

A large, ornate telescope is mounted on a tripod, pointing diagonally across the frame. The telescope has a dark body with gold-colored trim and lenses. It is positioned against a background of a city skyline, likely Paris, with the Eiffel Tower visible in the distance. The sky is overcast.

# Distant Viewing Toolkit

## Tutorial Slides 1

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](mailto:tarnold2@richmond.edu) and [ltilton@richmond.edu](mailto:ltilton@richmond.edu).

# Tutorial 1

# Movie Posters and Color Analysis

## 1.1 Setup

This 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.

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.



# 1.1 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**).

We will start by using **dvt** to load images into Python. The full power of the **dvt** module will become clearer as we move towards advanced computer vision algorithms towards the end of the first tutorial.

The screenshot shows the GitHub repository for 'distant-viewing / dvt'. The 'Code' tab is selected. The main content is the 'README.md' file, which has been updated by 'statsmaths'. The page includes a preview of the README content, which starts with a section titled 'Distant Viewing Toolkit for the Analysis of Visual Culture'. Below this, there are badge links for Python 3.10, PyPI v1.0.0, status stable, and JOSS 10.21105/joss.01800. The README text describes the Distant Viewing Toolkit as a Python package for computational analysis of images, mentioning its focus on minimal dependencies and providing links to example analyses, project theory, and a software whitepaper. It also encourages users to open GitHub issues for support.

## 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](mailto:tarnold2@richmond.edu) and [ltilton@richmond.edu](mailto: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.

