Interactive Illustration of the Sampling Properties of Estimators

Markus Mößler

August 9, 2023

Abstract

This is the documentation of the implementation of a learning module for an interactive illustration of the sampling properties of estimators

1 Introduction

This repository contains the implementation of a learning module with interactive and animated illustrations of fundamental statistical concepts and properties.

2 Goal of the learning module (Why?)

Understanding the concept and properties of sampling distributions of estimators is one of the most important concepts of statistical inference, i.e, of learning from data about the underlying data generating process (DGP) formalized in probabilistic (population) model of interest, and thus, a fundamental part of empirical studies in social science in general and econometrics in particular. Practice, however, has shown that students often have difficulty understanding the concept of sampling distributions and their properties. One potential reason for this is that the sampling properties of estimators are often formulated and analyzed in an abstract way only. The goal of this learning module is to give a more intuitive understanding of the sampling properties of estimators using interactive and animated illustrations of simulation results.

One example is to understand the effect of increasing the sample size n on the sampling properties of the sample average $\overline{X}$ as an estimator for the mean μ of a random variable of interest as stated in the law of large numbers (LLN) and the central limit theorem (CLT). Using interactive illustrations of simulation results the students can increase the sample size and observe how the sample average gets closer to the mean (LLN) and how the sampling distribution of the standardized statistic of the sample average gets closer to the standard normal distribution (CLT).

The topics, i.e. the sample distribution of estimators, is part of every statistics and econometrics course and can be found in any introductory textbook for this field. However, by using interactive and animated illustrations the results we aim to provide a deeper and more intuitive understanding of these concepts. We believe that this kind of understanding is hard to achieve by just attending a *lecture*, reading a *textbook*, or, by chatting with *ChatGPT* about this topic. Certainly, another way to achieve this is to provide the implementation of the simulation studies or to let the students to implement the simulation studies themselves. However, this is often too big a hurdle, especially for undergraduate students.

3 Subject of the learning module (What?)

3.1 General

Subject of this learning module are the sampling properties of estimators for different data generating processes (DGPs). The DGPs are based on a statistical model with a particular parameter of interest, e.g., the mean of a Bernoulli random variable. The parameter(s) of interest of the DGPs are estimated using a particular estimator, e.g., the sample average. This learning module shows the effect of changes in the DGP, e.g., increasing the sample size n, on the sampling distribution of an estimator.

Note, the interactive illustration of the sampling properties of the sample average for increasing the sample size n, i.e., the large sample properties of the sample mean, is only one subject of interest. Other subjects are to understand the effect of omitted variable bias (OVB) and heteroskedasticity on the sampling distribution of the OLS estimator in a simple linear regression model.

3.2 Univariate random variables and sample average

Bernoulli distribution and sample average

This illustration shows the effect of changing the sample size n and the probability of success p of the Bernoulli distribution on the sampling distribution of the sample average as estimator for mean of the Bernoulli distribution.

...

Continuous uniform distribution and sample average

This illustration shows the effect of changing the sample size n and the lower bound a and the upper bound b of the continuous uniform distribution on the sampling distribution of the sample average as estimator for mean of the continuous uniform distribution.

...

3.3 Linear regression model and ordinary least squares

Illustration of the properties of the OLS estimator to estimate the slope coefficient β_1 of a linear regression model, i.e.,

$$Y_i = \beta_0 + \beta_1 X_i + u_i \tag{1}$$

Sampling distribution and sample size

This illustration shows the effect of increasing the sample size n on the sampling distribution of the OLS estimator for the slope coefficient of a simple linear regression model.

...

Sample Size and parameterization of the DGP

This illustration shows the effect of changing the parameters of the DFP, i.e.,

1. changing the sample size n,
2. changing the variance of u_i, i.e., $\sigma^2_{u_i}$, and,
3. changing the variance of X_i, i.e., σ^2_X,

on the sampling distribution of the OLS estimator for the slope coefficient of a simple linear regression model.

...

Effect of Heteroskedasticity

...

Effect of Omitted Variable Bias

...

