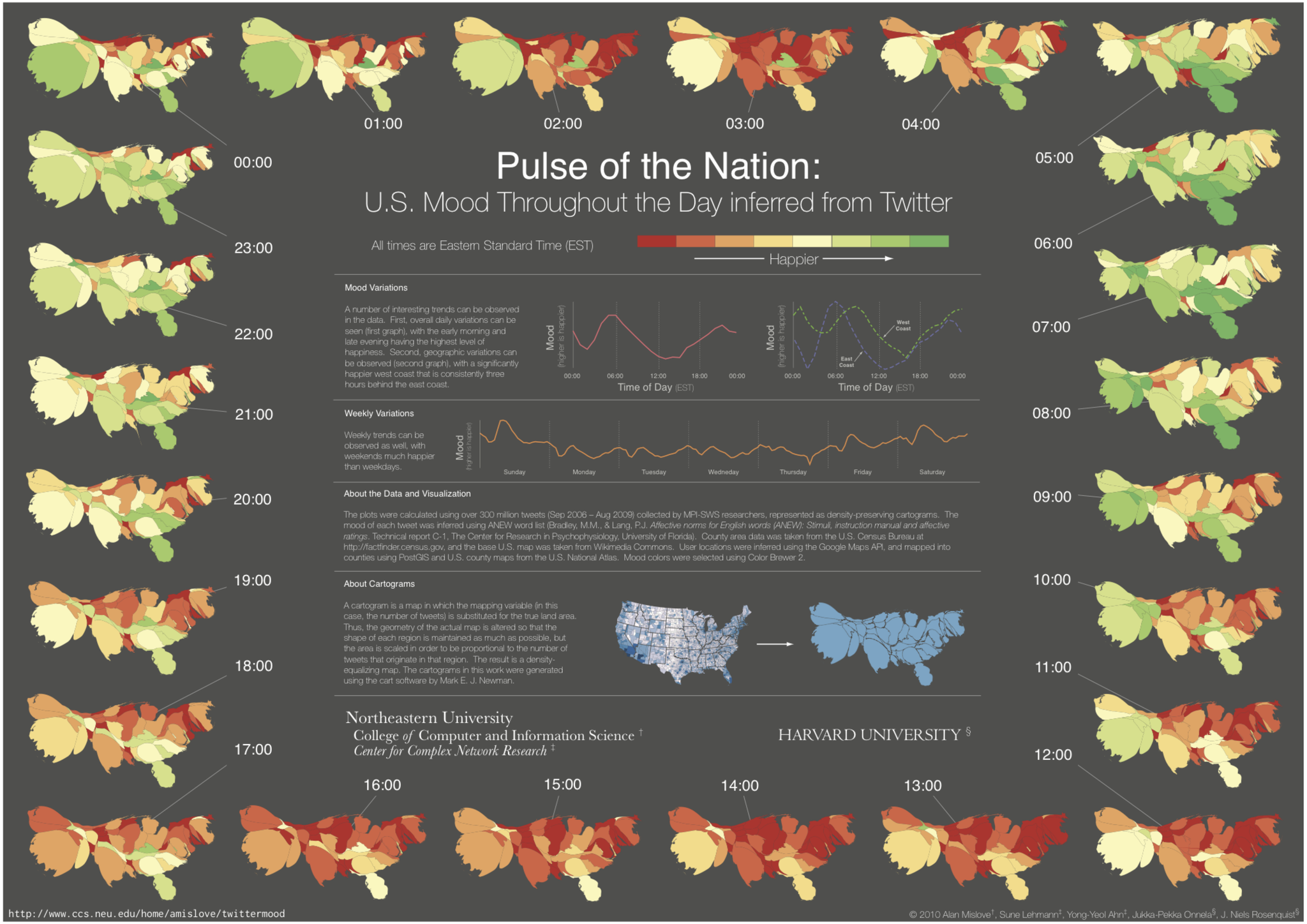
Reproducible Pipeline for Large-scale Data Analysis

Lecture for ISEA Training Program

Wei Ai <aiwei@umd.edu>
University of Maryland

So, you did some data science…

And you have a beautiful trained model / visualization / dashboard to show for it



But Behind the Theme?

Reproducibility and Reliability Matters?

