Introduction to Workflows

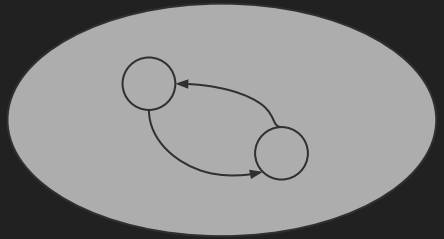
CMSE 890-402

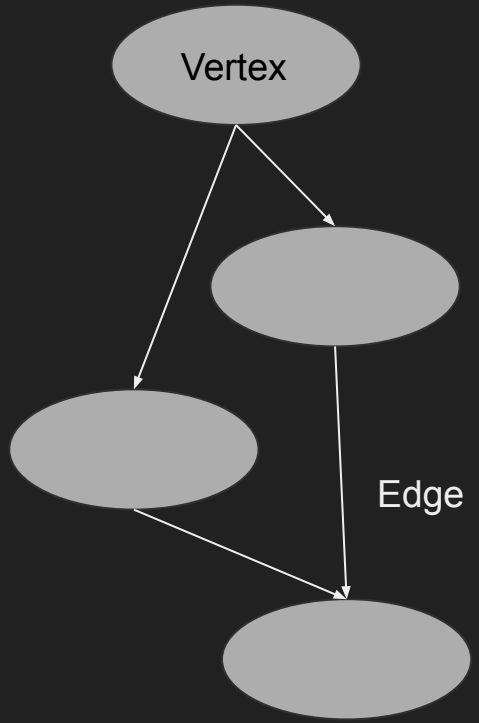
What is a workflow?

- Data In -> Process - > Result Out
- y = f(x) is a workflow!
- Recording data from an experiment and plotting it
- Downloading data and changing its format
- Running a simulation with multiple inputs
- Workflows can be described as a *Directed Acyclic Graph*

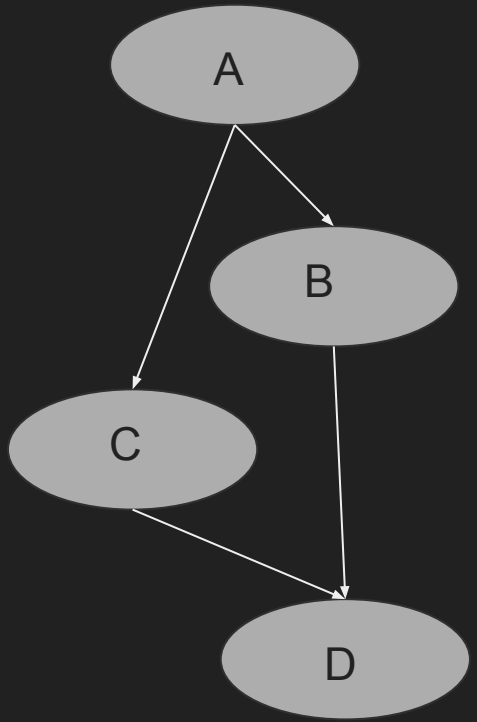
Directed Acyclic Graph (DAG)

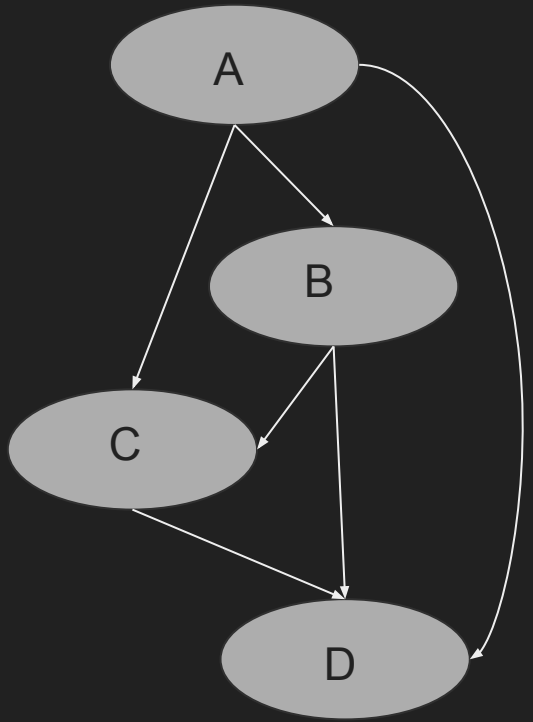
- Flowchart that goes in *one* direction
- Consists of *vertices* and *edges*
- Edges follow an *orientation*
- Edges do not return to a previous vertex **(no *cycles*)**
- Cycles can be encapsulated in a vertex (“condensation”)