# 1.1 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

Connect T4 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
```

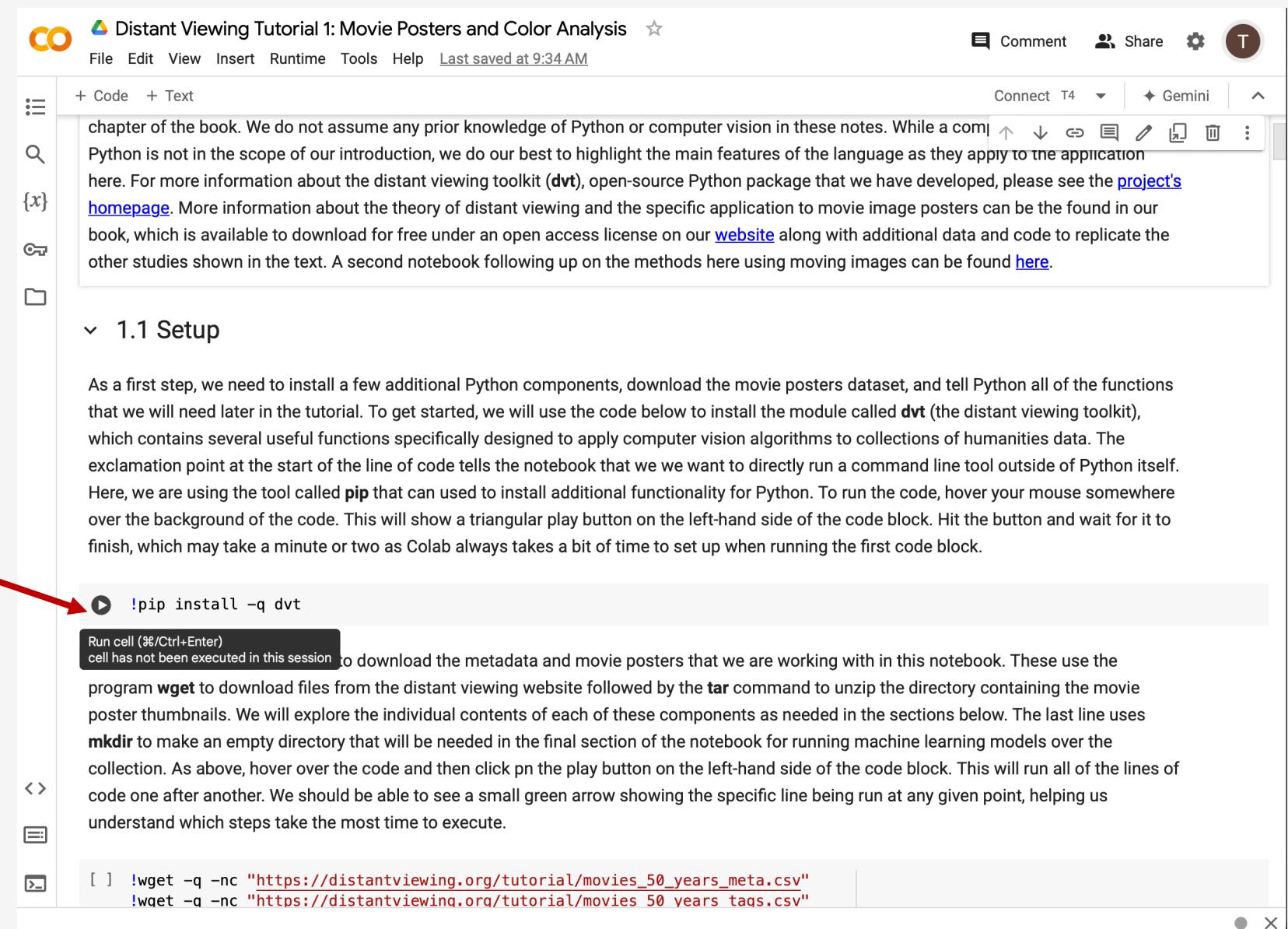
Run cell (⌘/Ctrl+Enter)  
cell has not been executed in this session

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

# 1.1 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 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.

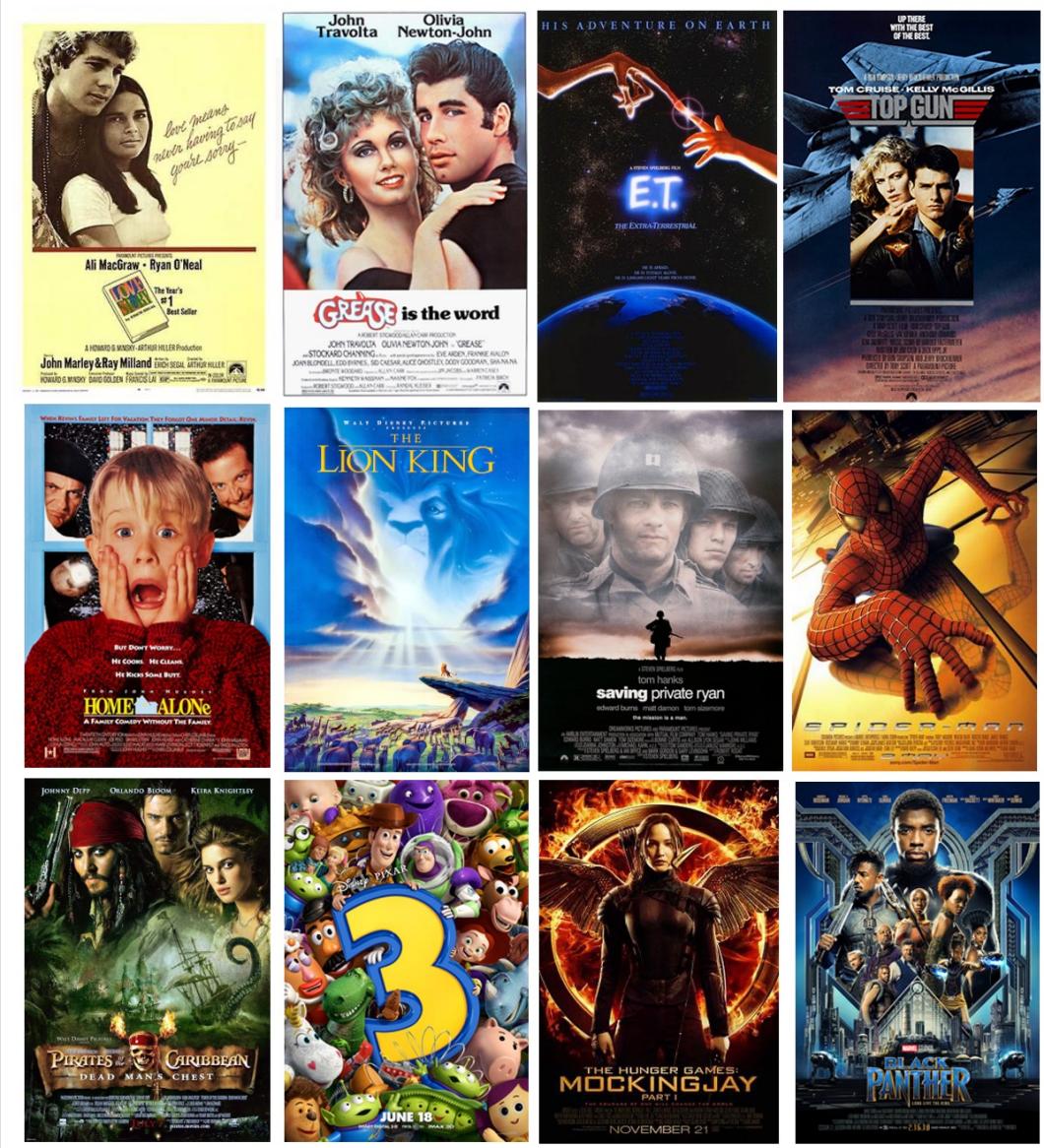
```
[ ] !wget -q -nc "https://distantviewing.org/tutorial/movies\_50\_years\_meta.csv"  
!wget -q -nc "https://distantviewing.org/tutorial/movies\_50\_years\_tags.csv"
```

## 1.2 Movie Poster Dataset

In this tutorial we are going to explore a collection of movie posters from the top-100 grossing films from every year between 1970 and 2019. Our corpus consists of nearly 5000 images (we were unable to locate a small set of older posters). Several examples from across the decades and genres are shown to the right.

Our goal is to understand how the color and composition of the posters has (1) changed over time and (2) is used to convey (or subvert) genre conventions. We want to use computational techniques to be able to identify and understand patterns across all the thousands of images in our collection.

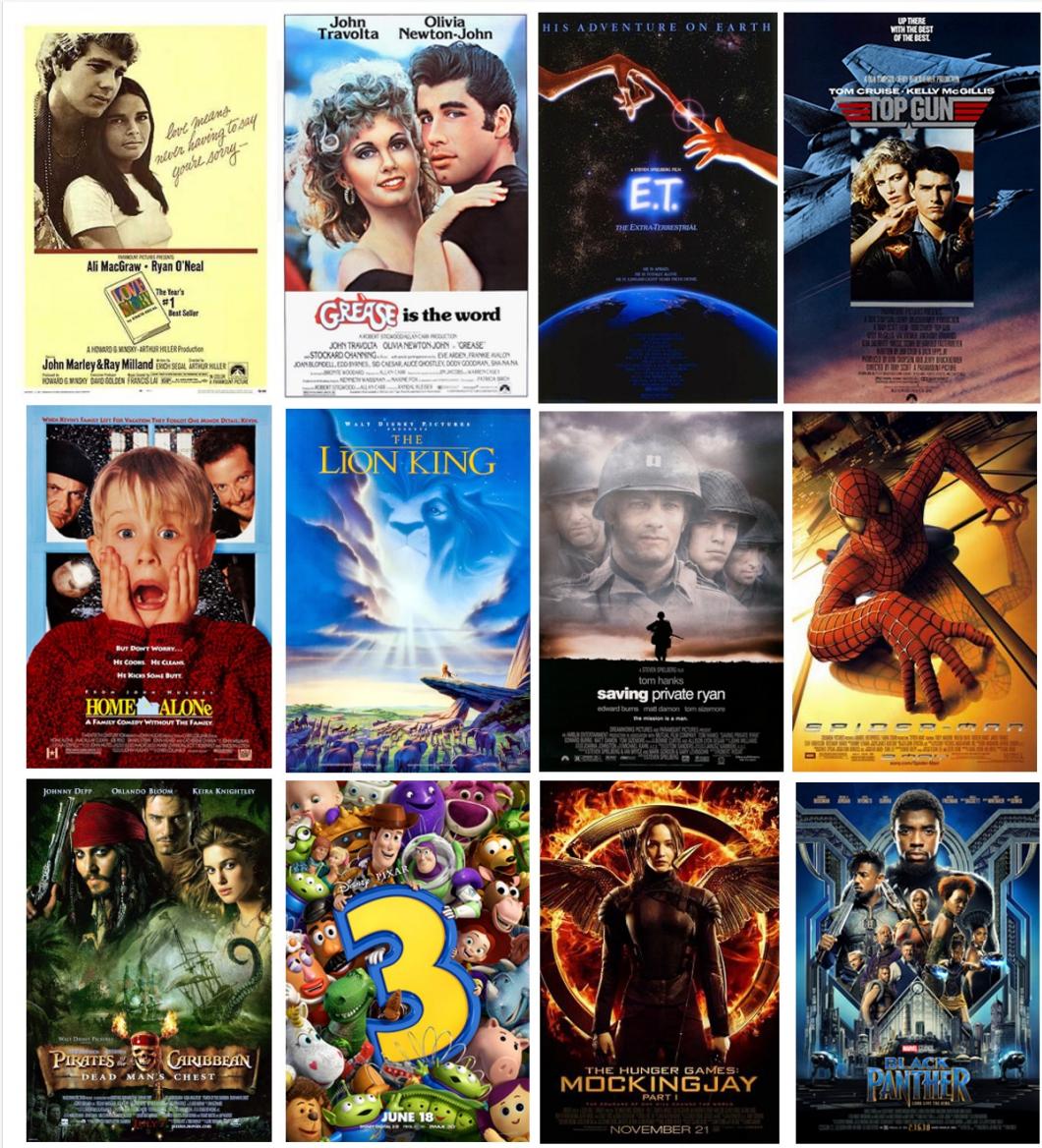
We will start by first understanding the ways that digital images are stored and how we can extract color information from them.



# 1.2 Movie Poster Dataset

Before we dive into the computational work, let's take a second to think about the underlying research questions of this work by considering the following questions:

1. What are the primary goals of movie posters? Have these goals changed over time?
2. What kinds of information do the makers of movie posters want to convey to the people who view them?
3. What kinds of information do you infer from the brightness, saturation, and the example movie posters on the right?
4. What hypotheses might we be able to produce about the relationship between the color and composition of a movie poster and genre?



## 1.3 Digital Images

On the right is a cropped image taken from a photograph credited to Russell Lee taken in August 1942 while working for the U.S. Federal government.

While the image was originally taken using physical color film, the version that we have here is a digitized scan stored and displayed on a computer.

When viewing the image with our eyes, we can identify a variety of elements such as the horse, the man, the sky, and the clouds. But how is this information understood by our computer?



# 1.3 Digital Images

Looking at just the green pixel values, we have sufficient space to see the actual numbers. As with the code in Python, these numbers are on a scale from 0 (no green) to 255 (as much green as possible). Note that there is a lot of green light here even though nothing in the original image is something that we would usually call “green.” The green light is blended with red and blue to create other colors such as brown, orange, and white.

