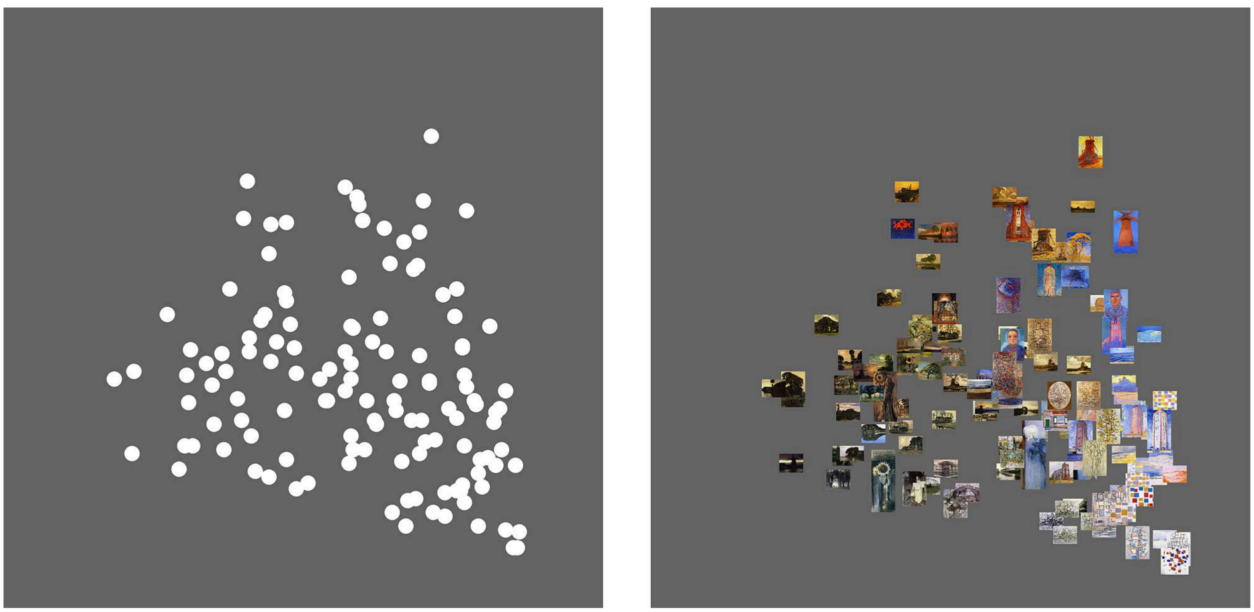
ImagePlot documentation

ABOUT IMAGEPLOT

ImagePlot is a free software tool that visualizes collections of images and video. It is implemented as a macro that runs inside the open source image processing application ImageJ. A copy of the ImageJ is included in the ImagePlot distribution. The ImagePlot macro is a single ASCII text file, so you can easily extend its functionality to meet the needs of your own projects.

ImagePlot creates new types of visualizations not offered by any other application. It displays your data and images as a 2D line graph or a scatter plot, with the images superimposed over data points.



Scatterplot (left) vs ImagePlot (right) of the same data.
127 paintings by Piet Mondrian created between 1905 and 1917.
X-axis = brightness median. Y-axis = saturation median.

An ImagePlot visualization can use any information about the images:

- existing metadata - for example, publication dates or author names.
- new metadata you can add manually - for example, you can use tags to describe images content.
- visual features of the images, which you can measure automatically with the ImageJ macros we provide in the ImagePlot distribution (see Measure Visual Features of Images section).

Visualizations created by ImagePlot are not interactive - if you want to change some aspects of a visualization's appearance, you need to render it again. The advantage is that your can visualize an image collection of any size. A folder with a few dozen images will take a few seconds, one million images can render over night. Your image sources can have any resolution: ImagePlot will automatically scale them while rendering a visualization.

Explore your image collection by creating multiple visualizations which combine this information in different ways. You are likely to discover interesting patterns and relationships.

Using ImagePlot with Video

Because a video is just a sequence of still frames, you can also use ImagePlot to visualize these frames. Usually it is better to only use a selection of keyframes or sample video at a lower frame rate (for instance, 1 frame per second, or 1 frame per shot), and only use this smaller set for visualization.

The next version of this documentation will describe how to do this in detail.

How to Use This Documentation

Use this documentation in a way which suits your learning style:

- To learn how to prepare data from your own images for use in ImagePlot, go to the Preparing Your Data and Images section.

- To learn how to use ImagePlot using the sample image set we provided, go to the Running ImagePlot section and then work through the Tutorial.

Because ImagePlot is a macro which runs from inside the ImageJ application, we recommend that you first spend a few minutes to familiarize yourself with ImageJ interface by checking a number of entries in the Basic Concepts section of ImageJ's online documentation: "Windows," Toolbar," "Status Bar," "Progress Bar" and "Images."

If your computer has more than 2 GB of RAM, we also recommend that you increase the memory setting for imageJ by following instructions in Change ImageJ Default Memory Setting section below.

All visualizations in this documentation are linked to their higher resolution versions in the ImagePlot documentation Flickr set. If you are curious to see all of them now, follow the link to this set.

Typographic Conventions Used in the Documentation

The documentation uses different fonts for particular content types:

- **Arial 11 bold red** is used for the links to software demo videos available on YouTube.

- **Arial 11pp bold** is used for command names.

- `Courier New 11pt` is used for file names and folder names.

- Arial 10 pp is used for additional tips and notes about using ImagePlot.

- *Arial 10 pp italic* is used for the parts of this documentation are more theoretical: the motivations for creating particular visualizations of the sample image set in the tutorial, interpretations of these visualizations, and general discussions of the advantages of exploring image sets via unique visualizations generated by ImagePlot.

Additional Resources

ImagePlot was developed by members of Software Studies Initiative as a part of their ongoing digital humanities research to analyze and visualize large cultural visual data sets.

Visualizations of Cultural Data Sets

Since 2009, we have used ImagePlot in our lab to explore collections of artworks, photos, comics, magazine pages, films, animations, video games, and other media content. Browsing the projects section of our website softwarestudies.com may give you new ideas as to how this tool can help you to explore and find patterns in your own media collections.

Most of 800+ visualizations available on our Flickr account http://www.flickr.com/photos/culturevis/ were created with ImagePlot (many use custom versions with additional code added to to achieve particular effects). Check them to see what ImagePlot can do.

Theory and Methodology Articles

We have also written a number of articles that address the theory and methodology for exploring large visual data sets. The articles also discuss concrete visualizations projects done in our lab using ImagePlot and the image measurement macros.

You can find these in the publications and cultural analytics sections of our website.

A number of these articles are included in the `theory` folder in `ImagePlot.zip` distribution:

`2008.Cultural_Analytics.pdf`
Manovich, Lev. "Cultural Analytics: Visualising Cultural Patterns in the Era of 'More Media.'" *Domus,* March 2009.

`2009.Visualizing_Temporal_Patterns.pdf`
Manovich, Lev and Jeremy Douglass. "Visualizing Change." *Visualizing the 21st Century*, ed. Oliver Grau. MIT Press, forthcoming 2012.

`2010.What_is_Visualization.pdf`
Manovich, Lev. "What is Visualization?" *Visual Studies*. Routledge, 2011.

`2010.Cultural_Analytics_at_Work--Presidential_Online_Video_Ads.pdf`
Zepel, Tara. "Cultural Analytics at Work: The 2008 U.S. Presidential Online Video Ads." *The Video Vortex Reader II*, eds. Geert Lovink and Rachel Somers Miles.

Amsterdam: Institute of Network Cultures, 2011.

`2011.How_To_Compare_One_Million_Images.pdf`
Manovich, Lev, Jeremy Douglass and Tara Zepel. "How to Compare One Million Images?"

We particularly recommend `2011.How_To_Compare_One_Million_Images.pdf` - it discusses a number of ImagePlot visualizations of image sets which range from a few hundred to a one million images.

The online article [Style Space: How to Compare Image Sets and Follow Their Evolution](#) presents more examples of ImagePlot visualizations of different image sets.

Acknowledgments and Credits

ImagePlot was developed by Software Studies Initiative (softwarestudies.com) and supported by a Digital Humanities Start-Up Grant Level II Grant (2010-2011) from the National Endowment for the Humanities, a grant from Andrew Mellon Foundation (2012-2015), the Center for Research in Computing and the Arts (CRCA), and California Institute for Telecommunication and Information (Calit2).

ImagePlot macro: Lev Manovich, Jeremy Douglass, Jay Chow, Matias Giachino, Nadia Xiangfei Zeng.

ImagePlot documentation: Tara Zepel, Lev Manovich, Jeremy Douglass.

ImagePlot documentation testing: William Huber, Eduardo Navas.

ImageJ measurement macros: Lev Manovich.

Vincent van Gogh paintings images, data preparation, initial exploration of the image set: Hernan Higuera, Phuc Duong Tran, Javad Ein Moghassemi.

Visualizations of van Gogh images included in the documentation: Lev Manovich and Tara Zepel.

INSTALLING IMAGEPLOT

Hardware requirements:

a computer running Mac OS, Windows or Lunix;

2 GB RAM (4GB or more recommended for high resolution visualizations with images).

Software requirements:

To use ImagePlot macro, you need the free open-source ImageJ application. ImageJ will run on Mac, Windows, and Linux. Application files for all three systems are included in the `ImagePlot.zip` file archive.

The file archive also contains `ImagePlot.txt` macro, ImageJ sample data sets, as well as theory and methodology articles by Software Studies Initiative. After you download and unzip it, you will have everything you need to start running ImagePlot.

32-bit versus 64-bit versions of ImageJ:

ImageJ distribution includes versions for PC, Linux, and two versions for Mac: ImageJ.app (32-bit version) and ImageJ64.app (64-bit version). 32-bit version can use the maximum of 1.7 GB of RAM; 64-bit version can use up to 70% of RAM in your computer (for example, 5.6 GB if the computer has 8 GB of RAM). We recommend using Image64 because it is faster and because you can give it more memory (see ImageJ installation documentation on how to do this).

If you are running ImageJ on Mac, follow these instructions on how to increase memory allocation.

If you are running ImageJ on Windows or Linux, follow these instructions on how to increase memory allocation.

RUNNING IMAGEPLOT

Follow these steps to open and run the `ImagePlot.txt` macro in ImageJ.

1. Start the ImageJ application by double clicking on one of these files located inside the `ImageJ` folder:

   If you are on Mac OS X, use `ImageJ64.app`.

   If you are on a PC running Windows, use `ImageJ.exe`.

   If you are on a computer running Linux, use `run`.

   Note: If the included version of ImageJ will not run on your operating system, there are many more versions available from the official ImageJ website:

   http://rsbweb.nih.gov/ij/download.html

   These include 64-bit only versions and versions with built-in Java if it is not pre-installed on your system.

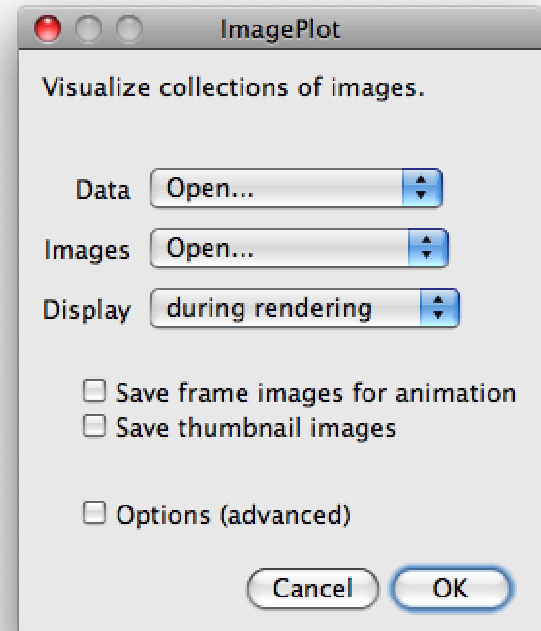
2. Select **File** > **Open** from the top menu bar, and navigate to the `ImagePlot.txt` macro file located inside the `ImagePlot` folder.

3. Choose the `ImagePlot.txt` file. The macro opens in its own window called **ImagePlot.txt.**

4. To run the macro, select **Macros** > **Run Macro** in the ImageJ menu bar. Alternatively, you can press ctrl-r (⌘R on a Mac).

   Note: If you don't see the **Macros** > **Run Macro** menu command, click once inside the **ImagePlot.txt** macro window to activate it.