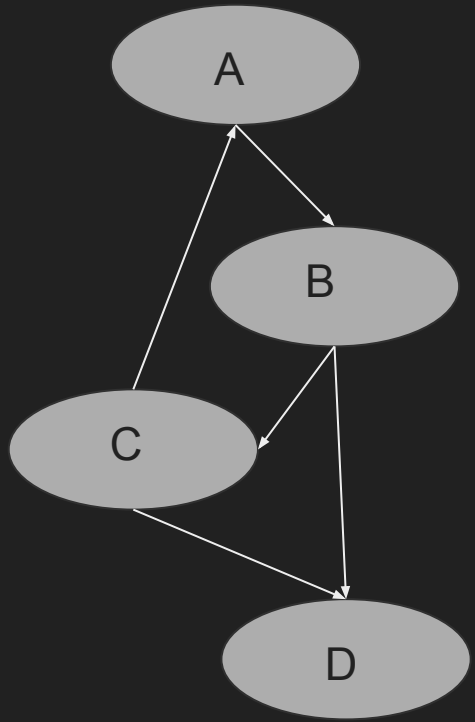




Examples

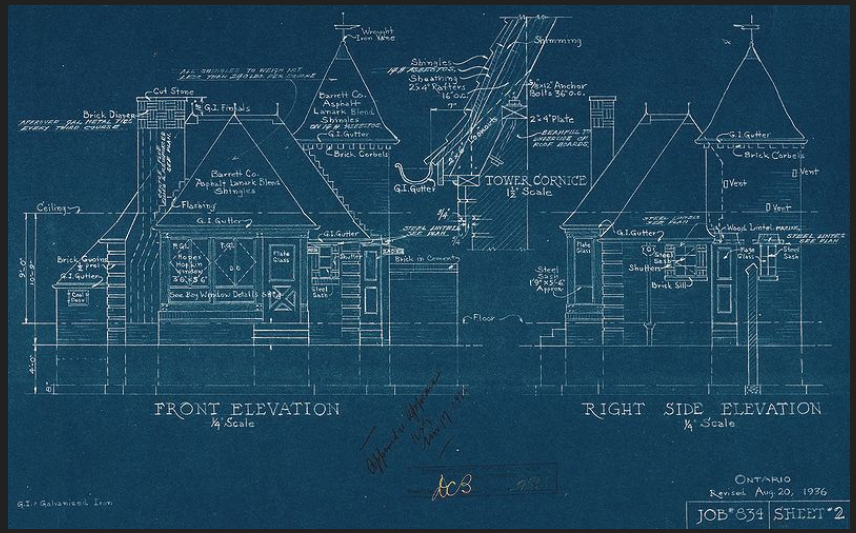






Why workflows are important

- Describe the research process
- Break work into manageable steps
- Provide a blueprint for future work
- Track data sources and outputs



