194      0.2335
AU12_r_mother.l1    0.543117   0.104372   5.204 0.000000376
AU12_r_daughter.l1  0.206293   0.101884   2.025      0.0438
AU09_r_mother.l1   -0.055698   0.040112  -1.389      0.1661
AU09_r_daughter.l1  0.109692   0.057374   1.912      0.0569
AU10_r_mother.l1    0.018599   0.101550   0.183      0.8548
AU10_r_daughter.l1  0.054606   0.098801   0.553      0.5809
const              -0.018202   0.051950  -0.350      0.7263

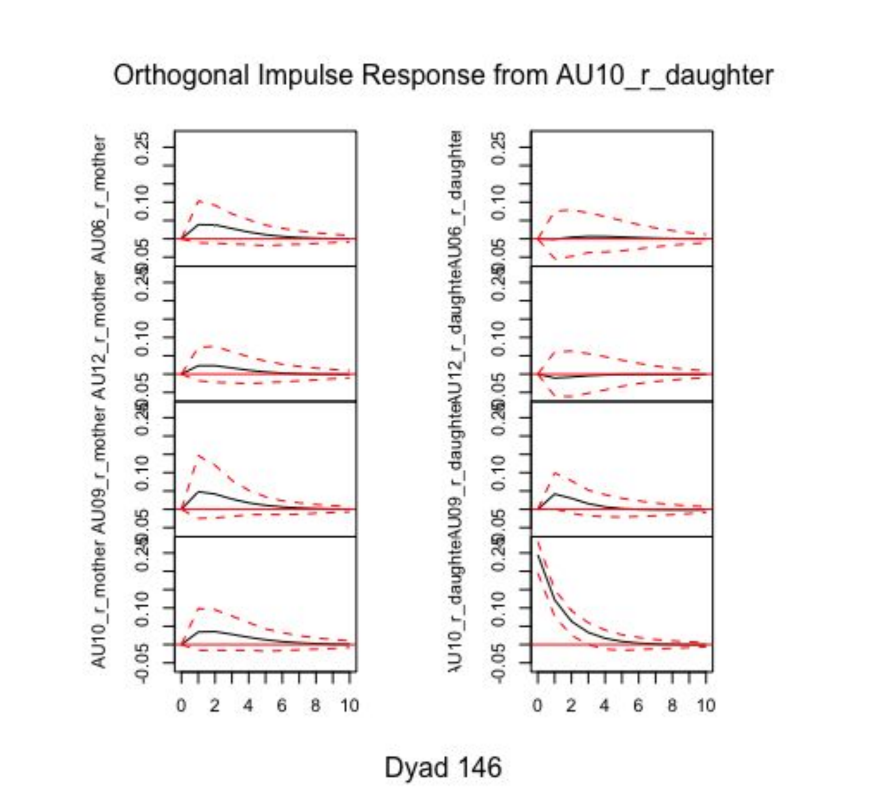

Residual standard error: 0.4868 on 283 degrees of freedom
Multiple R-Squared: 0.4831,     Adjusted R-squared: 0.4685
F-statistic: 33.06 on 8 and 283 DF,  p-value: < 0.0000000000000022
```



Dyad 147 (Control): Mother/Daughter AU12 plotted



Dyad 146 (Depressed) for Daughter AU10

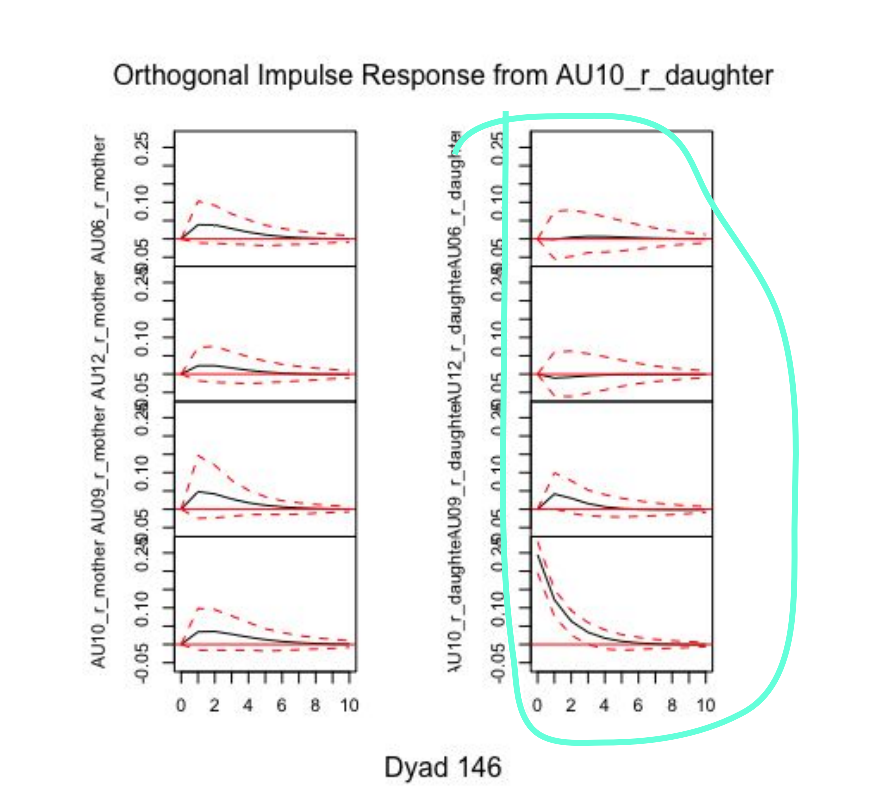



```
Estimation results for equation AU10_r_daughter:
================================================
AU10_r_daughter = AU06_r_mother.l1 + AU06_r_daughter.l1 + AU12_r_mother.l1 + AU12_r_daughter.l
1 + AU09_r_mother.l1 + AU09_r_daughter.l1 + AU10_r_mother.l1 + AU10_r_daughter.l1 + const

                    Estimate Std. Error t value          Pr(>|t|)
AU06_r_mother.l1     0.12950    0.09606   1.348          0.178723
AU06_r_daughter.l1  -0.13540    0.05452  -2.483          0.013596
AU12_r_mother.l1    -0.03545    0.06794  -0.522          0.602225
AU12_r_daughter.l1   0.23271    0.06747   3.449          0.000648
AU09_r_mother.l1    -0.01475    0.02652  -0.556          0.578618
AU09_r_daughter.l1   0.06592    0.03824   1.724          0.085821
AU10_r_mother.l1    -0.04111    0.06482  -0.634          0.526504
AU10_r_daughter.l1   0.49821    0.06447   7.727 0.000000000000194
const               -0.19674    0.03395  -5.794 0.000000018304798


