

A large, ornate telescope is mounted on a tripod, pointing diagonally across the frame. The telescope is made of brass and glass, with a large eyepiece at the front and a objective lens at the back. It is positioned against a dark, hazy background of a city skyline, possibly Paris, with the Eiffel Tower visible in the distance. The sky is overcast and grey.

Distant Viewing Toolkit

Tutorial Slides 2

Taylor Arnold
Lauren Tilton

These slides are designed to accompany the Python tutorials that we have built to introduce the theoretical and methodological foundations of *Distant Viewing* alongside an overview of the tools made available through the Distant Viewing Toolkit. The slides contain additional notes, references, and diagrams that we find useful when using the tutorials during in-person workshops. Most of the technical notes and materials are available direction in the notebooks, which can be accessed at the following links:

<https://colab.research.google.com/drive/1qQKQw8qHsTG7mK7Rz-z8nBfl98QBMWGf>

<https://colab.research.google.com/drive/1n7qWm47laCUJwg0-Rdx7pNw3dQWwuyxz>

For further information about our work, please visit Distant Viewing Lab website at <https://distantviewing.org> or feel free to email us directly at tarnold2@richmond.edu and ltilton@richmond.edu.

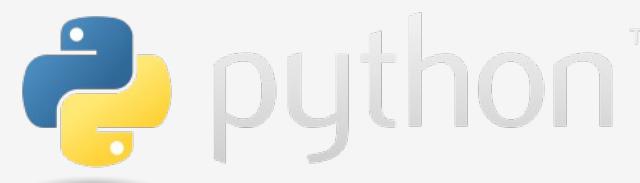
Tutorial 2

Network-Era Sitcoms and Visual Style

2.1 Setup

As with the first tutorial, the second tutorial makes use of Google's Colab environment, which allows us to use the Python programming language directly from the browser without needing to install any software locally. Colab can be used for small projects for free, with paid plans available to access more computing resources.

NOTE: If you already followed the first tutorial, the setup code and slides are very similar. Feel free to skim/skip the associated text and slides. However, you do need to re-run all of the code because in Colab we start with a fresh session each time we open a notebook. Also, here we are downloading new datasets.



2.1 Setup

We will make use of three common Python modules for data analysis: **NumPy**, **Pandas**, and **matplotlib**.

All the software libraries used in the tutorial are available under open-source licenses. The code we work through can all be run on your local machine with just a little bit of extra setup required to get started.



2.1 Setup

We will also be using a Python module that we built specifically for the humanistic analysis of visual materials: the distant viewing toolkit (**dvt**).

In this tutorial, we will use **dvt** to apply computer vision algorithm for face detection to digitized files containing moving images.

The screenshot shows the GitHub repository page for `dvt`. The main heading is "Distant Viewing Toolkit for the Analysis of Visual Culture". Below it, there are badge links for Python 3.10, PyPI v1.0.0, status stable, and JOSS 10.21105/joss.01800. The text describes the Distant Viewing Toolkit as a Python package for computational analysis of still and moving images, focusing on a minimal set of functions with few dependencies. It links to an "INSTALL.md" file for setup instructions and provides examples of use. A note at the bottom encourages users to open GitHub issues for troubleshooting or collaboration.

Distant Viewing Toolkit for the Analysis of Visual Culture

python 3.10 | pypi v1.0.0 | status stable | JOSS 10.21105/joss.01800

The Distant Viewing Toolkit is a Python package that facilitates the computational analysis of still and moving images. The most recent version of the package focuses on providing a minimal set of functions that require only a small set of dependencies. Examples of how to make use of the toolkit are given in the following section.

For more information about setting up the toolkit on your own machine, please see [INSTALL.md](#). More information about the toolkit and project is available on the following pages:

- Example analysis using aggregated metadata: ["Visual Style in Two Network Era Sitcoms"](#)
- Theory of the project: ["Distant Viewing: Analyzing Large Visual Corpora."](#)
- Software Whitepaper: [A Python Package for the Analysis of Visual Culture](#)

If you have any trouble using the toolkit, please open a [GitHub issue](#). If you have additional questions or are interested in collaborating, please contact us at tarnold2@richmond.edu and ltilton@richmond.edu.