Joy Oil Co Ltd, Public domain, via Wikimedia Commons

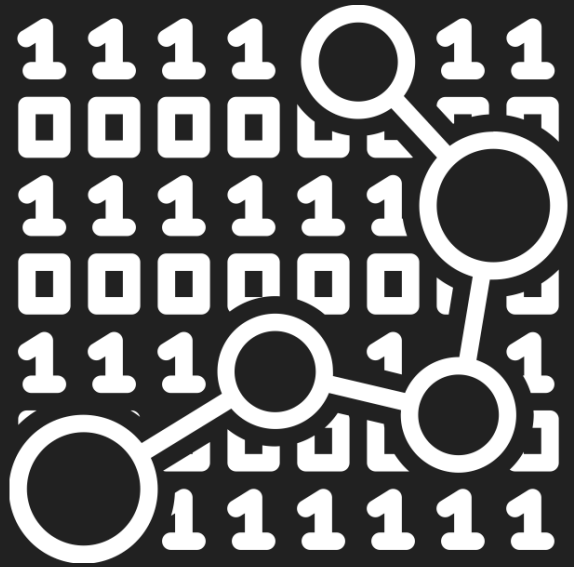
Why automation matters

- Reliability
  - Not prone to human error
- Reproducibility
  - Does the same thing every time
- Sustainability
  - Does not need human interaction to complete
- Speed
  - Does not have to wait for a human to complete

Achim Hering, Public domain, via Wikimedia Commons

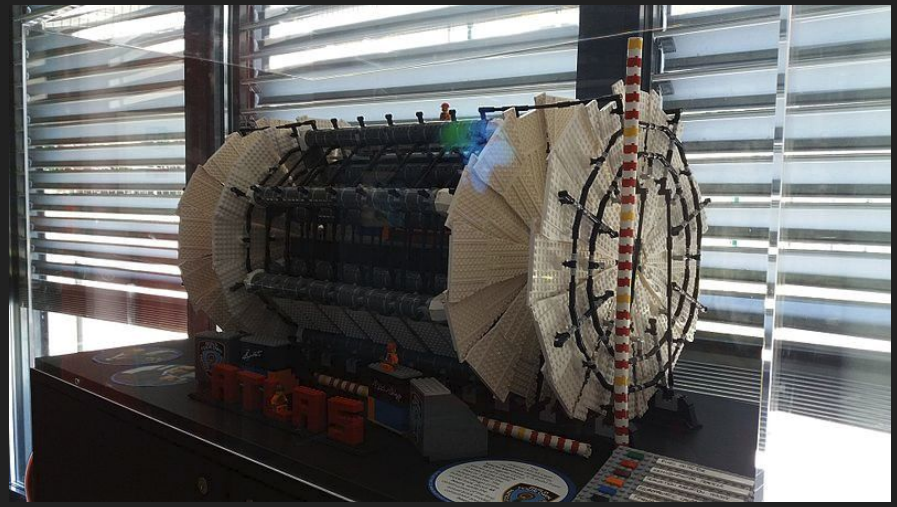
What computational workflows do

- Data processing!
  - Acquisition
  - Transformation
  - Reduction
  - Merging
  - Analysis
  - Presentation



Data Acquisition

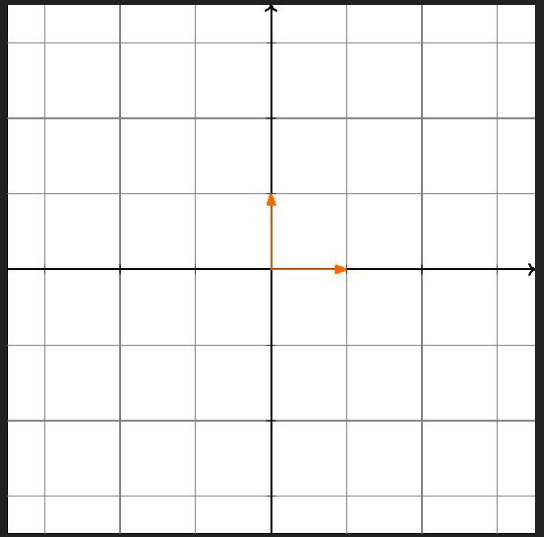
- Experimental equipment
- Download from a server
- Result from a simulation
- Result from a previous workflow step



