2022 ■ Vol. 18 ■ no. 2

image2data: An R package to turn images in data sets

Pier-Olivier Caron [a] and Alexandre Dufresne

[a]Université TÉLUQ

Abstract ■ Data visualization is an essential and powerful tool to generate hypotheses, uncover patterns, and disseminate findings. It is crucial that introductory statistics courses train students to become critical authors and consumers of data visualization. Ludic data sets might help teaching statistics to students by making graphics more enjoyable, using images as instantaneous feedback, encouraging to discover hidden patterns, and reducing their focus on traditional hypothesis testing. Those data sets that have hidden images can be difficult to come by for teachers. These considerations have led to the development of a package that could easily create data sets from images for educational purposes. The purpose of this study is to present `image2data`, an **R** package that generates data sets from images. In this study, we show how to install the package, explain the basic arguments, and show three examples on how it can be used for teaching. Future studies could evaluate the effectiveness and motivation generated by using hidden images in data sets. Our hope is that by using hidden image in data sets, students will be inspired to decrypt data sets and discover the unexpected.

Keywords ■ data visualization, R package, data extraction. **Tools** ■ R.

Acting Editor ■ Denis Cousineau (Université d'Ottawa)

pier-olivier.caron@teluq.ca

10.20982/tqmp.18.2.p186

Introduction

Data visualization is an essential tool to make sense of data (Tukey, 1977). It is a powerful tool to generate hypotheses and uncover patterns, and to disseminate findings efficiently, can be easily shared amongst researchers and decision-makers and, above all, laymans who might not consume data on a regular basis. *Data visualization unravels and tells stories*.

Creating effective graphs is both a science and an art though (Franconeri, Padilla, Shah, Zacks, & Hullman, 2021). It is crucial that introductory statistics courses train students to become critical authors and consumers of data visualization (Hudiburgh & Garbinsky, 2020). Not only do students and researchers should learn and teach data visualization, it should become automatic. They should be able to generate and play with data in a meaningful manner. Students and researchers should be enthusiastic about data visualization and analysis (Stander & Dalla Valle, 2017). Data manipulation and visualization should be rewarding and ludic.

There have been many studies showing examples on how to use data sets with hidden message or image to create humor, irony or interest in statistical courses. For instance, Stefanski (2007) showed how to hide text, which was only revealed by plotting the residuals versus predicted values from the least squares fit of the correct regression model. Farnsworth (2009) doubled down with hidden images. To emphasis further the importance of graphical representations, Matejka and Fitzmaurice (2017) showed how to generate datasets which are identical over a number of statistical properties but produce different graphs. Finally, Yanai and Lercher (2020) hid images in datasets to stimulate inductive reasoning and exploration of data as to avoid what they refered to as *the blindness of hypothesis-driven data analysis*.

Obtaining data sets that turn into an image can be difficult for teachers. Creating the plot by hand can be tedious and infuriating while computers can automatically do most of the work. These considerations have led to the development of a package that could easily create data sets for educational purposes. We developed `image2data`, an **R** package that creates data sets from images. There are many benefits to use images instead of *random data points*, such as

- *instantaneous feedback*. Finding a recognizable pat-

Figure 1 ■ Partial view of the mushroom data set in RStudio.

df

Filter

	x	y	g
1	-1.784687	0.0476676287	#000000
2	-1.784687	0.0436752586	#000000
3	-1.784687	0.0396828886	#000000
4	-1.784687	0.0356905185	#000000
5	-1.784687	0.0316981484	#000000
6	-1.784687	0.0277057783	#000000
7	-1.784687	0.0237134083	#000000
8	-1.784687	0.0197210382	#000000
9	-1.784687	0.0157286681	#000000
10	-1.784687	0.0117362980	#000000
11	-1.784687	0.0077439280	#000000

tern gives an instantaneous feedback that the goal has been accomplished, reinforcing automatically learners' habit of creating graphs;

- *enjoyable*. Decrypting data sets with hidden patterns is likely to be more interesting and surprising for learners than merely looking at scattered points;
- *critical thinking*. Testing hypotheses with graphs might break their habits on *p*-values (at least a little) and shows alternatives to traditional hypothesis testing.

