LeArEst: Length and Area Estimation from Data Measured with Additive Error

by Mirta Benšić, Petar Taler, Safet Hamedović, Emmanuel Karlo Nyarko and Kristian Sabo

Abstract This paper describes an R package **LeArEst** that can be used for estimating object dimensions from a noisy image. The package is based on the simple parametric model for data that are drawn from uniform distribution contaminated by an additive error. Our package is able to estimate the length of the object of interest on a given straight line that intersects it as well as to estimate the object area if it is elliptically shaped. The input data may be a numerical vector as well as an image in JPEG format. In this paper, background statistical models and methods for the package are summarized, and algorithms and key functions implemented are described. Also, examples that demonstrate its usage are provided.

Availability: LeArEst is available on CRAN.

Introduction

Image noise may arise by the physical processes of imaging, or can be caused by the presence of some unwanted structures (e.g. soft tissue captured in X-ray images of bones). Such problems can occur, for example, when the object is observed with a fluorescent microscope (Ruzin and others, 1999), ground penetrating radar, medical equipment (X-ray, ultrasound), etc. With the presence of additive noise, the object edge detection as well as determining length or area of the object becomes a non-trivial problem. The well known edge detection methods (Qiu, 2005; Canny, 1986) generally does not perform well.

Our approach does not use the mentioned edge detection methods, but looks at the problem in a different way. We start with a simple univariate model where the data represent independent realizations of a random variable X, $X = U + \varepsilon$. In the aforementioned equation U is supposed to be uniformly distributed over the object image and describes the object image without an error, while ε represents measurement error. It is shown that such a simple model can also be very useful in applications itself, not only related to the image analysis. For instance, in Tolić et al. (2017) the sum of uniform and normal distributions is confirmed to be the most representative distribution for modelling the transmission loss data.

Different aspects of this model are developed in Benšić and Sabo (2007b,a, 2010, 2016); Sabo and Benšić (2009); Schneeweiss (2004). The basic one-dimensional model is described in Section 2.2 together with the results that are used for statistical inference incorporated in the package. Although this model is not universal in all applications, we find it useful in some cases.

With the assumption that the observed object has a circular or elliptical shape, a two-dimensional approach has been developed, dealing with an object area estimation problem (Benšić and Sabo, 2007a; Sabo and Benšić, 2009). This approach utilizes many border estimations and performs parametric curve fitting on its results.

The package **LeArEst** (Bensic et al., 2017) uses these methods for length and area estimation of an object captured with noise. It supports numerical inputs, which is useful if a machine that records an object stores numerical data (coordinates of recorded points). However, if an object is shown in a picture file, the package includes a web interface with which one can load a picture, draw a line that intersects the object, adjust the parameters and perform an edge detection on the drawn line. Another web interface allows the user to draw a rectangle around the object and perform area estimation of the marked object. Description of functions dealing with numerical and graphical estimations and examples of their use are given in Section 2.3.

Basic statistical model

The basic model we deal with in this package is an additive error model

$$X = U + \varepsilon.$$

Here we suppose that the random variable U is uniformly distributed on the interval $[-a, a]$, $a > 0$, i.e. has a density

$$f_U(x; a) = \begin{cases} \frac{1}{2a}, & x \in [-a, a] \\ 0, & \text{else} \end{cases} \tag{1}$$

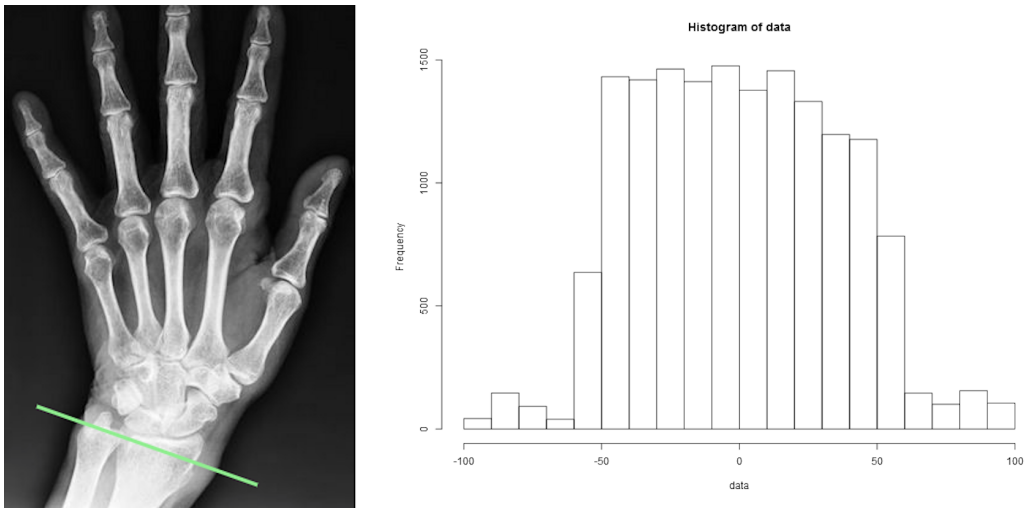

Figure 1: Line intersecting the object and histogram of recorded points for statistical inferences

and ε is an absolutely continuous random variable describing measurement error. Further, we assume that ε is independent of U with zero mean. Instead of the sample $U_1, U_2, \ldots, U_n$ from U, one can only observe the contaminated i.i.d. sample $X_1, X_2, \ldots, X_n$ from X. We are to estimate the unknown parameter a.