- Other people will thank you.
  - Collaborate more easily with you on this analysis
  - Learn from your analysis about the process and the domain
  - Feel confident in the conclusions at which the analysis arrives

- You will thank you.
  - Which notebook should I run first?
  - Where to go if I need to add one more column?
  - What if the questions are asked 4 years later?

```
cohort0.ipynb
cohort_new.ipynb
data_preprocessing.ipynb
dataset_preperation,ipynb
…
```

-

Cookiecutter Data Science

- A open-source project structure developed by DrivenData.
  - A template directory structure
  - Some opinions of good data science practice
  - And some suggested workflows.

```
├── LICENSE            <- Open-source license if one is chosen
├── Makefile           <- Makefile with convenience commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default mkdocs project; see www.mkdocs.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── pyproject.toml     <- Project configuration file with package metadata for
│                         {{ cookiecutter.module_name }} and configuration for tools like black
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.cfg          <- Configuration file for flake8
│
└── {{ cookiecutter.module_name }}   <- Source code for use in this project.
    │
    ├── __init__.py             <- Makes {{ cookiecutter.module_name }} a Python module
    │
    ├── config.py               <- Store useful variables and configuration
    │
    ├── dataset.py              <- Scripts to download or generate data
    │
    ├── features.py             <- Code to create features for modeling
    │
    ├── modeling
    │   ├── __init__.py
    │   ├── predict.py          <- Code to run model inference with trained models
    │   └── train.py            <- Code to train models
    │
    └── plots.py                <- Code to create visualizations
```

Managing a Data Science Project

- Manage data
  - Data Pipeline as a DAG
- Manage code
  - Notebook vs script
- Manage project
  - Version control
- Manage collaboration (internal and external)
  - Design doc
  - Issue & Kanban

Rethink about Plumbing —
Data Pipeline should be a Directed Acyclic Graph (DAG)

- Each step of your analysis is a node in a *directed graph with no loops*.
  - You can run through the graph forwards to recreate any analysis output, or you can trace backwards from an output to examine the combination of code and data that created it.
- Raw data must be treated as immutable.
  - it's okay to read and copy raw data to manipulate it into new outputs, but never okay to change it in place.
- The goal: anyone should be able to re-run your analysis using only your code and raw data and produce the same final products.

The data directory structure

```
├── LICENSE              <- Open-source license if one is chosen
├── Makefile             <- Makefile with convenience commands like `make data` or `make train`
├── README.md            <- The top-level README for developers using this project.
├── data
│   ├── external         <- Data from third party sources.
│   ├── interim          <- Intermediate data that has been transformed.
│   ├── processed        <- The final, canonical data sets for modeling.
│   └── raw              <- The original, immutable data dump.
│
├── docs                 <- A default mkdocs project; see www.mkdocs.org for details
│
├── models               <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks            <- Jupyter notebooks. Naming convention is a number (for ordering),
│                           the creator's initials, and a short `-` delimited description, e.g.
│                           `1.0-jqp-initial-data-exploration`.
```

- Most data should not be kept in version control
  - Use .gitignore to specify files not tracked by git.
  - Small and stable data files are fine (such as zipcode-city directory).

Discussion: Data Management

- Where do you store your data?
- How do you manage processed data or interim data?
- How do you share processed data with your teammates?

```
├── LICENSE            <- Open-source license if one is chosen
├── Makefile           <- Makefile with convenience commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default mkdocs project; see www.mkdocs.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
```

Notebooks are Good for Demonstration and Experimentation

- Small-scale data wrangling (manipulating dataset into desired structure).
- Running sanity check (checking summary statistics),
- Developing models with unit tests.
- Visualizing results and generating reports.

But Notebooks are bad for repetition (reproduction).

- The flexibility of running cells in arbitrary order creates additional dependency to manage
- Avoid writing in your documentation:
  - "In order to reproduce this figure, run cell A first, skip cell B, then run cell C, and then rerun cell A …"
- Notebooks are hard for version control

- One more reason to hate them: They don't work well with AI coding assistants.

If You Open a Jupyter Notebook with a Text Editor

opensmile.ipynb > ...

