Stat4DS / Homework 01

Pierpaolo Brutti

Due Sunday, December 04 (on Moodle)

General Instructions

I expect you to upload your solutions on Moodle as a **single running** `R Markdown` file (`.rmd`) + its `html` output, **named with your surnames**. Alternatively, a `zip`-file with all the material inside will be fine too.

You will give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Your responses must be supported by both textual explanations and the code you generate to produce your results.

R Markdown Test

To be sure that everything is working fine, start `RStudio` and create an empty project called `HW1`. Now open a new `R Markdown` file (`File > New File > R Markdown...`); set the output to `HTML mode`, press `OK` and then click on `Knit HTML`. This should produce a web page with the knitting procedure executing the default code blocks. You can now start editing this file to produce your homework submission.

Please Notice

- For more info on `R Markdown`, check the support webpage that explains the main steps and ingredients: R Markdown from RStudio or, equivalently, read about Quarto. For more info on how to write math formulas in LaTex: Wikibooks.
- Remember our **policy on collaboration**: *collaboration on homework assignments with fellow students is* ***encouraged****. However, such collaboration should be clearly acknowledged, by listing the names of the students with whom you have had* ***discussions*** *(no more) concerning your solution. You may* ***not****, however, share written work or code after discussing a problem with others. The solutions should be written by* ***you and your group*** *only.*

Exercise 1: Stat4Race (2nd ed.)

As you may remember, last week, while solving the most hated exercises from `Test-01`, I felt compelled to turn the coding part of the **Stopping Time** exercise into a drag-racing competition with prizes for the first 3 fastest teams.
Well, a promise is a promise, so here's the details!

Stopping Time

Process: suppose that $X \sim \text{Unif}(0, 1)$, and suppose we *independently* draw $\{Y_1, Y_2, Y_3, \ldots\}$ from yet another $\text{Unif}(0, 1)$ model until we reach the random stopping time T such that $(Y_T < X)$.

Question: it can be shown that the (marginal) PMF of T is such that $\Pr(T = t) = \frac{1}{t(t+1)}$ for $t \in \{1, 2, 3, \ldots\}$.
Setup a simulation in `R` that implements the sampling scheme above in order to numerically check this result.
Quantitatively check how the simulation size impacts the approximation. Make some suitable plot to support your comments and conclusions.

Rules: Some mandatory guidelines for you:

1. You must implement the actual process described above, no shortcuts!
2. In order to level the field, you must: 1. set the following `seed`: 13112221; 2. run it on `Colab` (with an `R` kernel). You will share your notebook with `stat4ta`.
3. The use of additional packages is allowed, but it will be subject to scrutiny by the TA-team.
4. Any sort of explicit code parallelization is allowed.
5. To see how your code scale with the simulation size `M`, you must run the simulation with `M` $= \{100, 1000, 10000, 100000, 1000000, 10000000\}$. Arrange your results in a nicely formatted table, pls!

Evaluation: The overall score for each team is composed as follow:

60% simulation speed;

40% originality and "depth" in the post-processing part (nice plots, interesting analyses, etc) as indisputably assessed by our flawless TA-team.

Prizes: In case of ties, the corresponding prize will be shared.

1st: 1kg of ice-cream;

2nd: cappuccinos and croissants;

3rd: an ultra-rare "*Be Positive*" t-shirt designed, made and signed by me!

In order to assess the speed of your implementation, you can use some simple code like this:

```
beg <- Sys.time()

#--                          --#
# R e l e v a n t   c o d e   h e r e #
#--                          --#

fin <- Sys.time() - beg
print(fin)
```

Exercise 2: Mind your *own* biz...

1. Background: Differential Privacy (more info)

Protecting privacy while performing statistical analysis/learning is quite challenging: the goal of statistics and machine learning is to be as informative as possible, protecting privacy is the complete opposite goal!

How do we formally define privacy? Can we protect privacy and still do an informative analysis? The definition of privacy that has become most common lately (also here) is **differential privacy** (Dwork, 2006; Dwork et al., 2006).