5. You will see the first ImagePlot dialogue box.



Now you're ready to start exploring!

Follow the step-by-step tutorial in the next section to learn how to create visualizations using the provided sample image set.

TUTORIAL: VISUALIZE AN IMAGE SET USING DEFAULT SETTINGS

Exploring Sample Image Set with ImagePlot

For this tutorial, you will explore a sample image set which we provide along with the software. The set contains digital images of most of Vincent van Gogh's paintings created between 1881 and 1890. The images which come from public web sites. For further details about this image set, and working with digitized images in general, see Working with Digitized Images section below.

Along with the images, we also provide text files which contains measurements of basic visual characteristics of these images: average brightness and average saturation. These measurements were obtained by running ImageMeasure.txt macro located in `extras` folder on the images.

Note: This macro also measures other visual properties of images. The instructions on its use are below in Measure Visual Properties of Images section.

In this tutorial, you will use ImagePlot to create high resolution visualizations which show images of 776 van Gogh paintings in our set organized according to their creation dates and their basic visual features (properties). This is the key idea behind ImagePlot: to allow us examine patterns in large image sets by visualizing all images using their metadata and various features measured with imageJ or other digital image analysis software. (For a more detailed explanation, see our article `2011.How_To_Compare_One_Million_Images.pdf` distributed with ImagePlot.)

The features can describe images brightness, saturation, colors, line orientations, number and types of shapes, composition, and so on. In some cases we can also use software to automatically detect some content properties For example, if our images contain faces, we can also use software to automatically detect these faces and use this information in visualization. To use our ImageJ macros to measure a number of features of your images, see Measure Visual Features of Images section.

The metadata can come with the images: for example, in our case we know a year and month than most paintings were created (this comes from van Gogh letters to his brother) and also the places where he lived. We can also add additional metadata about images to supplement the existing metadata and features measured automatically with software. For example, we can tag every images in our set as "portrait," "self-portrait," "still-life," "landscape," etc.

In this tutorial, we will only use creation dates and average brightness and average saturation measurements. In a Comparing Multiple Images Sets section below, we will also use information about the places where van Gogh lived.

Being able to see most of the paintings van Gogh created during his life together organized by their visual properties gives us new way to think about his art career. We will able to see how brightness and saturation values across 776 of his paintings change from 1881 until 1890. We will see which paintings follow the general trend, and which stand out as exceptions. We may also discover some "hidden" patterns running through his career which were not visible then his paintings are studies one by one.

There are lots of books and articles about van Gogh. We will also be able to see which existing narratives about his art hold true, and which need to be adjusted. For example, van Gogh painting style is often discussed in relation to the various places where he lived during his life.

Here is how Vincent van Gogh Museum in Amsterdam describes the changes in van Gogh style after he moves to Paris in 1886:

"His palette becomes brighter, his brushwork more broken." "Soon after arriving in Paris, Van Gogh senses how outmoded his dark-hued palette has become... His palette gradually lightens, and his sensitivity to color in the landscape intensifies." (Paris, 1886-1888, vangoghmuseum.nl, accessed July 31, 2011).

Here are some of the descriptions of artist's works created after he moved to Arles in the South of France in 1888:

"Inspired by the bright colors and strong light of Provence, Van Gogh executes painting after painting in his own powerful language. " "Whereas in Paris his works covered a broad range of subjects and techniques, the Arles paintings are consistent in approach, fusing painterly drawing with intensely saturated color." (Arles 1888-1889, vangoghmuseum.nl, accessed July 31, 2011).

What else can we say about the "language" of van Gogh's paintings created in Arles in addition to the oberservation that they are "fusing painterly drawing with intensely saturated color? Visualizing images of the paintings according to their visual features in the context of all his other paintings can help us make such statements more precise.

The Sample Image Set Files

The sample image files and the measurements data files are located in **sample_files** folder.

`Mondrian` folder contains images of 128 paintings by Piet Mondrian created between 1905 and 1917, and the data files containing feature measurements of these images. You can explore these files on your own.

`van_gogh` folder contains images of 776 Vincent van Gogh paintings created between 1881 and 1890, and a number of data files with features and metadata about these images. We will use these files for all how-to parts of this documentation. (For details on van Gogh image set, see WORKING WITH DIGITIZED IMAGES section below.)

Take a second to familiarize yourself with the contents of `van_gogh` folder:

> `van_gogh_images` folder which contains 776 images of van Gogh's paintings;
>
> a number of data files which contain feature measurements and metadata such as dates and places where the artist worked; all files use tab delimited format and have `.txt` extension;
>
> `additional_measurements` folder, which contains data files with additional images measurements (you can explore them on your own once you learn how ImagePlot works).
>
> `filepaths_example` folder, which contains images of van Gogh paintings created in Pariis in 1886-1888, and in Arles in 1888=18899 in separate subfolders (we explain how to visualize multiple image folders in Visualize Images Using File Paths section below.)

Open `van_gogh_data.txt` in any text editor or Excel to see the typical structure of a data file which ImageJ works with:

> "Filename" column contains names of image files - in this case, these are the names of the files in `van_gogh_images` folder.
>
> "Brightness_Median" and "Saturation_Median" are examples of visual features which can be measured with software (see Measure Visual Properties of Images section for instructions on measuring these and other features.)
>
> "Year_month," "Label_Place," and "Title" columns are the examples of metadata which often comes with images. The first is year and month of each van Gogh painting which art historians determined using his letters. To arrive at the values in this column, we used Excel to convert months to numbers 1-12 to decimal numbers 1-10, and then added these converted numbers to years. (To make this process more clear, we provide years and months separately in "year" and "month" column.) "Label_Place" column contains names of the key places where the artist lived and worked; the five labels correspond to typical categories used in discussing and exhibiting van Gogh's works (for instance, the Vincent van Gogh museum also presents an overview of artist career using same five periods.)

Now that you understand van Gogh image set, lets learn how to explore it with ImagePlot.

Visualize Data as Points

First you will use ImagePlot's default settings to visualize data as a scatter plot. Of course, you can also use many other programs to create scatter plots such as Excel; however ImageJ allows you to customize every possible graphical attribute of a plot.

Visualizing data about images with ImagePlot default settings is typically the first step in its exploration. Once you are sure that your data file is organized correctly and ImagePlot can read it, you can then use this data to create visualizations containing images.

In this example, you will plot the median brightness values of the images of 776 van Gogh paintings in relation to the dates of these paintings.

video demo: Thhis software demo video takes you through the steps below.

1. Follow instructions in Running ImagePlot section to open and run the `ImagePlot.txt` macro.

3. Select "Open..." from the **Data** drop down menu.

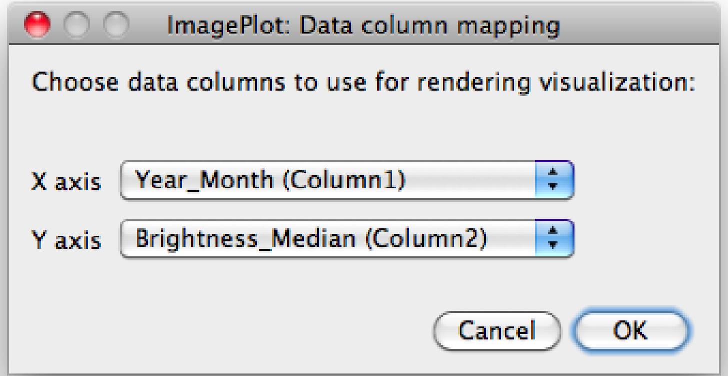
4. Select "None" from the **Images** drop down menu.

5. Click **OK** button.

6. When prompted to load the Data File, choose the file `van_gogh_data.txt` and click **Open** button.

   Note: ImagePlot saves the name and location of the the last data file and image directory you use. If you run the macro again and you want to use the same data file and/or image directory, you can select them in Data and Images down down menus.

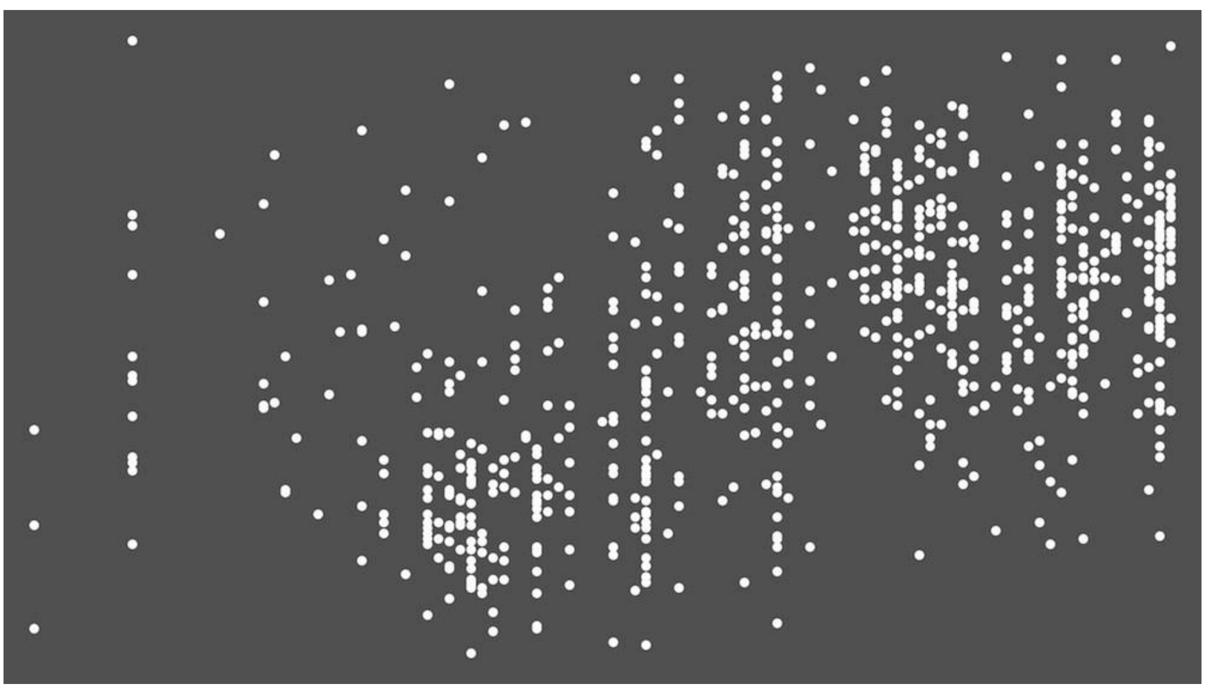
6. Next you will be presented with the dialog box "ImagePlot: Data column mapping." This allows you to select which columns to visualize. Select "Year_Month (Column1)"* for the **X axis** and "Brightness_Median (Column2)" for the **Y axis**.



Tip: By default, ImagePlot selects first two columns for X aixs and Y axis. To speed the process of repeatedly visualizing the same data columns with different graphical options, place these columns in the beginning of your data file.

6. The macro will open a new window which will contain the visualization. You will see the points being quickly drawn following the order of rows in the data file, and in a few seconds, they visualization will be finished. The macro also writes the values of visualization parameters (width, height, background color, points size, etc.) to **Log** window.

Your visualization should look like this:



776 Vincent van Gogh's paintings (1881-1890) plotted as points. X-axis = date (year and month). Y-axis = median brightness.

7. If the resolution of your visualization is larger than the resolution of your computer screen, ImagePlot displays the visualization in a zoomed-out view. To zoom into the the visualization, click on ImageJ toolbox window, and then click on the magnifying

glass icon in the middle. You can then repeatedly click inside the visualization window to zoom to %100 view. Alternatively, you can use ImageJ's **Zoom** command available from its top menu (**Image > Zoom**). You can even zoom in and out visualization while it's being rendered - try it when you are rendering a visualization with images in the next part of the tutorial.

8. To save the finished visualization, you will use ImageJ top down menu. Make sure that the window containing your visualization is active; then select **File > Save As**. Choose one of the available image formats, e.g. PNG; type a filename you want use, and click **Save**.

   Note: If you saving visualizations using jpeg format, by default ImageJ uses 75 quality setting (0 = min, 100 = max). To change this setting, see ImageJ documentation. Use 50 for medium quality, or 100 for maximum quality.

9. When ImagePlot starts renders your visualization, it outputs the values of all options set to Log window. If you like, you can save the contents of this window to a text file so you can have the record of all options used. To do this, select Log window, go to ImageJ top menu, and select **File > Save As**.