Example Usage

To use the toolkit on a still image, we first use the `load_image` function to load the image in Python. Then, we create an annotation model; below we'll use an annotation that detects and identifies faces. Finally, we apply the annotation to the image and save the results.

2.1 Setup

When opening the tutorial notebook in Colab, this should open a window in the browser similar to the one on the right-hand side here.

The notebook can be viewed by anyone; in order to run the code itself, we will need to sign into a Google account.

Note: You will be able to edit the code and text locally in your browser. These changes will not be saved for anyone else. So, at any point, feel free to play around with the code!

Distant Viewing Tutorial 1: Movie Posters and Color Analysis

File Edit View Insert Runtime Tools Help Last saved at 9:34 AM

Comment Share Gemini

+ Code + Text

chapter of the book. We do not assume any prior knowledge of Python or computer vision in these notes. While a command line tool like Python is not in the scope of our introduction, we do our best to highlight the main features of the language as they apply to the application here. For more information about the distant viewing toolkit ([dvt](#)), open-source Python package that we have developed, please see the [project's homepage](#). More information about the theory of distant viewing and the specific application to movie image posters can be found in our book, which is available to download for free under an open access license on our [website](#) along with additional data and code to replicate the other studies shown in the text. A second notebook following up on the methods here using moving images can be found [here](#).

1.1 Setup

As a first step, we need to install a few additional Python components, download the movie posters dataset, and tell Python all of the functions that we will need later in the tutorial. To get started, we will use the code below to install the module called **dvt** (the distant viewing toolkit), which contains several useful functions specifically designed to apply computer vision algorithms to collections of humanities data. The exclamation point at the start of the line of code tells the notebook that we want to directly run a command line tool outside of Python itself. Here, we are using the tool called **pip** that can be used to install additional functionality for Python. To run the code, hover your mouse somewhere over the background of the code. This will show a triangular play button on the left-hand side of the code block. Hit the button and wait for it to finish, which may take a minute or two as Colab always takes a bit of time to set up when running the first code block.

```
▶ !pip install -q dvt
```