Open in Notebook Editor

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 11,
   "metadata": {},
   "outputs": [],
   "source": [
    "# To run this notebook, please install librosa, opensmile, in addition to standard packages such
    "\n",
    "import librosa\n",
    "import opensmile\n",
    "from IPython.display import Audio, display\n",
    "\n",
    "import matplotlib.pyplot as plt\n",
    "import librosa.display"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Filename identifiers of the RAVDESS dataset\n",
    "# See: https://zenodo.org/records/1188976\n",
    "\n",
    "# Modality (01 = full-AV, 02 = video-only, 03 = audio-only).\n",
    "# Vocal channel (01 = speech, 02 = song).\n",
    "# Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust,
    "# Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neu
    "# Statement (01 = \"Kids are talking by the door\", 02 = \"Dogs are sitting by the door\").\n",
    "# Repetition (01 = 1st repetition, 02 = 2nd repetition).\n",
    "# Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).\n",
    "\n",
    "male_calm_normal_kids1 = \"Audio_Speech_Actors_01-24/Actor_01/03-01-02-01-01-01-01.wav\"\n",
    "male_happy_strong_kids1 = \"Audio_Speech_Actors_01-24/Actor_01/03-01-03-02-01-01-01.wav\"\n",
    "female_calm_normal_kids1 = \"Audio_Speech_Actors_01-24/Actor_02/03-01-02-01-01-01-02.wav\"\n",
    "female_happy_strong_kids1 = \"Audio_Speech_Actors_01-24/Actor_02/03-01-03-02-01-01-02.wav\""
   ]
```

```
   6d27t9XNAOCD2HsOgNciCAgo9PaRI0tqHdjZ6ACAGQhNALzGkSNHHN/PnDlTRowYITt27HAcK1eunL4AgBkYngPgNa
   +fOdXmvr1q16h3f1nOoxDzzwgJw4ccKCswbgKQhNAHzeRx99JJUrV5a1a9fqAPXYY4/JnXfeKW3btpXExETp2rWrDk
   Xr15eEhAQJCwvTIWrgwIH6mBrmO3nypPz666/6/hMmTNCB6ZVXXpG4uDj9/bRp02Tp0qWyc+dOq08HgEWoaQLg85o2
   +yz8nJycvTXs2fPSq9evWTs2LF5nqtatWqmtxeAZyI0AcBlrrvuOpk1a5bExsZKcDAfkwAuoqYJAC4zaNAgOXXqlNx
   FWAjaVzAQAArog/mQAAAAwgNAEAABhAaAIAADCA0AQAAGAAoQkAAMAAQhMAAIABhCYAAAADCE0AAAAGEJoAAAAMIDQ
    "text/plain": [
     "<Figure size 600x300 with 1 Axes>"
    ]
   },
   "metadata": {},
   "output_type": "display_data"
  },
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Extracting LLD features using OpenSMILE feature set: eGeMAPSv02\n",
    "Extracted 25 LLD features with 366 frames\n"
   ]
  },
  {
   "data": {
    "image/png":
    "iVBORw0KGgoAAAANSUhEUgAAAk0AAAGGCAYAAABmPbWyAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb
    R4nO2dB3hT5dvGH7pp6QAKHXSw9957CyKKDAVBGYoLcaCf/sWJiAoOxAGKooADRQEBBWQv2XuVXUoptKWM7j3yXc+b
    67rNMnJyTnvezLO3WdWUSgUCgIAAAAAAOXiVv7TAAAAAACAgWgCAAAAADACiCYAAAAAAAAOAaAIAAAAAMAKIjgAAAAA