Later we will show you how to add axis lines and labels to a visualization, so you will be able to add dates to X-axis. But even without the date labels, we can already make one interesting pattern in this visualization. Since the paintings in our image set were created from 1891 to the middle of 1890, van Gogh's move to Paris in March 1886 corresponds to approximately the middle of the visualization on X-axis. According to the description on Vincent van Gogh museum site, after the artist moves to Paris, "his palette gradually lightens." However, our visualization reveals that this trend already starts earlier in 1885.

Change imageJ Default Memory Setting

The real power of ImagePlot is its ability to render "image plots" - scatter plots which show original images superimposed over the points. Because ImageJ holds its renderings in memory, if you plan to render high resolution visualizations, you need to allocate maximum amount of RAM to the program.

We have changed memory setting in 64-bit ImageJ program version provide with this distribution to 2.8 GB (2800 MB). This is the recommended setting if your computer has 4GB of RAM. If this is the case, skip the rest of this section and go to the next one.

If your computer has more than 4 GB of RAM, you should change ImageJ memory setting to a larger number. If you computer only has only 2GB of RAM, you must change ImageJ memory setting to a 1.3 GB.

if you are using Mac, follow these steps to allocate more memory to ImageJ:

1. Select **Edit > Options > Memory & Threads....** in ImageJ top menu. if this menu is greyed out, click on ImageJ toolbox (it's the narrow horizontal window with icons for various tools.)

2. Type the new memory setting into **Maximum Memory** box. Memory is specified in megabytes. For example, to allow ImageJ to use 3 GB of RAM, type "3000." ImageJ documentation recommends setting the amount of memory for the program at maximum 70% of

the physical RAM. Thus, if your computer has 8 GB RAM, change the settings to “5600. For more details, see ImageJ documentation.

3. Restart ImageJ. ImageJ will save the new setting and will use it every time from now when you run it, so you don’t need to change it again - unless you put more RAM into your computer. If you want to use ImagePlot on a different computer, and you download ImageJ program from its web site, you will need to change its memory setting before creating high resolution visualizations using the same steps.

if you are using Windows or Linux, follow these instructions to allocate more memory to ImageJ.

Visualize Images

To turn our visualization into an “image plot” is quite easy. In this example you will use the same data and organize it in the same way: X-axis: median brightness, Y-axis: dates. However, this time we will replace the points with the scaled versions of the images of van Gogh’s paintings.

video demo: This software demo video shows the steps below.

1. Run the `ImagePlot.txt` macro.

2. Because the data file for this example is the same as that used for the last, select the “../van_gogh_data.txt” from the **Data** drop down menu.

3. Select “Open...” from the **Images** drop down menu. Click **OK**.

4. When prompted to load the Image Files, select `van_gogh_images` folder.

5. ImagePlot will next display “ImagePlot: Data column mapping” selection box. Because you selected “Open..” in **Images** drop down menu earlier, the selection box now contains a drop down menu for **Image filename**. Choose “Filename (Column0)” from this menu.

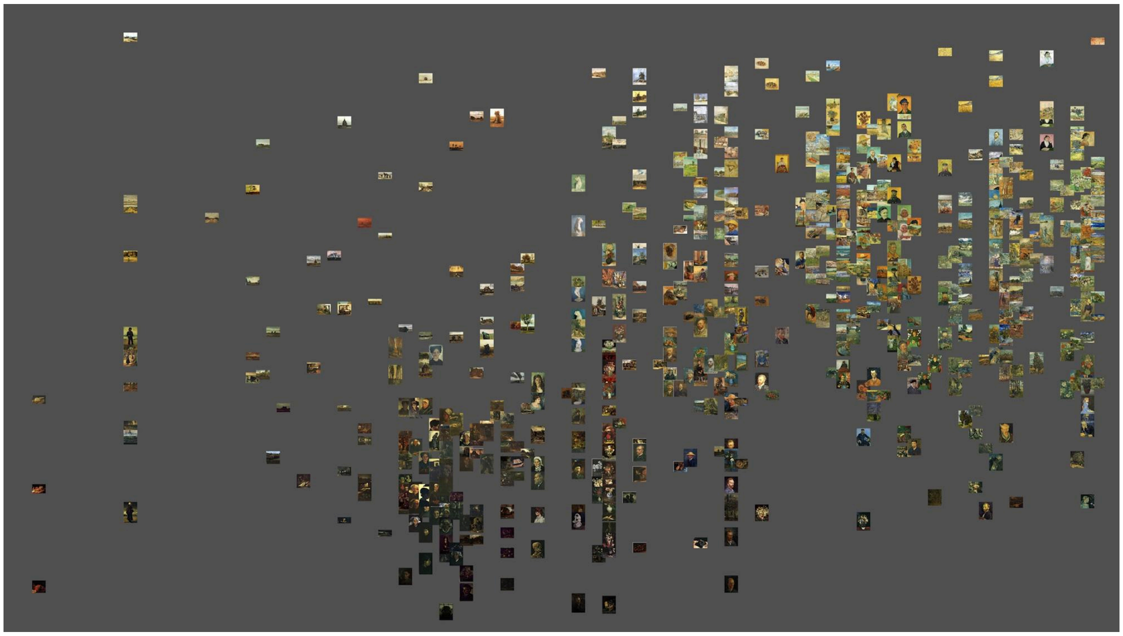
   Note: you can use any label instead of “Filename” for the column containing image files names; we choose this label to make sample data files more readable.

6. Select “Year_Month (Column6)” for the **X axis** and “Brightness_Median (Column7)” for the **Y axis**, as in the previous example. Click **OK**.

7. The macro will start running, and you will see images being added to the visualization. Because ImagePlot now has to manipulate much more data - every pixel in every image - the progress is slower. ImageJ shows this progress in real-time via the **progress bar** displayed in in the lower right hand corner of the toolbox window.

8. When the **progress bar** disappears, this means that visualization is finished. Go to the **File** menu in ImageJ and click on **Save** the visualization.

Your finished visualization should look like this:



776 Vincent van Gogh paintings (1881-1890).
X-axis = date (year and month).
Y-axis = median brightness.

Our first two visualizations show the changes in van Gogh's paintings brightness values over time. The same principle can be used to visualize any other sets of artifacts which have a temporal dimension. Here are some examples (links lead to high resolution visualizations on Flickr):

342 sequential pages of the web comic Freakanglels (www.freakangels.com) published over 15 months. X-axis: page publication order. Y-axis: mean brightness values.

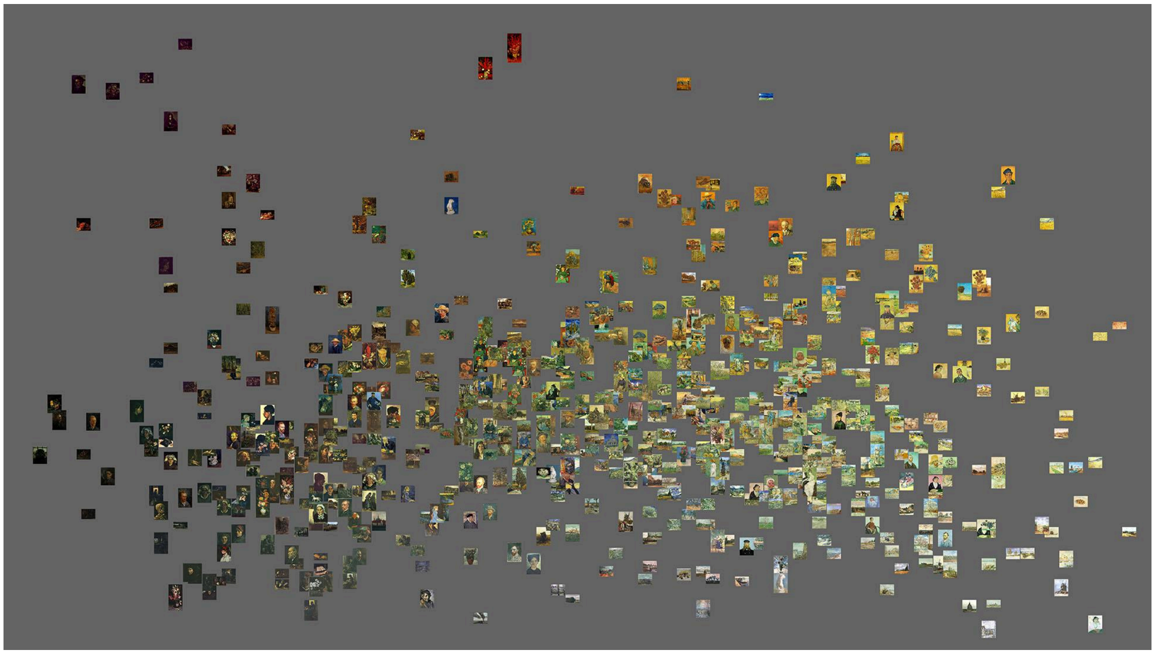
4535 covers of all Time magazine issues published from 1923 to 2009. X-axis: Publication order. Y-axis: brightness mean (for black and white covers), saturation mean for color covers.

We can also use ImagePlot to compare all images in a set to each other using two visual features at the same time. One feature is mapped to X-axis, another is to Y-axis. To illustrate this, let's create a new visualization where the brightness values of all van Gogh paintings we have are mapped to X-axis, and the saturation values are mapped to Y-axis.

1. Run the `ImagePlot.txt` macro again and select the "../van_gogh_data.txt" for **Data** and "../van_gogh_images" for **Images.**

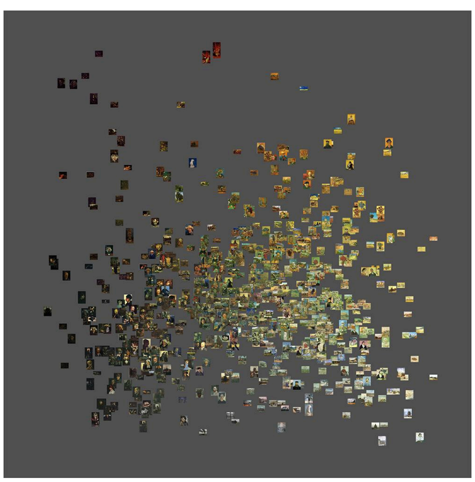
2. In the next dialog box, select "Filename (Column0)" from the **Image Filename** drop down menu, "Brightness_Median (Column2)" for the **X axis**, and "Saturation_Median (Column3)" for the **Y axis**.

3. After ImagePlot finishes rendering the visualization, it should look like this:



776 Vincent van Gogh paintings (1881-1890).
X-axis = median brightness.
Y-axis = median saturation.

Note: Measurements of brightness and saturation fall within 0-255 range. When plotting two variables which have the same scale, it is best to make the width and height of the visualization the same. To learn how to change the default dimensions of the canvas, see Canvas section below. If you visualize the data in this example using the same width and height, your visualization will look like this:



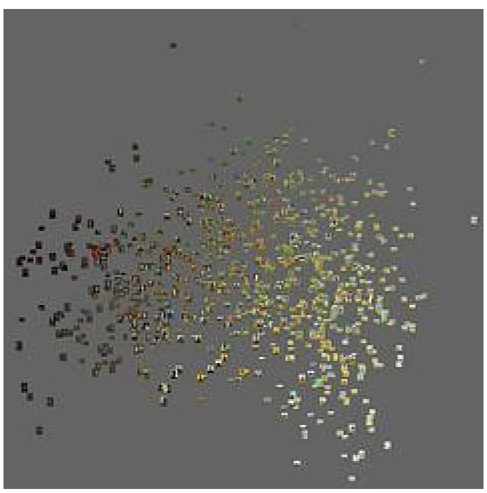
We use the term “style space” to refer to a 2D coordinate space where each cultural artifact is represented by its properties (features) - such as average brightness and saturation in this case. Of course, these two features do not cover all aspects of of van Gogh paintings; if we want to characterize his paintings styles more fully, we will need to create a number of such representations using different combinations of features. (For a detailed discussion of style space and additional examples, see Manovich, Style Space: How to Compare Image Sets and Follow Their Evolution.)