Run cell (⌘/Ctrl+Enter)
cell has not been executed in this session

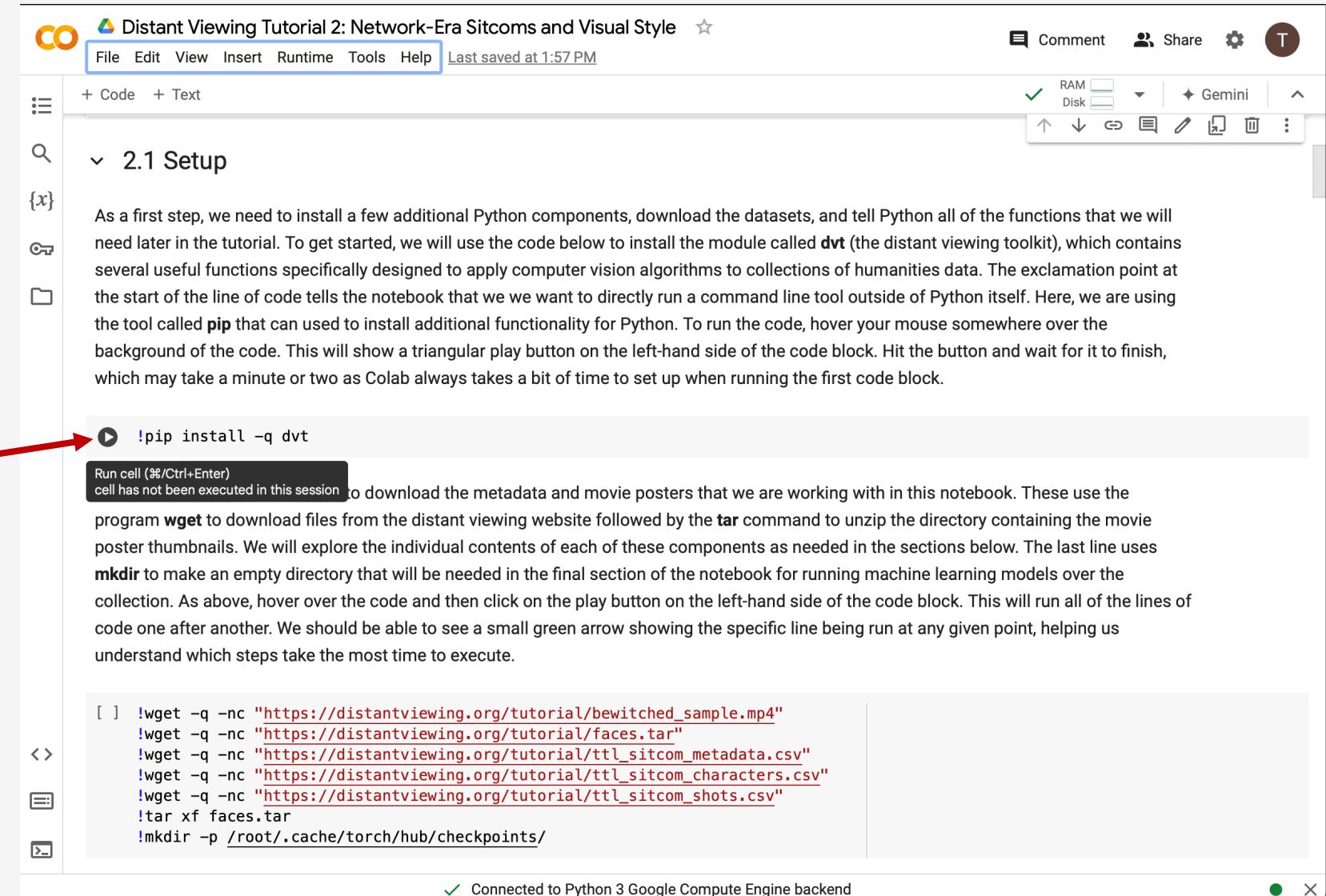
to download the metadata and movie posters that we are working with in this notebook. These use the program **wget** to download files from the distant viewing website followed by the **tar** command to unzip the directory containing the movie poster thumbnails. We will explore the individual contents of each of these components as needed in the sections below. The last line uses **mkdir** to make an empty directory that will be needed in the final section of the notebook for running machine learning models over the collection. As above, hover over the code and then click on the play button on the left-hand side of the code block. This will run all of the lines of code one after another. We should be able to see a small green arrow showing the specific line being run at any given point, helping us understand which steps take the most time to execute.

```
[ ] !wget -q -nc "https://distantviewing.org/tutorial/movies\_50\_years\_meta.csv"  
!wget -q -nc "https://distantviewing.org/tutorial/movies\_50\_years\_tags.csv"
```

2.1 Setup

After signing into a Google account, we will be able to run the code block (the parts of the notebook on a grey background). To run the code, hover over the grey background and click on the play button as shown by the red arrow here.

Throughout the tutorial, we need to run each of the code blocks in order. You can go ahead now to read and run all of the setup code shown in the first section of the notebook.