    AHAQIJoAsCNLly4VF2VF2V9y/Tp0+09PFmTmJgoRM+JEyesfiwWmobex40bN1r9+AAA2+Bho+MAAMrh/
    fffp3r16mmta9mypd3G4yyiaebMmcJ61rZtW6sfz9vbm3744YcY9hgaoBB4
    +u1CoWCPvjgA4qIiCBfX1/q168fxcTElDnG3bt36dVXX6VWrVpRtWrVKCAgQAi8kydPam23c+dOcew///1xTWHfq1Kl
    kodOnSgqlWrUo0aNeiRRx6hhIQErW369u0rrHBnz54V4+bx8/E++eQTrfF06tRJ3OdjSa4yPkfMf//9Rw8//DBFRUU
    +FaTgwcP0r333kuBgYHiXPbp00d8hvSN//Lly+IzFxQUJLbnc5uTk1NmnDwfHh/vj+fau3dv2rx5s3hu4sSJFBwcTI
    +++K0TTfffdJ5Zjx46JC1ZBQYHWvq5cuUJr1qwRAoPdhzdv3qTvvvtOXFhZAISHh2ttP3v2bHFB53gsvsB+/fXX5On
    +5cqV9Prrrwtxx6KuWbNmwuXJ+3/66aepV69e4nXdu3cXtytWrBAX+ylTplDNmjXp0KFD+Xljy0tZs2VJRU
    cXgoyFjyY8Rn7f+H3mzwy7I1kIf/zxx+ptWDTz+8zj5PfAy8tLCDM+Fn/Gxo8fTz///DNt2rTY8
    55ZcV7u7uirS0NPE4JSVF4eXlpRg6dKjWdm+++aZ4veY+8/LyFMXFxVrHjYuLU3h7eyvef/999
    +fn5itDQUMWoUaPU6w4fPiy24/OiS05OTpl1s2fPFmONj49Xr+PzbszPJJ8zfe8hj1VzP3w+NDF23nxOa9eurWjbtq
```

Refactor codes from .ipynb to .py

- Consciously and continuously extracting common building blocks
  - From notebooks (.ipynb) to source files (.py)
- Prevents duplication of code across multiple notebooks
- Separates the multiple concerns into logical units.
  - data layer (ETL code) -> modeling and experimentation -> output layer
- Make sure each notebook is always correct if “run all”
- Use naming conventions to indicate “owner” and give a sense of the order the analysis
  - e.g. `<step>-<initials>-<description>.ipynb`
  - `3.1-aiwei-extract-audio-features.ipynb`

Discussion: Do You Love/Hate Jupyter Notebooks?

- How long do you keep your notebook running?
- Have you tried Google CoLab? AWS SageMaker Studio? JupyterLab?
- Have you tried running Jupyter Notebook on a remote server?

Always use Version Control

- The most widely-acknowledged norm of software engineering.
- It is a good idea to initiate a project folder with a template
  - e.g. the Cookiecutter Data Science
- It makes it natural for you to adopt other good habits:
  - Setting up an environment (reduce the dependency error, increase consistency)
  - Enabling code reviews.
  - Making it easy to document decisions and design choices.
- It is also how AI coding assistant works (by creating a “diff”)

Keep the Following Away from Version Control

- Large data file,
- Secrets
  - OpenAI API keys, AWS Credentials, SSH keys, Database Passwords, …
  - Store them in a configuration file. (.env, .config, etc.)
- Other configuration file
  - Personal IDE setting (keyboard mapping)
  - Machine-dependent variables
    - Username, path-to-project-folder, etc.
- Use a package to load them automatically.