On the average brightness dimension (X-axis) in the visualization above, van Gogh’s earlier (1881-1885) dark paintings occupy the left part; the paintings created between 1885 and 1890 *occupy the center and right part. On the average saturation dimension (Y-axis), his mature paintings occupy the lover part, and even more saturated Arles paintings still rarely end in the upper part. This is a surprising finding given the museum characterization of these painting as having "intensely saturated color."*

By default, imagePlot uses the minimum and the maximum data values to set the range on X-axis and Y-axis. In other words, if the minimum value of median brightness is 17, min of X-axis will be set to 17. If you want to create a "style space" visualization which show where your image set lies in relation to all possible images (according to particular visual features used), you need to change this default. Use Range option to set X-axis and Y-axis start and end values to the min and max values of the features used. For instance, if you use ImageMeasure.txt macro to measure visual features, use 0 as X-axis start. value and Y-axis start value, and 255 as X-axis end and Y-axis end value. Use these settings to create a new visualization will show the parts of the space occupied by 776 van Gogh paintings in relation to all possible brightness and saturation values: from very dark to very light and from no color to pure color.

To see examples of style spaces by Piet Mondrian and Mark Rothko, check our MondrianRothko.viz project. To see a visualization of a style space of one million manga pages, visit Manga.viz project.

In addition to using two features to create 2D image plots, we can also use multivariate statistics methods such as Principal Component Analysis (PCA), cluster analysis, etc., and also dimension reduction techniques to translate many features into more compact representations. For example, the following visualization uses PCA calculated over 400 features extracted from van Gogh images. While on the first glance it may look similar to brightness/saturation image plot above, a closer look reveals that it positions images in terms of their visually similarity in a more precise way.



Visualize Images Using File Paths

The previous examples used a single folder which contained all images in a set. ImagePlot also supports visualizing images located in different places on your computer. To do this, you need to add a column to your data files containing full file path for each image, and then select the appropriate option when running ImagePlot.

This feature of ImagePlot is especially useful for very large data sets that may be spread across multiple directories. For example, our visualization of 1,074,790 manga pages which uses data located in 40,000 separate folders - one for every chapter of 883 manga tiles.

To illustrate file paths option, ImagePlot distribution includes a **filepaths_example** folder. The folder contains a data file `filepaths_data.txt` and two folders with images:

> `van_gogh_images_Paris`: van Gogh's paintings created in Paris in 1886-1888;
> `van_gogh_images_Arles`: van Gogh's paintings created in Arles in 1888.

Before you can visualize images in these two folders together, you need to change the file paths in the data file to reflect the directory structure of your own computer. Open `van_gogh_data_filepaths.txt` in Excel or a text editor, and replace all occurrences of "/Volumes/SWS02/projects/ImagePlot-Release/" with the actual path to **filepaths_example** on your computer.

For example, lets assume that you copied ImagePlot distribution files to your computer to "Documents" folder located at the root of your home drive. The path will look like this:

> /Documents/ImagePlot/sample_files/van_gogh/filepaths_example/

Replace all occurences of "/Volumes/SWS02/projects/ImagePlot-Release/" in `filepaths_data.txt` file with "/Documents/ImagePlot/".

After this replacement, the entries in "Filepath" column will look like this (only first three rows are shown):

> /Documents/ImagePlot/sample_files/van_gogh/filepaths_example/sample_Paris/197.jpg
> /Documents/ImagePlot/sample_files/van_gogh/filepaths_example/sample_Paris/198.jpg
> /Documents/ImagePlot/sample_files/van_gogh/filepaths_example/sample_Paris/199.jpg

After you correct the file paths, follow these steps to visualize images from both folders:

1. Run the `ImagePlot.txt` macro.
2. Select "Open…" from the **Data** drop down menu.
3. Select "Paths in datafile" option from the **Images** drop down menu. Press **OK**.
4. When prompted to load the Data File, select `filepaths_data.txt`.
5. When "ImagePlot: Data column mapping" selection box appears, Choose "Filename (Column0)" from this menu.
6. Select "Year_Month (Column6)" for the **X axis** and "Brightness_Median (Column7)" for the **Y axis**.
7. The macro will render a visualization which will contain images from both `van_gogh_images_Paris` and `van_gogh_images_Arles` folders.

Change the Display Mode

By default, ImagePlot displays visualization while it is being rendered. You can change this behavior by selecting a different option from **Display** pull-down menu in ImagePlot first dialog box. Two options are available:

Display > during rendering

This is the option selected by default. ImagePlot creates a new window which will contain visualization, and updates this window after its adds each new point or image.

Note: if you check **Save images for animation** option, you also need to use **rendering display** option.

Display > when finished

If you select "when finished" option, a log dialogue will open but the completed visualization will only displayed after the rendering is complete. In this mode, ImagePlot uses less RAM. If you get an error message when creating a high resolution visualization, try using this option.

Although you can't see visualization while it is being created, you can follow ImagePlot progress via its progress bar.

Generate Image Thumbnails

When you run ImagePlot with images option on, the macro spends most of its time resizing your source images to the size specified in Image size option. (This option is described in images section in Advanced Options part below.)

To avoid having ImagePlot scale your images every time you create a visualization, you can select **Save thumbnail images** option the first time you run the macro. The macro will scale down your images to place them in the visualization, and save these scaled versions into a new folder. The next time the macro is run, the thumbnail folder will be listed in the **Images** drop down menu. Selecting this folder will significantly speed up the render process since the macro will not have to resize images.

When you select **Save thumbnail images** option, the macro will prompt you to select a folder where to save the scaled down versions of your images. The new images are saved in JPEG format. The size of the scaled down images is determined by **Thumbnail width** setting; the default is 100 pixels. See **Advanced Options > Images** section below on how to change this setting.

You can also use ImageJ built-in **Convert** command to scale any folder of images at any time without running ImagePlot. To do this, select **Process > Batch > Convert** from ImageJ top menu. The command can also change image format. For details on how to use this command, see ImageJ documentation.

Tip: To create best looking resized images for use in visualizations for publications and exhibitions, we recommend using Photoshop rather than ImageJ. Photoshop's **Image Size..** commands allows you to chose between five different algorithms for resizing images; bicubic options create better results than ImageJ's algorithm for resizing. To resize all images in a folder with Photoshop, open the program and use **File > Scripts > Image Processor.**

New features in ImagePlot 1.1:

Each new visualization is given automatically generated meaningful unique filename. It includes the names of data file and the data columns used for x-axis and y-axis.

Option to automatically save the visualization after it have been rendered (appears in the first application dialog box).

Option to render the visualization using a better resize algorithm (runs slower but generates nicer images; the option appears in the Image Options dialog box). See **Images** in **Advanced Options** below.

File open error checking: if ImagePlot can't find a particular image, the filename is printed in the Log window, but rendering continues.

ADVANCED OPTIONS

Overview

ImagePlot allows you to configure dozens of different settings to customize the appearance of your visualizations. To allow you quickly explore image collections without having to go through many options every time, the option to configure these settings is checked off by default. To enable their configuration, select the **Options** checkbox by clicking its checkbox in the ImagePlot first dialogue box.

The options are organized into groups. Each group is configured through its own menu: **Canvas, Points, Lines, Labels, Images, Axis and Range**.

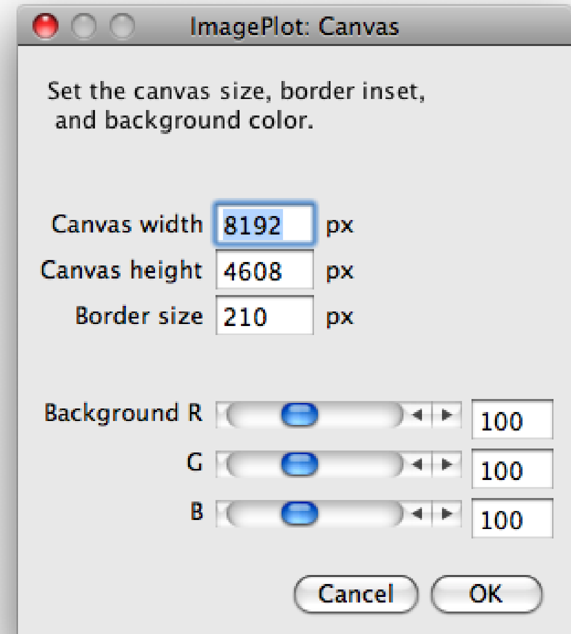
When you select **Options** and click OK, you will first be prompted to select the Data File and Image Files (if applicable). After you made these selections, you will be presented with an Options dialogue box where you can choose which options you want to configure. For example, you want to change the size of the visualization and the background color, check **Canvas**. If you want to change the size of points, check **Points**. To enable rendering lines which connect points and control their appearance, check **Lines**. To change canvas, points and lines options, select all three.

The options available in each menu are described below.

> Note: when ImagePlot starts rendering your visualization, it outputs the values of all options settings to Log window. You can save the contents of this window to a text file so you can have the record of all options used. To do this, select Log window, go to ImageJ top menu, and select **File > Save As**.

video demo: this software demo video shows using some of the options to render a visualization of a sample image set.

Canvas



- Change the size of the visualization you want to render by adjusting the **Canvas width** and **Canvas height**.

- **Border size** specifies the distance between X and Y axis and the the edge of the visualization. If your visualization will include images, we recommend setting this number to at least twice the size of your image thumbnail size to avoid the possibility of any images being cut off. Since the default thumbnail size is 100 pixels, the default border size is set to a little over twice this number - 210 pixels.

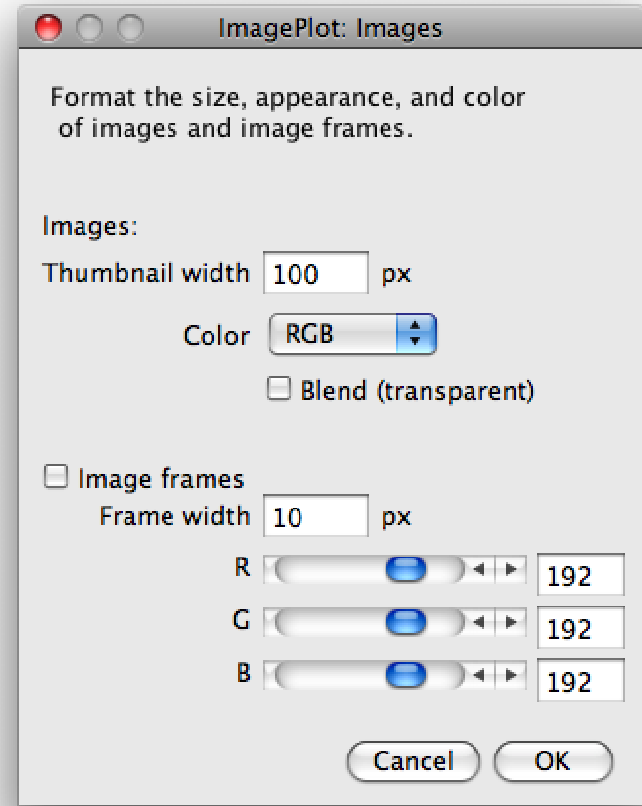
- **Background** RGB values allow you to change the color of visualization background. Each value can range between 0 and 255. The default setting (100, 100, 100) corresponds to a neutral gray.

Here are the examples of values to enter to get particular colors, listed in RGB order:

black:	0, 0, 0
grey:	127, 127, 127
white:	255, 255, 255
blue:	0, 0, 255

Tip: Changing background color of a visualization can often make a big difference in seeing the patterns in your image set. If you images are in color, experiment with generating visualizations with black, gray, and white background to see which works best.

Images



- Change the size of images in a visualization by adjusting the **Thumbnail width**.

  Note: **Thumbnail width** setting controls both the size of images in a visualization, and the size of thumbnails that will be saved if you select **Save thumbnail images** option in the first ImagePlot dialog box.