Residual standard error: 0.3126 on 282 degrees of freedom
Multiple R-Squared: 0.5151,     Adjusted R-squared: 0.5014
F-statistic: 37.45 on 8 and 282 DF,  p-value: < 0.00000000000000022
```

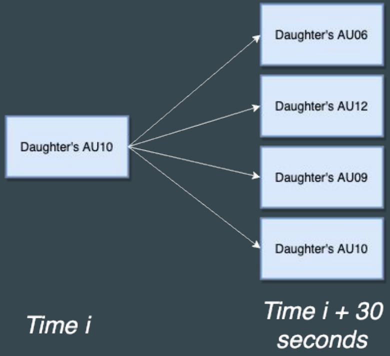
Dyad 146 (Depressed) for Daughter AU10



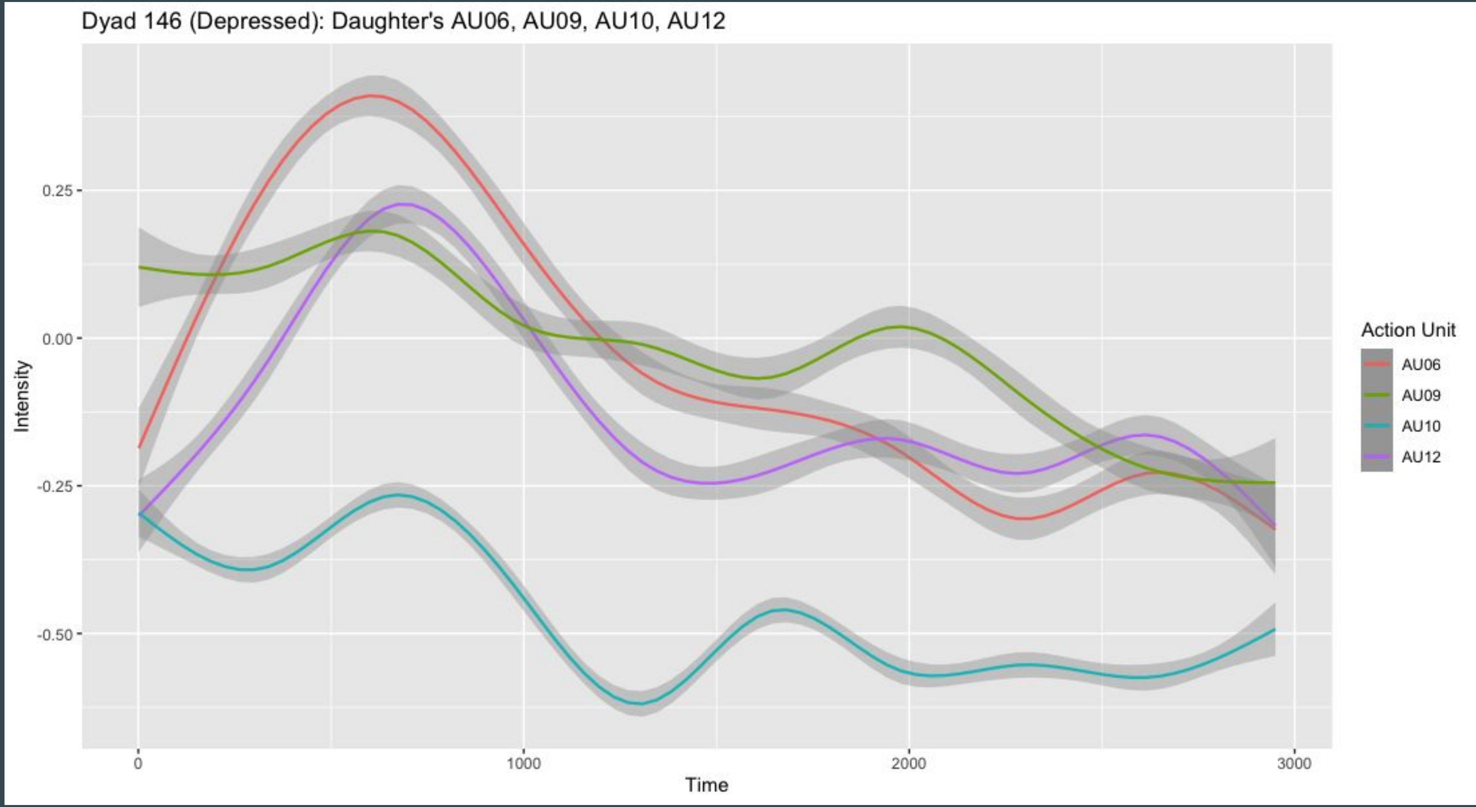

```
Estimation results for equation AU10_r_daughter:
================================================
AU10_r_daughter = AU06_r_mother.l1 + AU06_r_daughter.l1 + AU12_r_mother.l1 + AU12_r_daughter.l
1 + AU09_r_mother.l1 + AU09_r_daughter.l1 + AU10_r_mother.l1 + AU10_r_daughter.l1 + const

                   Estimate Std. Error t value          Pr(>|t|)
AU06_r_mother.l1    0.12950    0.09606   1.348          0.178723
AU06_r_daughter.l1 -0.13540    0.05452  -2.483          0.013596
AU12_r_mother.l1   -0.03545    0.06794  -0.522          0.602225
AU12_r_daughter.l1  0.23271    0.06747   3.449          0.000648
AU09_r_mother.l1   -0.01475    0.02652  -0.556          0.578618
AU09_r_daughter.l1  0.06592    0.03824   1.724          0.085821
AU10_r_mother.l1   -0.04111    0.06482  -0.634          0.526504
AU10_r_daughter.l1  0.49821    0.06447   7.727 0.000000000000194
const              -0.19674    0.03395  -5.794 0.000000018304798