Romainbehar, CC0, via Wikimedia Commons

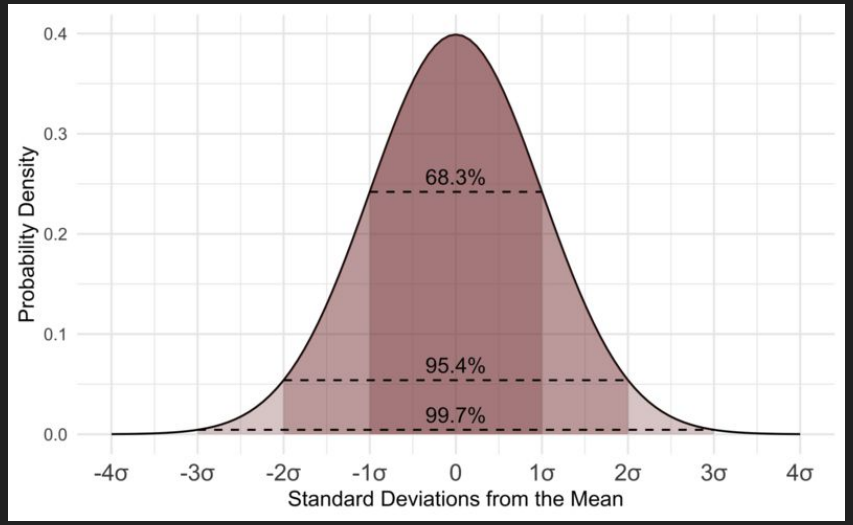
Data Transformation

- Rotate an image
- Convert a file format
- Transpose a table



Data Reduction

- Compute statistics from a column
- Extract individual frames from a video
- Cut out noise from audio
- Extract citations from text
- Filter a database



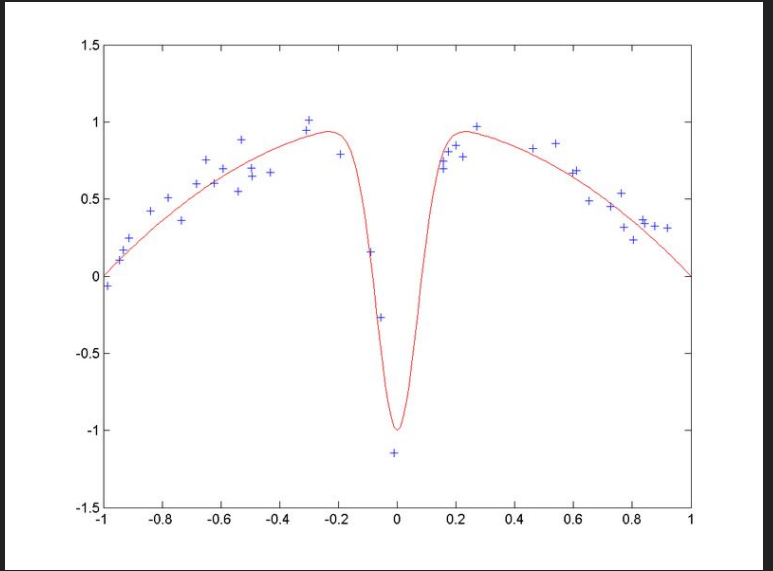
Data Merging

- Add columns to a table
- Put tables into a database
- Collect images into a video
- Place text into a single document