4 Method of the learning module (How?)

4.1 General

For an interactive and immediate user experience it is useful to separate the simulation study and the presentation of the results. This two-step procedure allows also a flexible implementation, i.e., the simulation studies can be conducted with any suitable software, e.g., *R*, *python*, etc. and the results can be presented interactively using basic web development languages, i.e., *html*, *css* and *javascript*.

Appealing, flexible, interactive and animated presentation using html*,* css *and* javascript

While the setup and the implementation of the simulation or other studies depends on the topic of the learning module, the presentation of the results should be based on core web development tools. The content of each learning module is presented using *html*, the most popular language to build web pages.[1] This allows a very flexible presentation of the results. The styling of the learning module is based on *css*, a popular language for styling web pages, and on *bootstrap*, a popular *css* framework. The interactive and animated feature of the learning module is based on *javascript*, a popular language for programming web pages. The *bootstrap* framework with its grid system allows the development of responsive, mobile-first websites with an appealing design. This enables an appealing experience of the learning also on mobile devices.

Structure of the html, css *and* javascript *files*

What follows in Section 4.3 to 4.3 is an explanation of the structure of the *html*, *css* and *javascript* files. The explanations are based on the illustration of the properties of the sample average as estimator for the mean of a Bernoulli random variable, i.e., based on module `ber-dis-sam-ave` which has two parameters (the sample size n and the probability of success p) and three figures (the bar plot of number of ones and zero, the histogram of sample average and the histogram of the standardized sample average). Note, the implementation is flexible in the sense that it can be used for any learning module with two parameters and three figures.

4.2 Structure of the file with the simulation study

Depending on the subject of the learning module the number of interactive parameters and the number of figures and/or tables for the illustration of the results are chosen. For the learning module `ber-dis-sam-ave` there are two interactive parameters, i,e, sample size n and probability of success p, and three figures, i.e., the bar plot of number of ones and zero, the histogram of sample average and the histogram of the standardized sample average. In this setup the variable and fixed inputs of the simulation study can be defined as in *R* Code Snippet 1 in the Appendix. The simulation study is based on a loop which runs over the variable inputs (here `n.vec` and `p.vec`). For each variable

[1] An introduction to the most important web page development languages can be found here: https://www.w3schools.com/

input combination `RR=10000` realizations of the DGP are simulated and `RR=10000` estimates are calculated. Based on the simulation results for `RR=10000` realizations the illustration of interest is stored as `figure_01_y_z.svg`, where `0x` indicates the `0x`th figure, `y` indicates the `y`th index value of the first parameter and `z` indicates the `z`th index value of the second parameter. E.g. for `ii = 1` and `jj = 1` the results for the parameters $n = 5$ and $p = 0.2$ are stored in `figure_01_1_1.svg` (see also *R* Code Snippet 2 in the Appendix).Note, for the integration into the *html* file and the animation using *javascirpit* file later the naming of the figure and/or tables is important (see Section 4.3).

4.3 Structure of the *html* part

4.3.1 General structure

Distribution of the content across bootstrap containers

The content of the learning module is divided across *bootstrap containers.* The general scheme of the structure is outlined below.

- 1st Container: Header of the module
- 2nd Container with three sections
  - Topic of the module
  - Data generating process (DGP)
  - Estimator and parameter of interest
- 3rd Container:
  - Parameter panel with the slider to change the parameter of interest
  - Illustration panel with four tabs for the illustration of
    * Sample draw: Particular outcome
    * Residual: Particular diagnostic

    * Estimates: Histogram of parameter estimates
    * Std. Estimates: Histogram of standardized parameter etimates

- 4th Container: More Details
  - Simulation Exercise:
  - ...

4.3.2 Interactive and animated implementation

The core components for the interactive and animated implementation in the *html* part consist of six tags. These six tags are introduced below.

Tag 1) Integration of the illustrations

The illustrations are integrated using *img* tags with the *class* attribute `figureCl`. The *src* attributes of the *img* tags point to a particular figure, i.e., the illustration of a particular simulation result. The *id* attributes of the *img* tags are set to, `figure1Id`, `figure2Id`, ..., and used to refer to the *src* attribute of the *img* tag using *javascript*. Code snippet 3 in the Appendix below shows the code of the integration of figure 1 based on a DGP with two interactive parameters, i.e., two slider inputs.