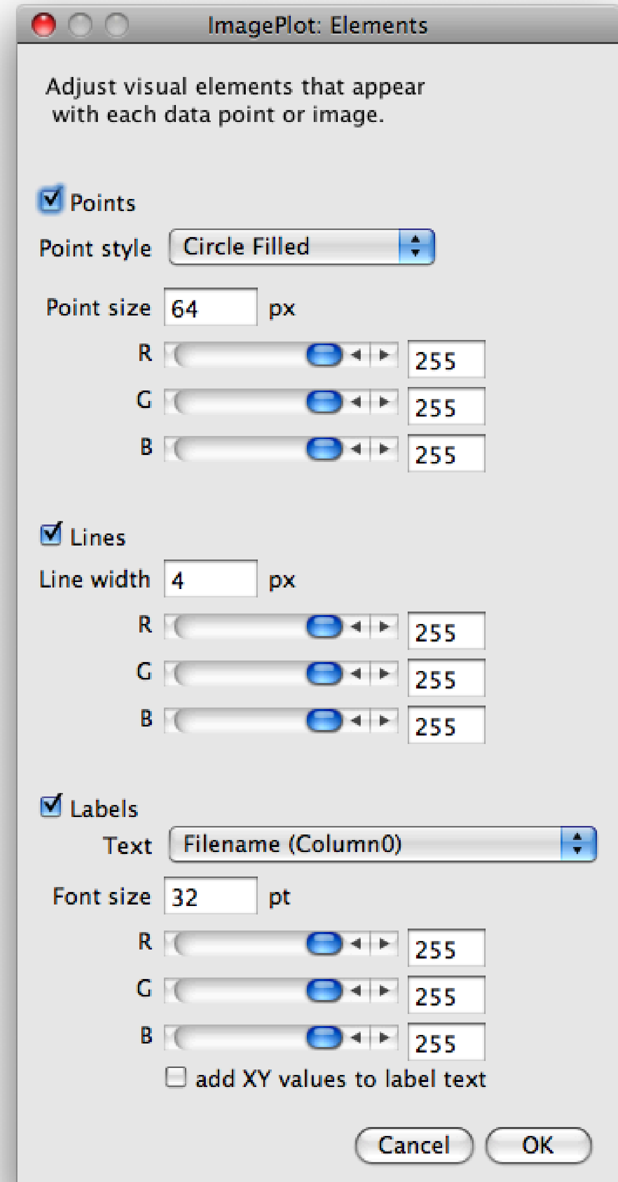
- **Color** allows you to choose between creating a full-color ("RGB") and grayscale visualization. The color visualization will take up four times more RAM than gray scale one. Therefore, if your images are black and white, select grayscale option to be able to render higher resolution visualization.

- If **Blend** is selected, images are rendered with %50 degree transparency. Note that this setting applies to all images, so if visualization's background is black, their will look darker than originals. If visualization's background is white, all images will look lighter than the originals.

- To render frames around the images, check **Image frames** box. Adjust width and color of the frames with **Frame width** and **RGB** settings.

- **New in ImagePlot 1.1:** By default, ImagePlot uses faster but less accurate algorithm to resize images. If you want to use another algorithm which runs slower but generates better resized versions, select **Smooth Version**.

Points/Lines/Labels



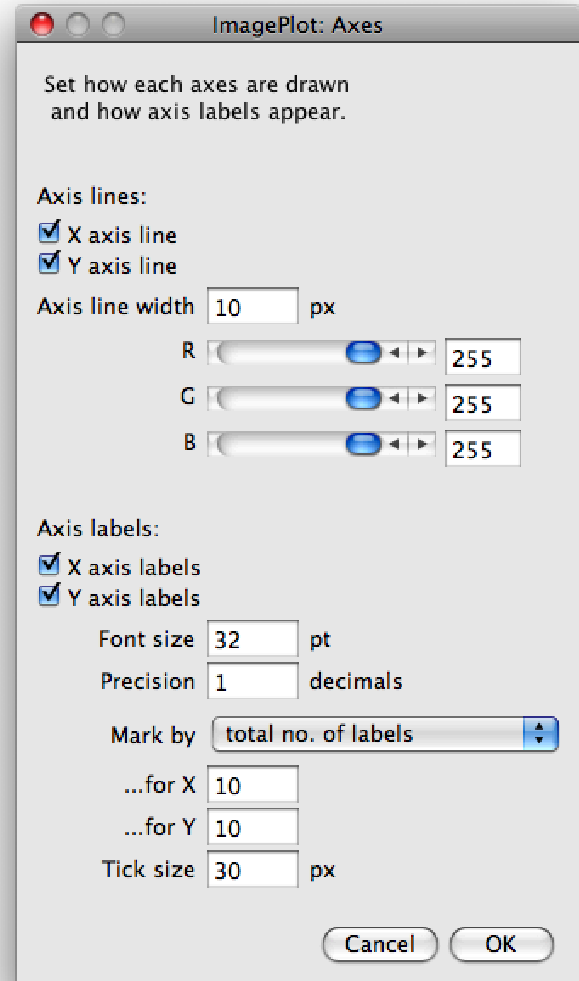
- Modify the style, size and color of points by adjusting the options under **Points** checkbox.

- Checking **Lines** instructs ImagePlot to render lines between the points in a visualization. Modify the style and color of lines by adjusting the options under **Lines** checkbox.

  Note: **Points** and **Lines** operate independently. You can check either one, or both.

- **Labels** add labels and/or XY values next to points or images in a visualization. Select a column to use as labels from the **Text** drop down menu. Modify the style and color of the label text by adjusting the options under the **Labels** checkbox. You can select text labels, XY coordinates, or both together.

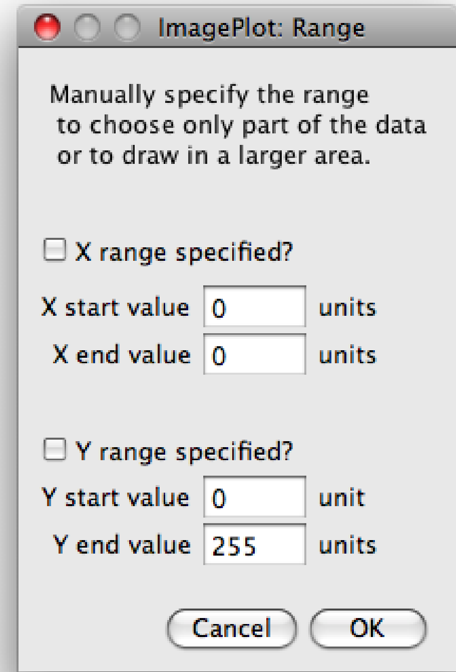
Axes



- Use the options under **Axis labels** to add the labels, and modify their size and numerical precision.

- You can also control how the labels are spaced out. Select **total no. of labels** from the **Mark by** drop down menu to specify the number of label divisions. Alternatively, select **distance between labels** to control the distance between labels. Adjust the **Tick size** to the desired size.

Range



- By default, the start and the end values of X and Y axes are set to the min and max of the data being visualized. You can override this default behavior by manually specifying start and end values for X, Y, or both. To do this, select **X range specified?** and/or **Y range specified?** options, and then enter the desired numbers into **start value** and **end value** boxes.

- These controls can be used in a number of ways. You can limit your visualization to a part of the data. For instance, `van_gogh_data.txt` data file provides measurements of brightness and saturation of 776 van Gogh paintings created between 1881 and 1890. To visualize only the images of paintings created in Paris according to their brightness values, select Year_Month column for X, and Brightness_Median as Y when you run ImagePlot, and also check **Range** menu. When you get to this menu, enter 1886.2502 as X start and 1888.0834 as X end.

  In other situations, it is desirable to set the start and/or the end values larger than the min and max of a data column. For example, to start X axis labels in the last example at 1880.0, enter this number into **X start value** box.

  Setting ranges manually is also important if you want to compare two or more image sets. For example, let's say you want to compare van Gogh's paintings created in Paris and Arles according to their average brightness and average saturation values. If you don't set X and Y ranges manually, each visualization will have its own start and end values determined by min and max values of the data. Therefore the two visualizations will not be aligned. To prevent this, enter 0 as start value and 255 as end value for both X and Y for each visualization. The resulting visualizations will have exactly the same scale.

PREPARING YOUR DATA AND IMAGES

ImagePlot works with the most common data formats for visual exploration: a set of image files in any of the popular formats (.jpg, .png, etc.) and the data about these images saved in a tab-delimited text file (.txt).

If you want to use ImagePlot to generate line graphs and scatter plots without images, you only need a data file.

To include images in your visualization, they need to be on the same computer as the ImagePlot macro. The images can be located in a single or multiple folders.

ImagePlot Data File Format

The data for ImagePlot's visualizations is stored in a external tab-delimited text file (.txt). The file can have any name and any location. When you run ImagePlot, the macro will prompt you for the location of this file. If you like, you can prepare multiple files with different information for the same image set.

> Tip: If you are using Excel to prepare the data, use **File > Save As..** and choose **Tab Delimited Text (.txt)** from format pull-down menu.

File Structure

The information for each image in your collection occupies a single row. The columns contain information about the images - for instance, image file name, author name, date, average image brightness, average image saturation, image width, image height, etc.

> Tip: To create a table containing the filenames of images in a folder, open the folder window, sort the images in the desired order, select all contents, and press "copy"; then open a new worksheet in Excel, select the first cell, and press "paste."

The first row of the data file should contain text labels describing the columns. (ImagePlot will work if you don't include the labels rows but doing including one makes it easier to select the desired data for a specific visualization.)

The data file must have a minimum of two columns: one containing X values, another containing Y values as shown by the highlighted columns in the data file below. The values can be in integer numbers (e.g. 1, 2, ...) or floating point numbers (e.g. 1.11, 1.12, ...)

Filename	X_axis	Y_axis	Addl_Column1	Addl_Column2
xxx.jpg	1	1.11	111	1_xxx
xxx.jpg	2	1.12	222	2_xxx
xxx.jpg	3	1.13	333	3_xxx
xxx.jpg	4	1.14	444	4_xxx
xxx.jpg	5	1.15	555	5_xxx
xxx.jpg	6	1.16	666	6_xxx
xxx.jpg	7	1.17	777	7_xxx
xxx.jpg	8	1.18	888	8_xxx

If you want to include images in your visualizations, you need to add a third column with image file names (if all images are in a single folder) or full image file paths (if images are located across multiple folders).

You can also add as many additional columns as you want. Additional columns can contain numbers (integer or floating point values), text, or a combination of the two.

You can organize columns in a data file in any order. When you run ImagePlot , it will ask you which columns to use as X and Y data coordinates in a visualization. You can choose any column, as long as it contains only numbers in every cell. (If by mistake you run ImagePlot and by mistake you choose a column which contains text in some cells, you will get an error message.)

You can mix integers and floating point numbers in a single column. Any column can be used as a source of X or Y values to position points or images in a visualization.

Text Columns

If you check labels option when running ImagePlot, it will prompt you for a column to use for labels. The labels will be rendered next to points (or images) in your visualization. A column used for labels can contain text (one or a few words), integer numbers or floating point values. A column which will be used for labels can mix these data types.

Text columns are useful for describing content or other properties of your images. Any text file can be used for labels which which are rendered next to the images in a visualization. However, text columns cannot be directly used as X or Y values. If you want to use a column with text tags which you want to use in this way, add an additional column which would contain numbers corresponding to text tags. For example, let's say you are described images content with tags "portrait," "still-life," and "landscape." Add a column where 1 stands for "portrait," 2 stands for "still-life," 3 stands for "landscape," and use this column for X or Y.

You can use characters and numbers in text columns. You can have spaces between words.

Common Errors

If ImagePlot prints an error message when you use with your data file, make sure that your data file does not have any empty cells.

You can't use commas anywhere in your data file as they will cause errors. Periods are

OK.

Here are two examples:

123,45 (wrong)
123.45 (right)

apple, the (wrong)
the apple (right)

Data File Examples

In this section we provide a few examples of how to organize your data files for ImagePlot. (These examples are different versions of the `van_gogh_data.txt` file included with the sample image set in `van_gogh` folder located inside `sample_files` folder.)

The first example has only two columns. Year_Month contains dates of a few van Gogh paintings; Brightness_Median contains measurements of these paintings (only first part of the data file of the complete data is shown). When you run ImagePlot, you can select Year_Month as X, and Brightness_Median as Y. The result will be a visualization which shows the evolution of average brightness in van Gogh paintings over time. (For detailed instructions on working through this example, see Tutorial section below.)

Year_Month	Brightness_Median
1881.834	96
1881.834	23
1881.834	61
1882.5838	240
1882.5838	84
1882.5838	171
1882.5838	86
1882.5838	124
1882.5838	54
1882.5838	153

If you want to include images in your visualization and these images are located in a single folder, include additional column with the filenames. When you run ImagePlot, it will prompt for the location of the image folder; it will then use filenames in the data files to find and load images. Note: The folder can contain additional images or other files; they will be ignored by ImagePlot.

Filename	Year_Month	Brightness_Median
001.jpg	1881.834	96
002.jpg	1881.834	23
003.jpg	1881.834	61
004.jpg	1882.5838	240
005.jpg	1882.5838	84
006.jpg	1882.5838	171
007.jpg	1882.5838	86
008.jpg	1882.5838	124
009.jpg	1882.5838	54
010.jpg	1882.5838	153

If you want to include labels next to points or images in the visualization, add another columns with text or numbers you want to use for labels. In this example, we added a column which contains titles of van Gogh paintings which can be used as labels.