Transportstyrelsen; CC0, via Wikimedia Commons

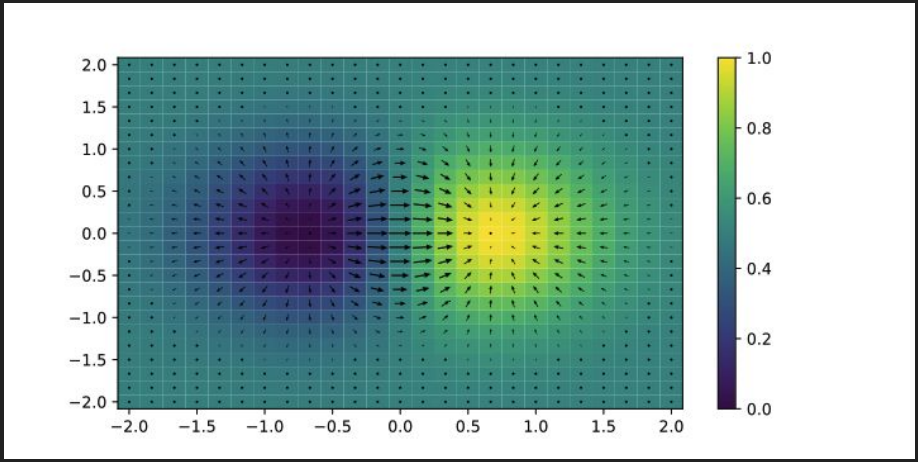
Data Analysis

- Compute statistics from multiple sources
- Extract interesting features
- Compare a model to an observation



Data Presentation

- Produce a plot from data
- Output a table in a human readable format
- Render an animation



How to build a computational workflow

1. Design: **The most difficult and important step**
    a. Break workflow into modules
    b. Connect modules together
2. Choose software description
    a. General scripting language (bash, Python etc.)
    b. Workflow description language (SnakeMake, NextFlow etc.)
3. Construct software description

Dataflow diagrams (DFDs)

- Created in the 1970s (*Structured Design*, Yourdan & Constantine)
- Common in business analysis
- Multiple systems of symbols
  - Yourdon and Coad
  - Yourdon and DeMarco
  - Gane and Sarson
- Four components:
  - External entity
  - Process
  - Data store
  - Data flow

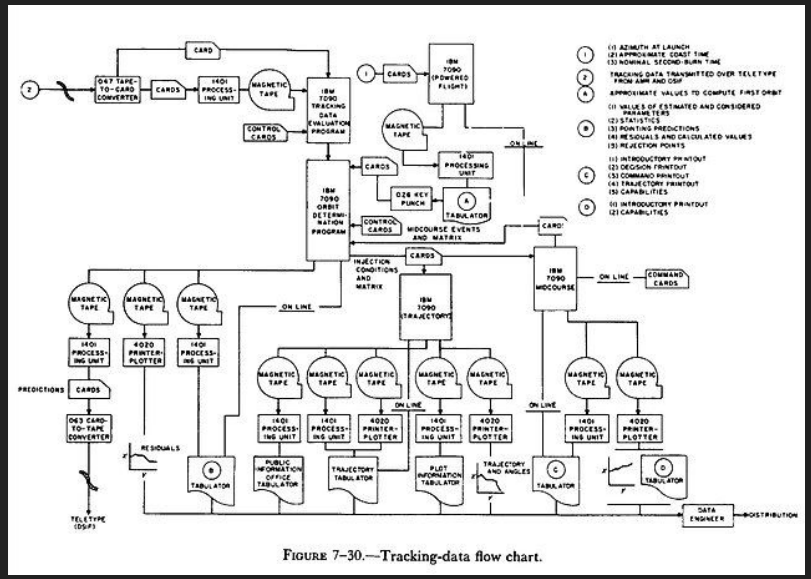


FIGURE 7–30.—Tracking-data flow chart.

NASA, Public domain, via Wikimedia Commons

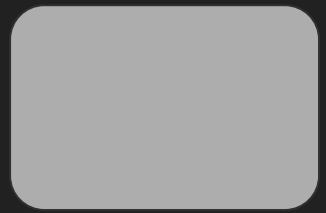
Symbols and Connectors

- Data flow connectors are DAG edges
- Symbols are DAG vertices
- DFDs do not have to be DAGs
  - But we are using them in this way
- Different sets of symbols can be used (but mean the same things)

Yourdon/Coad symbols (*Object Oriented Design*, Coad and Yourdon 1991)