The purpose of this study is to present `image2data`, a **R** package to generate data sets from images for effective teaching and research. We show how to install the package, explain the basic arguments and show some examples on how it can be used for teaching.

image2data

The goal of `image2data` is to extract images and return them into a data set, especially for teaching data manipulation and data visualization. Basically, the eponymous function takes an image file (extension: png, tiff, jpeg, bmp) and turn it into a data set, pixels being rows (subjects) and columns (variables) being their bicoordinate positions (x- and y-axis) and their respective color in hex color codes (variable `g`).

The output of the function is a $n \times 3$ data frame containing three variables `x`, `y`, `g`, which are respectively, the pixels' horizontal (`x`) and vectical (`y`) coordinates and its hex color (`g`). This last variable is really important to keep when many colors are extracted[1]. The data frame has the number of rows $\times$ the number of columns pixels (n), which becomes quite large for high-quality images. Figure 1 shows an example of the generated data frame.

Data can thereafter be manipulated as would any data sets by either creating other related variables (to hide the image) or as a genuine toy data set. The data set can be exported to other software if teaching is not done in **R**. It can easily be achived by the functions `write.csv("df", file = "df.csv")` (dot is used for the decimal point) or `write.csv2("df", file = "df.csv")` (comma is used for the decimal point) for a csv file, where `df` is the variable containing the data frame.

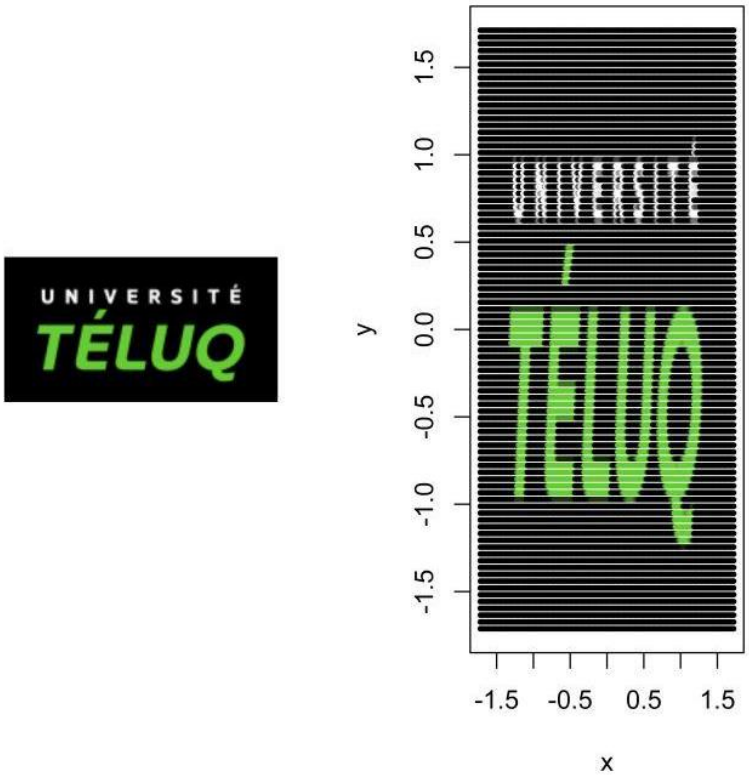
Download the package

To use the package, users can install the package from the CRAN directory

```
install.packages("image2data")
```

[1] One can easily convert between hex color code and RGB code in **R** using `rgb()` or inversely `col2rgb()`. For instance, pure black is `"#000000"` in hex and `R = 0; G = 0; B = 0`.

Figure 2 ■ Turning the TELUQ University logo into a data set.



or development version from GitHub repository (be sure to have the `remotes` package installed).

```
remotes::install_github(repo =
   "quantmeth/image2data")
```

Once the package is installed, users can either load the library via `library(image2data)` or use the `::`, like `image2data::image2data()`, to access the `image2data()` function.

Minimal examples

Before going more into the details on how the function works and its arguments, we first show two minimal examples.