Randomized Response

The predecessor to *differential privacy* is **randomized response** which is a method used in surveys that we know VERY well indeed. It was proposed by Warner in 1965. Let me remind you what we are talking about here in a shorter form.
Imagine I want to know how many of you have ever cheated on a test. Suppose that proportion is p. If I ask this question directly, in all likelihood I will *not* get truthful responses. So, I tell everyone to flip a coin C with $\Pr(C = 1) = \theta$ and $\Pr(C = 0) = 1 - \theta$. To protect your privacy, if the coin is `Tails` I ask you to answer `YES` no matter what, whereas if the coin is `Heads`, you *should* answer honestly the question "*have you every cheated?*".
The observation Y is thus $Y = (1 - C) + C \cdot Z$ where $Z = 1$ if they have cheated and $Z = 0$ otherwise. So $\pi \equiv \Pr(Y = 1)$ is $\pi = (1 - \theta) + \theta \cdot p$ so that $p = (\pi - 1 + \theta)/\theta$. I can then estimate p by estimating π (knowing θ).

Differential Privacy

Let $\mathcal{D}_n = \{X_1, \ldots, X_n\}$ be a generic dataset where $X_i \in \mathcal{X}$. Notice that *knowing the measurement space* $\mathcal{X}$ explicitly is critical for differential privacy!

- **Goal:** report some informative function $Z = T(\mathcal{D}_n)$ of the data.

- **How:** we will use some data dependent *randomization*; that is, we will take $Z \sim Q(\cdot\,|\mathcal{D}_n)$ for some distribution Q.

Let's now qualify better the distribution Q we are after by introducing the following definition. We say that two datasets $\mathcal{D}_n$ and $\mathcal{D}'_n$ are **neighbors**, and we write $\mathcal{D}_n \approx \mathcal{D}'_n$, if they differ in only one random variable. In symbols,

$$\mathcal{D}_n = \{X_1, \dots, X_{i-1}, X_i, X_{i+1}, \dots, X_n\} \quad \text{and} \quad \mathcal{D}'_n = \{X_1, \dots, X_{i-1}, X'_i, X_{i+1}, \dots, X_n\}.$$

Now, we say that the distribution $Q(\cdot)$ satisfies the **ϵ-differential privacy** if

$$Q(Z \in A \,|\, \mathcal{D}_n) \leqslant \mathrm{e}^{\epsilon} \cdot Q(Z \in A \,|\, \mathcal{D}'_n),$$

for all possible sets A and all pairs $\mathcal{D}_n \approx \mathcal{D}'_n$ of neighbors datasets. If $Q(\cdot)$ has a density $q(\cdot)$ this means that

$$\sup_z \frac{q(z \,|\, \mathcal{D}_n)}{q(z \,|\, \mathcal{D}'_n)} \leqslant \mathrm{e}^{\epsilon}. \tag{1}$$

Meaning: this definition means that whether you are in or not in the database has little affect on the output Z. For example, suppose I think you are person i in the database and I want to guess if your value is $X_i = a$ or $X_i = b$. Before I see *any* information, suppose my *prior odds* are $\Pr(X_i = a)/\Pr(X_i = b)$. After I see Z, my *posterior odds* are

$$\frac{\Pr(X_i = a \,|\, Z)}{\Pr(X_i = b \,|\, Z)} = \frac{q(z \,|\, X_i = a)\Pr(X_i = a)}{q(z \,|\, X_i = b)\Pr(X_i = b)} \quad \Rightarrow \quad \mathrm{e}^{-\epsilon}\frac{\Pr(X_i = a)}{\Pr(X_i = b)} \leqslant \frac{\Pr(X_i = a \,|\, Z)}{\Pr(X_i = b \,|\, Z)} \leqslant \mathrm{e}^{\epsilon}\frac{\Pr(X_i = a)}{\Pr(X_i = b)}.$$

Since e^{ϵ} is approximately equal to $1 + \epsilon$ and ϵ is small, we see that knowing Z does **not** change my odds much. So, when differential privacy holds, we cannot learn much about whether a particular person *is* in the dataset or not.