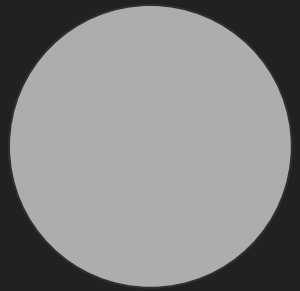
External entity

- Represents data from outside sources e.g. a physical experiment
- Named with a noun
- Sends or consumes information
- Data flows to and from entities only via processes



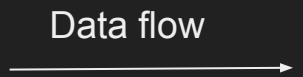
Process

- Named to describe *what* the process does (but not how)
  - E.g. a verb-object phrase “Merge tables”
- Must have both input and output



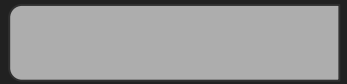
Flow

- Connects vertices together
- Labeled with the meaning of the data moving along the flow
  - E.g. “Decay energy”
- Same flow may have a different meaning for different parts of the system

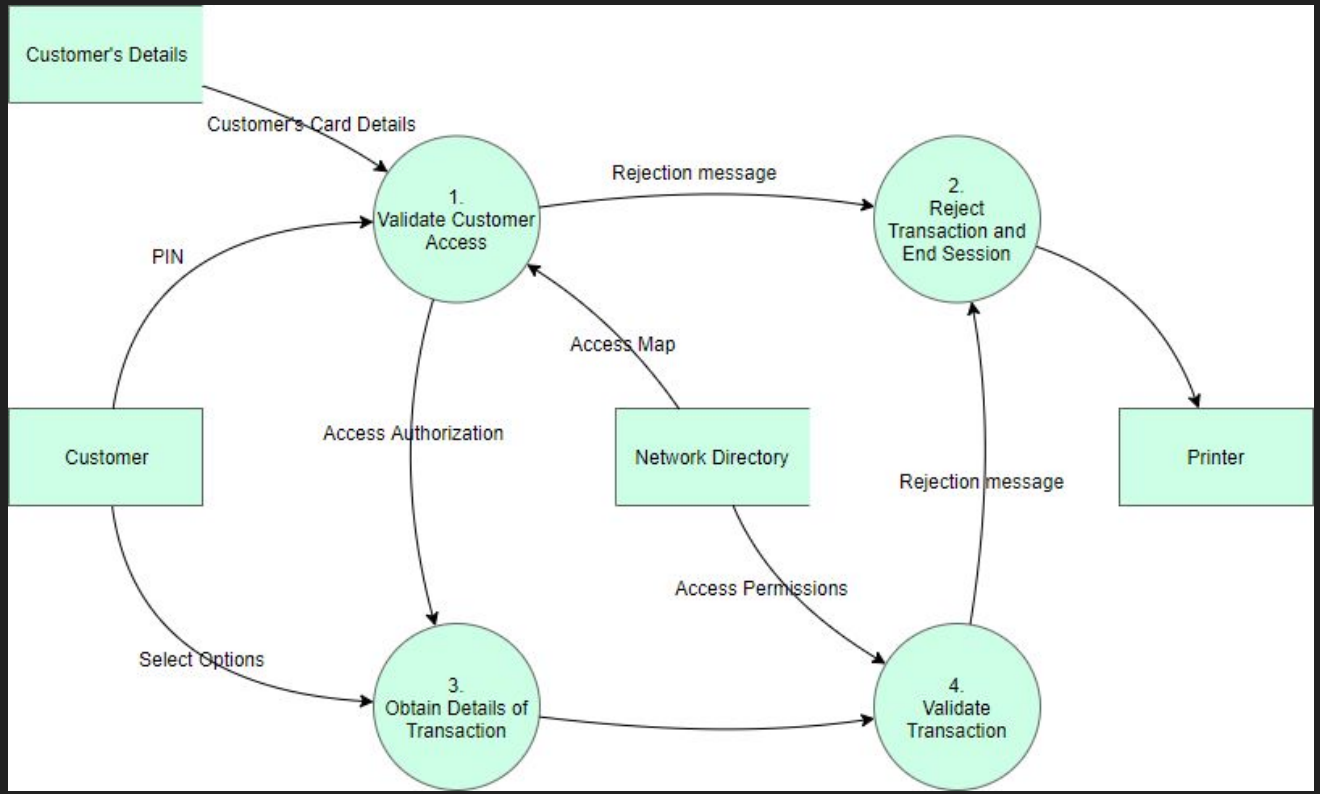


Data store

- Final resting place of data
- Represents a collection of data
- Stores are passive
  - Flow in: write, update
  - Flow out: read
- Data flows MUST come from processes to the data store



DFD Examples



https://online.visual-paradigm.com/de/knowledge/software-design/dfd-tutorial-yourdon-notation/

Avoid

- Processes with inputs and no output
- The inverse, processes with outputs and no input
- Unlabeled flows and processes

Possible source of confusion:

- DFDs *may* have loops if you search for examples. But for the purpose of a workflow DAG, those loops should be encapsulated in a process (which itself may be a DFD when “zoomed in”)

Design a Dataflow Diagram for your research project

- **Use at least one of each symbol**
- **The DFD should consist of at least 5 vertices**
- **The DFD should follow the properties of a DAG**