A complete image

The default operation of `image2data()` is to extract all the pixels of the original images. Figure 2 shows the TELUQ University logo[2], a green and white text on a black background on the left panel and the extracted data on the right panel. Extracting a complete image is the simplest possible option as no further arguments, beside the image path, is necessary. It is worth to note that all pixels were recorded, green, white and black ones. The invisible pixels were not kept. A total of 14863 rows are in the data sets.

```
df <- image2data(path = "teluq.png")
```

A screen capture

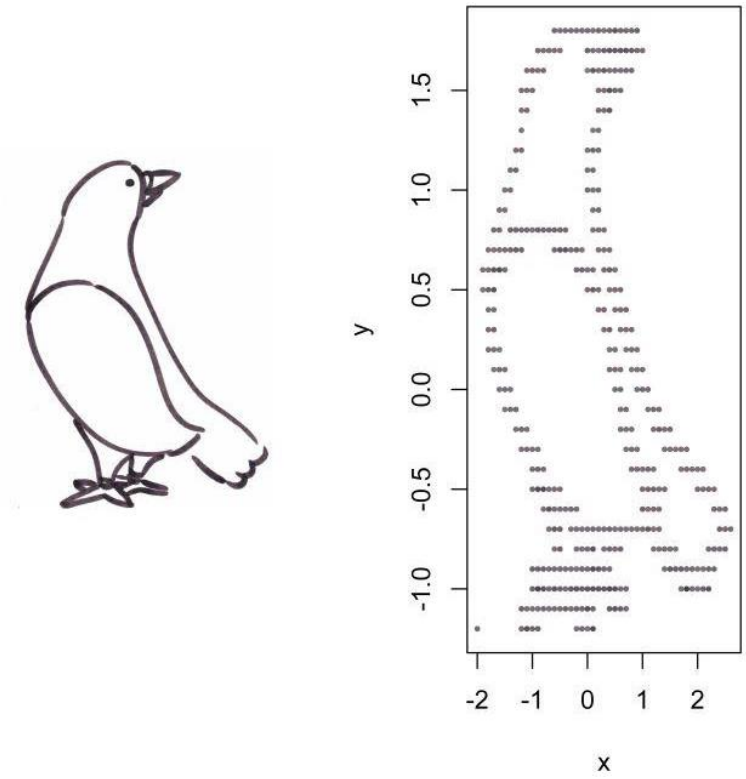
Another possible option is to extract the outline of an image, such as texts, urls, or drawings. Left panel of Figure 3 shows a hand drawn pigeon which was scanned, truncated and turned into a png file.[3]

```
df <- image2data(path = "pigeon.png",
                 type = "line",
                 reduce = "unique",
                 Grey = c(0,.5))
```

Some tweakings are needed to correctly turn this image in a data set, which will be discussed later on. First, the argument `type = "line"` specifies that only a range of color (such as an outline) has to be extracted. As the im-

[2] Available at https://www.teluq.ca/site/documents/presse/logo_teluq_rvb.png
[3] It is the original hand drawn version of the pigeon by Geneviève Dufresne found in Caron (2017), Figure 1, p. 33

Figure 3 ■ Turning a hand drawn pigeon into a data set.



age is a range of grey and not purely black, a wider range of grey is specified by `Grey = c(0, .5)`. Finally, to reduce the amount of rows (subjects in the data set), the argument `reduce = "unique"`. Right panel of Figure 3 show the resulting image. There is 1124 data points. As it will be discussed later, the precision of the image can be increased as to be closer to the original. Only the grey pixels has been kept, the white background has not been extracted.

Basically, the pigeon file was a screen capture of a scanned images. It could have been any other screen captures, such as a part of a text, a message, and an url or image. The tweakings are image-specific, such as the range of color to extract or the inclusion of background or not, and output-specific, like how many data points and the shape of the data. All these specifications are described below.

Basic arguments

The mandatory argument is the `path` to the image, which can be a png, a tiff, a jpeg, or a bmp file. In doubt, users can find the file's path using the `path <- file.choose()` function from **R**-base which turns the file's path into a variable. For simplicity, it is recommended to save the image in the current **R** working directory. The software looks for files in the current working directory by default, which simplifies the path to the file name between quotes, such as `"file_name.extension"` (no complicated directory needed). Here is an example using the TELUQ University logo.

```
df <- image2data(path = "teluq.png")
```

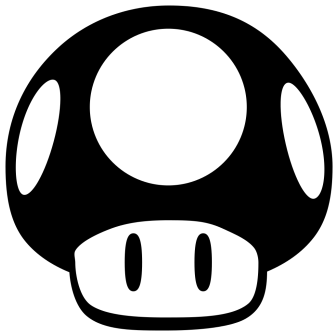
For readers who are not used to **R** programming language, the `df <-` assign the data set to a variable, `df`, which can then be manipulated subsequently.