Releasing a Whole Dataset

Real data analysis involves: looking at the data, fitting models, testing fit, making predictions, constructing confidence sets etc. This requires access to the whole data set. This leads to the following questions. Can we release a *privatized* version of the whole dataset? In fact, there are several ways to do this.

A first famous approach due to McSherry and Talwar (2007) is called **Exponential Mechanism** and it is a very general method for preserving differential privacy.

Another way to release an entire privatized dataset starting from an original dataset $\mathcal{D}_n = \{X_1, \dots, X_n\}$ is to compute a *privatized density estimate* $\widehat{q}(\cdot)$. Then we can draw a sample $\mathcal{Z}_k = \{Z_1, \dots, Z_k\} \sim \widehat{q}$ for some suitable value of k. It is easy to show that if $\widehat{q}(\cdot)$ *is* differentially private then so is $\mathcal{Z}_k$ for any choice of k.

Dwork et al. (2006) suggested using a **Privatized Histogram** which was analyzed in Wasserman and Zhou (2010). Here's the details for *one* version of this idea: the **Perturbed Histogram** approach.

The Algorithm: suppose that the data are on $\mathcal{X} = [0, 1]^d$. Divide the space into $m = 1/h$ bins $\{B_1, \dots, B_m\}$ and form the usual *histogram*[1]

$$\widehat{p}_{n,m}(\boldsymbol{x}) = \sum_{j=1}^{m} \frac{\widehat{p}_j}{h^d}\mathbb{I}(\boldsymbol{x} \in B_j) \quad \text{with} \quad \widehat{p}_j = \frac{\{\#\text{ of observations in bin } j\}}{n} = \frac{\widehat{n}_j}{n}. \tag{2}$$

To *privatize* $\widehat{p}_{n,m}(\cdot)$, let $\{\nu_1, \dots, \nu_m\} \overset{\text{iid}}{\sim} \text{Laplace}(\texttt{mean} = 0, \texttt{variance} = 8/\epsilon^2)$. Thus the density of ν_j is $f(\nu) = (\epsilon/4)\mathrm{e}^{-(\epsilon/2)\cdot|\nu|}$. Then define

$$\widehat{q}_{\epsilon,m}(\boldsymbol{x}) = \sum_{j=1}^{m} \frac{\widehat{q}_j}{h^d}\mathbb{I}(\boldsymbol{x} \in B_j) \quad \text{with} \quad \widehat{q}_j = \frac{\widetilde{D}_j}{\sum_{s=1}^{m}\widetilde{D}_s} \quad \text{where} \quad \widetilde{D}_j = \max\{0, D_j\} \quad \text{and} \quad D_j = \widehat{n}_j + \nu_j. \tag{3}$$

Important Result: Wasserman and Zhou (2010) showed that, under some smoothness condition on the true distribution $p_X(\cdot)$ of the original data $\boldsymbol{X}$, if we choose the number of bins m of the order of $n^{d/(2+d)}$ where n is the sample size and d the observation size, we can see that there is no (informational/statistical) loss by releasing the whole privatized histogram $\{\widehat{q}_1, \dots, \widehat{q}_m\}$ **or** a privatized dataset $\mathcal{Z}_k = \{Z_1, \dots, Z_k\}$ sampled from $\widehat{q}_{\epsilon,m}(\cdot)$ for *some* (large enough) value of k.[2]

Comment: this is *not* the whole story. Suppose that the original histogram $\widehat{p}_{n,m}(\cdot)$ is *sparse* i.e. has many empty cells. The privatized histogram $\widehat{q}_{\epsilon,m}(\cdot)$ is forced to "fill in" these empty cells. So in these cases, $\widehat{q}_{\epsilon,m}(\cdot)$ will look very different from $\widehat{p}_{n,m}(\cdot)$. In particular, much of the clustering/sub-population structure will be lost. And if the (*nominal*) data-size d is very large and there is any meaningful *lower dimensional* structure in the data, this will be destroyed.