Filename	Year_Month	Brightness_Median	Title
001.jpg	1881.834	96	Still Life with Cabbage and Clogs
002.jpg	1881.834	23	Still Life with Earthenware Bottle and Clogs
003.jpg	1881.834	61	Still Life with Yellow Straw Hat
004.jpg	1882.5838	240	Beach at Scheveningen in Calm Weather
005.jpg	1882.5838	84	Cluster of Old Houses with the New Church in The Hague
006.jpg	1882.5838	171	Dunes
007.jpg	1882.5838	86	Edge of a Wood
008.jpg	1882.5838	124	Fisherman on the Beach
009.jpg	1882.5838	54	Fisherman's Wife on the Beach
010.jpg	1882.5838	153	Girl in the Street Two Coaches in the Background

You can add as many additional columns as you want. In this example, we added one column containing measurements of paintings' average saturation, and a column containing names of places where van Gogh was working (a part of the file corresponding to artist's move from Paris is Arles is shown; titles below are cropped.)

Filename	Year_Month	Brightness_Median	Saturation_Median	Label_Place	Title
393.jpg	1888.0834	169	138	2_Paris	Pork-Butcher's Shop Seen fro
394.jpg	1888.0834	153	103	2_Paris	Self-Portrait in Front of the Ea
395.jpg	1888.0834	210	39	2_Paris	Snowy Landscape with Arles i
396.jpg	1888.1668	163	88	3_Arles	Avenue of Plane Trees near Ar
397.jpg	1888.1668	176	78	3_Arles	Blossoming Almond Branch in
398.jpg	1888.1668	193	73	3_Arles	Blossoming Almond Branch in
399.jpg	1888.1668	148	82	3_Arles	Gleize Bridge over the Vigueir
400.jpg	1888.1668	224	76	3_Arles	Langlois Bridge at Arles with I
401.jpg	1888.1668	169	66	3_Arles	Langlois Bridge at Arles with \
402.jpg	1888.1668	155	115	3_Arles	Orchard in Blossom

Image Formats

ImagePlot will work with any image type supported by ImageJ: TIFF, GIF, JPEG, PNG, BMP, PGM.

Many more formats are supported via ImageJ plugins. They are described in Non-native Supported Formats section of ImageJ documentation.

You can mix different image formats in the same folder. (For instance, you can have some JPEG images, and same PNG images.)

Images which appear in a visualization can be located in a single folder, or in different folders on your computer (see Visualize Images Using File Paths.)

There are no limitations on image sizes. When ImagePlot runs, it will automatically scale images to the default size (100 pixels) or a different user-defined size (see Images section in Advanced Options part of the documentation). If you images are big, most of the rendering time will be spend for scaling these images before they are placed in the visualization. In this case, you may want first to generate scaled down copies of your images using a special ImagePlot option described in Generate Image Thumbnails section, and then use these versions in future visualizations.

Check Sizes of All Images in a Folder

Before you start visualizing a set of images using ImagePlot, it may be useful to display their sizes and proportions. To do this, place your individual images and/or image folders into a single folder and run **ImageSizeReport macro** `(ImageSizeReport.txt)` on this folder. The macro is located in `extras` folder.

The macro processes all images files in a user-specified folder and any subfolders in this folder, saving the results to a tab-delimited text file in the same folder. The file is named `image_dimensions.txt.`

To show its progress, the macro prints the following information to a **Log** window as it runs: image width, image height, image ratio (width/height), and image file path.

To run this macro, follow these steps:

1. Start ImageJ by double clicking on `ImageJ64.app` if you on the Mac, or `ij44-nojre-setup.exe` if you are on the PC running Windows.

2. Select **File** > **Open** from the top menu bar, and navigate to the `ImageSizeReport.txt` macro file located inside the `extras` folder of the distribution.

3. Choose `ImageSizeReport.txt` file. The macro opens in its own window in ImageJ, titled **ImageSizeReport.txt**.

4. To run the macro, select **Macros** > **Run Macro** in ImageJ menu bar. Alternatively, press ctrl-R (Windows) or ⌘R (Mac).

   Note: If you don't see **Macros** > **Run Macro** menu command, click once inside the **ImageSizeReport.txt** macro window to activate it.

5. You will be prompted to select a folder containing images and folders to process. Select the folder which you want to process.

6. The macro will run, outputting its progress to the **Log** window in ImageJ. Once complete, the tab-delimited text file `(image_dimensions.txt)` containing the

information about the images will be save in the folder you selected for processing. The following information is saved for each image: image width (jn pixels), image height (jn pixels), image proportion (height/width), and a file path.

Measure Visual Properties of Images

ImagePlot allows you examine patterns in large image sets by visualizing all images using their metadata and visual characteristics.

ImageJ has a built-in command which can measure a number of grayscale characteristics of all images in a folder. The command is available from the top ImageJ menu: **Process > Batch > Measure.** For details on how to use it, see ImageJ documentation.

We also provide two additional ImageJ macros we wrote to automatically measure a number of other characteristics of images. **ImageMeasure** measures brightness, saturation, and hue. **ImageShapes** counts a number of shapes per image.

The macros files `ImageMeasure.txt` and `ImageShapes.txt` are located in `extras` folder.

The measurements generated by macros are saved in a tab delimited text files with .txt extensions. You can combine these file with other metadata using Excel, other spreadsheet software, or Unix commands, and use the new data file to create visualizations with ImagePlot.

In computer science, the properties of images which can be quantified and measured automatically are often called features. The examples of features include images brightness, saturation, colors, line orientations, number and types of shapes, textures, composition, and so on. In some cases we can also use software to automatically detect certain content properties. For example, if our images contain faces, we can also use software to detect these faces and use this information in a visualization.

Usually we have some metadata about the images we want to visualize. For example, in the case of our sample image set, we know a year and month for most of van Gogh's paintings (this comes from van Gogh's letters to his brother) and also the places where he lived. This metadata can be also used in ImagePlot visualizations. For example, we can plot images according to their dates (X) and average brightess (Y).

We can also add additional metadata about images to supplement the existing metadata and features measured automatically with software. For example, we can tag every images in our set as "portrait," "self-portrait," "still-life," "landscape," etc.

ImageMeasure macro `(ImageMeasure.txt)`

This macro measures basic visual properties of every image in a user-specified folder, saving the results to a tab-delimited text file named `measurements.txt.` The file is saved in the user-specified folder.

The macro file `ImageMeasure.txt` is inside the `extras` folder. To run it, follow the directions provided above for `ImageSizeReport.txt`, but chose `ImageMeasure.txt` instead as the macro file to open and run.

video demo: this software demo video shows using some of the options to render a visualization of a sample image set.

Measuring many images may take some time, especially if they are large. To show its progress, the macro prints the following information to **Log** window as it runs:

- current image number being measured;
- total number of images in the folder;
- current image filename;

This information for each image is printed on its line in this format:

current number / total number   image filename

The macro saves the measurements for each image in `measure.txt` file. The following information is saved:

- image filename;
- imageID (this is a sequential number of the image in the folder);
- brightness_median (average of grayscale values for the pixels in an image);
- brightness_stdev (standard deviation of grayscale values of the pixels in an image);
- saturation_median (average of saturation, i.e. purity of color for each pixel in an image);
- hue_median (average of hue values of all pixels in an image);
- hue_stdev (standard deviation of hue values of the pixels in an image).

> Tip: If you are not familiar with brightness-saturation-hue representation, you can find detailed information in this Wikipedia article: http://en.wikipedia.org/wiki/HSL_and_HSV.
>
> Tip: we use median method of describing central tendency of data because it is less sensitive to outlier values; see http://en.wikipedia.org/wiki/Central_tendency for description of other measures of central tendency. Standard deviation and other measures of data dispersion are described in http://en.wikipedia.org/wiki/Standard_deviation.

For brightness measurements, color images are internally converted grayscale using this formula: gray=(red+green+blue)/3. (For information on how to chose a different formula, see Conversions... in the ImageJ documentation).

All measurements are done on 0-255 scale. For example, if an image is pure black, brightness_median will 0; if an image is pure white, brightness_median will 255.

Low saturation_median value indicates that image colors are mostly desaturated; a high value indicates that most colors are close to being pure (very saturated).

Hue_median is the average of hue values of every pixel. Normally hue values are represented as degrees of a circle, i.e. they range from 0 to 360. `ImageMeasure.txt` macro maps these values to 0-255 scale.

Here is an example of how `measurements.txt` might look:

filename	imageID	brightness_median	brightness_stdev	saturation_median	saturation_stdev	hue_median	hue_stdev
001.jpg	1	96	45.1122	91	45.4321	30	13.1285
002.jpg	2	23	60.9367	170	65.4442	8	86.6207
003.jpg	3	61	78.7791	174	59.9569	12	55.5634
004.jpg	4	240	69.3129	51	53.8758	42	35.0705
005.jpg	5	84	60.7108	57	29.0426	51	64.154
006.jpg	6	171	55.204	157	54.1331	33	6.6025
007.jpg	7	86	42.8795	65	49.4814	37	45.1777

ImageShapes macro (ImageShapes.txt)

The macro uses the ImageJ command "Analyze Particles.." to count the number of shapes in every image in a folder.

To run the macro, follow the the directions provided above for ImageSizeReport.txt, but chose ImageShapes.txt instead as the macro to open and run.

Macro operation:

The macro first converts each image to a binary black and white image, and then counts the number of distinct white areas. (For more details, see Analyze Particles.. section of ImageJ documentation.)

The results are printed to a "Summary" window. To save the results, select all contents of "Summary" window and paste into a spreadsheet or a new text file.

"Count" column contains the number of shapes found in every image (a "shape" is a distinct white areas in a binary black an white version).

Total Area" and "Average" size results depend on the resolution of images - so you should discard them if you are running ImageShapes.txt on images which significantly vary in size.

The results produced by the macro are approximate and depend on the kinds of images being measured. While they do not always accurately reflect the actual number of shapes which you may see in an image, they are useful in comparing a number of images to each other .

To control how the macro analyze your images, edit the text in the macro file:

To increase the minimum size of shapes being counted, edit "size=10-Infinity." Shapes with size (area) outside the range specified in this setting are ignored.

For instance, "size=100-Infinity" will skip shapes with area smaller than 100 pixels.

We recommend setting minimum size in proportion to the resolution of your images. For instance, with small images, very small shapes are relevant to count, but with large images, you may want to skip them.

You can also skip counting shapes larger than a particular size. For example, to skip shapes with areas larger than 400 pixels (but larger than 10 pixels), use "size=10-400".

To make the macro only count more circular shapes, or more rectangular shapes, change "circularity=0.00-1.00" option.

A value of 1.0 indicates a perfect circle. As the value approaches 0.0, it indicates an increasingly elongated polygon. (Note that these values may not not valid for very small particles.)

You can also control what information about the shapes is displayed in "Summary" window than the macro runs. From ImageJ top menu, select **Analyze > Set Measurements..** and choose desired options such as "Perimeter" and "Shape Descriptions" to display more detailed information.

Using Other Digital Image Processing Software

There are other ways to measure your images besides ImageJ built-in commands and our macros. If you are comfortable with writing computer scripts or programming, you can use other digital image processing software and libraries such as Matlab and openCV to measure many other properties of your images.

WORKING WITH IMAGEPLOT

Maximizing Output Resolution

To maximize the size of of your visualization, change the canvas default size setting. ImagePlot can render visualizations up to 2.5 GB in size. (This limit is set in imageJ software.)

You can calculate the maximum possible dimensions for **Canvas Width** and **Canvas Height** using the following formula:

Visualization size = **width** (in pixels) x **Height** (in pixels) x **pixel size in memory** (*)

* Pixel size in memory is 1 byte for 8-bit grayscale images, or 4 bytes for 24-bit color images.

Example 1: color visualization (24-bit image)
25,000 (width) x 25,000 (height) x 4 bytes (memory used for a single pixel = 2,500,000,000 bytes = 2.5 GB

Example 2: grayscale visualization (8-bit image):
100,000 x 25,000 x 1 byte = 2,500,000,0000 bytes = 2.5 GB 8-bit grayscale image

In practice, you often have to use much lower resolution numbers. If you try to render a visualization which approaches 2.5 GB in size, you may get an error message indicating that that ImagePlot run out of memory. If this happens, restart ImageJ to free used memory and then run ImagePlot again, using use smaller numbers for canvas size. If you get an error message again, reduce canvas size again until ImagePlot successfully renders the visualization.