Distant Viewing Tutorial 2: Network-Era Sitcoms and Visual Style

File Edit View Insert Runtime Tools Help Last saved at 1:57 PM

RAM Disk Gemini

2.1 Setup

As a first step, we need to install a few additional Python components, download the datasets, and tell Python all of the functions that we will need later in the tutorial. To get started, we will use the code below to install the module called `dvt` (the distant viewing toolkit), which contains several useful functions specifically designed to apply computer vision algorithms to collections of humanities data. The exclamation point at the start of the line of code tells the notebook that we want to directly run a command line tool outside of Python itself. Here, we are using the tool called `pip` that can be used to install additional functionality for Python. To run the code, hover your mouse somewhere over the background of the code. This will show a triangular play button on the left-hand side of the code block. Hit the button and wait for it to finish, which may take a minute or two as Colab always takes a bit of time to set up when running the first code block.

```
!pip install -q dvt
```

Run cell (⌘/Ctrl+Enter)
cell has not been executed in this session

To download the metadata and movie posters that we are working with in this notebook. These use the program `wget` to download files from the distant viewing website followed by the `tar` command to unzip the directory containing the movie poster thumbnails. We will explore the individual contents of each of these components as needed in the sections below. The last line uses `mkdir` to make an empty directory that will be needed in the final section of the notebook for running machine learning models over the collection. As above, hover over the code and then click on the play button on the left-hand side of the code block. This will run all of the lines of code one after another. We should be able to see a small green arrow showing the specific line being run at any given point, helping us understand which steps take the most time to execute.

```
[ ] !wget -q -nc "https://distantviewing.org/tutorial/bewitched_sample.mp4"  
!wget -q -nc "https://distantviewing.org/tutorial/faces.tar"  
!wget -q -nc "https://distantviewing.org/tutorial/ttl_sitcom_metadata.csv"  
!wget -q -nc "https://distantviewing.org/tutorial/ttl_sitcom_characters.csv"  
!wget -q -nc "https://distantviewing.org/tutorial/ttl_sitcom_shots.csv"  
!tar xf faces.tar  
!mkdir -p /root/.cache/torch/hub/checkpoints/
```

Connected to Python 3 Google Compute Engine backend

2.2 Network-Era Sitcom Dataset

Our analysis will study two popular U.S. sitcoms that ran from the mid-1960s into the early 1970s. A short summary of each reveals why they are often compared and contrasted with one another:

Bewitched: Central premise focuses on the marriage of the ordinary Darrin Stevens to the supernatural witch Samantha. Samantha does her best to live a “normal” American life, but difficulties from her magical world crop up to make this process challenging.

I Dream of Jeannie: Features a 2,000-year-old female genie, referred to as Jeannie, discovered by the astronaut Tony Nelson, whom she refers to as her “master.” A recurring plot device involves Tony attempting to fulfill his job responsibilities while hiding from and dealing with Jeannie’s magical antics.



Bewitched (ABC; 1964–1972)

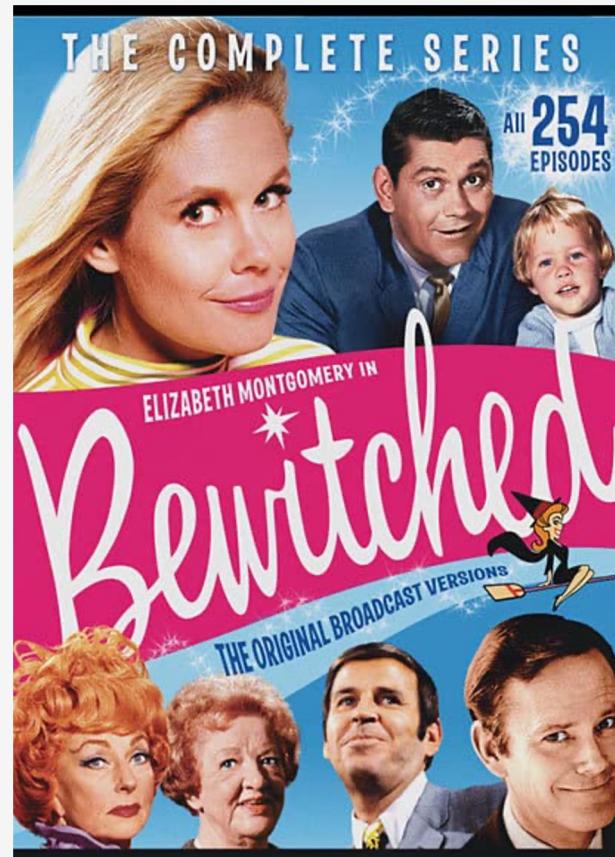


I Dream of Jeannie (NBC; 1965–1970)

2.2 Network-Era Sitcom Dataset

In order to work computationally with these shows, we first purchased DVD sets that contained all 393 episodes from the two series. Then, we extracted the video files from the DVDs, a process now allowed in the U.S. through the DMCA §1201 of the U.S. copyright code.

