STEPS:

01) **download elan and wav files from google drive:**

prerequisite: google account, enable drive api, download credentials and install corresponding libraries

input: path of the download directory, path to the elan directories, path to the wav directories, speaker files.

program name: GDownload.py The Google Drive downloading script. a) At the first run, the elan and wav directories will be empty. The script will read in the trans_id in speaker files into a list. It will search and download all the related files from Google Drive containing the trans_id into the download directory. The files with multiple speaker names in it will not match the trans_id and need to be downloaded manually and put into the corresponding elan or wav directory. b) After the first run, the elan and wav directories will not be empty. The script use the path to the elan or wav directories to put all the file names within the directory into a list. It will search Google Drive with the file names, and compare the last modified time of the local file and the drive file. If the file is updated,it will be downloaded using ID into the download directory.

program name: select_move_file.py The script will match the download directory files with the speaker file trans_id and move the elan and wav files to the corresponding directories.

output: Up to date Elan files and wav audio files in elan directory(format:.eaf, .wav). There are five directories for Elan files: panel_elan/1982, panel_elan/2017, panel_wav, trend_elan, trend_wav, twin_elan, twin_wav and Manual.

02) **processing elan files: manually and automatically**

Manual: some elan files have multiple speakers which requires some manual processing after transforming them into textgrids. These elan files are in the Manual directory.
manual file example: S124_S125_S126-17-I-1-Laura_Sonja_Michaela
These files need to be manually processed using the Elan app.
Open it in ELan -> Export As -> Praat TextGrid -> Select the SWG tier -> Change to file name to only contain one speaker -> Save it to the correct TextGrid directory.

program name: Eaf2TextGrid.py

prerequisite: install pympi

input: path of elan directory, path of TextGrid directory, name of tier that will be kept in TextGrid
The program will use the path of the elan directory, read in one Elan file, then use the pympi library to select tiers and convert the file into TextGrid and save it to the TextGrid directory.

output: TextGrid files in TextGrid directory(format:.TextGrid). There are four TextGrid directories: panel_tg/1982, panel_tg/2017, trend_tg, twin_tg.
Note:some elan have additional layers, which will cause the program to fail. delete tier “3-MORPHEME”. Example : S008-82-I-1-Rupert, S008-82-I-2-Rupert.

03) **checking tiers and skip-tags:**

check the tier name for TextGrid:
program name: check_tiers.py
input: path of TextGrid directory.
Read in each TextGrid and iterate over its tier names. If there is a tier named “SWG”, the file is good. If there are no “SWG” tier within the TextGrid. Print out the file name and all the tier names of the file.
To get rid of the additional name in the tier name, open the TextGrid in any text editor. Then rerun the check program to make sure no further change is needed.
check the skip-tags like [BEGIN-READING]:
program name: tg_inspect.py
input: path of TextGrid directory.
Read in each TextGrid and check for tags which are letters surrounded by square brackets, like [Any-Content-Within]. Then check if the token is a valid tag. If not, print out the TextGrid name and the invalid tags.
Open the TextGrid in any text editor and correct those tags.
Then rerun the check program to make sure no further change is needed.

04) **create extracts:**
prerequisite: stanford-corenlp(for pos tagging)
input: date, extract type, path of lex table, path of output directory, path of TextGrid
program name: words_extract.py, clauses_extract.py, clauses_rel_extract.py
The output extracts name will be determined by the date, extract type.

- **process the lex table using lex table method**:
- if the lex table is new, the program will check it for invalid var_code in the word_vars column and add four new columns: 'word_stem_freq', 'word_lemma_freq', 'word_standard_freq', 'word_variant_freq'. The output of the lex table method is a new .csv file which will be used when generating new extracts. The invalid var_code will be omitted in the new lex table. They will be put into a separate txt file for correction.
- **processing and producing the extract**:
- Read in all the TextGrids, keep the file name and all the text annotations within the SWG tier.
- For words extract, all the text will be tokenized and filter out all the non-words(filler words like -mhm-, punctuations like ---, half words like word---.etc). Then all the words will be checked by the ddm tagging method. If the word is in the word_variant column of the lex table, it will be attached with its ddm tag(s) and POS. If the word is not found in the lex table, it will be POS tagged using the stanford-corenlp POS tagger.
- After ddm tagging, all words from annotations will be checked by the skip method in order to skip over the part of fairy-tale reading, word reading and word game. **There are two ways to skip**:
- 1. skip_by_tag: if a Begin tag is detected, all the words between the Begin tag and its corresponding End tag will be skipped.
- 2. skip_word_list: if there is no Begin tag, the method will find possible words to skip by match word sequences within annotations with the first/last ten words of word lists/fairy-tales.