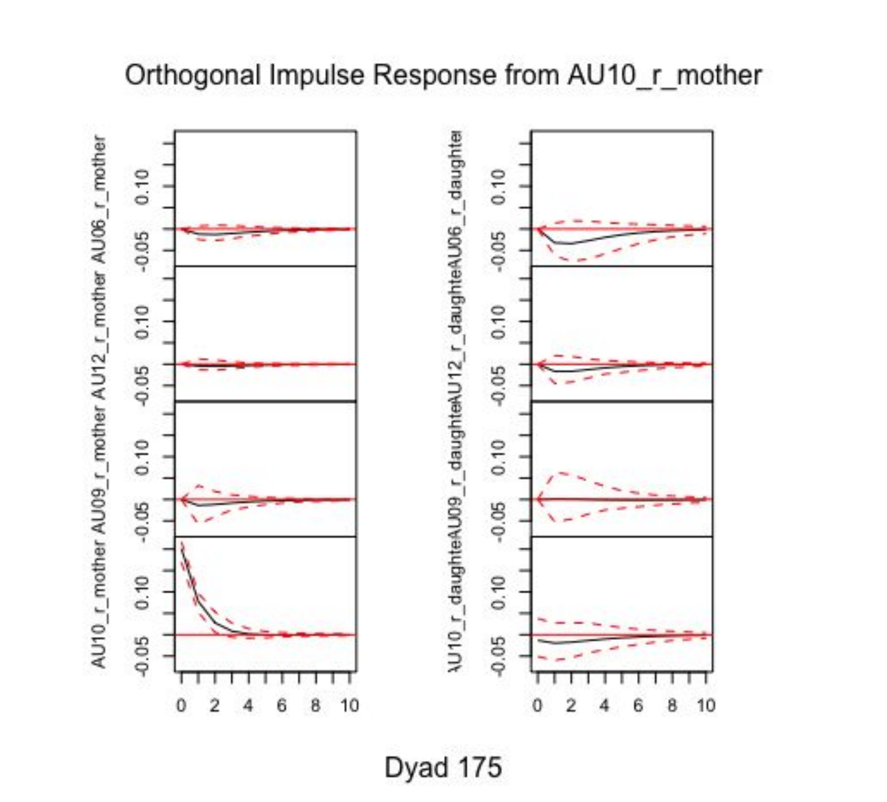

Residual standard error: 0.3126 on 282 degrees of freedom
Multiple R-Squared: 0.5151,     Adjusted R-squared: 0.5014
F-statistic: 37.45 on 8 and 282 DF,  p-value: < 0.0000000000000022
```



Dyad 146 (Depressed) for all Daughter AUs



Dyad 175 (Daughter) for Mother AU10

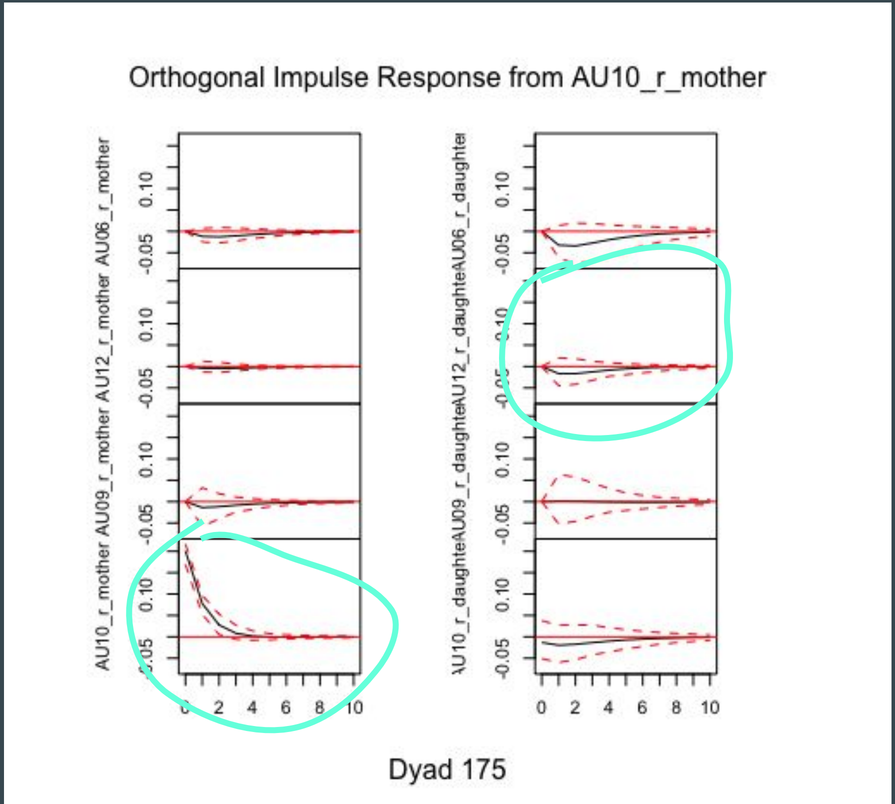