The total collection contains over 150 hours of material consisting of over 13 million individual frames.



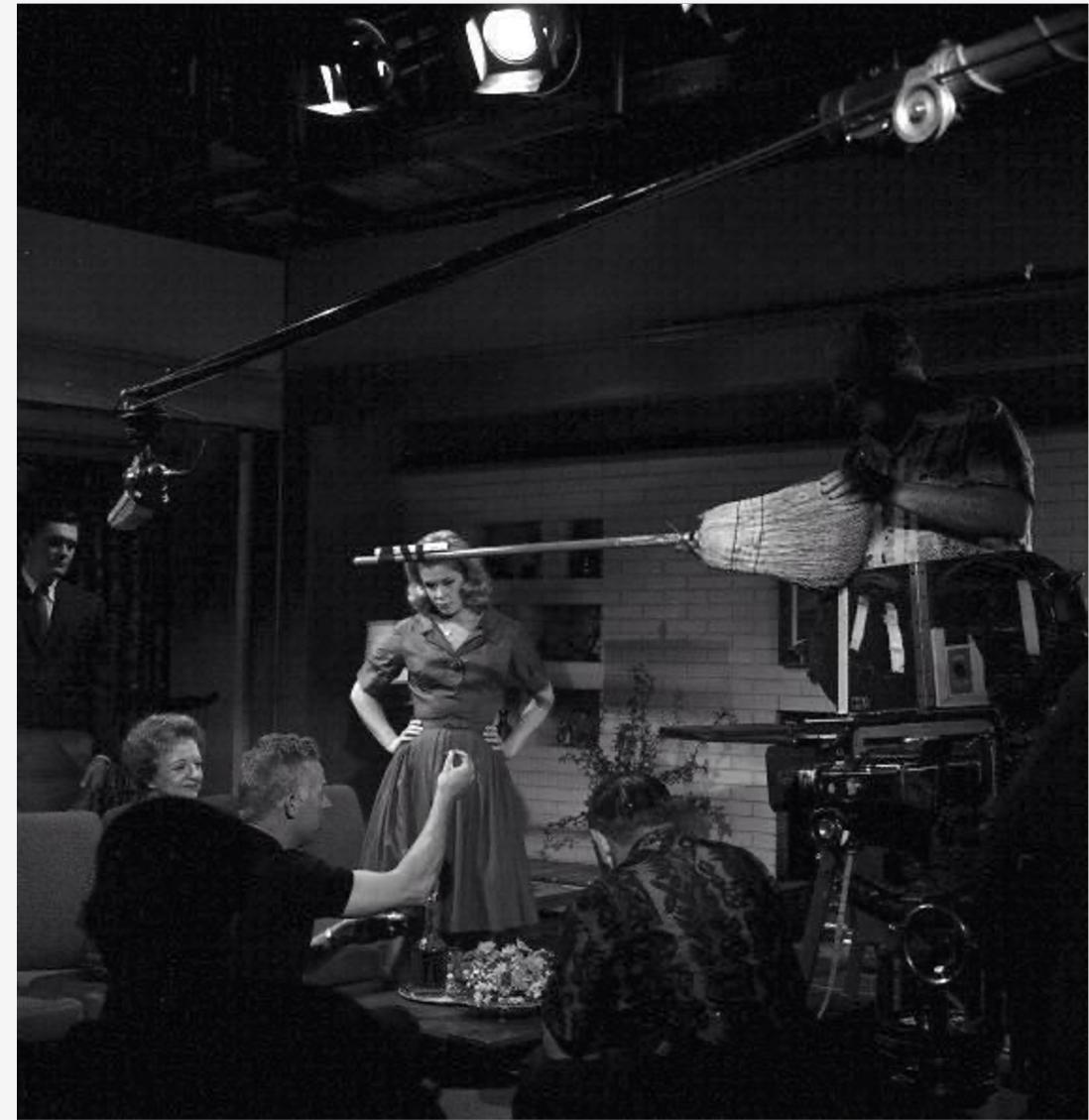
Dombrowski, Q., & Tilton, L. (2024). Access and Advocacy: Text & Data Mining and DMCA § 1201. *Digital Studies/Le champ numérique*, 13(3).