127	117	121	132	116	115	114	114	116	115	116	113	112	113	111	115	112	114	115	112	113	111	111	111		
151	127	130	128	126	124	126	157	124	124	127	116	117	117	117	115	118	116	115	115	117	115	113	114	114	
183	174	157	160	140	131	132	166	200	187	175	126	102	116	122	120	120	118	117	120	117	115	118	116	114	114
182	183	173	173	185	186	195	198	203	196	184	146	124	144	145	123	122	124	124	120	123	118	118	118	118	118
174	174	175	181	192	201	203	214	210	197	248	12	34	34	33	26	144	128	137	124	123	125	118	124	121	121
200	190	187	185	193	195	195	196	13	27	28	30	31	26	24	31	33	201	190	167	131	126	124	126	122	122
199	196	192	193	189	190	189	192	33	33	16	131	95	33	34	28	35	193	184	170	147	145	140	130	127	127
193	188	188	192	198	204	203	195	37	19	97	86	73	49	19	28	187	170	158	147	139	130	137	131	131	131
188	175	168	176	181	178	177	174	174	167	34	186	92	40	34	124	157	145	143	137	138	137	137	137	131	131
147	144	148	151	155	153	152	155	165	160	125	102	33	33	57	217	155	144	143	144	142	141	140	139	140	140
148	150	148	154	150	158	156	162	161	157	159	35	40	32	218	173	208	112	148	148	150	143	144	148	145	145
150	151	148	150	152	148	151	154	151	152	156	97	38	215	218	129	28	160	172	161	147	151	148	150	148	148
152	154	168	154	152	153	157	153	152	236	230	139	126	203	30	31	127	138	125	132	32	155	154	155	153	153
155	154	156	156	155	159	159	153	141	230	200	121	174	34	87	109	137	127	57	77	60	162	160	158	155	155
159	159	160	157	158	157	163	157	137	81	196	190	155	66	77	130	91	16	105	80	72	170	168	172	162	162
162	166	162	162	162	170	181	132	132	25	76	40	23	63	110	106	52	114	68	79	124	211	215	203	193	193
167	169	169	165	170	167	172	123	127	121	52	75	58	66	107	70	50	117	91	107	108	158	204	192	183	183
173	170	172	174	175	175	170	117	85	99	98	35	97	69	88	47	54	122	26	120	78	71	171	168	173	173
185	201	238	230	228	193	85	105	109	30	97	19	127	129	133	142	146	70	85	67	60	36	176	176	174	174
226	225	223	227	131	72	75	37	60	182	148	45	131	117	115	101	95	84	71	137	58	188	178	183	174	174
218	217	208	247	216	39	31	122	35	135	153	66	110	102	62	102	111	82	63	56	41	184	181	183	190	190
188	207	218	200	91	22	43	30	33	21	41	26	47	24	42	45	32	64	62	43	34	195	199	204	206	206
71	205	196	184	106	121	28	28	27	27	68	25	24	26	28	25	26	27	31	28	205	200	195	189	188	188
67	90	186	182	72	28	19	166	27	25	26	27	24	52	92	22	20	25	27	27	37	124	197	196	198	198
77	60	138	100	200	27	207	73	19	26	20	24	68	78	118	67	43	40	26	26	34	201	189	216	219	219

# 1.3 Digital Images

To understand the way that digital images are stored, we will further zoom into our image and consider a lower resolution version.

On the top-left is the modified image centered on the man. The other three parts of the diagram show how the image is represented as *pixel intensities* through the blending of red, green, and blue light. Each point in the combined image is created by defining the amount of red, green, and blue light needed to re-create the given color.



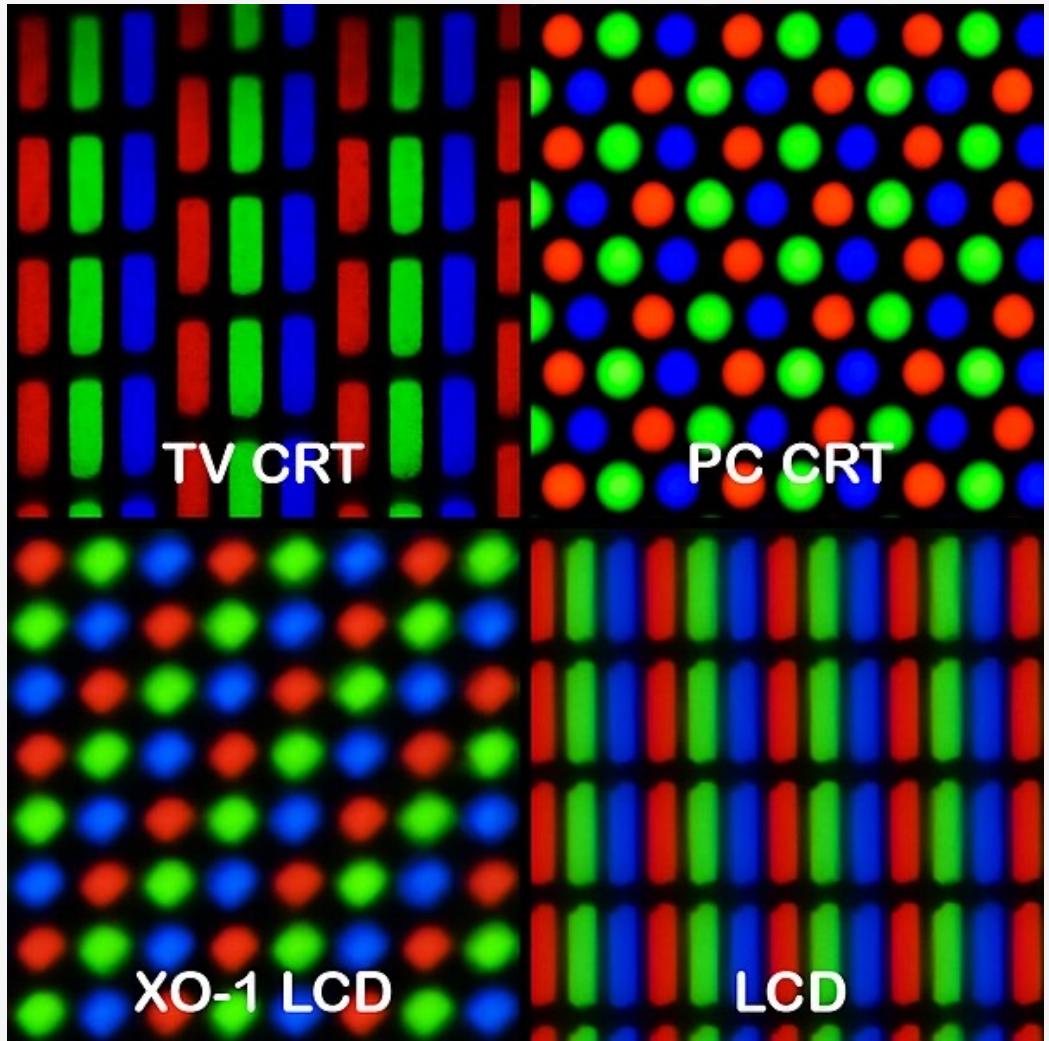
127	117	121	132	116	115	114	114	116	115	116	113	112	113	111	115	112	114	115	112	113	112	113	111	111	111		