The function has two primary mode to extract the image. By default, the function returns the complete image using `type = "fill"`. The previous line of code use this default value to extract the image, so black, green and white colors are extracted. Right panel of Figure 2 shows the logo once it has been extracted. Users can also extract a range of color using `type = "line"`.

```
df <- image2data(path = "teluq.png",
                 type = "line")
```

By default, `type = "line"` returns a range of black only. In the University TELUQ logo, only the black pixels

Figure 4 ■ Mushroom from the Nintendo's franchise Mario.



are extracted. This can be modified by the users with the argument `R`, `G`, `B`, `A`, which are red, green, blue and alpha (transparency). They range between 0 to 1 and their default values are .00 to .05, which are shades of black. There is also a `Grey` argument which overwrites `R`, `G`, `B` for simplicity. This argument was used previously in the minimal example of the pigeon. If only the green color would be desirable, like in the University TELUQ logo, the function can be rewritten. Unless the specific color range is already known, it may take some trials and errors to find the correct range of color.

```
df <- image2data(path = "teluq.png",
                 type = "line",
                 B = c(.2, .5),
                 R = c(.2, .7),
                 G = c(.5, .8))
```

Using the `type = "line"` may lead to an error message like `"No data were returned."`. This problem occurs when no pixel correspond to the current range of color. The solution is to increase the range of `R`, `G`, `B` or `Grey`.

The function `image2data()` has many optional arguments. For instance, it produces by default a plot to see the results (`showplot`). The argument `showplot` can be turned off by setting it to `FALSE`. The `scaling` can be either `standardized` (default, x- and y- axes have average 0 and standard deviation of 1), `normalized` (convert to unit vector), or `original` (let the data in its original form, *i.e.*, pixels' coordinate).

Sample sizes of data sets are directly related to the number of pixels, which can be quite large for high-quality images (even if a single range of color is extracted). There is three options to reduce the generated sample size with the argument `reduce`. By default, `reduce = 1` returns all (100%) data. If `reduce` is set between 0 and 1, it returns a percentage of the data. For instance, `reduce = .05` returns 5% of the original data. If `reduce` is an integer over or equal to 2, it returns that number of data. For instance, `reduce = 100` returns 100 data points. Data are randomly selected without replacement. The seed can be fixed directly from **R** or in the function (`seed`). A third option is to set `reduce = "unique"` with a certain *precision threshold* (`precision = 1` by default), which is roughly similar to the pixelated aspect of the image. Precision can range between $0 < precision < \infty$. The higher the `precision` (`precision` > 1), the more data points are returned; the lower the `precision` ($0 <$ `precision` < 1), the less data points are returned. It is worth noting that users could use any other technique they prefer via the unreduced data sets.

Extracting an image

Figure 4 is an image composed of $1028 \times 1024 = 1052672$ pixels[4]. It is worth noting that white pixels are actually transparent in this file, which simplifies the procedure.[5].

The function can easily extract the complete non-transparent pixels with the following code.

```
df <- image2data(path = "mushroom.png")
```

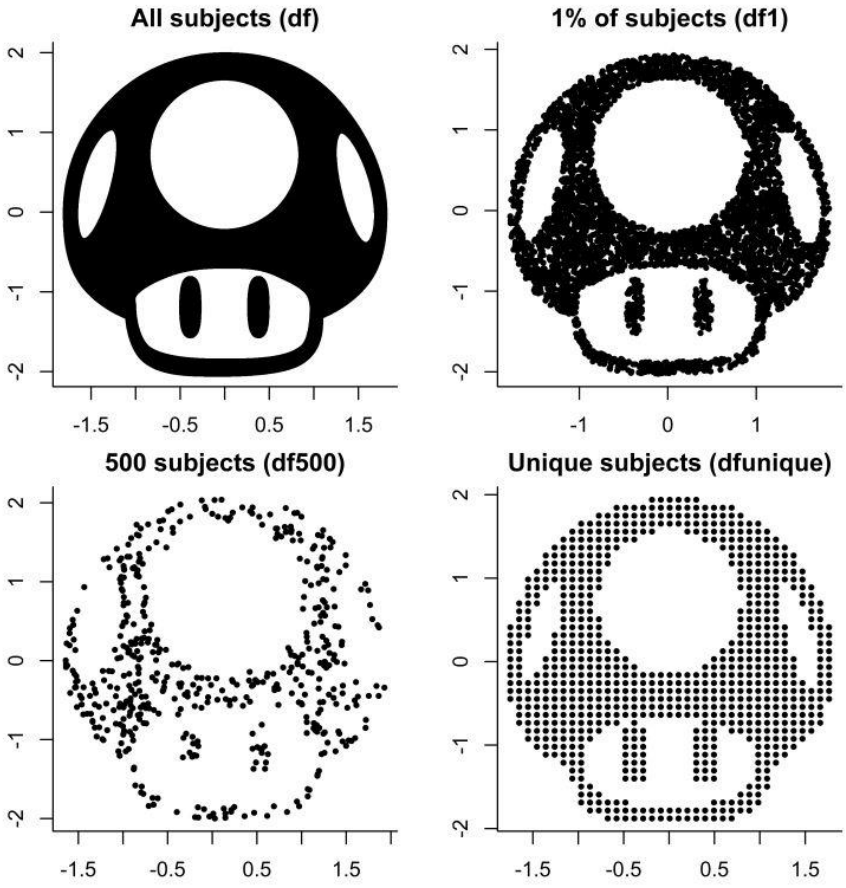
Figure 1 shows what the data looks like in `View()`. The actual data would look similar if they were exported to another software.