2.2 Network-Era Sitcom Dataset

Finally, before continuing on to the computation analysis, consider the following research questions that motivated our work on these shows:

1. How does the framing of a shot contribute to the pacing of the episode and how do these elements differ across series, time, and other features?
2. What and how do the visual and aural form of the television shows establish the centrality and role of each character?
3. What and how do the visual and aural form of the television shows establish relationships and intimacy between pairs of characters?

Consider your own hypotheses about these questions. Can you add any other similar questions that we could use this collection to address?



Filming on the set of *Bewitched* (1965).

2.3 Sample Movie File

When doing computational analyses with visual and audiovisual materials, it is important to frequently go back and look at specific examples to ensure that we understand how our large-scale analysis connects back to the actual human experience of viewing.

Make sure to take some time watching the 45 second clip from *Bewitched*.

After watching the short clip, do you have any updated thoughts about our research questions? Are there any new research questions you might consider adding?



2-inch videoreels from UCLA Film & TV Archive

2.4 Working with Video Frames as Images

While video files are stored in highly compressed formats, often the best way to think of a digital video file is as a sequence of individual frames (along with a soundtrack), each made from an image of the same size.

As an example, the image on the right was made by taking every 10th frame from a forty-second clip at the start of the Bewitched episode “The Magic Cabin” (season 2, episode 16).



2.5 Shot Boundary Detection

As noted in the code, detecting the boundaries between shots is a frequent and important step in the computational analysis of moving images.

Looking at the sample frames on the right, can you identify where the shot breaks are? Notice that using brightness here as in the example in the notes would be challenging because at least one shot does not change the overall brightness much and the end of the sequence finishes with a fade-out.



2.5 Shot Boundary Detection

Here are the detected shots from the algorithm included in the distant viewing toolkit. Do they correspond to the breaks that you found?



Li, Yanghao, et al. (2021) "Benchmarking Detection Transfer Learning with Vision Transformers." arXiv e-prints: arXiv-2111.