151	127	130	128	126	124	126	157	124	124	127	116	117	117	115	118	116	115	115	117	115	113	114	114	114	114		
183	174	157	160	140	131	132	166	200	187	175	126	102	116	122	120	120	118	117	120	117	115	118	116	114	114		
182	183	173	173	185	186	195	198	203	196	184	146	124	144	145	123	122	124	124	120	123	118	118	118	118	118		
174	174	175	181	192	201	203	214	210	197	243	12	34	34	33	26	144	128	137	124	123	125	118	124	121	121		
200	190	187	185	193	195	195	196	13	27	28	30	31	26	24	31	33	201	190	167	131	126	124	126	122	122	122	
199	196	192	193	189	189	192	192	33	33	16	131	95	33	34	28	35	193	184	170	147	145	140	130	129	129	129	
193	188	188	192	198	204	203	195	37	19	76	73	49	19	28	187	170	158	147	139	130	137	131	131	131	131	131	
188	175	168	176	181	178	177	174	174	167	34	186	92	40	34	124	157	145	143	137	133	137	137	131	131	131		
147	144	148	151	155	153	152	155	165	160	125	102	33	33	57	217	155	144	143	144	142	141	140	139	140	140		
148	150	148	145	150	158	156	162	161	157	159	35	40	32	218	173	208	112	148	148	150	143	144	148	148	148		
150	151	148	150	152	148	151	154	151	152	157	56	97	38	215	215	129	28	160	172	161	147	151	148	150	148		
152	154	168	154	152	153	157	157	152	236	230	139	126	203	30	31	127	138	125	132	32	155	154	155	153	153		
155	154	156	155	159	159	153	141	230	200	121	174	34	87	109	137	127	57	77	60	162	160	158	155	155	155	155	
159	159	160	157	156	157	163	157	137	81	196	190	155	66	77	130	91	16	105	80	72	170	168	172	162	162	162	
162	166	162	162	170	181	132	132	25	76	40	23	63	110	106	52	114	68	79	124	211	215	203	193	193	193	193	
167	169	169	165	170	167	172	123	127	121	52	75	58	66	107	70	50	117	91	107	108	158	204	192	183	183	183	
173	170	172	174	175	175	170	117	85	99	98	35	97	69	88	47	54	122	26	120	78	71	171	168	173	173	173	
185	201	238	239	228	193	85	105	109	30	97	19	127	129	133	142	146	70	85	67	60	36	176	176	174	174	174	
226	228	223	227	131	72	75	37	60	182	148	45	131	117	115	101	95	84	71	137	58	188	178	183	174	174	174	
218	217	208	237	216	39	31	122	35	135	153	66	110	102	62	102	111	82	63	56	41	184	181	183	190	190	190	
188	207	218	200	91	22	43	30	33	21	41	26	47	24	42	45	32	64	62	43	34	195	199	204	206	206	206	
71	205	196	184	106	121	28	28	27	27	68	25	24	26	28	25	26	27	31	28	205	200	195	188	188	188	188	188
67	90	186	182	72	28	19	166	27	25	26	27	24	52	92	22	20	25	27	27	37	124	197	196	198	198	198	198
77	60	138	100	200	27	207	73	19	26	20	24	68	78	118	67	43	40	26	26	34	201	189	216	219	219	219	219

102	87	92	107	86	83	81	81	83	85	86	82	82	83	78	78	84	81	83	82	81	80	79	80	78	78		
129	99	102	101	98	92	98	130	96	93	96	84	84	84	82	85	86	86	84	84	87	82	83	81	81	81		
168	157	139	139	131	101	106	146	184	186	183	183	169	143	163	157	91	87	91	87	86	86	85	85	85	85		
158	160	160	166	176	187	189	199	193	191	255	20	41	41	40	35	122	96	107	91	90	92	86	90	87	87		
183	174	172	170	177	178	178	180	17	34	36	35	35	27	30	39	39	185	175	147	101	94	91	91	91	91	91	
180	180	176	177	173	174	175	186	39	35	37	188	137	40	49	28	37	176	167	150	126	121	113	98	93	93	93	
170	156	151	157	162	163	162	168	158	158	67	230	148	65	48	113	129	120	114	105	105	105	105	105	105	105	105	