The data set, `df`, has 407357 rows (non-transparent black pixels only), which is equivalent to the number of subjects in the data sets. Since this *sample size* is actually quite large, it might be more appropriate to reduce it to make it look similar to real data sets (if desired). The

[4] License: Creative Commons (CC BY-NC 4.0); retrieved from https://freepngimg.com/thumb/mario_bros/92503-sonic-olympic-smiley-head-bros-mario-games.png

[5] If an image has an unwanted background, `type = "line"` should be used, like in the pigeon example.

Figure 5 ■ Four data sets from the mushroom image. Subjects are technically pixels of the original image.



following code shows three methods to reduce the sample size : choose a percentage of random subjects; a number of random subjects; or choose *unique* subject according to a specified precision.

```
df1 <- image2data(path = "mushroom.png",
                  reduce = .01,
                  seed = 42)
df500 <- image2data(path = "mushroom.png",
                  reduce = 500,
                  seed = 42)
dfunique <- image2data(path = "mushroom.png"
                  reduce = "unique")
```

Figure 5 shows the plots of the four original data sets contained within `df`, `df1`, `df500` and `dfunique` (complete, 1% subjects, 500 subjects, unique subject).

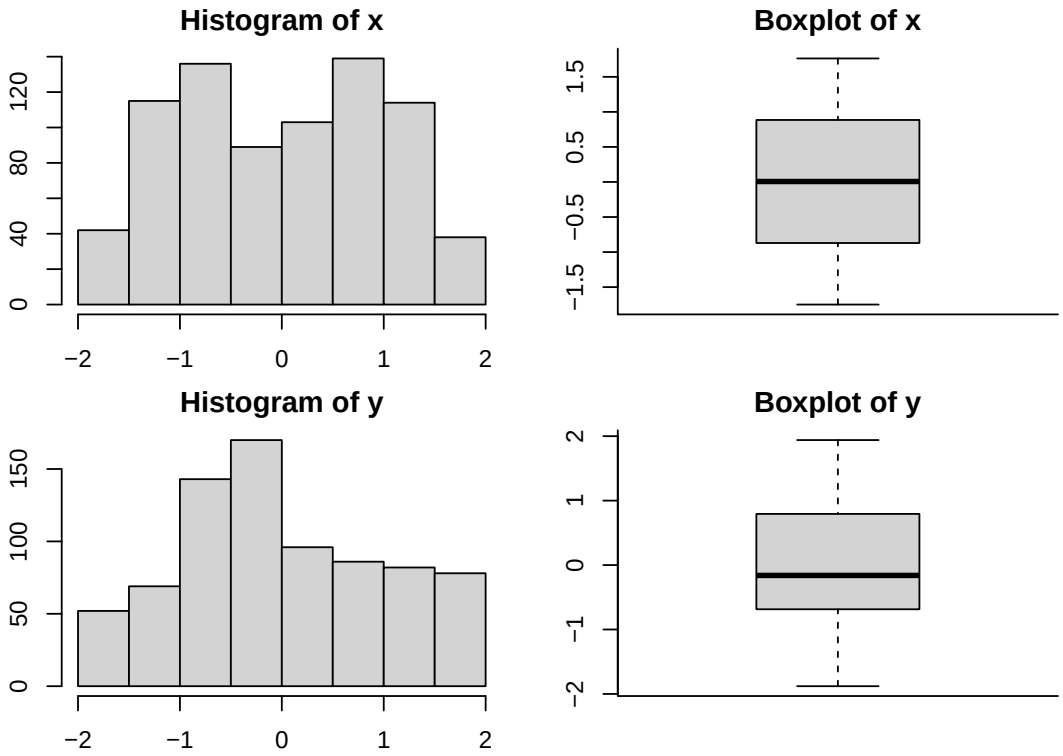
To share the data set among students or to use it with another software, the function `write.csv("df", file = "df.csv")` saves the data frame `df` to a csv file that can be imported by most statistical software. The file is named `"df.csv"` and saved in the current directory. To reuse the saved data set in **R**, the function `df <- read.csv(file = "df.csv")` will import the data by assigning it the variable name `df`. Users can use their prefered importation methods as well.