- After skipping, output the extract into the output directory.

For word extract, each row is the information about one word in the annotation.
For clauses extract, there is no tokenization or filter. All the segments will be ddm tagged and pos tagged using the same methods and tools. The content within skip-tags will be skipped. The output extract has one segment per row. * is used as a placeholder for not tagged words and uninformative POS tags.
For clauses_rel extract, only the clauses which contain tag [REL] or [ANT] are included.

output: extracts without social information(format:.csv)

05) **split the TextGrid and wav files for Aligner:**
prerequisite: pydub
input: path to directory which contains wav and TextGrid files
program name: split.py
The script will find the matching TextGrid and wav files. For every TextGrid Interval, it will write the Interval text to a txt file in the format: "TextGridName_Number.txt". Meanwhile, it will cut out the corresponding wav audio: "TextGridName_Number.wav", using the Interval xmin and xmax time stamps.
If the text is empty, the txt and wav will be saved in an empty folder. Else, they will be saved in none_empty_SPEAKERTYPE folder.
output: txt and wav files(format: .txt, .wav) in directories: none_empty_panel, none_empty_trend, none_empty_twin.
Note: for the next step, it's better to load all the files on the Suebi server. Use unix command scp. example: scp /local/path/to/the/target/directory yourusername@suebi.sfs.uni-tuebingen.de:~/SWG/target/directory/on/server

06) **Run Aligner**:
prerequisite: R, Aligner, htk (please use the Suebi Server samantha's account. The Aligner and the scripts are on there.)
input: path to the none_empty directories(working directory), path to the output directories(NEWFOLDER), path to the intermediate aligner files(ALIGNFOLDER), TRANSFOLDER, path to the output folder(GRIDFOLDER)
program name: SWG_run_aligner.R
The R script set working directory to the none_empty folder. Read all the file names into a list. Then for each filename it will down sample the wav file. It will read in the txt file. It will remove punctuations and replace all the special characters. Then the Aligner will align the cleaned text with the audio on two levels: segment and word. Segment level align phones within the words. This information is stored in the intermediate aligner files.
Then the script will read in the intermediate files and write the phones, words and time stamps into a TextGrid.
output: TextGrids and wav audio files(format: .TextGrid, .wav) in the done_panel, done_trend, done_twin directories.

07) **create extract**:
input: date, extract type, path of lex table, path of output directory, path to the TextGrid directories, path to the wav directories
program name: phone_extract.py, formant_extract.py, Praat_extract_formants.praat
The phone_extract.py script will read in the Aligner output TextGrids and the original txt. There will be one phone/segment each row with it's start time and end time and the word that contains the phone/segment and the word's start time and end time. The word is tagged like other extracts. Then, add four columns: previous_seg, following_seg, previous_word, following_word.
Run the praat script to process all the wav in the wav directories and generate .Formant files to the output directory.
The formant_extract.py script will first read in the phone extract and select all the lines which contain AIS in var_code. These lines are written into formant_raw csv file for further processing. All the Formant files will be read into a dictionary with the filename as key. For each line in the formant_raw file, get the relevant formant's time, F1, F2 using the file name and the seg start and end time.
Create the seg and word duration column by subtracting start_time from end_time column.

- **Normalized time columns**: Normalized time always starts from 0 and ends at 1. 1) getting the number of lines of each seg or word interval(same start and end time). 2) divide 1 by the number of lines to get the length of the step. 3) for each line within the interval the normalized time is step_length * line_num.
- **zeroed time**: The zeroed end time is time(from Formant) minus the start time. It's the difference between the current time and the start time of the file.
- Then write the formant extract to a .csv file.

output: phone extracts and formant extracts without social information.(format:.csv)

08) **add social informations:**
input: path to extract, path to speaker file, extract type
program name: add_social_info_to_csv.py
Read in the extract and speaker file into dataframes. Merge the two files using the trans_id columns.
output: extract with social information(format:.csv).