120	115	123	126	128	128	128	130	146	144	173	137	68	51	90	227	127	111	110	111	109	107	105	105	106	106	106	
117	116	115	120	121	131	135	144	142	137	143	54	56	63	222	222	145	139	187	190	155	158	117	115	114	113	113	
114	118	115	116	120	120	129	127	123	150	110	53	236	222	145	139	187	190	155	158	117	115	114	113	113	113	113	
117	124	142	122	116	120	125	121	121	231	231	148	195	218	40	38	162	168	153	161	24	122	121	122	121	121	121	
121	120	122	124	124	133	133	122	154	233	206	131	168	45	106	143	173	154	85	103	67	128	128	124	122	122	122	
125	125	126	123	128	125	128	135	169	169	203	193	175	95	117	162	124	30	138	106	90	132	135	143	129	129	129	
131	131	128	128	128	145	163	159	163	46	102	60	32	90	143	137	83	152	97	108	148	196	203	186	186	186	186	
132	135	130	136	130	137	133	156	155	155	72	111	87	97	140	105	70	153	121	139	129	154	185	175	165	165	165	
138	135	137	142	140	139	139	150	120	128	139	49	131	95	117	54	73	153	54	153	105	67	148	150	152	152	152	
156	175	222	219	220	223	110	138	144	42	132	24	156	160	163	173	170	99	116	98	82	33	147	145	144	144	144	
217	210	210	213	213	135	96	106	48	86	204	197	56	162	156	152	136	128	114	100	163	84	162	150	156	156	156	
205	205	205	236	236	210	51	47	161	39	180	199	78	146	138	83	133	144	108	86	79	53	161	157	165	165	165	
79	196	162	166	169	129	132	34	28	33	37	92	26	31	37	37	32	31	32	36	173	166	166	166	165	165	165	
71	90	168	176	176	96	36	30	30	27	31	27	71	124	39	30	27	26	28	28	38	96	165	166	165	170	170	170
85	66	135	112	192	27	234	82	36	25	24	28	98	110	148	100	65	48	28	27	38	182	168	200	206	206	206	

<tr

## 1.3 Digital Images

Usually, particularly on modern screens, the individual pixels are too small to individually see. But know that if we zoomed in far enough, we would see the individual red, green, and blue lights as shown in the image on the right.



Peter Halasz, CC-SA 3.0, Wikimedia Commons

## 1.4 Distant Viewing: Theory

On the right is another set of zoomed versions of our example image. Each image is centered on the same point but zoomed out at different levels. For simplicity, we have converted the image into grayscale, which requires only one number for each pixel.

Notice that it is impossible to understand that the numbers in the upper-left hand corner represent the eye of a horse until we see those numbers in the context of a wider version of the entire image.

In fact, there is no explicit way that the digital image records that this is a horse. Rather, we recognize this as a horse because it *looks similar to that a horse looks when we see it with our own eyes*.

30	29	31	35	72	112	132	97	71	56	46
29	25	23	24	26	41	107	197	152	70	51
26	25	24	32	23	17	9	111	180	150	56
25	28	27	19	27	28	21	20	64	134	121
26	24	26	26	28	30	26	24	17	33	93
27	26	26	23	26	25	25	23	30	34	28
27	27	23	28	23	26	23	27	37	41	29
28	24	25	22	26	23	34	33	40	35	30
29	23	24	28	26	31	37	39	43	29	31
20	29	29	33	31	34	31	41	38	34	32
51	34	32	30	43	45	48	49	31	32	35

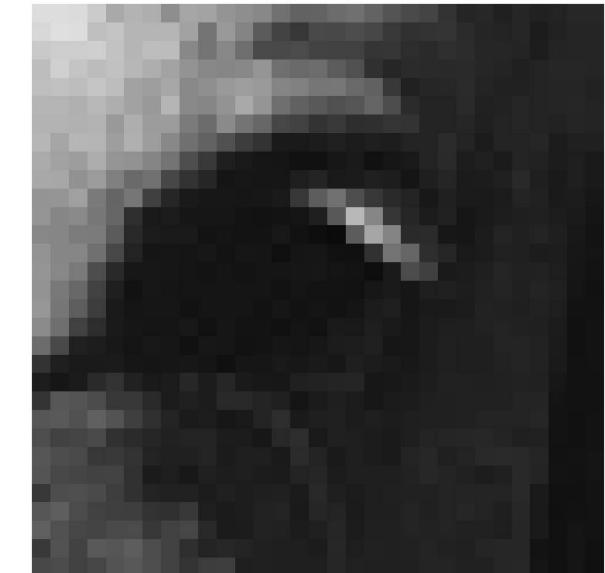