Using the data sets

Once the data set is obtained, it can be used as is or added to other data sets. The variable `g` can be dropped or transformed into a grouping variable depending on whether the color is necessary or not. Variables can also be unstandardized by multiplying by a specified standard deviation and adding a specified mean. Users could easily add outliers.

Any descriptive statistics can be computed (mean, standard deviation, covariance, correlation), as usual. Table **??** shows the means and the covariance matrix. As expected, the averages are 0 and the variance are 1. The covariance matrix (and consequently the correlation matrix) depends on the actual image, some may have an *accidental* correla-

Figure 6 ■ Histograms and boxplots for the variables of the mushroom data sets.



tion between variables whereas other do not.

Basic plotting techniques can also be done. No univariate visualization technique will unravel the actual image nor look suspicious. Figure 6 shows the histograms and boxplot of the data sets.

Without any other modificiations, data can be readily used. However, it can be pedagogically relevant to hide the image. In the following section, possible ways to hide the image to be retrieved by students are explained. Only the `dfunique` will be used.

Transformation

Variable transformation is one of the many methods to hide the image. Users must be cautious not to alter the variable in a way that can be easily undone, for instance, by squaring negative values. In this example, we transform `y` as `exp(y)`, or mathematically e^y. The variable is added to the original data set.

```
dfunique$ey <- exp(dfunique$y)
```

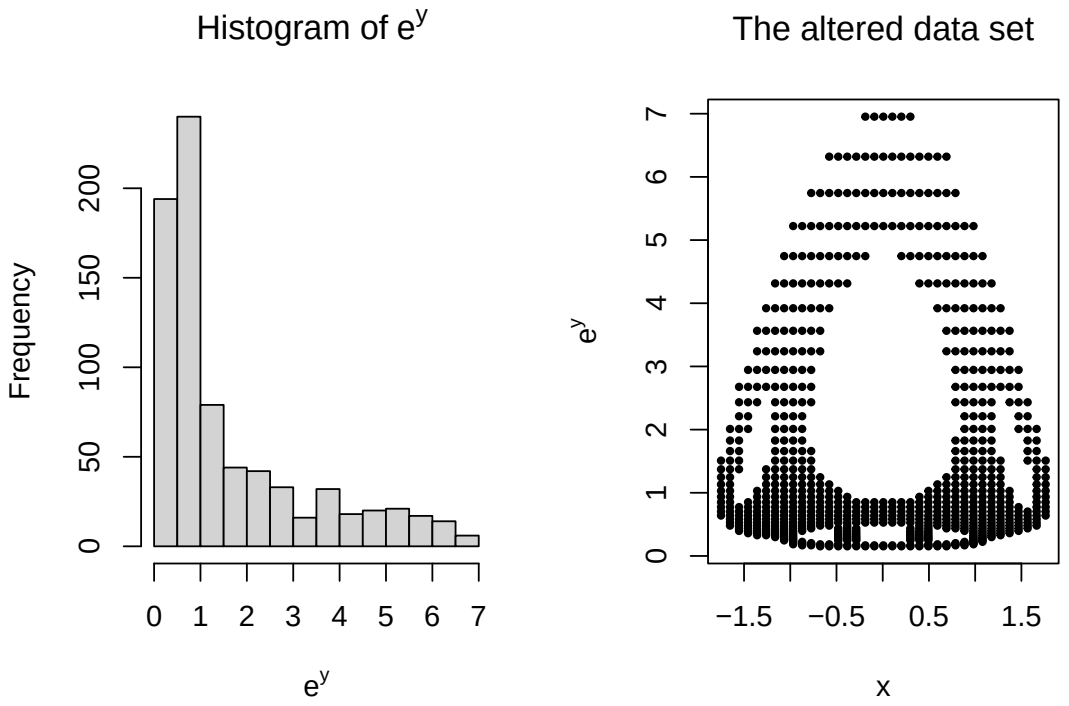
Figure 7 shows the histogram of the new variable and how it has altered the image. To untransform the image, users simply need to log the transformed variable as `log(dfunique$ey)` or equivalently $\log(e^y) = y$.