```
Estimation results for equation AU10_r_mother:
==================================================
AU10_r_mother = AU06_r_mother.l1 + AU06_r_daughter.l1 + AU12_r_mother.l1 + AU12_r_daughter.l1
 + AU09_r_mother.l1 + AU09_r_daughter.l1 + AU10_r_mother.l1 + AU10_r_daughter.l1 + const

                   Estimate Std. Error t value     Pr(>|t|)
AU06_r_mother.l1   -0.02844    0.08278  -0.344     0.731371
AU06_r_daughter.l1 -0.06417    0.03722  -1.724     0.085513
AU12_r_mother.l1    0.25029    0.16500   1.517     0.130132
AU12_r_daughter.l1  0.21637    0.06105   3.544     0.000443
AU09_r_mother.l1    0.02856    0.03241   0.881     0.378913
AU09_r_daughter.l1  0.02207    0.02305   0.957     0.339000
AU10_r_mother.l1    0.38644    0.06802   5.681 0.0000000266
AU10_r_daughter.l1 -0.01438    0.03083  -0.466     0.641173
const              -0.03085    0.06716  -0.459     0.646220


Residual standard error: 0.3086 on 381 degrees of freedom
Multiple R-Squared: 0.2988,     Adjusted R-squared: 0.2841
F-statistic:  20.3 on 8 and 381 DF,  p-value: < 0.00000000000000022
```

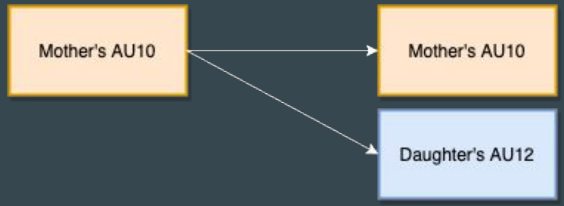
Dyad 175 (Daughter) for Mother AU10




```
Estimation results for equation AU10_r_mother:
==================================================
AU10_r_mother = AU06_r_mother.l1 + AU06_r_daughter.l1 + AU12_r_mother.l1 + AU12_r_daughter.l1
 + AU09_r_mother.l1 + AU09_r_daughter.l1 + AU10_r_mother.l1 + AU10_r_daughter.l1 + const

                   Estimate Std. Error t value     Pr(>|t|)
AU06_r_mother.l1   -0.02844    0.08278  -0.344     0.731371
AU06_r_daughter.l1 -0.06417    0.03722  -1.724     0.085513
AU12_r_mother.l1    0.25029    0.16500   1.517     0.130132
AU12_r_daughter.l1  0.21637    0.06105   3.544     0.000443
AU09_r_mother.l1    0.02856    0.03241   0.881     0.378913
AU09_r_daughter.l1  0.02207    0.02305   0.957     0.339000
AU10_r_mother.l1    0.38644    0.06802   5.681 0.0000000266
AU10_r_daughter.l1 -0.01438    0.03083  -0.466     0.641173
const              -0.03085    0.06716  -0.459     0.646220