Our model is the special case of a general additive error model $X = Y + \varepsilon$, where Y and ε are assumed to be independent continuous random variables, but only X is observable. Estimating the unknown density f_Y from i.i.d. sample $X_1, X_2, \ldots, X_n, X_1 \sim X$, is known as the deconvolution problem. Usually, the error part ε is supposed to have a known density f_ε. Several nonparametric methods have been developed to estimate f_Y (Meister, 2009), but the most popular and studied is the deconvolution kernel density estimator (Carroll and Hall, 1988; Stefanski and Carroll, 1990). The packages **decon** (Wang and Wang, 2013) and **deamer** (Stirnemann et al., 2012) provide functions for estimating density f_Y in a nonparametric way. Two different approaches in estimating the support of a density from a contaminated sample can be seen in (Meister, 2006) and (Delaigle and Gijbels, 2006). One is based on the deconvolving kernel density estimator (Delaigle and Gijbels, 2006) and the other on the moment estimation (Meister, 2006). To our knowledge, none of them is implemented in some R package submitted to the CRAN repository.

For our purpose (e.g. estimating borders of some object from a noisy image), we find that the model with $Y \sim \mathcal{U}[-a, a]$ is useful in some instances. Namely, in many cases we have a relatively high contrast between an object and its background as it is the case in Figure 1. It seems reasonable to assume that the data extracted from the green line in Figure 1 come from a uniform distribution, but contaminated by an additive error.

Let f_ε and F_ε be the density and distribution function of the error part ε. Then the density of $X = U + \varepsilon$ is

$$f_X(x; a) = \int_{-\infty}^{\infty} f_U(t) f_\varepsilon(x - t) dt = \frac{1}{2a} \left(F_\varepsilon(x + a) - F_\varepsilon(x - a) \right). \tag{2}$$

If we suppose that the distribution of ε belongs to a scale family, with scale parameter σ, then (2) may be rewritten as

$$f_X(x; a, \sigma) = \frac{1}{2a} \left(F\left(\frac{x + a}{\sigma} \right) - F\left(\frac{x - a}{\sigma} \right) \right),$$

with $F(x)$ being the standard ($\sigma = 1$ and zero mean) distribution function.

Let $\mathbf{x} = (x_1, \ldots, x_n)$ denote the realization of the i.i.d. sample $\mathbf{X} = (X_1, \ldots, X_n)$. The likelihood function has the form

$$L(a, \sigma; \mathbf{x}) = \prod_{i=1}^{n} f_X(x_i; a, \sigma) = \frac{1}{(2a)^n} \prod_{i=1}^{n} \left(F\left(\frac{x_i + a}{\sigma} \right) - F\left(\frac{x_i - a}{\sigma} \right) \right)$$

and the log-likelihood function is given by

$$l(a, \sigma) = -n \log(2a) + \sum_{i=1}^{n} \log \left(F\left(\frac{x_i + a}{\sigma} \right) - F\left(\frac{x_i - a}{\sigma} \right) \right).$$

If the distribution of ε is symmetric around zero, then the Fisher information is

$$I(a) = \frac{-1}{a^2} + \frac{1}{a\sigma^2}\int_0^\infty \frac{\left(f\left(\frac{x+a}{\sigma}\right) + f\left(\frac{x-a}{\sigma}\right)\right)^2}{F\left(\frac{x+a}{\sigma}\right) - F\left(\frac{x-a}{\sigma}\right)}\, dx.$$

Supposing that parameter σ is known or consistently estimated then, under regularity, we have

$$\sqrt{n}\left(\hat{a}_{ML} - a_0\right) \xrightarrow{D} \mathcal{N}\left(0, \frac{1}{I(a_0)}\right), \tag{3}$$

where a_0 is the true value of a.

Some flexibility of our model is achieved by changing the error distribution. Three types of error distributions are available in the package for now. The normal distribution $\varepsilon \sim \mathcal{N}\left(0, \sigma^2\right)$ is sometimes a natural choice. Properties of maximum likelihood (ML) and method of moments (MM) estimators with known σ^2 are given in Benšić and Sabo (2007b). The model with $Y \sim \mathcal{U}[0, a]$ is treated in Schneeweiss (2004). Benšić and Sabo (2010) considered the unknown σ^2 case. The possibility of this (one-dimensional) model in two-dimensional problems is given in Benšić and Sabo (2007a) and Sabo and Benšić (2009). For the sake of robustness Laplace and scaled Student (with 5 degrees of freedom) distributions are also incorporated in the package as a choice of the error distribution. Estimating issues with Laplacian error can be seen in Benšić and Sabo (2016), as well as discussion connected to robustness.

Two procedures for deriving confidence intervals for a are described in (Hamedović et al., 2017). The first one is based on the asymptotic distribution of ML estimator in (3). For a specified $0 < \alpha < 1$, an asymptotic $(1-\alpha)100\%$ confidence interval for a is[1]

$$\left(\hat{a}_{ML} - \frac{z_{\alpha/2}}{\sqrt{nI(\hat{a}_{ML})}}, \hat{a}_{ML} + \frac{z_{\alpha/2}}{\sqrt{nI(\hat{a}_{ML})}}\right).$$

The second method is based on the likelihood ratio statistic

$$\lambda(\mathbf{X}) = \frac{\sup\limits_{H_0} L\left(a; \mathbf{X}\right)}{\sup\limits_{(0,\infty)} L\left(a; \mathbf{X}\right)} = \frac{L\left(a_0; \mathbf{X}\right)}{L\left(\hat{a}_{ML}; \mathbf{X}\right)}.$$

From the asymptotic distribution of log-likelihood ratio statistic

$$-2\log\lambda(\mathbf{X}) \xrightarrow{D} \chi_1^2$$

an approximate $(1-\alpha)100\%$ confidence interval for a is

$$\left\{a \middle| \, l(\hat{a}_{ML}) - l(a) \leq 0.5\chi_1^2(1-\alpha)\right\},$$

where $\chi_1^2(1-\alpha)$ is the $1-\alpha$ quantile of χ_1^2 distribution.