When ImagePlot starts running, it needs to reserve enough memory (RAM) for the visualization. This comes from the total allocation you specify for ImageJ in the program settings (**Edit** > **Options > Memory & Threads.**) You can specify any size up to 70% of the physical RAM of your computer.

For example, if your computer has 4 GB of RAM, you can allocate 2.8 GB to ImageJ. This would allow ImagePlot to use maximum 2.5 GB to keep visualization in memory while it's being rendered. However, if your computer only has 3 GB of RAM, you can only allocate 1.7 GB to ImagePlot; consequently the macro will have less memory for storing visualization. In this case, limit your visualization size to 1.5 GB. (Remember that 32-bit version of ImageJ for Mac can only use up to 1.7 GB of RAM, while 64-bit version does not have this limit. Thus, if you want to allocate 2.8 GB of RAM, make sure to use 64-bit version of the program on the Mac.)

When you are rendering a very high resolution visualization, it is best not to multi-task. Let ImagePlot finish rendering before switching to another program. This allows ImageJ to use maximum available memory.

Comparing Multiple Image Sets

Creating visualizations of related image sets and placing them next to each other allows you to compare multiple images sets in a single glance. To illustrate this, we will visualize separately van Gogh's Paris and Arles paintings using the same data columns: Brighhtness_Median for X-axis, and Saturation_Median for Y-axis).

Follow these steps to first visualize Paris painting, and visualize Arles paintings:

1. Run the `ImagePlot.txt` macro.
2. Select "Open..." from the **Images** drop down menu.
3. Select the **Options (advanced)** checkbox.
4. When prompted to load the Data File, choose the file `van_gogh_data_paris.txt`.
5. When prompted to load the Images, select the `van_gogh_images` folder.
6. In the Data Selection dialogue box, select the following from the drop down menus:

   Image filename - "Filename (Column0)"
   X axis - "Brightness_Median (Column2)"
   Y axis - "Saturation_Median (Column3)"

7. When prompted to select advanced options menus, click **Canvas** and **Range** checkboxes.

8. In the **Canvas** dialogue box, enter 4000 pixels for **Canvas width** and 4000 for **Canvas height**.

9. In the **Range** dialogue box, select the **X range specified?** and **Y range specified?** checkboxes and set the range from 0 to 255 for both axes.

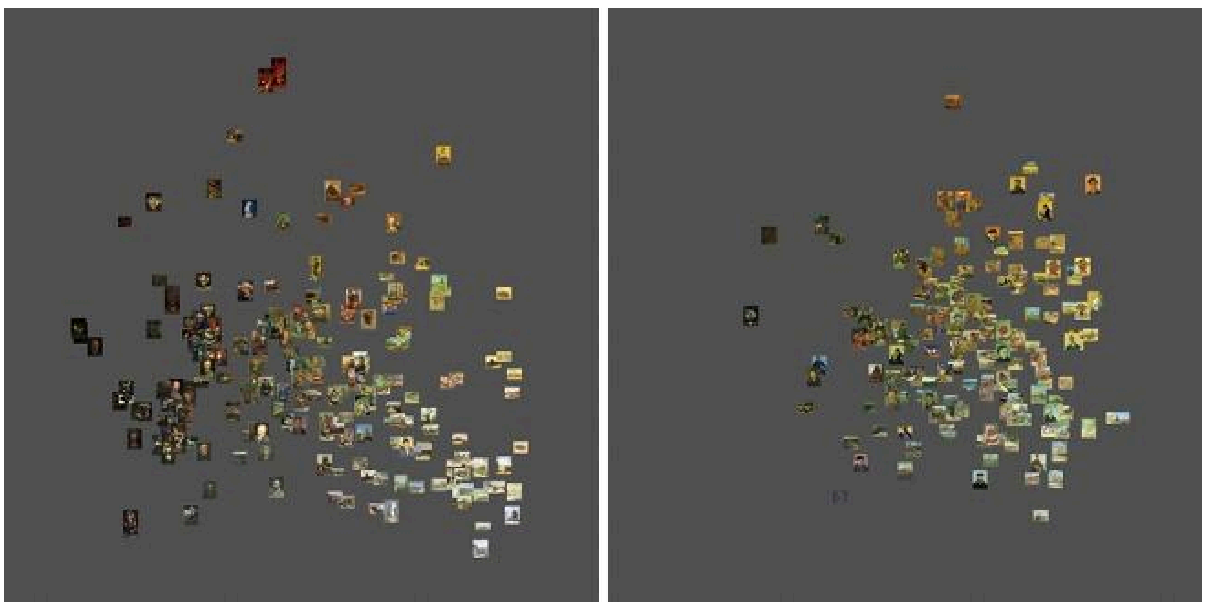
10. The macro will run and render a visualization.

11. When visualization is completed, go to the **File** menu in ImageJ and **Save** the visualization.

12. To generate a comparable visualization of van Gogh's Paris and Arles paintings, repeat steps 1-11 but use `van_gogh_data_arles.txt` file instead.

    Note: `van_gogh_data_arles.txt` includes images of paintings created between March 1888 and April 1889.

The two visualizations placed side by side should look like this:



Left: 199 van Gogh paintings created in Paris (1886-1888). X-axis = median brightness. Y-axis = median saturation.
Right: 161 van Gogh paintings created in Arles (1888-1889). X-axis = median brightness. Y-axis = median saturation.

Notice how the brightness and saturation values of the Paris and Arles paintings significantly overlap. This suggests that that the usual division of van Gogh's works into stylistic periods based on where he lived may need to be reconsidered. Often, than we think about a particular artist, we only compare his most famous works. These works may exaggerate the differences between the periods. However, when we systematically compare most of the paintings created in Paris and Arles using image plots, we can see that the differences between the two sets are smaller than we could have expected by only looking at the famous works.

We also can better understand the nature of these differences. First, van Gogh's paintings *created in Paris have significantly more variety in both brightness and saturation values than the paintings created in Arles. Second, the center of the "cloud" formed by Arles paintings (i.e. the point which corresponds to the means of the brightness and saturation values for all Arles paintings) is shifted to the left and to the top. In other words, Arles paintings are overall both lighter and more saturated than Paris paintings.*

Calculating the average values of brightness and saturation for the paintings created in two periods allows us to further quantify these observations:

> *Paris_brightness_mean = 129.83.*
> *Arles_brightness_mean = 158.51.*
>
> *Paris_saturation_mean = 95.70.*
> *Arles_saturation_mean = 109.28.*

We can also calculate standard deviation for these values for each period to quantify their variability:

> *Paris_brightness_standard_deviation = 51.65.*
> *Arles_brightness_standard_deviation = 34.71.*
>
> *Paris_saturation_standard_deviation = 40.59.*
> *Arles_saturation_standard_deviation = 36.30.*

These numbers agree with the interpretation of van Gogh's Arles period provided by Vincent van Gogh Museum web site: "Whereas in Paris his works covered a broad range of subjects and techniques, the Arles paintings are consistent in approach, fusing painterly drawing with intensely saturated color." What is interesting, however, is that the changes in saturation (both median and standard deviation values) turn out to be less significant than the changes in brightness values.

Examples Using Other Data Sets

Visit the ImagePlot gallery on Flickr which contains examples of visualizations of other data sets created with ImagePlot.

See hundreds of other visualizations created by members of Software Studies Initiative at http://www.flickr.com/photos/culturevis/collections.

Read our article "How to Compare One Million Images?" included with ImagePlot distribution: it presents and discusses a number of visualizations of image sets which range from a few hundred to a one million images - all created with ImagePlot.

ANIMATE VISUALIZATIONS

ImagePlot can also create animated visualizations. When you select the appropriate option, ImagePlot will save the canvas every time it adds a new point or image into a separate file. The sequence of files can be then turned into an animation using QuickTime or any video editing software.

Animated visualizations of image sets are useful to show developments over time. For instance, we can animate images of Piet Mondrian paintings using dates for X-axis, and brightness_median, saturation_median, or other features for Y-axis. You can find Mondrian images and the data file with the measurements in `sample_files` > `Mondrian` folder.

Before we start this tutorial: When you use animation option, each frame will have the resolution specified by the **Canvas width** and **Canvas height**. Since the default settings are 8192 pixels x 4608 pixels, you should change them to smaller numbers so your frames will play smoothly. Since you are using small canvas, you may also need to change default **Thumbnail width** size of 100 pixels to a different value.

In this tutorial, we will use 1280 for **Canvas width and** 720 for **Canvas height**. (We chose these numbers because 1280x720 is the recommended resolutions for YouTube HD videos, so we will get good quality playback if we upload the animated visualization to YouTube). We will reduce **Thumbnail width** to 30 pixels.

Follow these steps to render frames for the animated visualization:

1. Run the `ImagePlot.txt` macro.

2. Select the option "Open..." in both **Data** and **Images** drop down menus.

   Note: You can also select "None" in **Images** drop down menu to create an animated visualization which shows data as points.

3. Select the **Save images for animation** checkbox. Click **OK.**

4. When prompted to select Data File, select the `Mondrian_images_data.txt.` When prompted to select Image Files, select `Mondrian_images` folder.

5. Because you checked "save images for animation" option in the first menu, ImagePlot will next show **Save Animation Frames** window asking you to select the folder where it will save the image sequence. Select "create new folder" button to create a new folder now. Name the new folder `Mondrian_frames`. (You can also create a new folder before running ImagePlot, and then select this folder in this step).

6. We have organized the data file in such a way that the right columns will be automatically selected In the next dialog box: "filename (Column 0" for I**mage Filename**, "imageID (Column 1") for **X axis**, and "brightness_median ("Column 2)" for **Y axis**.

7. Next you will see **Options** menu. Select **Canvas** and **Image** to configure these options, and click **OK.**

8. Next you will see I**magePlot: Canvas** menu. Set **Canvas width** to 1280, and **Canvas height** to 720. Click **OK.**

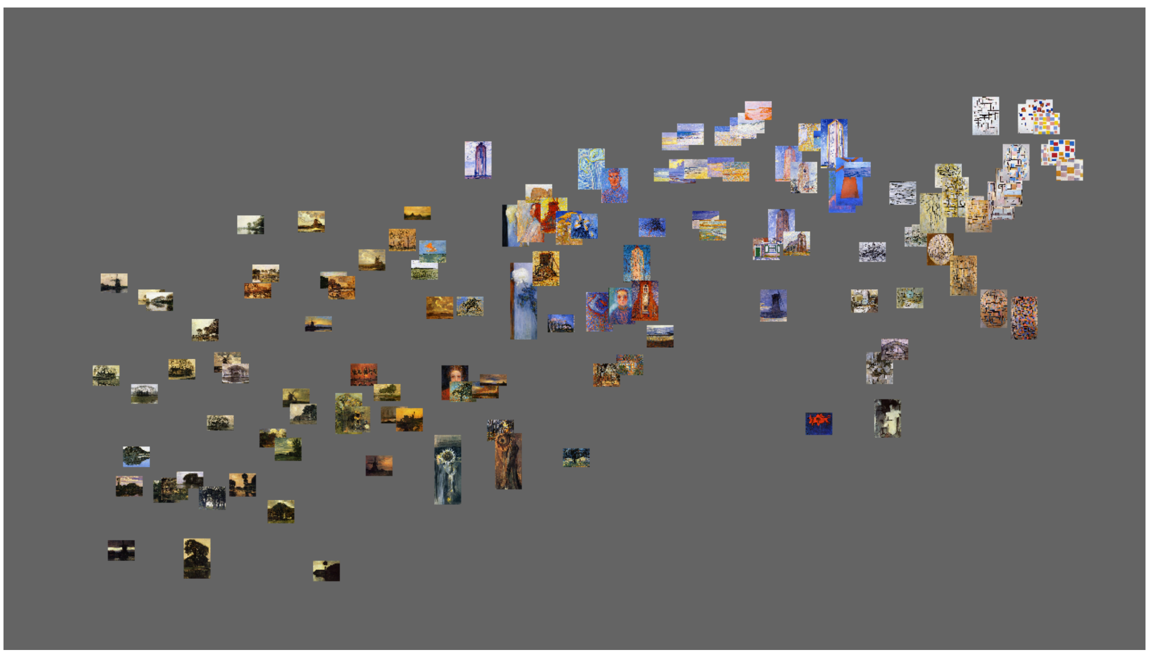
9. Next you will see **ImagePlot: Images** menu. Set **Thumbnail width** to 30. Click **OK**.