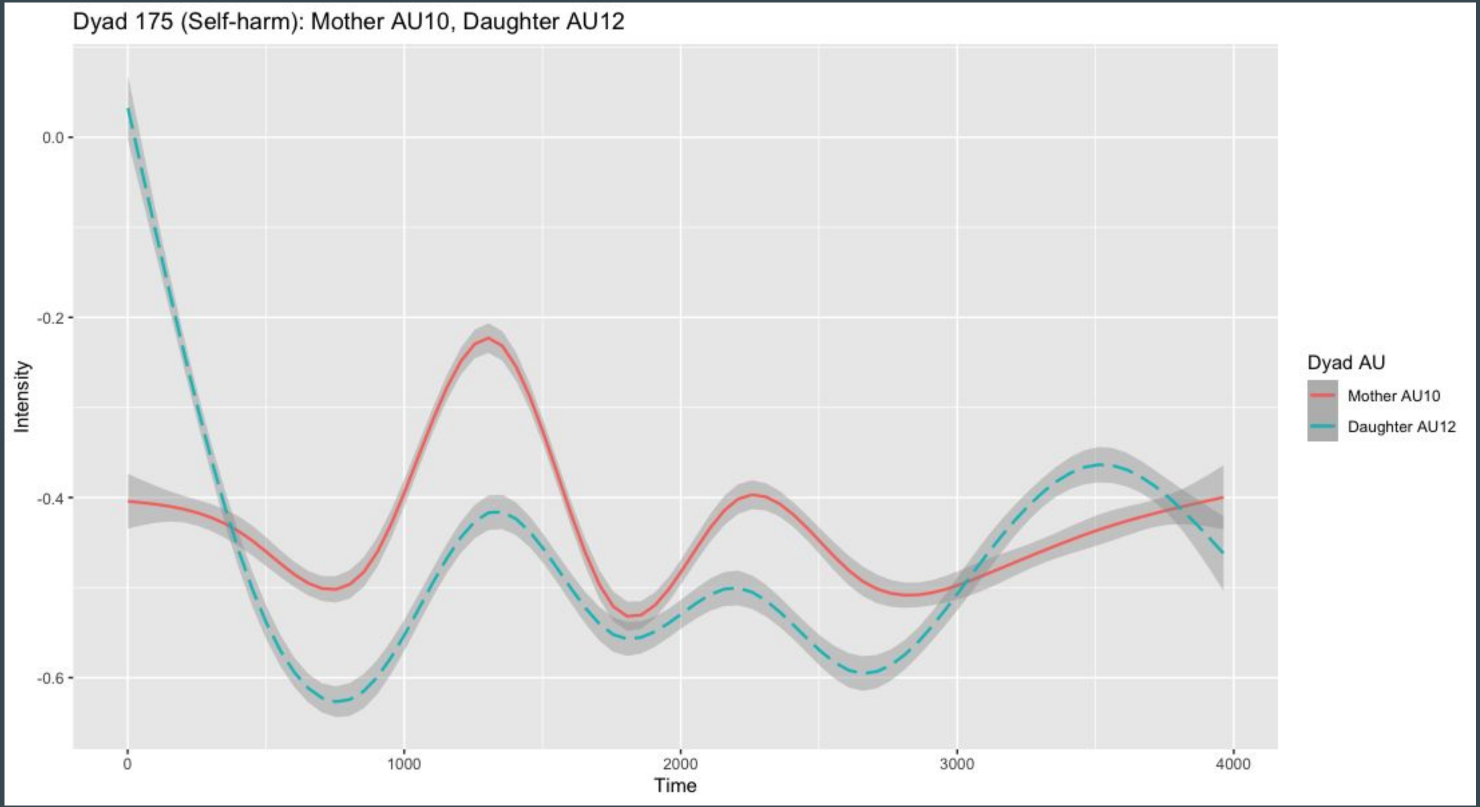

Residual standard error: 0.3086 on 381 degrees of freedom
Multiple R-Squared: 0.2988,     Adjusted R-squared: 0.2841
F-statistic:  20.3 on 8 and 381 DF,  p-value: < 0.00000000000000022
```



Time i

Time i + 30 seconds

Dyad 175 (Daughter): Mother AU10, Daughter AU12



Limitations & Future Directions

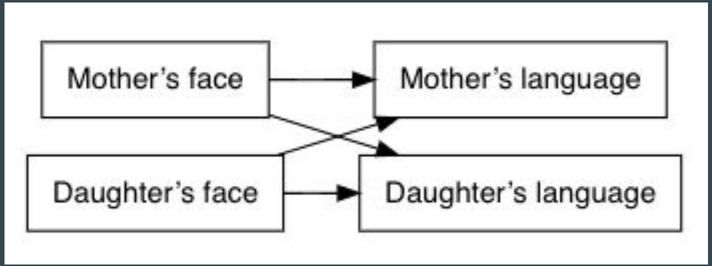
- What went wrong?
- What can be done differently in the future?

Limitations

- Exclusion criteria
  - To replicate Haines et al. (2019), we used all videos from 47 dyads
  - However, other studies that use OpenFace often apply an exclusion criteria to videos with low confidence or low success rates (e.g., Drimilia et al, 2020)
  - If we used the criteria from Drimilia et al. (2020), we would've gone from 48 dyads to 33 dyads
    - Remove dyads with <.75 average confidence and where 10%+ of observations had success=0
  - Low confidence/success is often due to people moving out of the frame, covering their face with their hands or an object, or turning their heads so less than half their face can be seen, so any videos that impacted OpenFace would have also impacted iMotions

Future Directions

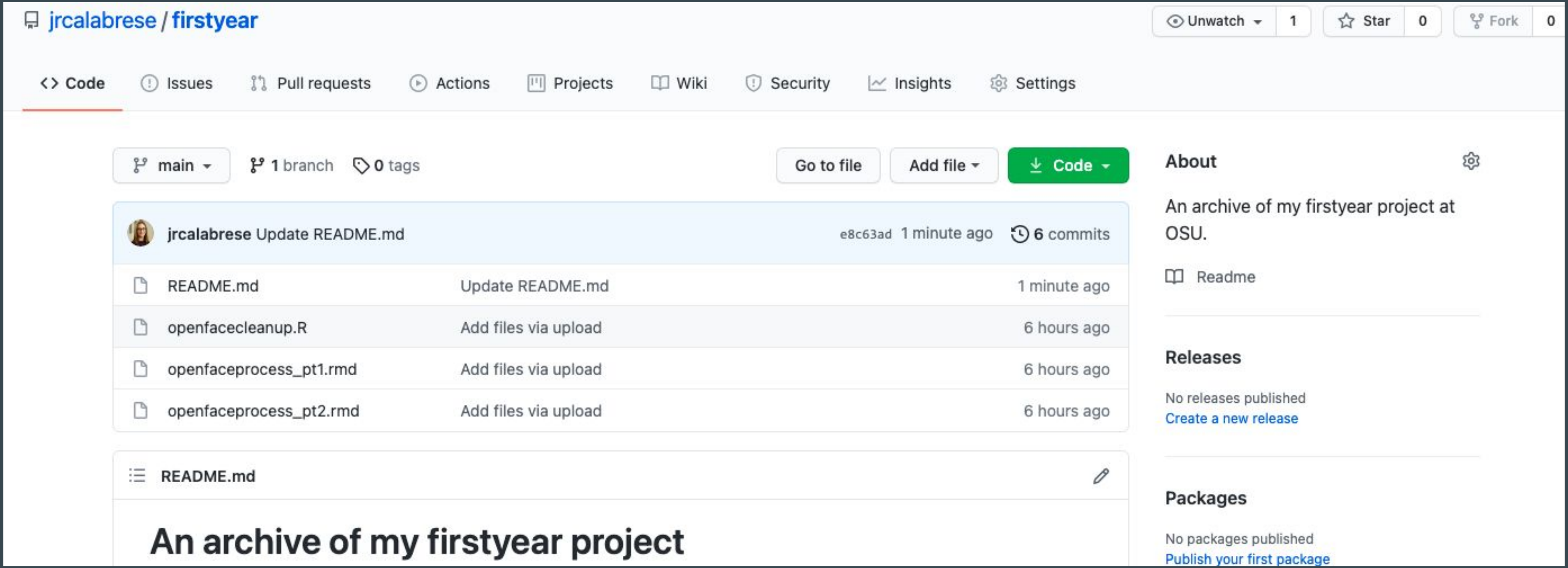
- Make composite emotion values (“Happiness”, “Fear”) from OpenFace’s raw Action Units and try again at replicating Haines et al. (2019)
    - Would require machine learning but definitely doable
    - Random Forest (RF)
- The data used in Haines et al. (2019) was originally collected in 1999
    - It’d be nice to get more recent data
- More vector autoregression with stronger computer
    - Might try again when we return to campus