These two approaches can be used to test hypotheses regarding the parameter a. For example, in the case of two-sided hypotheses $H_0 : a = a_0$ versus $H_1 : a \neq a_0$, the critical regions of asymptotic size α are

$$\left\{\mathbf{x} \middle| \, \sqrt{nI(a_0)}|\hat{a}_{ML} - a_0| \geq z_{\alpha/2}\right\}, \text{ and}$$

$$\left\{\mathbf{x} \middle| \, -2\log\lambda(\mathbf{x}) \geq \chi_1^2(1-\alpha)\right\},$$

respectively. Note that both methods are asymptotically equivalent.

Overview of the package

The package **LeArEst** depends on the following packages that should be used, i.e. installed in addition to **LeArEst** installation: **conicfit** (Gama and Chernov, 2015), **jpeg** (Urbanek, 2014) and **opencpu** (Ooms, 2014). The stable version of the package is available on the Comprehensive R Archive Network repository (CRAN; `https://CRAN.R-project.org/`) and can be downloaded and installed by typing the following command in the R console:

```
> install.packages("LeArEst")
```

[1] as usual, z_α is the $1-\alpha$ quantile of standard normal distribution

Function	Short description
`lengthest()`	Performs length estimation from a numerical data set.
`lengthtest()`	Performs one-sided and two-sided tests for uniform distribution half-length.
`areaest()`	Performs area estimation of a numerically described object in plane.
`startweb.esttest()`	Opens default web browser and loads a web page for length estimation and testing (the object of interest is shown in an image).
`startweb.area()`	Opens default web browser and loads a web page for area estimation of the object shown in an image.

Table 1: Overview of **LeArEst** functions

Arguments	Description
`x`	Vector of input data.
`error`	Error distribution.
`var`	Error variance.
`var.est`	Method of error variance estimation.
`conf.level`	Confidence level of the confidence interval. Defaults to 0.95.
Results	**Description**
`radius`	Estimated half-length of the uniform support.
`var.error`	Error variance, estimated or explicitly given by argument `var`.
`conf.int`	Confidence interval for half-length of the uniform support.
`method`	Method used for computing a confidence interval (asymptotic distribution of ML or likelihood ratio statistic).

Table 2: Summary of arguments and results of `lengthest`

The package is loaded using the following command:

```
> library(LeArEst)
```

An overview of the package's functions is given in Table 1.

Length estimation — a numerical data set

The function `lengthest` computes the length of an interval which is the domain of uniform distribution from data contaminated by an additive error according to the model described in the previous section. The function's arguments and results are given in Table 2.

In order to perform length estimation, a type of the error distribution must be chosen with the argument `error` among three possibilities: '`laplace`' (Benšić and Sabo, 2016), '`gauss`' (Benšić and Sabo, 2007b, 2010) or '`student`' (scaled Student distribution with 5 degrees of freedom).

The variance of the additive error may or may not be known. If the variance is known, argument `var` should be used and the variance should be assigned to it. In the case of unknown variance, function `lengthest` implements two methods for its estimation: *Method of Moments* and *Maximum Likelihood*. Value '`MM`' of the argument `var.est` instructs the functions to use Method of Moments, while the corresponding value '`ML`' triggers Maximum Likelihood Method. There is a possibility, depending on input data, that the Method of Moments estimate of error variance does not exist. That being the case, the function stops and outputs the message instructing the user to use Maximum Likelihood estimator or to give an explicit variance. It is important to mention that arguments `var` and `var.est` must not be used simultaneously – one of them must be omitted.

The last argument, `conf.level`, specifies the confidence level of the confidence interval calculated by the function.

Results of this function are the estimated half-length of uniform distribution (i.e. of an object), estimated or explicitly given error variance, confidence interval for half-length (with regard to the given confidence level) and the statistical method for computing a confidence interval.

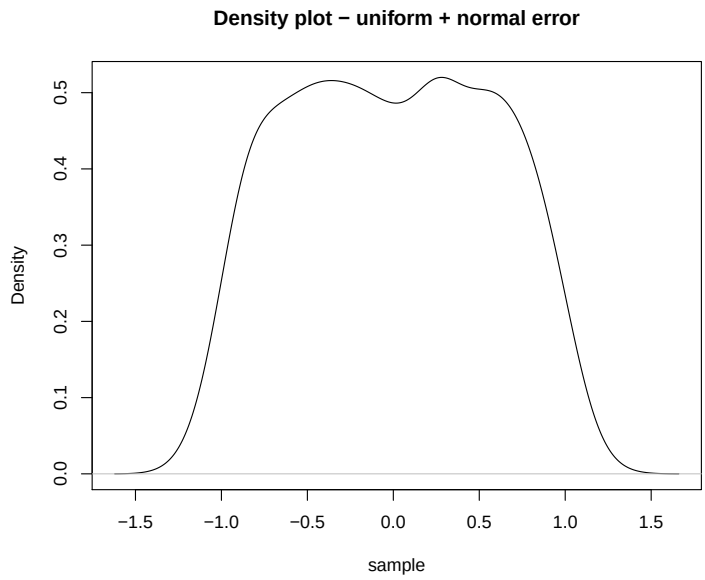

Figure 2: Estimation of the density function

Usage example. Let us generate a sample of size 1000 from $X = U + \varepsilon$, where $U \sim \mathcal{U}[-1,1]$ and $\varepsilon \sim \mathcal{N}\left(0,(0.1)^2\right)$:

```
set.seed(12)
sample_1 <- runif(1000, -1, 1)
sample_2 <- rnorm(1000, 0, 0.1)
sample <- sample_1 + sample_2
```