## 1.4 Distant Viewing: Theory

In fact, there is no explicit way that the digital image records that this is a horse. Rather, we recognize this as a horse because it *looks similar to that that a horse looks when we see it with our own eyes.*

To use the language of semiotics, digital images (just as with their analog counterparts) make meaning through their mimetic properties. They convey information by sharing similarities with the things they represent. This is very different from the way that textual data conveys information.

30	29	31	35	72	112	132	97	71	56	46
29	25	23	24	26	41	107	197	152	70	51
26	25	24	32	23	17	9	111	180	150	56
25	28	27	19	27	28	21	20	64	134	121
26	24	26	26	28	30	26	24	17	33	93
27	26	26	23	26	25	25	23	30	34	28
27	27	23	28	23	26	23	27	37	41	29
28	24	25	22	26	23	34	33	40	35	30
29	23	24	28	26	31	37	39	43	29	31
20	29	29	33	31	34	31	41	38	34	32
51	34	32	30	43	45	48	49	31	32	35



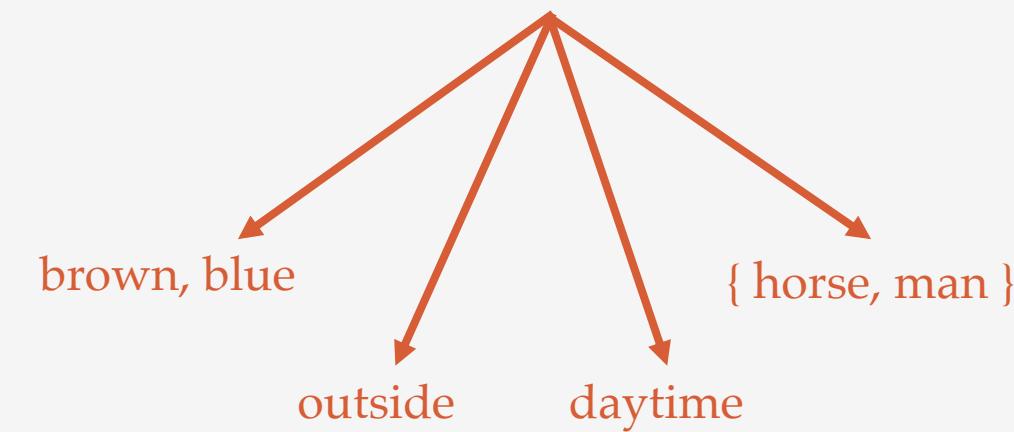
## 1.4 Distant Viewing: Theory

Because the digital images do not explicitly code semantic elements, we need to construct representations of the information that we want to study as a part of the analytical process.

For example, on the right we see a number of different ways that we can summarize the information about our example image through structured *annotations*. Ideally, we would be able to construct this algorithmically, which we will work towards in the remainder of the notebook.

Can you think of other ways to annotate the image with semantic information?

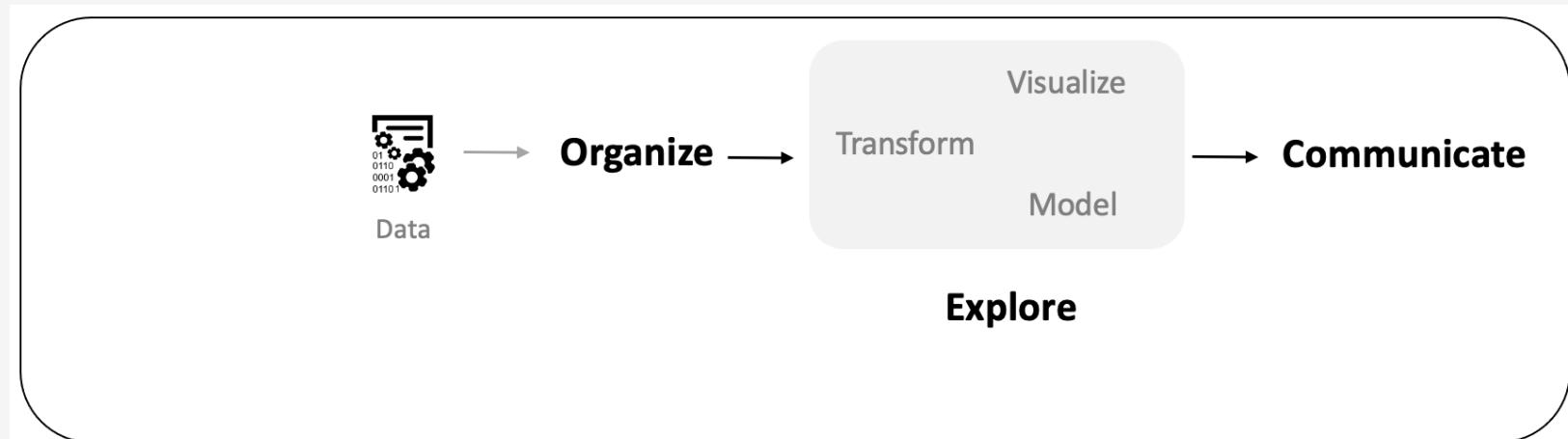
Notice that there is no way to perfectly represent the image through the annotations. We are always both missing information present in the image and making assumptions (and potential errors) in the way that we construct the annotations.



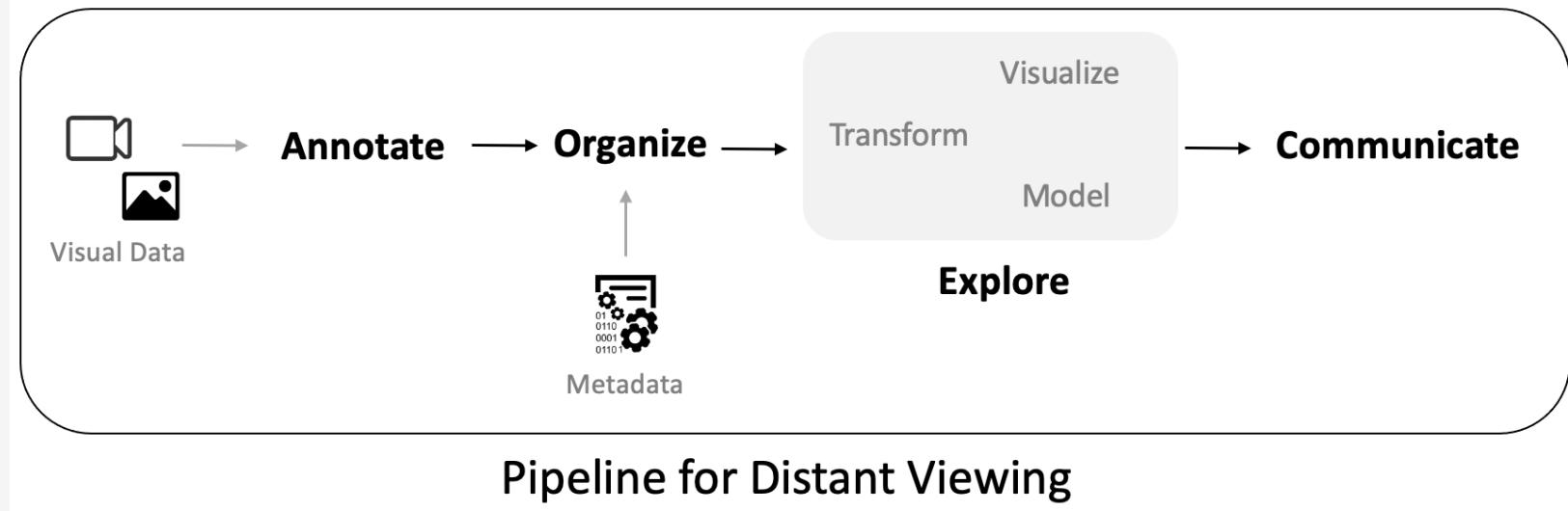
## 1.4 Distant Viewing: Theory

We can summarize how the process of working with digital images differs from working with structured tabular datasets that are common in the sciences and social sciences.

We need to *annotate* our data before doing any exploratory work. Also, we need to constantly return to the visual data itself to ensure that our annotations capture the information that we are interested in studying.



Data Science Pipeline (Structured Data)



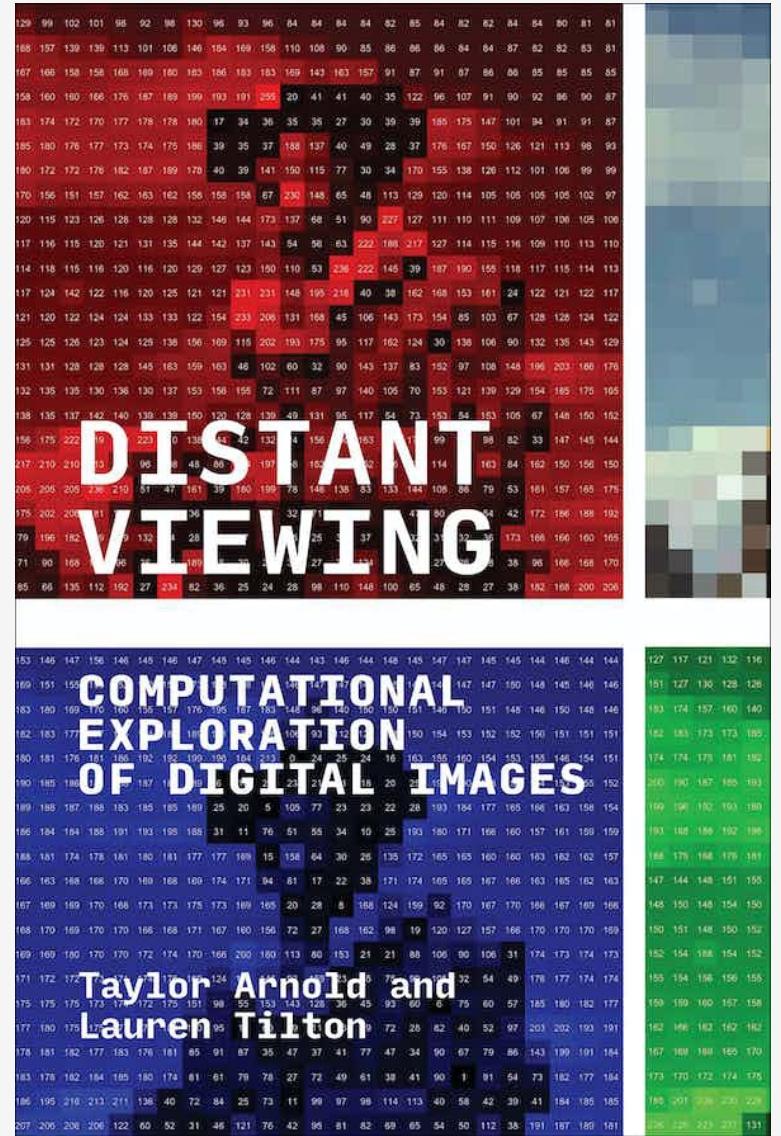
Pipeline for Distant Viewing

# 1.4 Distant Viewing: Theory

The *theory of distant viewing* combines the observations here with scholarship in semiotics and visual culture to present a theory of how computational exploration of digital images through the application of computer vision works, and why it is needed. We can summarize these results by the following:

1. Mimetic nature of images necessitates that the computational analysis of digital images starts with the extraction of semantic metadata (“annotations”) using computer vision.
2. The process of constructing annotations is a computationally mediated form of viewing.
3. The application of computer vision is not a neutral process. It engages in socially mediated ways of seeing and viewing that are encoded into the algorithms and the ways that they are used.