2.6 Face Detection and Facial Recognition

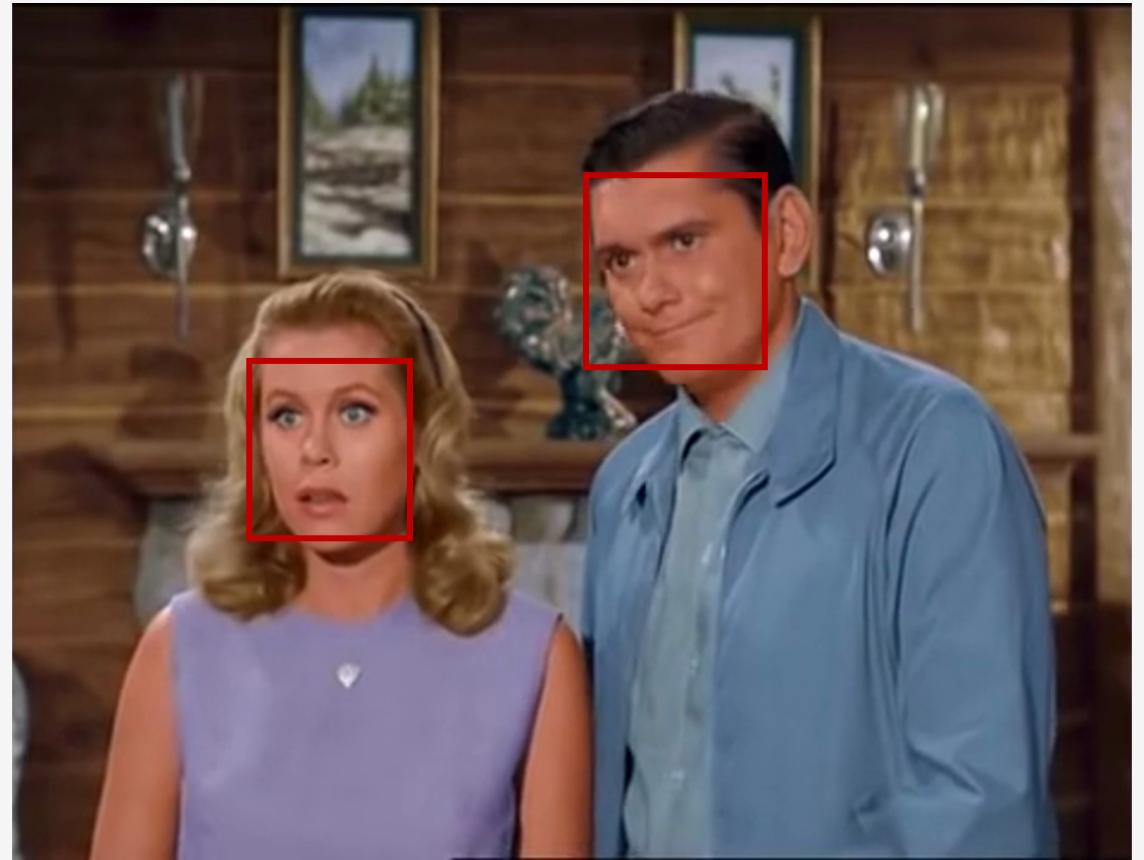
As you may have also seen in the first tutorial, detecting the location of faces in an image is often a useful annotation for many applications.

For moving images, noting the location of the character's faces can often serve as a starting point for understanding the editing and blocking of shots. These are exactly the kinds of relationships that align with our research questions.



2.6 Face Detection and Facial Recognition

Using the algorithm contained in the distant viewing toolkit, here are the two faces detected in this shot from *Bewitched*.