Figure 2 shows density estimation from the generated data obtained with the R function `density`. A half-length estimation of the uniform support for these data can be done with the following command:

```
lengthest(x = sample, error = "gauss", var.est = "MM", conf.level = 0.90)
```

The most important part of its output is:

```
$radius
MLE for radius (a) of uniform distr.: 0.9916513
$var.error
MM estimate for error variance: 0.01279636
$method
[1] "Asymptotic distribution of LR statistic"
$conf.int
[1] 0.9724316 1.0116479
```

Testing hypothesis — a numerical data set

Function `lengthtest` performs one-sided and two-sided tests against hypothesized half-length of the uniform support as it is described in Section 2.2. Since the actual calculations inside this function are based on the ML approach most input arguments are similar to those in the function `lengthest` (see Table 3). Argument `null.a` is a positive number representing hypothesised half-length of the uniform support, while argument `alternative` defines the usual forms of alternatives (‘`two.sided`’, ‘`greater`’ or ‘`less`’).

Function `lengthtest` also performs length estimation, so all values from its output, except `p.value` and the calculated value of the test statistic (`tstat`), are the same as that of the function `lengthest`.

Usage example. Generate the data in a similar manner as in the `lengthest` example:

```
set.seed(12)
sample_1 <- runif(1000, -1, 1)
sample_2 <- rnorm(1000, 0, 0.1)
sample <- sample_1 + sample_2
```

Arguments	Description
`x`	Vector of input data.
`error`	Error distribution.
`null.a`	Specified null value being tested.
`alternative`	The form of the alternative hypothesis.
`var`	Error variance.
`var.est`	Method of error variance estimation.
`conf.level`	Confidence level of the confidence interval. Defaults to 0.95.
Results	**Description**
`p.value`	*p-value* of the test.
`tstat`	The value of the test statistic.
`radius`	Estimated half-length of the uniform support.
`var.error`	Error variance, estimated or explicitly given by argument `var`.
`conf.int`	Confidence interval for half-length.
`method`	Method used for computing a confidence interval (asymptotic distribution of ML or likelihood ratio statistic).

Table 3: Summary of arguments and results of `lengthtest`

To test that the uniform support half-length equals 1 against that it is less than 1 the function `lengthest` can be used in the following way:

```
lengthtest(x = sample, error = "gauss", alternative = "less", var.est = "MM",
           null.a = 1, conf.level = 0.95)
```

Part of the output dealing with a testing procedure is:

```
$p.value
[1] 0.2418929
$tstat
[1] -0.7002265
```

Area estimation — a numerical data set

An input for the function `areast` is supposed to be a data set of points in plane representing independent realizations of the two-dimensional random vector

$$X = U + \varepsilon.$$

It is assumed that U has uniform distribution on an ellipsoid and ε is a two-dimensional error term independent of U. Arguments and results of this function are listed in Table 4.

The algorithm implemented in the function `areast` is explained in details in Benšić and Sabo (2007a). The main task in area estimation is to estimate edge points of the uniform support. In order to achieve this, the original problem is reduced to several corresponding one-dimensional problems, which can in turn be solved by function `lengthest`.

Let us denote the data set with $D = \{(x_i, y_i), i = 1, \ldots, n\}$. The function `areast` transforms this data set in two different ways: through the y-axis and through the x-axis. The algorithm for transformation through the y-axis is presented below, while the transformation through the x-axis is done analogously.

ALGORITHM 1 (*Transformation through the y-axis (Benšić and Sabo, 2007a)*)

Step 1. Separating through y-axis.

Choose an integer $m < n$ and real numbers $\eta_1 < \eta_2 < \cdots < \eta_m$ such that

(i) $\eta_1 \leq \min\{y_i : i = 1, \ldots n\}$, $\max\{y_i : i = 1, \ldots n\} \leq \eta_m$ and

(ii) $C_k := \{(x_i, y_i) \in D : y_i \in [\eta_k, \eta_{k+1}]\}$, $k = 1, \ldots, m-1$ is a nonempty set.

Step 2. Centring through y-axis.

Let us denote

$$c_k := \frac{1}{|C_k|} \sum_{(x_i,y_i)\in C_k} x_i,\ d_k := \frac{1}{|C_k|} \sum_{(x_i,y_i)\in C_k} y_i,$$

Arguments	Description
`data`	Two-column data matrix containing the points that describe observed object. First column represents x coordinate of the point, while second column represents y coordinate.
`nrSlices`	Number of slices applied for plain data cutting. Defaults to 10.
`error`	Error distribution.
`var`	Error variance.
`var.est`	Method of error variance estimation.
`plot`	Logical parameter that determines whether to plot data set, calculated edge points and the resulting ellipse. Defaults to `FALSE`.
Results	**Description**
`area`	Estimated area of the object.
`points`	Set of estimated object's edge points.
`semiaxes`	Resulting ellipse's semi-axes.

Table 4: Summary of arguments and results of `areaest`

$$k = 1, \ldots, m-1,$$

for $k = 1, \ldots, m-1$ define $\overline{C}_k := \{x_i - c_k : (x_i, y_i) \in C_k\}$.

Using this algorithm the data are transformed in the way that we have sets $\overline{C}_k$, $k = 1, \ldots, m-1$ that represent centred tiny strips. Argument `nrSlices` corresponds to $m-1$ and specifies the number of strips. Lengths of these strips (in x-direction) can be estimated using the function `lengthest` (parameters `error`, `var` and `var.est` are used in `lengthest` call in the way described earlier). After doing so, the algorithm needs to be repeated through the x-axis. Finally, at this point of the procedure, the data that is a noisy version of points from the curve is created – it represents estimated points from border of the object.