10. ImageJ will start rendering the visualization. After each new image is added to the visualization canvas, ImagePlot will save it as a separate file inside `Mondrian_frames`. Saving each image takes additional time, and therefore the rendering of the complete visualization will also take longer. To speed up rendering, it is recommend that you don't trust to multi-task: let ImagePlot finish rendering all frames before switching to another program,.

11. After all frames have been rendered (you will know that visualization is finished when the progress bar disappears), look inside `Mondrian_frames` folder. The files are named sequentially. Each file contains the visualization which shows all images specified in the previous rows. For example, the file `frame_25.png` contains images specified in the columns 2 to 25. Since `Mondrian_images_data.txt` has 128 rows not counting the columns labels row (one row for each image) you will find 128 sequentially named files.

12. Use QuickTime or any video editing software to turn these files sequence into a video. If you use QuickTime player, select **File > Open Image Sequence...** and then choose the folder which contains the file sequence created by ImagePlot. (See Apple's [QuickTime documentation] for details on how to import image sequences.)

If you follow these steps and make the video, its last frame should look like this:



Click inside the image to go to the [YouTube video]

If you want to see changes in saturation in Mondrian's paintings in our set, repeat the steps above, but instead of choosing "average_brightness" column for **Y-axis** in step 6, choose "saturation_median."

Animated visualizations are also great for showing evolution of an image sequence according to two two visual features. For instance, we can animate our earlier visualization of van Gogh images using brightness median for X-axis) and saturation median for Y-axis.

We can also statistical techniques such as Principal Component Analysis to combine many features into two. For an example, see this [video]. Like in the previous examples, images of Mondrian's paintings are rendered over time according to their dates. However, rather than using single features, X-axis and Y-axis use combinations of 18 features to position the images by their visual similarity. The images that are visually similar are situated next to each other; the images which are different are situated further away. The distance between images corresponds to the degree of their visual difference. Animated visualization shows how over time Mondrian moves from his early style, while at the same time expanding the range of visual options in his paintings. (For more detailed analysis, see our article [Style Space: How to compare image sets and follow their evolution](#).)

WORKING WITH DIGITIZED IMAGES

Our data set contains 776 images of van Gogh paintings; art historians estimate that the artist produced approximately 900 paintings during his life. Digital images of van Gogh paintings are available publicly on the web at flickr.com, commons.wikimedia.com, vggallery.com, vangoghgallery.com, and vangoghmuseum.nl, among others. These images are the most likely to appear in image search results for van Gogh, and as such, are significant for being the mostly widely viewed contemporary representations of his work. The digitized images vary substantially in terms of how accurately they represent the originals. Some are as good as the reproductions appearing in the most expensive print publications, while others may have inaccuracies in how they represent contrast, hue, saturation, and texture. Higher quality digital reproductions are available through artstor.org (only accessible to institutional subscribers) and scalarchives.com (a commercial art reproduction service).

This situation is typical today for a variety of cultural artifacts: often their digital reproductions (of varying quality) are easy to access and freely available online. Whether these images should be used in research depends on the questions being asked. If we are interested in making arguments about particular works, we want to use the highest quality images. If we we want to investigate larger patterns and cultural trends which manifest themselves only across large image collections (hundreds or more) and are careful about our procedures, the presence of scanning artifacts, color inaccuracy, and other problems resulting from digitization of these images may not affect our results. For example, we can use "median" rather than "mean" to represent the average value, because median measurement is less sensitive to outliers. In another example, if many images in our data set have inaccurate colors this would not affect the measurement of a numbers of "shapes" in images (by shapes we mean parts parts of an image which have different grey scale values). We can also use statistical methods, which deal with uncertainty and chance to precisely quantify the levels of confidence in our results.

To illustrate this, we are going to compare a selection of 15 images of Paul Gauguin paintings taken from one of the web galleries about the artist, and 15 images of the same paintings from the Wikipedia article List of paintings by Paul Gauguin.Using ImageMeasure.txt macro, we measure a number of visual features in both image sets, and then calculate the averages (mean) of the feature values for each set. The following table shows the results of this comparison.

image set	brightness median	saturation median	hue median	brightness stdev	saturation stdev	hue stdev
web gallery set: mean value of 15 images	140.73	124.80	52.47	57.64	60.57	61.52
Wikipedia article set: mean value of 15 images	142.27	112.60	65.73	56.18	51.97	55.59
absolute difference between the two means	1.53	12.20	13.27	1.46	8.61	5.93
absolute difference between the two	**0.60**	**4.78**	5.20	1.15	6.78	4.67

means as a percentage						

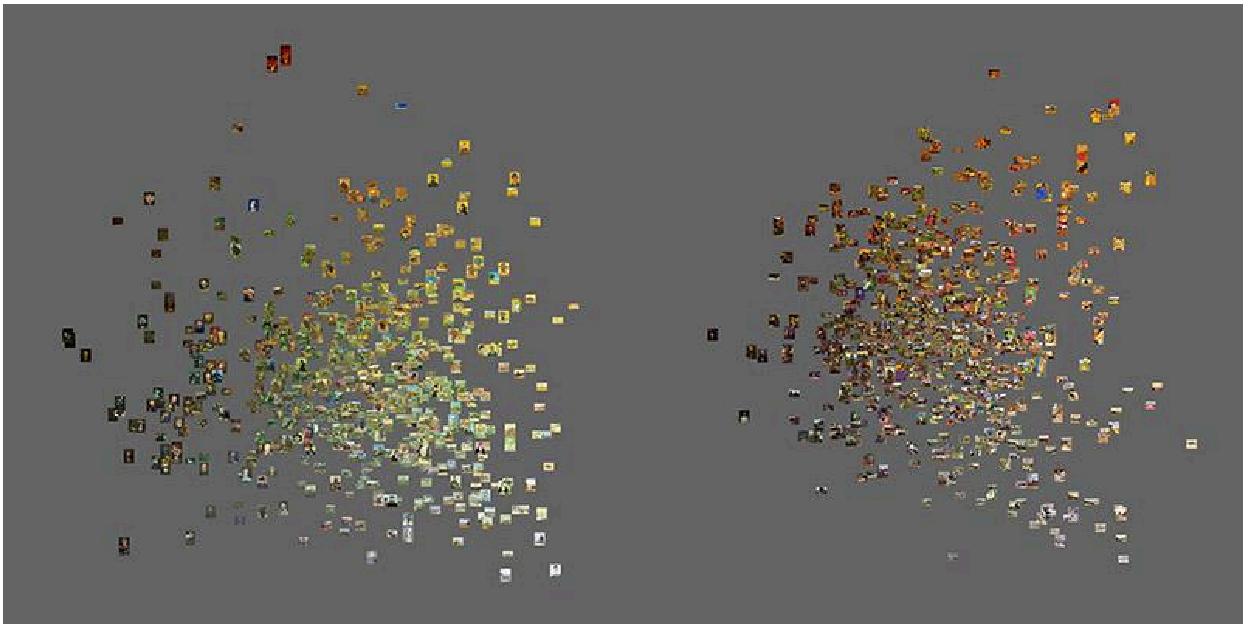
Range of possible values for median measurement (columns 2, 3, 4): 0-255.
Range of possible values for standard deviation measurement (columns 5, 6, 7): 0-127.
The 2nd row: averages (mean) for the first image sample set of Gauguin images.
The 3rd row: averages (mean) for the second image set of the same Gauguin paintings.
The 4th row: the difference between the means of the two sets expressed as a percentage of the total possible range of values..

Names of the images in the sample set from Wikipedia source:

Woman-In-The-Waves-Aka-Ondine-I.jpg
Arearea-Aka-Joyousness.jpg
Ferme_à_Arles.jpg
Madame_Roulin.jpg
Of-Vincent-Van-Gogh-Painting-Sunflowers-Aka-Villa-Rotunda-By-Emma-Ciardi.jpg
Otahi.jpg
Portrait-Of-A-Woman-With-Cezanne-Still-Life.jpg
Seed-of-the-Areoi-(Te-aa-no-areois).jpg
Snow--Rue-Carcel.jpg
Still-Life-With-Apples--Pear-And-Ceramic-Portrait-Jug.jpg
Vahine-No-Te-Tiare.jpg
Vision_after_the_sermon.jpg
Washerwomen_at_the_Roubine_du_Roi_Arles_1888_Paul_Gauguin.jpg
Where-Do-We-Come-From-What-Are-We-Doing-Where-Are-We-Going.jpg
Yellow-Christ.jpg

In this example, the differences in brightness, saturation and colors between the two sets of digital images of the same paintings vary from 0.6% (brightness median) to 6.78% (saturation standard deviation). Are these differences important? It all depends on the particular questions being asked. For example, if we compare this set of Gauguin images with the similar number of images of another artist whose works are visually similar, and we want to quantify the degree of similarity, 5% percent matter. But if we are interested to trace the evolution of modern art across thousands of images of hundreds of artists, such a difference will not affect the result.

To see how this may work in practice, let us compare van Gogh's paintings with the works of another post-impressionist artist - Paul Gauguin. Since our Gauguin set contains 580 images, we will select the same number of images for Van Gogh, dropping paintings created before he moved to Paris in Spring 1886.



Left: 580 van Gogh paintings (4/1886-7/1890). Right: 580 Gauguin paintings. X = brightness (median). Y = saturation (median).

Both van Gogh's paintings (left) and Gauguin's paintings (right) form single clusters, with most works in the center, and fewer works on the periphery. The center of van Gogh's cluster is to right and lower than the center of Gauguin's cluster. Since X-axis represents average brightness, and Y-axis represents average saturation, this means that hypothetical "average" van Gogh's painting is lighter and less saturated than the hypothetical "average" Gauguin painting.

To confirm this, we can calculate the averages (mean) of the features of all van Gogh and all Gauguin images:

image set	brightness median	saturation median	hue median	brightness stdev	saturation stdev	hue stdev
van Gogh: mean value of 580 images	**145.04**	**100.81**	51.46	50.72	47.17	33.61
Gauguin: mean value of 580 images	**131.14**	**122.39**	40.07	56.88	53.43	58.03

Placing the biographies and art careers of the two artists within the larger context of the development of modern art provides a reasonable explanation for these differences. Vincent van Gogh (1953-1890) "mature" period occupies just five years: 1886-1890. Gauguin (1948-1903) starts painting in 1870s, reaching similar saturation as van Gogh at around the same time (mid 1885s) - but he still continues to develop for another 18 years. The higher saturation values of his later work fit with the larger trajectory of European modernist modernist canon towards highly saturated colors which culminates in Fauvism (1904-1908). Both van Gogh and Gauguin also also went south from Paris to reach higher saturation in their works. However, while van Gogh only went as far as south of France (Arles), Gauguin went to French Polynesia..

Would these conclusions be invalidated if one of our image sources contains images which are too light or over-saturated? For example, imagine that instead of a web gallery we used as a source for Gauguin's gallery, we would use Gauguin's images from Wikipedia. Let's also assume that overall difference in feature values between the two sources is the same as we found with the sample of 15 images: 0.6% (average brightness) and %4.78 (average saturation). If we increase average brightness and average saturation values of every of 580 Gauguin's images by 0.6% and %4.78, respectively, and make new *visualizations, Gauguin's "image cloud" will move a tiny bit to the right and up - but this will not change our conclusion.*

Using born-digital images and video often offers an advantage: since they are originally created in digital form, we don't need to worry about changes in color, contrast and other visual characteristics introduced than analog images and digitized. However, born-digital data has its own problem: compression. Images on on Flickr are available as high-quality jpeg files, so compression artifacts are not significant. However, lots of many user-generated video on YouTube is compressed quite significantly. This may seriously affect certain measurements such as color features.

These issues are not limited to working with images and video. Humanists who use qualitative methods to analyze texts may appear to be more lucky. Since texts consist from discrete signs drawn from a small alphabet, moving text from one format to another does not introduce artifacts; text files are small so they are not stored using lossy compression. However, this is only true if we deal with born-digital texts, or if the texts originally published in other media were carefully transcribed into a digital text file. The bulk of hand-written and printed texts from the past remain in their original form - although Google Books is making good progress in digitizing books and using its custom OCR to extract text. (Available commercial and free OCR software usually only produces %100 accurate text if original source use standard fonts and is scanned at high resolution.)