For a more detailed presentation of the theory, please see *Distant Viewing: Computational Exploration of Digital Images* (MIT Press, 2023), which is available under an Open Access license from our website: <https://distantviewing.org/book>

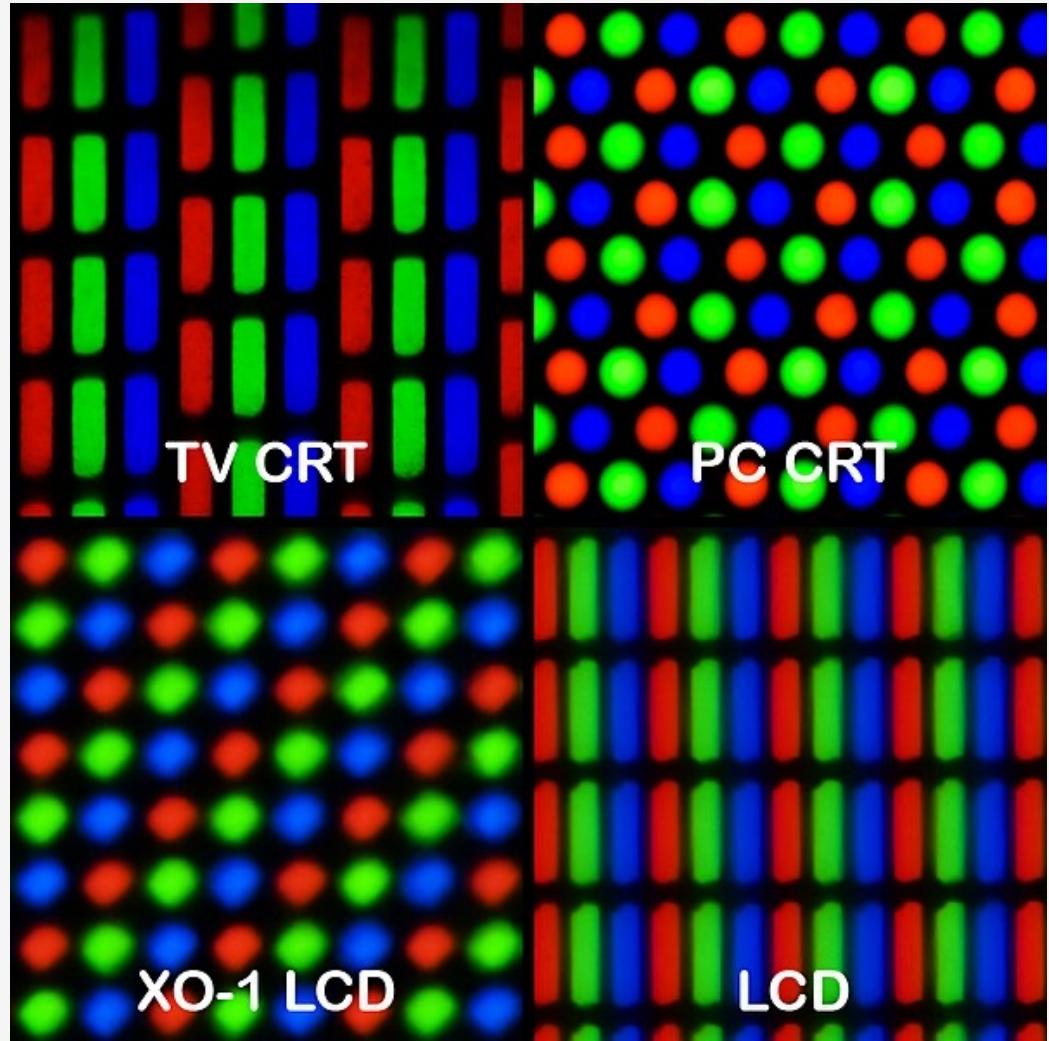


# 1.5 Annotating Image Brightness

We start with a relatively straightforward annotation that measures the how bright one of our posters is. Before we move to the code consider the following questions:

1. What changes might there be in the brightness of movie posters over time? What may contribute to these changes?
2. How might brightness relate to the genre of a movie poster? Which genres do you expect to be the brightest? Which do you expect to be darkest?

How should we measure the brightness of an image? Recall that the pixel intensities represent how much red, green, and blue light to use to create a specific color. From this, we see that taking the average pixel values across an image will tell us how much light is used in representing the image, therefore presenting one way of measuring the *brightness* image.

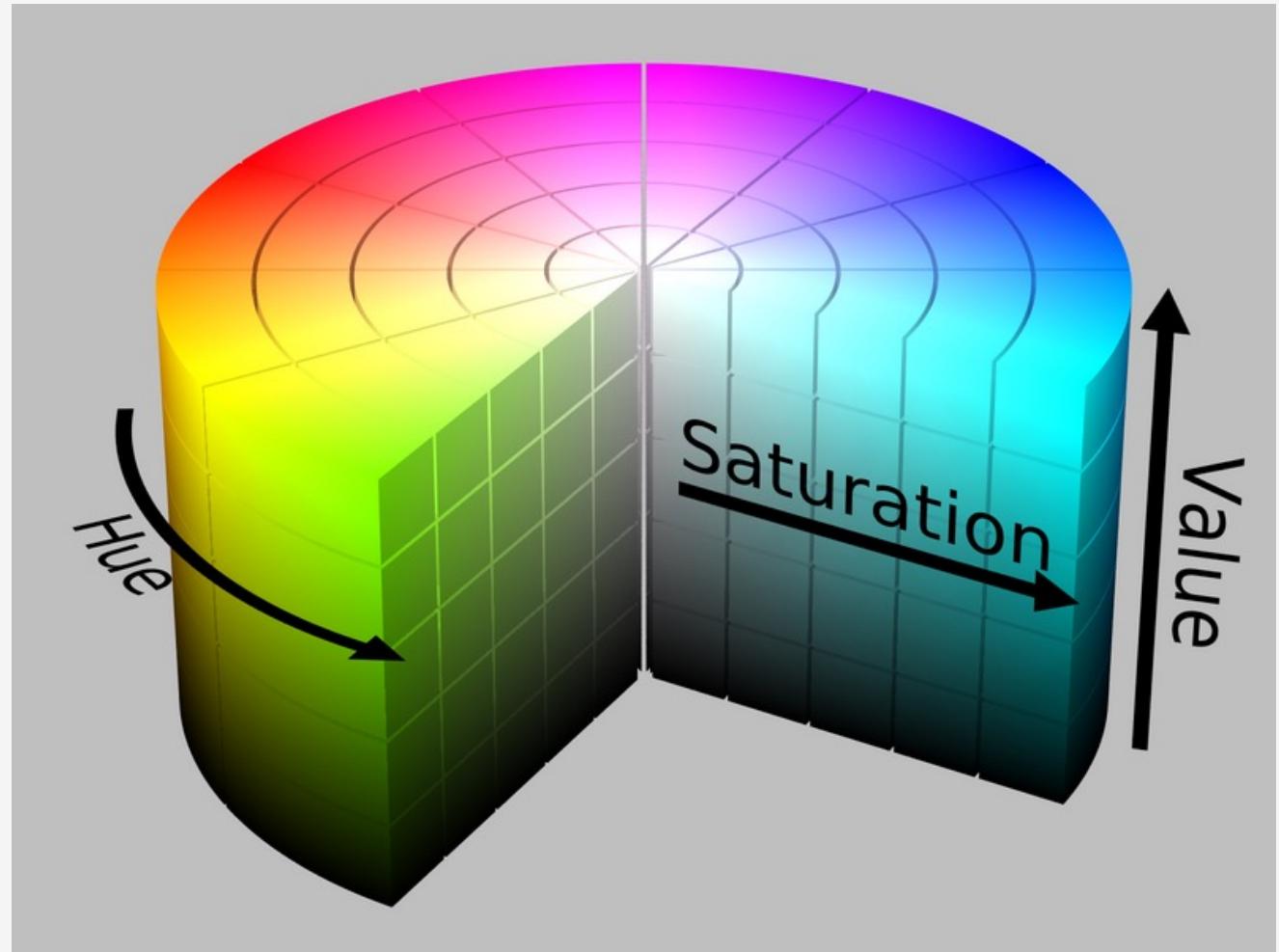


Peter Halasz, CC-SA 3.0, Wikimedia Commons

## 1.6 Saturation and Chroma

On the right is a visualization of color as hue, saturation, and value (HSV), an alternative way of representing color to the RGB intensities that we have so far been working with.

*Saturation* is 0 in the middle of the cylinder (shades of grey) and 1 on the exterior of the cylinder. *Value* is an alternative measurement of brightness. We'll return to hue in the next section.

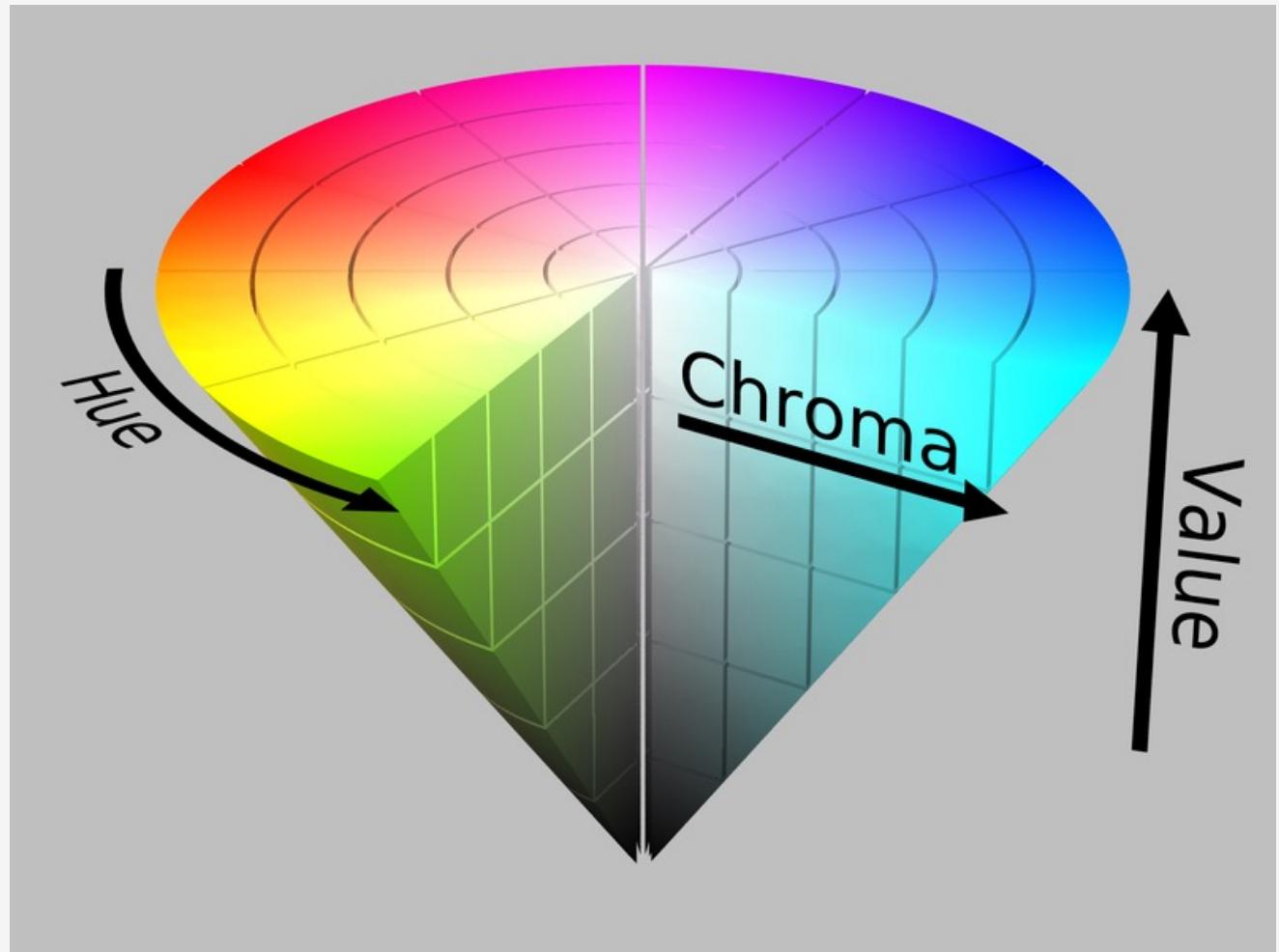


Jacob Rus, CC-SA 3.0, Wikimedia Commons

# 1.6 Saturation and Chroma

*Chroma* is an alternative representation of the saturation of a color. It can be geometrically visualized by pinching the bottom of the HSV cylinder to form a cone.

We work with chroma in our analysis because we think it better represents the richness of a color. The choice between these two measurements is a good example of the kinds of subtle subjective decisions that influence applications of distant viewing and need to constantly be accounted for in our analysis.

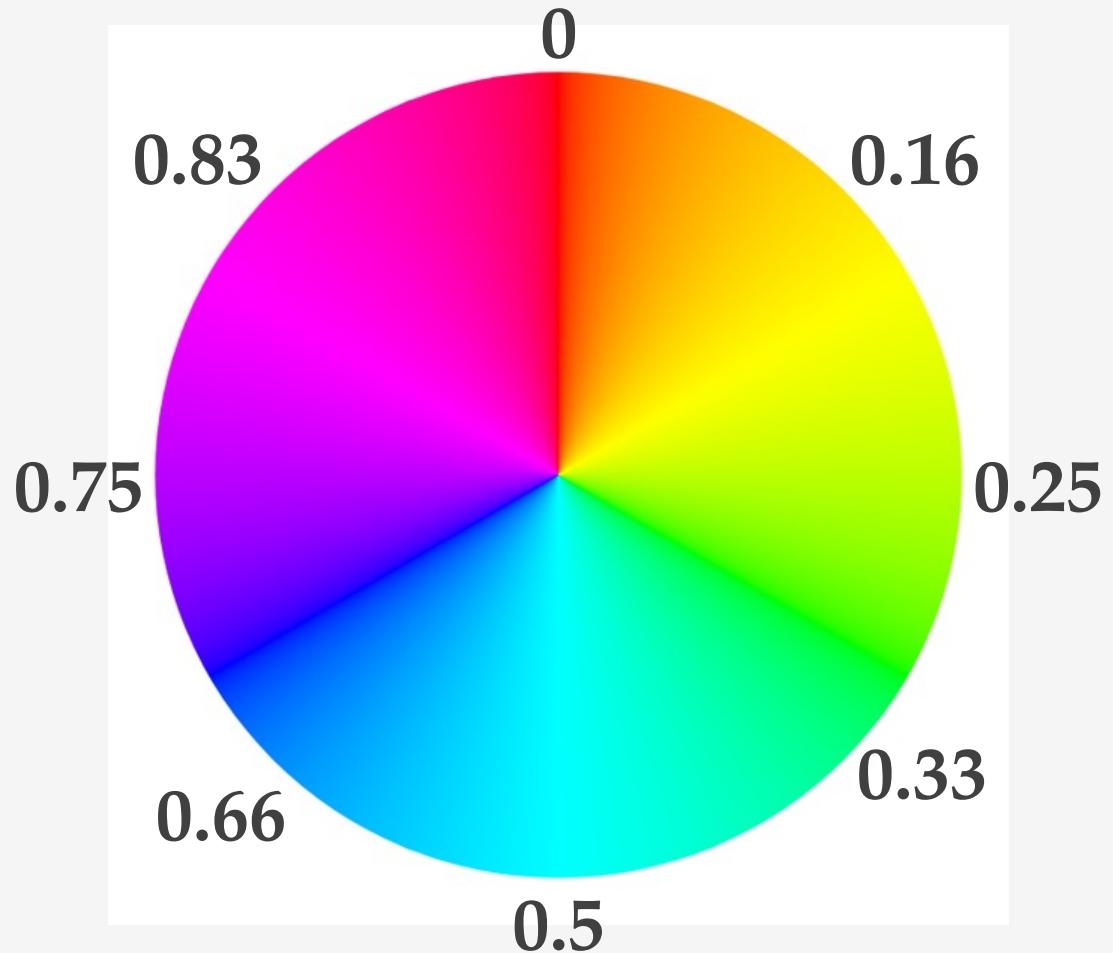


Jacob Rus, CC-SA 3.0, Wikimedia Commons

## 1.7 Dominant Color

The *hue* of a color represents the common name that we would typically give to a color, such as red, green, or purple. We will use a number between 0 and 1 to represent hue, corresponding to the numbers placed on the color wheel on the right.

Notice that hue is a circular measurement, with values close to 0 being a similar shade of red to values near 1. This means that it rarely makes sense to take average values when working with hue.



CC0 1.0 Public Domain, Wikimedia Commons + Added Annotations

# 1.7 Dominant Color

The Python notebook contains code to find the posters that have the most amount of any given dominant color name according to the buckets that we defined. For reference, on the right are the top six posters most associated with each hue.