The next task is to choose one of the well-known curve fitting procedures for parameter estimation. Here we are dealing with a nonlinear parameter estimation problem.

Let us suppose that we have an elliptical domain, i.e.

$$\mathcal{D}(\mathbf{p}) = \left\{ (x, y) \in \mathbb{R}^2 : \frac{(x-p)^2}{\alpha^2} + \frac{(y-q)^2}{\beta^2} \leq 1 \right\},$$
$$\mathbf{p} = (p, q, \alpha, \beta)^T.$$

On the basis of data obtained so far, the vector of unknown parameters p needs to be estimated, and by doing so, the optimal ellipse that fits into our points is to be defined. For this purpose `EllipseDirectFit` function from the **conicfit** package is used. This function implemets the algebraic ellipse fit method by Fitzgibbon-Pilu-Fisher (Fitzgibbon et al., 1999). Having parameters p, it is a trivial task to calculate the area of the ellipse that approximates the observed object.

Usage example. Two internal files are provided with the package: 'ellipse_3_4_0.1_gauss.txt' and 'ellipse_3_4_0.1_laplace.txt'. Both of them represent an ellipsoidal object with center in point $(1, 1)$, half-axes 1.5 and 2, with added measurement error. In the first file, error distribution is two-dimensional normal with independent margins and variance 0.01, while in the other it is Laplacian ($\lambda = 0.1$) in both directions, also with independent margins.

In order to use one of these files, data need to be loaded into the data frame `inputdata`:

```
inputfile <- system.file("extdata", "ellipse_3_4_0.1_laplace.txt", package = "LeArEst")
inputdata <- read.table(inputfile)
```

Area estimation of the uniform support can be done with the command:

```
areaest(inputdata, error = "laplace", var.est = "ML", nrSlices = 5, plot = TRUE)
```

In the previous example, the parameter `plot` is set to `TRUE`, so the function plots the given input data (black dots), estimated border points (red dots) and the resulting ellipse (cyan ellipse), see Figure 3.

The most important parts of numerical output are:

```
$area
[1] 9.938305
```

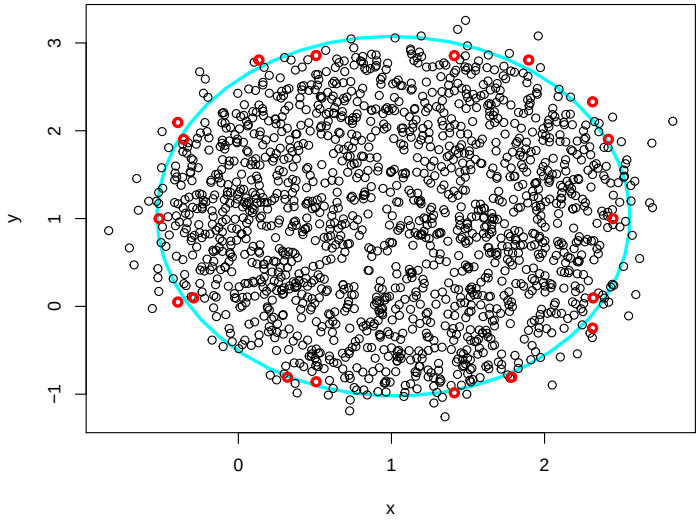

Figure 3: Data points, estimated border points and resulting ellipse obtained by function `areaest`

```
$semiaxes
[1] 2.048028 1.544638
```

Length estimation and testing for an object shown in a picture

In order to apply the described methods to a picture of an object, two web interfaces have been built and embedded into the package.

As far as we know, **shiny** (Chang et al., 2017) provides the simplest way of building web applications using R. However, limitations of its free version discouraged us from using it, so we decided to use **opencpu** package. This package provides a reliable and interoperable HTTP API for data analysis based on R. Basically, it provides an interface between functions in R package and a custom-made web page bundled with the package, using JavaScript and AJAX. Building web interfaces using **opencpu** is more complex than using **shiny**, but at the same time, it provides more flexibility in application design. It is assumed that developers are familiar with HTTP protocol, HTML, and JavaScript language, in order to develop such web applications.

Function `startweb.esttest` will be described in this section. This function takes no arguments and returns no results, its task is to start a web interface for length estimation and hypothesis testing (Figure 4).

To start the analysis using the web interface the picture in JPG format should be loaded (*Load Picture* button). Then, a line should be drawn that intersects the object of interest by clicking on two points in the picture - length estimation will be performed on that line. Finally, parameters for a data set preparation should be set.

`Levels of gray` parameter determines how many levels of grey the algorithm should take into account. It is important to mention that, although color images can be loaded, they are internally converted to grey-scales prior to any calculations. Since JPG format supports 2^{24} different colors, the number of possible colors should be reduced in order to optimize estimation speed and memory consumption.

`Line thickness` specifies how many picture pixels around the drawn line are taken into account in length estimation. For instance, if `Line thickness` is set to 3, the algorithm takes pixels which are direct left neighbours of the line, pixels on the line itself, and the ones which are direct right neighbours of the line. By doing so, the matrix of (length of the line) $\times$ `Line thickness` pixels – *PixelMatrix* is obtained.

By doing so, we have obtained the matrix of (length of the line) $\times$ `Line thickness` pixels – *PixelMatrix*.

By clicking on the *Prepare data* button the data set will be prepared for the inference.