Tag 2) Distribution of the illustrations across tabs

The *img* tags with the *class* attribute `figureCl` are embedded into *div* tags with the *class* attribute `tabContentL1Cl` to distribute the illustrations across tabs, i.e., one tab for each figure. The content of the *div* tags, i.e., the tabs, can be displayed or not displayed using the *display* attribute of the *div* tags. The *id* attributes of the *div* tags for the tabs are set to, `tabContentL1N1Id`, `tabContentL1N1Id`, ..., and used to change the *display* property of the *style* attribute of the *div* tags, i.e, to display or not to display the content of the tabs based on *buttons* using *javascript*. Code snippet 4 in the Appendix below shows the code of the integration of tab 1.

Tag 3) Interaction using sliders

The parametrization of the DGP cintegrationan be changed using *input* tags[2] with the *class* attribute `sliderCl`. The values of the *input* tags will be used to change the *src* attribute of the *img* tags, i.e., to change to the illustration of a particular simulation result. The *id* attributes of the *input* tags are set to, `slider1Id`, `slider2Id`, ... and used to interact with particular sliders using *javascript*. Code snippet 5 Appendix below shows the code for the integration of slider 1.

Tag 4) Displaying the values of the sliders

The value of the sliders, i.e., the underlying specification of the DGP, is displayed using *p* tags with the *class* attribute `sliderValueCl`. The *id* attribute of the *p* tags are set to, `sliderValue1Id`, `sliderValue2Id`, ..., and used to change the `.innerHTML` property of the *p* tags to display the slider values using *javascript*. Code snippet 6 in the Appendix below shows code to display the slider value of slider 1.

Tag 5) Explanations of the figures and the effect of the parametrization

For each figure an overall explanation can be added. Furthermore, for each figure and each slider, i.e., each interactive parameter, a explanation of the effect of the parametrization can be added. All explanations are collected in *div* tags with the *class* attribute `audioTextCl` and the *id* set to, `audioTextFigure1OverallId`, `audioTextFigure2OverallId`, ..., for the explanations of the figures and,
`audioTextFigure1Slider1Id`, `audioTextFigure1Slider2Id`, ...,
`audioTextFigure2Slider1Id`, `audioTextFigure2Slider2Id`, ..., for the explanations of the effects of different parametrization of the DGP. The *id* attributes of the *div* tags are used to refer to specific explanations. Longer explanations can be distributed across multiple *span* tags inside the *div* tag. The distribution across multiple *spans* invokes a break when the explanations is read aloud. Code snippet 7 in the Appendix below shows the integration of an overall explanation text for figure 1. Code snippet 8 in the Appendix below shows the integration of an explanation of the effect of changing

[2] The *bootstrap-slider* library (see `https://github.com/seiyria/bootstrap-slider`) is used to display and interact with the learning module using sliders

parameter 1, i.e., slider 1, based on figure 1.

Tag 6) Displaying the explanations of the figures and the effects of the parametrization The explanation of the figures or the effects of the parametrization are displayed using *p* tags with the *class* attribute `audioShowTextCl`. The *id* attribute of the *p* tags are set to, `audioShowTextFigure1Id`, `audioShowTextFigure2Id`, ..., and used to change the `.innerHTML` property of the *p* tags to display the explanations using *javascript*. Code snippet 9 in the Appendix below shows code to display the explanation for figure 1.