- Actor-Partner Interdependence Model
    - Would require audio transcription
    - Client-therapist interactions



Conclusions

- Automated facial expression coding (AFEC) is a worthwhile alternative to manual coding of affective behavior
- Although it may not be easier to use compared to commercial, "out-of-the-box" AFEC software like iMotions, OpenFace provides a free tool for facial expression detection

Code available on Github under jrcalabrese/firstyear



Thank you!

Acknowledgements

- Nate Haines for providing the R code from Haines et al. (2019) and helping me with the lme4 models and vector autoregression
- Shane Ruland for helping out with all the technical issues I encountered

→ *If you are interested in automated facial expression coding/OpenFace or any of the R code/data wrangling techniques, please contact me at: calabrese.75@osu.edu

References

Baltrusaitis, T., Zadeh, A., Lim, Y. C., & Morency, L.-P. (2018). OpenFace 2.0: Facial Behavior Analysis Toolkit. 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), 59–66. https://doi.org/10.1109/FG.2018.00019

Crowell, S. E., Baucom, B. R., McCauley, E., Potapova, N. V., Fitelson, M., Barth, H., Smith, C. J., & Beauchaine, T. P. (2013). Mechanisms of Contextual Risk for Adolescent Self-Injury: Invalidation and Conflict Escalation in Mother–Child Interactions. Journal of Clinical Child & Adolescent Psychology, 42(4), 467–480. https://doi.org/10.1080/15374416.2013.785360

Darwin, C. (1899). The Expression of the Emotions in Man and Animals. D. Appleton and Company.