As we look at the posters and think about cultural significant of color, consider the following questions before looking at the aggregative results:

1. What connections to you expect exist between individual colors and genre? Do you expect specific colors to be frequently indicative of a specific genre?
2. What colors do you think are most commonly used in movie posters? Which colors do you think are the least used?

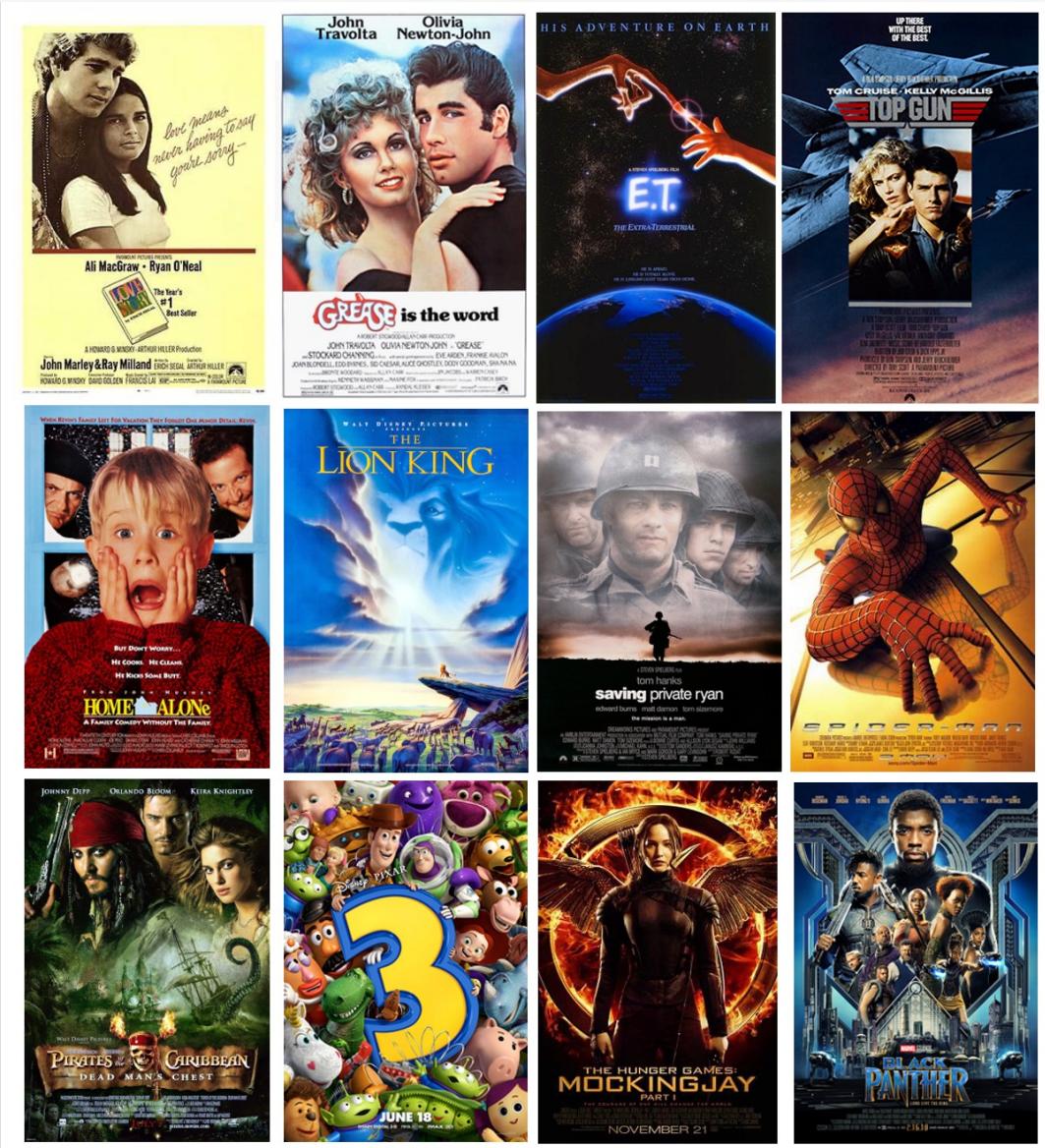
Then, run the code in the Python notebook to see how your hypothesis match the results.



# 1.8 Face Detection

Looking again some of the example movie posters, we see that one common feature of movie posters are images of one or more main characters from the movie. As we did with color, let's start by thinking about some questions about how the faces convey meaning through the movie poster:

1. What kinds of information do you think is being conveyed through use of character images on posters?
2. What kinds of genre-specific patterns do you expect there to be between the number and placement of the faces on the poster?
3. Do you think there are changes over time between the number and size of the faces on the poster?

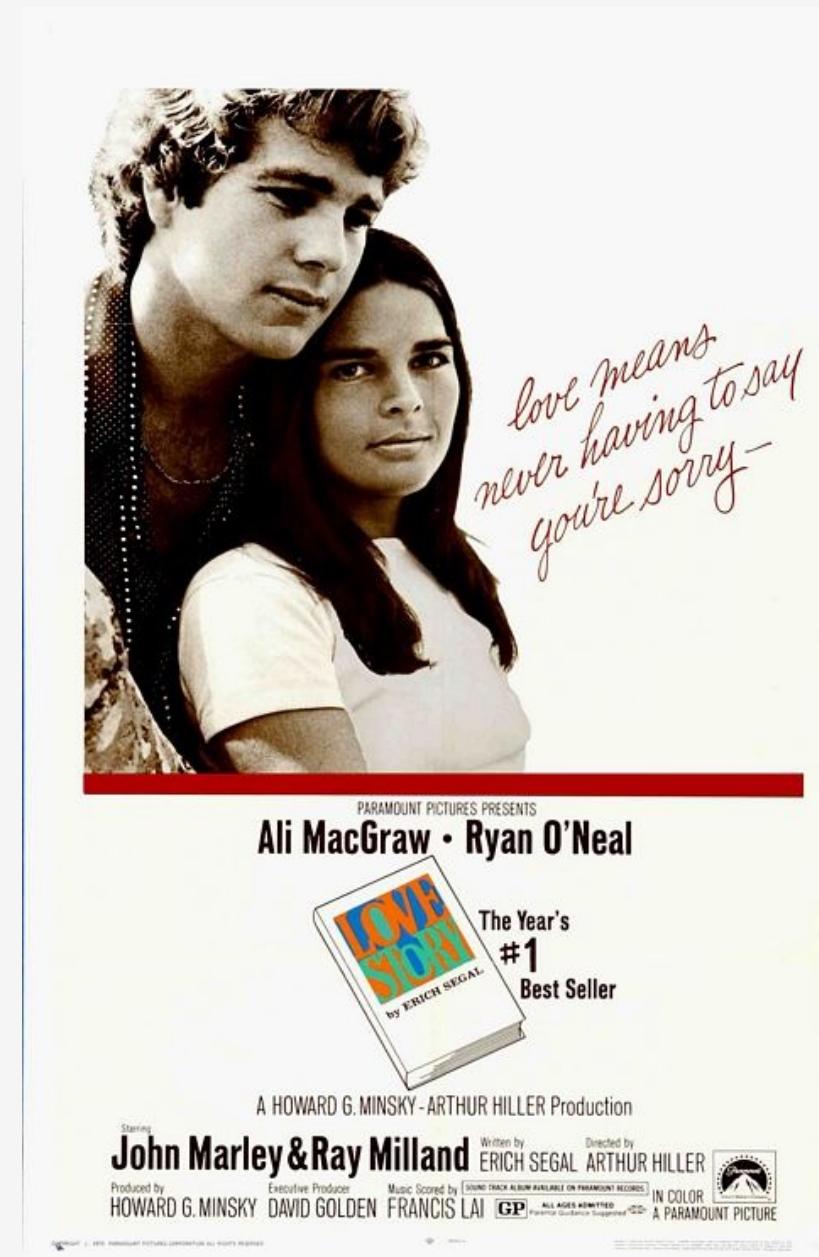


## 1.8 Face Detection

In order to study these questions, we need to create new annotations that show the location of the faces in each poster. Unlike the color analyses that we have worked on so far, the detection of faces cannot be done by simple mathematical summaries of the pixel intensities. A more complex computer vision algorithm is needed.

This is where the distant viewing toolkit (**dvt**) becomes particularly powerful by allowing us to quickly download and apply a modern computer vision algorithm to detect faces in an image.

As shown in the notebook, for example, we are able to find the two faces of the characters shown in the movie poster for the movie *Love Story* (1970).



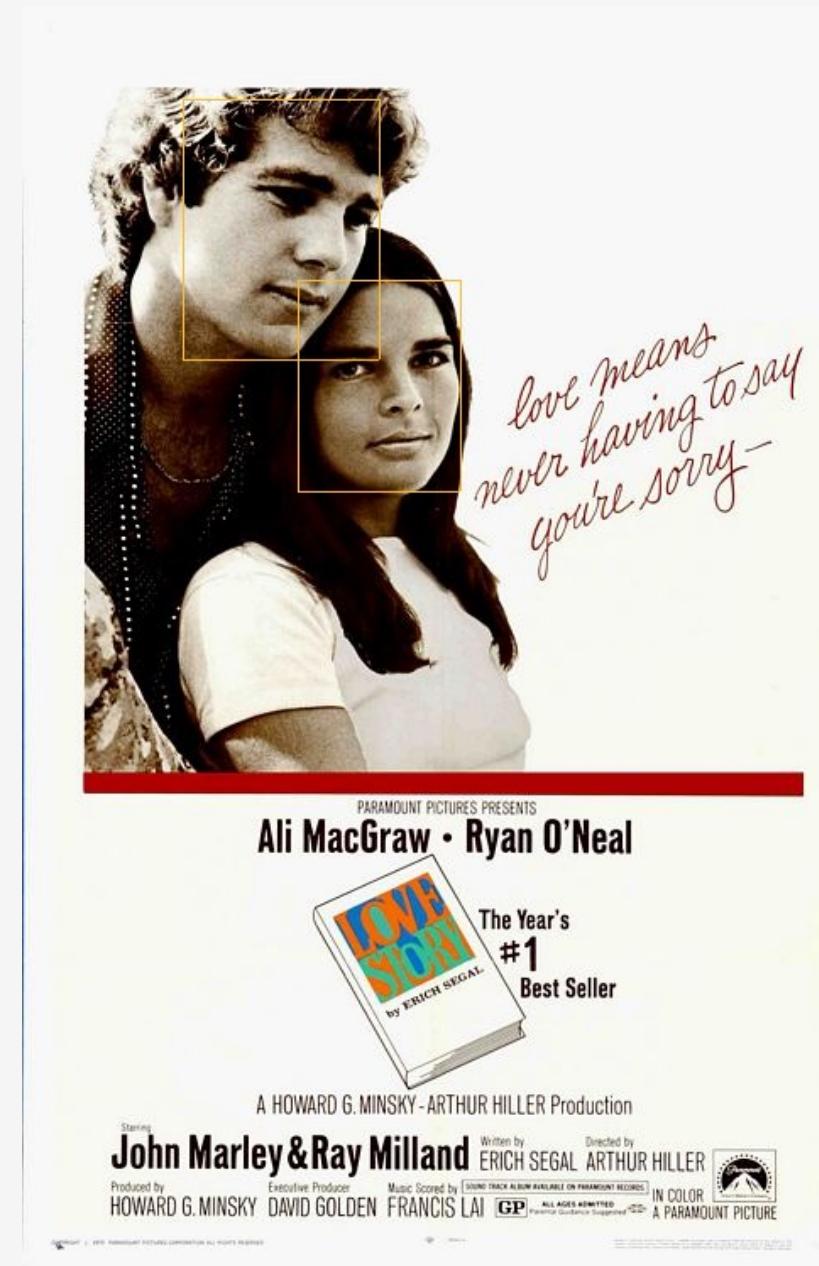
## 1.8 Face Detection

The output of the annotation can be seen visually in the image on the right, which can also be created within the Python notebook.

We also can represent the detected faces as a structured dataset, as shown here:

face_id	x	xend	y	yend	prob
0	0	95	211	55	0.999969
1	1	163	259	162	0.999953

These show the locations of the faces in pixels from the top-left corner of the image along with a probability showing how confident the algorithm is in its determination that it has found a face.



# 1.9 Conclusions and Next Steps

There are many directions to explore after following these notes:

1. To see how to complete the analysis presented here, check out Chapter 3 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. To learn more about the use of computer vision algorithms and how to work with moving images, see the second tutorial, which works through an example using a set of U.S. television shows from the 1960s and 1970s.

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