Users can delete the original variable before distributing the data set. At this point, students are asked to look at the shape of the distribution and to transform the variable.

Table 1 ■ Some caption

	Mean	Min	Max	x	y
x	0	-1.749	1.763	1.000	0.004
y	0	-1.879	1.939	0.004	1.000

Figure 7 ■ Histogram an scatterplot of the transformed variable.



Interaction

Another technique to hide the image is to create new variables based on the image's variables. One such example is an interaction, i.e., the product of two variables (Caron, Valois, & Gellen-Kamel, 2020). The interaction model without intercept is

$$y = \beta_a a + \beta_b b + \beta_{ab} ab + \epsilon.$$

There is two new variables in this model, a, b, and their product $a \times b = ab$, where ab is either of the variables in the data, the other being the dependent variable in the regression model (y). We can create a or b from a random distribution and then compute the other by dividing $ab/a = b$ or $ab/b = a$. The random variable can be created from the dependent variable plus some random noise as to add a correlation. The standard deviation of the noise will define the covariance between variables. The following code shows an example. First, define the interaction and the dependent variable (optional).

```
dfunique$interaction <- dfunique$x
dfunique$DV <- dfunique$y
```

Then, create an independent variable from the dependent variable (DV) and some noise as to add some relation between the new independent variable (IV) and the dependent variable. A completely random variable could be created also, but would be unrelated.

```
dfunique$IV <- dfunique$DV +
                    rnorm(nrow(dfunique))
```

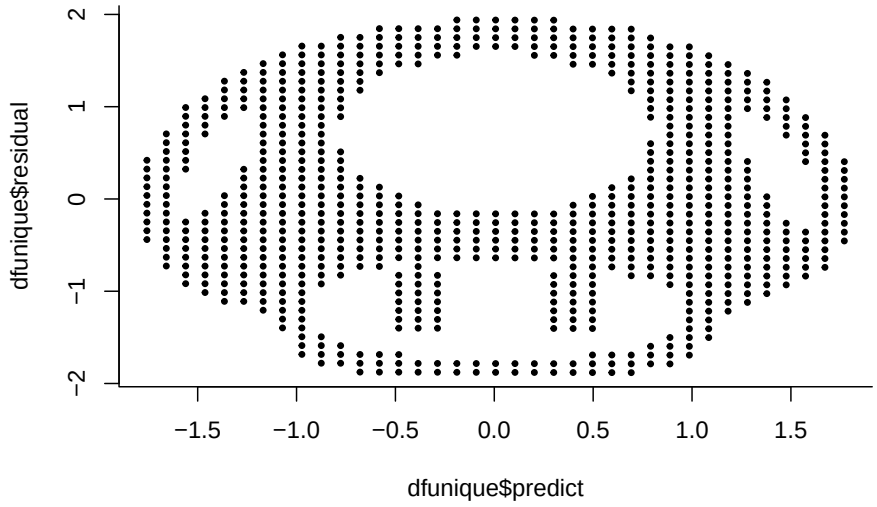
Finally, create the moderator variable.

```
dfunique$moderator <-
   dfunique$interaction / dfunique$IV
```

At this point `dfunique$interaction` (and the original variables) can be discarded as it can be easily retrieved from the two other variables. Students can now be asked to look for a moderation effect between the independent variable and moderator on the dependent variable.

In this example, there is no covariance between the variables `x` and `y` of the data set so there will not be a significant regression coefficient between the interaction and the dependent variable. If there was a correlation in the original data set, then they would be correlated. Despite there being no correlation between the interaction and the

Figure 8 ■ Plot of predicted values and residuals.



dependent variables, plotting the variables will reveal the mushroom.