  - One data flow direction
  - No cycles
- Online software option: https://online.visual-paradigm.com/knowledge/software-design/dfd-tutorial-yourdon-notation/ (scroll down to Yourdon and Coad and click the “edit this example” button)
- Google apps or MS office should do the job as well
- If you don’t have a suitable project, examples follow. If the example says data comes from many locations, think about how to represent that.

Astronomy feature detection

- Download multiple catalogs from different astronomy databases
- Merge the catalogs by object
- Plot data about the objects
- Detect important features on the plot
- Save the plot

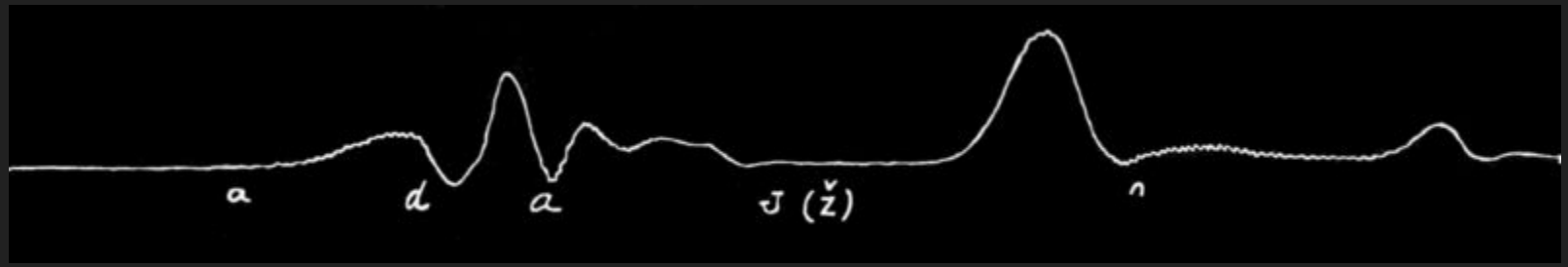
NASA, Public domain, via Wikimedia Commons

Genetics database matching

- Collect genetic data from samples of multiple individuals
- Convert the data format
- Clean the data
- Match the data to an existing database
- Save the matched table

Audio linguistics study

- Obtain audio files from multiple recordings
- Validate audio files
- Process audio (e.g. remove noise)
- Extract linguistics information
- Save linguistics information



Popular Science Monthly Volume 79, Public domain, via Wikimedia Commons

Homework

- Finish and submit DFD by midnight today
  - Save, photograph, or scan your DFD and submit it on D2L
- Create a GitHub account if you have not already and post it as a comment in your D2L submission

Pre-class 2:

- Complete the Git & GitHub fundamentals assignment
- Link available in D2L and class calendar