The following step deals with data preparation and is a crucial step of the algorithm. Each pixel

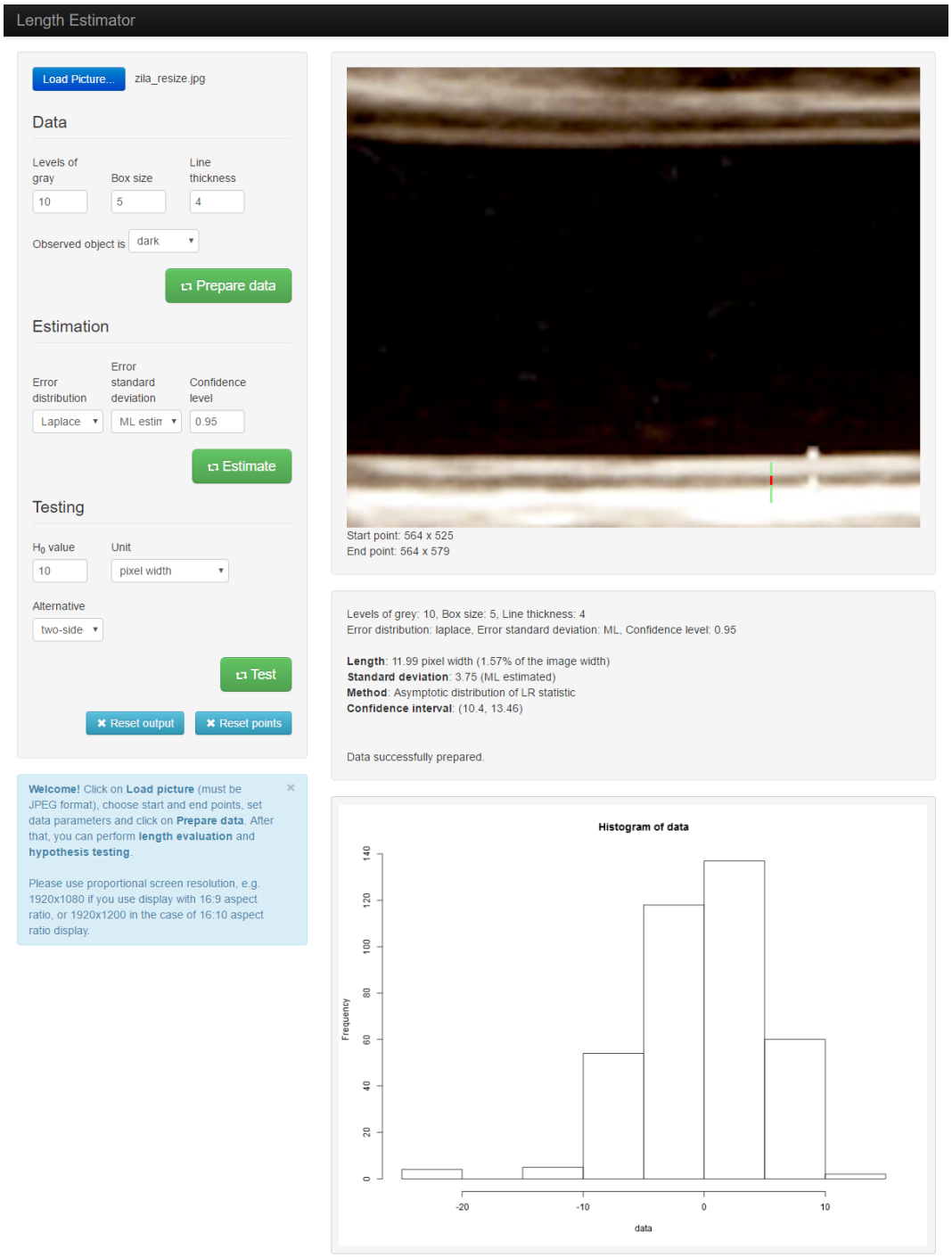

Figure 4: Web interface for length estimation and hypothesis testing. Loaded image shows arterial wall of the carotid artery; we are trying to estimate its intima media thickness (darker layer below artery cavity).

of *PixelMatrix* is mapped to a new matrix of `Box size` $\times$ `Box size` booleans – *DotMatrix* (note that `Box size` is a parameter). Further, every *DotMatrix* is filled with uniformly distributed dots (i.e. `TRUE` values) in a way that total number of dots in each *DotMatrix* corresponds to brightness of pixel it represents. Then, *DotMatrices* are tiled up with respect to the position of corresponding pixel and, by doing so, a new matrix of (length of the line $\cdot$ `Box size`) $\times$ (`Line thickness` $\cdot$ `Box size`) booleans is obtained – *FinalDotMatrix*. The last step is to summarize rows of the *FinalDotMatrix* to obtain a vector of (length of the line) $\cdot$ `Box size` integers. The vector's histogram is shown on a web interface (Figure 4, at the bottom) and the vector itself serves as an input to the functions `lengthest` or `lengthtest`.

Parameters in *Estimation* section of the web interface are transferred to `lengthest`, as well. After

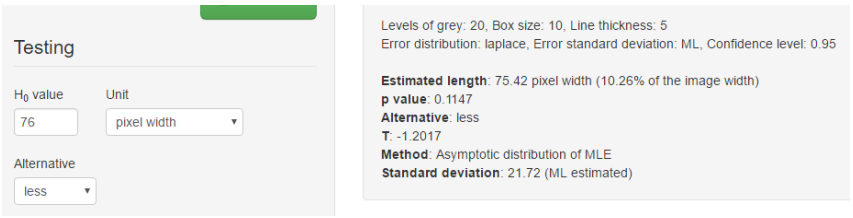

Figure 5: Web interface for length estimation and hypothesis testing - hypothesis testing output

the user clicks on *Estimate* button, `lengthest` is executed, and its results are displayed below the picture. Additionally, the estimated uniform support is marked red on the intersecting line.