Predict and residuals

Another technique is to use one of the original variables as the residuals (ϵ) in the linear model. Students can be asked to look at the distribution of the residuals or the homogeneity of variances (assumptions on linear models). The model without intercept and an implicite slope of 1 ($\beta = 1$) is

$$y = x + \epsilon.$$

This only works if the original variables are not correlated. Otherwise, the image obtained after a multiple regression will be distorted because of the least squares estimations. Techniques to decorrelated the variables do exist, but will inevitably distort the image, notably by rotating it.

To create the data, first define the residuals and the independent variable (optional).

```
dfunique$residuals <- dfunique$y
dfunique$IV <- dfunique$x
```

Then, create the dependent variable.

```
dfunique$DV <- dfunique$IV +
               dfunique$residuals
```

The effect size of the relation can be modified by changing β to the predictor or the standard deviation of the residuals. Once these operations are carried, `dfunique$residuals` and the original variables can be dropped as they are not longer useful. Students are asked to carry a regression of the independent variable on the dependent variable and then to plot the predicted values and the residuals to look for homoscedasticity and normality of residuals.

To do this analysis in **R**, the regression analysis is carried out first.

```
res.lm <- lm(DV ~ IV, data = dfunique)
```

Then, the predicted values and the residuals of the regression can be extracted easily and added to the data frame.

```
dfunique$residual = resid(res.lm)
dfunique$predict = predict(res.lm)
```

Plotting both variables will reveal the mushroom, as shown in Figure 8.

```
plot(dfunique$predict,
     dfunique$residual,
     bty = "l",
     pch = 19,
     cex = .5)
```

Conclusion

The purpose of this study is to present `image2data`, a **R** package to generate data sets from images. Using ludic data sets might help teaching statistics to students by making graphics more enjoyable, using images as instantaneous feedback, encouraging to discover hidden patterns and reducing their focus on *p*-values. Future studies could assess the effectiveness and motivation generated by using hidden images in data sets. Our hope is that by using hidden images in data sets, students will be inspired to decrypt data sets with data visualization and discover what we never expected.

References

Caron, P.-O. (2017). Sur la loi de l'appariement. *Psychologie Française*, *62*, 29–55. doi:10.1016/j.psfr.2015.10.003

Caron, P.-O., Valois, P., & Gellen-Kamel, A. (2020). Some computational descriptions of moderation analysis. *The Quantitative Methods for Psychology*, *16*(1), 9–20. doi:10.20982/tqmp.16.1.p009

Farnsworth, D. L. (2009). Playing with residuals. *Teaching Statistics*, *31*(3). doi:10.1111/j.1467-9639.2009.00348.x

Franconeri, S. L., Padilla, L. M., Shah, P., Zacks, J. M., & Hullman, J. (2021). The science of visual data communication: What works. *Psychological Science in the Public Interest*, *22*(3), 110–161. doi:10.1177/15291006211051956

Hudiburgh, L. M., & Garbinsky, D. (2020). Data visualization: Bringing data to life in an introductory statistics course. *Journal of Statistics Education*, *28*(3), 262–279. doi:10.1080/10691898.2020.1796399

Matejka, J., & Fitzmaurice, G. (2017). Same stats, different graphs: Generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 chi conference on human factors in computing systems* (pp. 1290–1294). CHI '17. doi:10.1145/3025453.3025912

Stander, J., & Dalla Valle, L. (2017). On enthusing students about big data and social media visualization and analysis using R, RStudio, and RMarkdown. *Journal of Statistics Education*, *25*(2), 60–67. doi:10.1080/10691898.2017.1322474

Stefanski, L. A. (2007). Residual (sur)realism. *The American Statistician*, *61*(2), 163–177. doi:10.1198/000313007X190079

Tukey, J. W. (1977). *Exploratory data analysis*. Pearson.

Yanai, I., & Lercher, M. (2020). A hypothesis is a liability. *Genome Biology*, *21*(1), 231. doi:10.1186/s13059-020-02133-w

Citation

Caron, P.-O., & Dufresne, A. (2022). image2data: An R package to turn images in data sets. *The Quantitative Methods for Psychology*, *18*(2), 186–195. doi:10.20982/tqmp.18.2.p186

Received: 25/04/2022 ~ Accepted: 03/07/2022