4.4 Structure of the *javascript* part

The core components for the interactive and animated implementation in the *javacript* part consist of a module specific internal *javascript* and a general external *javacript* file called `interactionAndAnimationScript.js` located in the assets subdirectory. The module specific internal *javascript* contains one object called `sliders` which contain the module specific information on the sliders, i.e. (1) the values of the sliders, (2) the initial index of the sliders and (3) the parameter in mathjax syntax. Code snippet 10 in the Appendix below shows module specific *javascript* code for the module `ber-dis-sam-ave`. The interactive and animated implementation is part of the external *javacript* file `interactionAndAnimationScript.js`. The most important functions are (1) `updateFiguresAndSliderValues()` to update the figures based on the current slider value(s), (2) `animateButtonClick(slider)` to animate a slider, e.g., for slider 1, `animateButtonClick(slider = 1)`, and (3) `explainButtonClick()` to explain a figure. Important, the interactive and animated implementation relies on the correct *class* and *id* naming convention of the *html* tags introduced in Section 4.3.

4.5 Integration in the lecture

- The material of this learning module can be provided on a gradual basis using links to the specific illustrations/sub modules or as a complete course/module with a starting page and links to the sub modules.

- The material can be hosted on *GitHub* or on a learning platform such as *ILIAS*. The easiest way to host the material on a learning platform such as *ILIAS* is using a import interface for `.html` structures. In the case of the learning platform *ILIAS* this procedure is quite easy and flexible.

A Appendix

Code Snippet 1: *R* code snippet simulation setup

```
# inputs (variable)
n.vec <- c(5, 10, 25, 50, 100)
p.vec <- c(0.2, 0.4, 0.6, 0.8)

# inputs (fixed)
RR <- 10000
```

Code Snippet 2: *R* code snippet simulation and illustration implementation

```
# simulation/illustration

for (ii in 1:length(n.vec)) {

  for (jj in 1:length(p.vec)) {

    NN <- n.vec[ii]
        p <- p.vec[jj]

        simulation
        tmp.sim <- Y_bar_ber_sim_fun(RR = RR, NN = NN, p = p)

        # plot no 01
        plt.nam <- paste(fig.dir, "figure_01_", ii, "_", jj, ".svg", sep = "")
        svg(plt.nam)

        ...

        dev.off()

        ...

  }

}
```

Code Snippet 3: *Html* code snippet for the integration of figure 1 with two slider inputs

```
<img src="./figures/figure_01_1_1.svg" alt=""
class="figureCl" id="figure1Id" style="max-width: 75%;">
```

Code Snippet 4: *Html* code snippet for integration of tab 1

```
<div id="tabContentL1N2Id" class="tabContentL1Cl">

% content of the tab, i.e., figure

</div>
```

Code Snippet 5: *Html* code snippet for the integration of slider 1

```
<input class="sliderCl" id="slider1Id" type="text"
data-slider-min="0" data-slider-max="4" data-slider-step="1"
data-slider-value="2"/>
```

Code Snippet 6: *Html* code snippet to display the value of slider 1

```
<p style="font-size: 12pt; text-align: center;">
Sample Size \(n\)
</p>
<p style="font-size: 12pt; text-align: center"
class="sliderValueCl" id="sliderValue1Id">
</p>
```

Code Snippet 7: *Html* code snippet for adding an overall explanation of figure 1

```
<div id="audioTextFigure1OverallId" class="audioTextCl">
<span>The figure shows:<br></span>
<span>Blablabla... .</span>
</div>
```

Code Snippet 8: *Html* code snippet for adding an explanation of the effect of changing slider 1 based on figure 1

```
<div id="audioTextFigure01Slider1Id" class="audioTextCl">
<span>Blablablab ... .<br></span>
<span>Blablablab ... .</span>
</div>
```

Code Snippet 9: *Html* code snippet to display the explanation of figure 1

```
<p id="audioShowTextFigure1Id" class="audioShowTextCl"
style="font-size: 10pt; color: red; font-style: italic;
text-align: center; display: none;">
</p>
```

Code Snippet 10: *Javascript* code snippet containing the module specific for the module `ber-dis-sam-ave`

```
// object with all sliders
let sliders = {
        value: {
                slider1: [5, 10, 25, 50, 100],
                slider2: [0.2, 0.4, 0.6, 0.8],
        },
        initialIndex: [2, 2],
        sliderValueParStr: ['\\(n = ', '\\(p = '],
};
```

References

[SW19] James H. Stock and Mark W. Watson. *Introduction to Econometrics*. Pearson, 2019.