Estimated length is expressed in width of a pixel and in percentage of whole image's width as well. As stated in the info box, it is important to use a proportional screen resolution on user's display, so the pixels on the screen are square-shaped.

Testing section of this web interfaces serves for hypothesis testing. Procedures related to image loading, choosing an intersecting line and data preparation are the same as described above. For the purpose of testing, null value `H0`, `unit` and `alternative` ('greater', 'less' or 'two-sided') need to be specified. Part of the web interface dealing with output of hypothesis testing procedure is shown in Figure 5.

Area estimation of an object shown in a picture

The function `startweb.area` starts a web interface for area estimation (Figure 6). Again, first step is to load an image. To select an object whose surface needs to be evaluated, a rectangle should be drawn around it. It is done by clicking on its upper-left and lower-right corners, after which a green rectangle is drawn on the picture.

Data parameters are similar to ones in the length estimation web interface, with the exception of `number of slices`.

The first step in the area estimation algorithm for this function is to roughly isolate the object in the selected rectangle. In order to do that, pixels from the selected rectangle are divided into two clusters by using the `kmeans` function from base-R **stats** package (criterion for clustering is pixel brightness). Further, only pixels from the 'object cluster' are observed and divided into horizontal and vertical stripes, as described earlier in Algorithm 2.3.3. Number of stripes is dictated by the `number of slices` parameter. Length estimation procedure is conducted on each stripe, obtaining two estimated edge points of the object for each stripe (red dots in Figure 5). Two parameters in *Estimation* section of the web interface are related to length estimation procedure of the stripes.

Finally, an optimal ellipse that fits into edge points is found using `EllipseDirectFit` function from the **conicfit** package, as well as in the `areaest` function described earlier. The resulting ellipse is drawn red in Figure 5. Its area is printed below, measured in pixels and percentage of the whole image area.

Concluding remarks

The R package **LeArEst** provides routines for estimating the support of the random variable U, $U \sim \mathcal{U}[-a, a]$, based on a sample from $X = U + \varepsilon$. The random variable ε represents additive measurement error and is supposed to have normal, Laplace or scaled Student distribution with 5 degrees of freedom. It also includes functions for estimating the borders or area of some object from a noisy image. The package may be useful for this purpose mainly in the case of images with reasonable contrast between the object of interest and background. For greater robustness, we find it convenient to use some error distributions with heavier tails. Sometimes we have different amount of noise in the tails, so it would be useful to include some asymmetric error distributions as well. These are some features we are going to add in the package in order to improve flexibility of our models.

Acknowledgement

This work was supported by the Croatian Science Foundation through research grant IP-2016-06-6545. We would like to thank Krunoslav Buljan from Osijek Clinical Hospital Center for providing an image of the carotid artery.

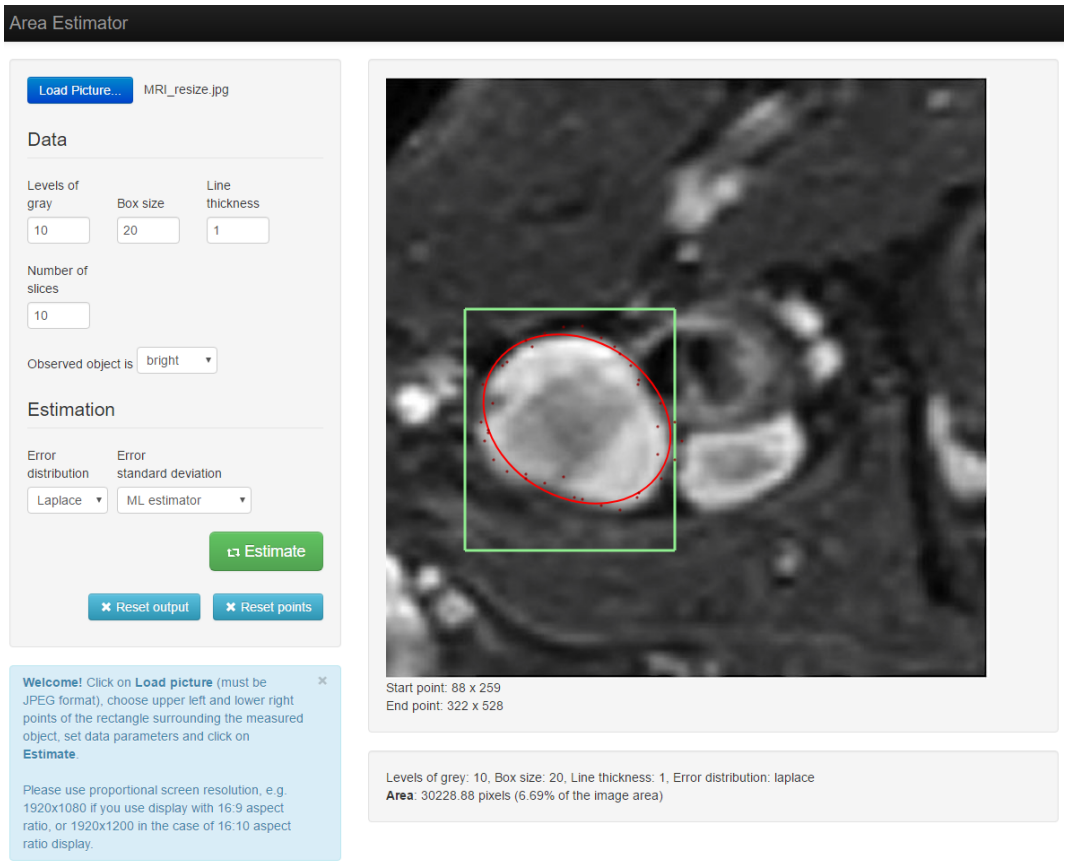

Figure 6: Web interface for area estimation showing an MRI scan detail (taken from Bankman (2008))