```python
import os
from dotenv import load_dotenv, find_dotenv

# find .env automagically by walking up directories until it's found
dotenv_path = find_dotenv()

# load up the entries as environment variables
load_dotenv(dotenv_path)

database_url = os.environ.get("DATABASE_URL")
other_variable = os.environ.get("OTHER_VARIABLE")
```

More on Jupyter Notebook

- Some IDE (e.g. VS Code, Cursor, PyCharm), with proper plugin, supports pseudo-notebooks.
- Use #%% to indicate new cells.

tooluse.py > calculator

```
Run Cell | Run Below | Debug Cell
#%%
# Block 1: Import necessary libraries
import json
from typing import Dict, List, Optional, Union
import openai
from dotenv import load_dotenv
import os

Run Cell | Run Above | Debug Cell
#%%
# Load environment variables
load_dotenv()

Run Cell | Run Above | Debug Cell
#%%
# Block 2: Define the calculator tool
def calculator(operation: str, numbers: List[float]) -> float:
    """
    A simple calculator that performs basic arithmetic operation

    Args:
        operation: One of '+', '-', '*', '/'
        numbers: List of numbers to operate on

    Returns:
```

opensmile.ipynb > ...

Open in Notebook Editor


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 11,
   "metadata": {},
   "outputs": [],
   "source": [
    "# To run this notebook, please install librosa, opensmile, in addition to standard packages such\n",
    "\n",
    "import librosa\n",
    "import opensmile\n",
    "from IPython.display import Audio, display\n",
    "\n",
    "import matplotlib.pyplot as plt\n",
    "import librosa.display"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Filename identifiers of the RAVDESS dataset\n",
    "# See: https://zenodo.org/records/1188976\n",
    "\n",
    "# Modality (01 = full-AV, 02 = video-only, 03 = audio-only).\n",
    "# Vocal channel (01 = speech, 02 = song).\n",
    "# Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust,\n",
    "# Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neu\n",
    "# Statement (01 = \"Kids are talking by the door\", 02 = \"Dogs are sitting by the door\").\n",
    "# Repetition (01 = 1st repetition, 02 = 2nd repetition).\n",
    "# Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).\n",
    "\n",
    "male_calm_normal_kids1 = \"Audio_Speech_Actors_01-24/Actor_01/03-01-02-01-01-01-01.wav\"\n",
    "male_happy_strong_kids1 = \"Audio_Speech_Actors_01-24/Actor_01/03-01-03-02-01-01-01.wav\"\n",
    "female_calm_normal_kids1 = \"Audio_Speech_Actors_01-24/Actor_02/03-01-02-01-01-01-02.wav\"\n",
    "female_happy_strong_kids1 = \"Audio_Speech_Actors_01-24/Actor_02/03-01-03-02-01-01-02.wav\""
   ]
  }
```

Discussion: Version Control

- Have you heard about git?
- Have you used git?
- What do you think is the most effective way to learn git?
-

Project Management — More Than Version Control

- Issues
- Project
- I personally keep two documents:
  - A “Design Doc”
  - A “Lab Log”

Start with a simple pipeline

- Data ETL,
- Feature extraction,
- Preliminaries - summary statistics, visualizing distribution, etc.
- Model fitting (a linear model is OK!)
- Model evaluation

- Allow you to think about problem formulation and evaluation criteria early on.

Other notes

- Test
  - Unit Test, Integration Test
- Log
- Sanity check
  - Rule-of-thumb for validating findings from exploratory data analysis:
    - Findings that match intuition
    - Findings that are serendipitous but can be explained
    - Findings that warrant further investigation or intervention

```
2022-11-01 14:10:02 INFO starting run on branch master (HEAD @ 37c02ab)
2022-11-01 14:10:02 DEBUG set random seed of 42
2022-11-01 14:10:02 DEBUG reading config file
2022-11-01 14:10:02 DEBUG … settings: {"n_threads": 4, <...snip...>}
2022-11-01 14:10:03 INFO reading in and merging data files
2022-11-01 14:10:39 INFO finished loading data: 2,835,824 rows
2022-11-01 14:10:40 WARNING … dropped 126,664 rows where ID was duplicated (4.47%)
2022-11-01 14:10:41 WARNING … dropped 2,706 rows where column `total` was null (0.09%)
2022-11-01 14:10:41 INFO creating train/test split
2022-11-01 14:10:41 INFO … train: 0.5 [1,353,227 rows]
2022-11-01 14:10:41 INFO … test: 0.5 [1,353,227 rows]
2022-11-01 14:10:42 INFO starting grid search cross validation ...
2022-11-01 14:21:01 DEBUG … 30/120
2022-11-01 14:31:46 DEBUG … 60/120
2022-11-01 14:42:31 DEBUG … 90/120
2022-11-01 14:53:03 DEBUG … 120/120
2022-11-01 14:53:03 INFO finished cross validation, writing best parameters
2022-11-01 14:53:03 DEBUG … runs/2022-11-01_14-10-02/parameters/xgboost.yml
2022-11-01 14:53:03 DEBUG … runs/2022-11-01_14-10-02/parameters/random_forest.yml
2022-11-01 14:53:03 DEBUG … runs/2022-11-01_14-10-02/parameters/adaboost.yml
2022-11-01 14:53:03 INFO training voting classifier on ensemble of 3 best models…
2022-11-01 14:53:19 INFO making predictions
2022-11-01 14:53:22 INFO writing results to runs/2022-11-01_14-10-02/results.yml
2022-11-01 14:53:22 DEBUG … results: precision=0.9624 recall=0.9388 f1=0.9514
2022-11-01 14:53:23 INFO writing predictions to runs/2022-11-01_14-10-02/predictions.csv
```

Missing semester: did you ever

- Ssh to a machine, but figuring out that you need to type in password every time?
- Run a job on a remote machine, but afraid that the job will get lost if you disconnect

- If so, check out The Missing Semester of Your CS Education
  - https://missing.csail.mit.edu/

Open Discussion

- How comfortable do you find using AI assistant in coding?
- Are there “best” practices for using AI assistants?