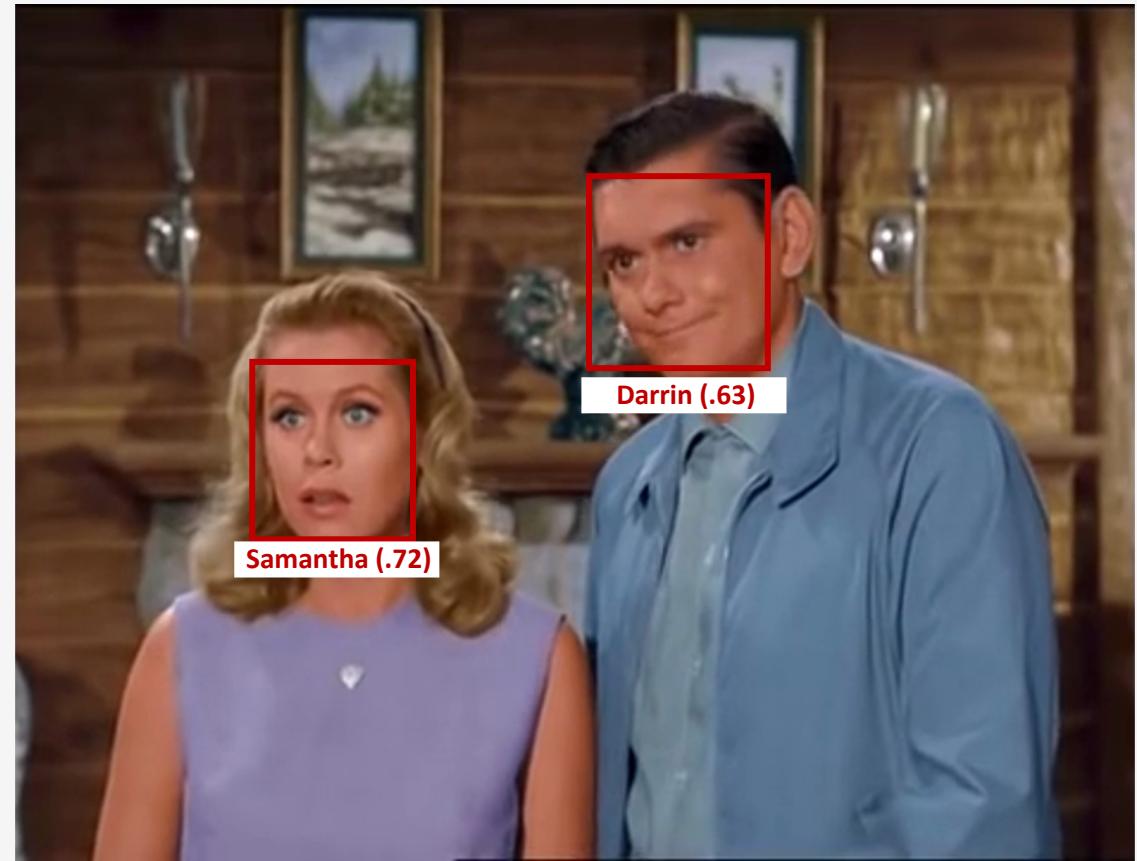


Zhang, K., et al. (2016). "Joint face detection and alignment using multitask cascaded convolutional networks." *IEEE Signal Processing Letters*, 23(10):1499–1503.

2.6 Face Detection and Facial Recognition

As described in the Python notebook, when working with film and television data we often also want to identify which character is associated with each of the faces in a frame.

Using the same technique shown in the example in the notebook, here are the estimated character names and similarity scores for the two characters.



Cao, Qiong, et al. (2018). "VGGface2: A dataset for recognising faces across pose and age." *13th IEEE international conference on automatic face & gesture recognition*.

2.7 Building Annotations from a Video File

Once we understand how to run the annotations over a short clip of the corpus, the next step is to run the algorithms over the entire collection. When working with large sets of moving images this can take a significant number of computational resources and/or time, so we have cached the results for the notebook to read in directly.

One step that we are skipping here is the method for selecting the thresholds and breakpoints when filtering the data. For example: At exactly what probability do we determine that a face is really found by the algorithm? How close does a face need to be to a portrait to consider having found a character? If we run the algorithm over many frames from a shot, how do we aggregate them? Answering these requires a lot of going back to the original materials and manually figuring out what thresholds work best for our application.



2.8 Full Corpus Analysis

In this section of the notebook, we see how to use the extracted data to address some of the research questions we previously posed.

Note that only a small set of the analyses contained in the book chapter are presented here. We encourage interested readers to check out the longer analyses featured in the book.

Can you think of ways to extend this analysis using newer techniques (LLMs?) or by including additional television shows into the corpus?



Stage 2 of General Service Studio, where *I Love Lucy* TV was produced, showing the three-camera setup used on stage. *American Cinematographer*.

2.9 Conclusions and Next Steps

There are many directions to explore after following the second tutorial:

1. To see how to complete the analysis in the second tutorial here, check out Chapter 5 of the *Distant Viewing*.
2. If you are new to programming, see the links in the notebook to learn a bit more about working with Python.
3. For those who skipped the first tutorial, it may be useful to now go back to it in order to gain a deeper understanding of how digital images work and the implications for the theoretical underpinning of distant viewing.

Thank you for taking time to engage with us. Please reach out with any questions or feedback!



<https://distantviewing.org>

tarnold2@richmond.edu
ltilton@richmond.edu