Bibliography

I. Bankman. *Handbook of Medical Image Processing and Analysis*. Academic Press series in biomedical engineering. Elsevier Science, 2008. ISBN 9780080559148. [p11]

M. Bensic, S. Hamedovic, K. Sabo, and P. Taler. *LeArEst: Border and Area Estimation of Data Measured with Additive Error*, 2017. R package version 0.1. [p1]

M. Benšić and K. Sabo. Border estimation of a two-dimensional uniform distribution if data are measured with additive error. *Statistics*, 41(4):311–319, 2007a. [p1, 3, 6]

M. Benšić and K. Sabo. Estimating the width of a uniform distribution when data are measured with additive normal errors with known variance. *Computational statistics & data analysis*, 51(9):4731–4741, 2007b. [p1, 3, 4]

M. Benšić and K. Sabo. Estimating a uniform distribution when data are measured with a normal additive error with unknown variance. *Statistics*, 44(3):235–246, 2010. [p1, 3, 4]

M. Benšić and K. Sabo. Uniform distribution width estimation from data observed with laplace additive error. *Journal of the Korean Statistical Society*, 45(4):505–517, 2016. [p1, 3, 4]

J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-8(6):679–698, 1986. [p1]

R. J. Carroll and P. Hall. Optimal rates of convergence for deconvoluting a density. *J. Amer. Statist. Assoc.*, 83:1184–1186, 1988. [p2]

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *Shiny: Web Application Framework for R*, 2017. URL https://CRAN.R-project.org/package=shiny. R package version 1.0.3. [p8]

A. Delaigle and I. Gijbels. Estimation of boundary and discontinuity points in deconvolution problems. *Statistica Sinica*, 16:773–788, 2006. [p2]

A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):476–480, 1999. [p7]

J. Gama and N. Chernov. *Conicfit: Algorithms for Fitting Circles, Ellipses and Conics Based on the Work by Prof. Nikolai Chernov*, 2015. URL https://CRAN.R-project.org/package=conicfit. R package version 1.0.4. [p3]

S. Hamedović, M. Benšić, K. Sabo, and P. Taler. Estimating the size of an object captured with error. Technical report, Department of Mathematics, University of Osijek, 2017. URL http://www.mathos.unios.hr/images/homepages/ksabo/hambensabtal.pdf. Submitted to Central European Journal of Operations Research. [p3]

A. Meister. Support estimation via moment estimation in presence of noise. *Statistics*, 40:259–275, 2006. [p2]

A. Meister. *Deconvolution Problems in Nonparametric Statistics*. Springer-Verlag, 2009. [p2]

J. Ooms. The opencpu system: Towards a universal interface for scientific computing through separation of concerns. *arXiv:1406.4806 [stat.CO]*, 2014. URL http://arxiv.org/abs/1406.4806. [p3]

P. Qiu. *Image Processing and Jump Regression Analysis*, volume 599. John Wiley & Sons, 2005. [p1]

S. E. Ruzin and others. *Plant Microtechnique and Microscopy*, volume 198. Oxford University Press New York, 1999. [p1]

K. Sabo and M. Benšić. Border estimation of a disc observed with random errors solved in two steps. *Journal of computational and applied mathematics*, 229(1):16–26, 2009. [p1, 3]

H. Schneeweiss. Estimating the endpoint of a uniform distribution under measurement errors. *Central European Journal of Operations Research*, 12(2), 2004. [p1, 3]

L. Stefanski and R. J. Carroll. Deconvoluting kernel density estimators. *Statistics*, 21:169–184, 1990. [p2]

J. Stirnemann, A. Samson, F. Comte, and C. Lacour. *Deamer: Deconvolution Density Estimation with Adaptive Methods for a Variable Prone to Measurement Error*, 2012. URL https://CRAN.R-project.org/package=deamer. R package version 1.0. [p2]

I. Tolić, K. Miličević, N. Šuvak, and I. Biondić. Non-linear least squares and maximum likelihood estimation of probability density function of cross-border transmission losses. *IEEE Transactions on Power Systems*, 2017. [p1]

S. Urbanek. *Jpeg: Read and Write JPEG Images*, 2014. URL https://CRAN.R-project.org/package=jpeg. R package version 0.1-8. [p3]

X.-F. Wang and B. Wang. *Decon: Deconvolution Estimation in Measurement Error Models*, 2013. URL https://CRAN.R-project.org/package=decon. R package version 1.2-4. [p2]

Mirta Benšić
Department of Mathematics, University of Osijek
Trg Lj. Gaja 6, HR-31 000 Osijek
Croatia
mirta@mathos.hr

Petar Taler
Department of Mathematics, University of Osijek
Trg Lj. Gaja 6, HR-31 000 Osijek
Croatia
petar@mathos.hr

Safet Hamedović
Faculty of Metallurgy and Materials Science
Travnička cesta 1, BA-72 000 Zenica
Bosnia And Herzegovina
safet.hamedovic@famm.unze.ba

Emmanuel Karlo Nyarko
Faculty of Electrical Engineering, Computer Science and Information Technology, University of Osijek
Kneza Trpimira 2B, HR-31 000 Osijek
Croatia
nyarko@etfos.hr

Kristian Sabo
Department of Mathematics, University of Osijek
Trg Lj. Gaja 6, HR-31 000 Osijek
Croatia
ksabo@mathos.hr