Drimalla, H., Scheffer, T., Landwehr, N., Baskow, I., Roepke, S., Behnia, B., & Dziobek, I. (2020). Towards the automatic detection of social biomarkers in autism spectrum disorder: Introducing the simulated interaction task (SIT). Npj Digital Medicine, 3(1), 25. https://doi.org/10.1038/s41746-020-0227-5

Eason, E. G., Carver, N. S., Kelty-Stephen, D. G., & Fausto-Sterling, A. (2020). Using Vector Autoregression Modeling to Reveal Bidirectional Relationships in Gender/Sex-Related Interactions in Mother–Infant Dyads. Frontiers in Psychology, 11, 1507. https://doi.org/10.3389/fpsyg.2020.01507

Ekman, P., & Friesen, W. V. (1975). Unmasking the face: A guide to recognizing emotions from facial clues. Prentice-Hall.

Ekman, P., & Rosenberg, E. L. (Eds.). (2005). What the face reveals: Basic and applied studies of spontaneous expression using the facial action coding system (FACS) (2nd ed). Oxford University Press.

Haines, N., Bell, Z., Crowell, S., Hahn, H., Kamara, D., McDonough-Caplan, H., Shader, T., & Beauchaine, T. P. (2019). Using automated computer vision and machine learning to code facial expressions of affect and arousal: Implications for emotion dysregulation research. Development and Psychopathology, 31(3), 871–886. https://doi.org/10.1017/S0954579419000312

Kring, A. M., & Sloan, D. M. (2007). The Facial Expression Coding System (FACES): Development, validation, and utility. Psychological Assessment, 19(2), 210–224. https://doi.org/10.1037/1040-3590.19.2.210

Schmidt, K. L., & Cohn, J. F. (2001). Human facial expressions as adaptations: Evolutionary questions in facial expression research. American Journal of Physical Anthropology, 116(S33), 3–24. https://doi.org/10.1002/ajpa.20001

Stock, J. H., & Watson, M. W. (2001). Vector Autoregressions. Journal of Economic Perspectives, 15(4), 101–115.