[1]Here we are thinking in terms of fixing the bin-width h first, and then getting the number of equal-size bins m. Of course we could easily also go the other way around.

[2]Please notice (again), that, in building the dataset $\mathcal{Z}_k$, privacy is assured for *any* k. The problem is achieving statistical accuracy!

2. The Exercise: Comment every line of code + pick nice, meaningful plots to support your results/comments.

$\rightsquigarrow$ Your job $\leftsquigarrow$

1. Let's focus on the univariate case with $d = 1$ so that the the measurement space is the unit interval, $\mathcal{X} = [0, 1]$. Assume also that the true density $p_X(\cdot)$ behind your data X is known and equal to a Beta$(\alpha = 10, \beta = 10)$.
In this part of the exercise you have to setup up a simulation to compare the Mean Integrated Squared Error (`MISE`, see below) between the true model $p_X(\cdot)$ and its two approximations $\widehat{p}_{n,m}(\cdot)$ and $\widehat{q}_{\epsilon,m}(\cdot)$.

$$\texttt{MISE}(p_X, \widehat{p}_{n,m}) = \mathbb{E}_{P_X}\Big(\int_0^1 \big(p_X(x) - \widehat{p}_{n,m}(x)\big)^2 \mathrm{d}x\Big) = \{\texttt{MISE}\text{ between the true model and the original histogram}\},$$
$$\texttt{MISE}(p_X, \widehat{q}_{\epsilon,m}) = \mathbb{E}_{P_X,Q}\Big(\int_0^1 \big(p_X(x) - \widehat{q}_{\epsilon,m}(x)\big)^2 \mathrm{d}x\Big) = \{\texttt{MISE}\text{ between the true model and the privatized histogram}\}.$$

It is **crucial** to understand that here we are dealing with **two** sources of randomness: 1. the randomness due to IID-sampling from the population model $P_X(\cdot)$; 2. the randomness due to the privacy mechanism $Q(\cdot) \rightsquigarrow$ for us, this is the IID-sampling from the `Laplace`. Consequently, for a generic transformation $r(\cdot)$, the expectation $\mathbb{E}_{P_X,Q}(\cdot)$ above should be parsed as

$$\mathbb{E}_{P_X,Q}\big(r(Z_1,\ldots,Z_k)\big) \overset{\texttt{LLS}}{=} \int \Big(\int r(z_1,\ldots,z_k)\mathrm{d}Q(z_1,\ldots,z_k \,|\, x_1,\ldots,x_n)\Big)\mathrm{d}P_X(x_1)\cdots\mathrm{d}P_X(x_n).$$

Once this is clear (and you must ask questions if it's not!), the following, are the relevant simulation parameters to try:

- Pick a large enough simulation size M that does not cook your CPU;
- $n \in \{100, 1000\}$;
- $\epsilon \in \{0.1, 0.001\}$;
- $m \in \text{grid}\big([5, 50]\big)$, size the grid wisely: not too coarse to achieve decent resolution, not too fine to save CPU-time.

2. Repeat the exercise above by replacing the single Beta model with a mixture of 2 Beta's (free to choose their parameters) that must induce some "sparsity" in the resulting histogram $\widehat{p}_{n,m}(\cdot)$. In pseudo-`R` notation, pick

$$p_X(x) = \pi \cdot \texttt{dbeta}(x|\alpha_1, \beta_1) + (1 - \pi) \cdot \texttt{dbeta}(x|\alpha_2, \beta_2),$$

where $\pi \in (0, 1)$ is the probability to pick observations from the first sub-population.
Comparatively comment the results you got under these two different population scenarios: is there informational loss? Explain, possibly also evaluating $\texttt{MISE}(\widehat{q}_{\epsilon,m}, \widehat{p}_{n,m})$.

3. Think hard. Can you figure out a simple/small ($n < 100$, only one variable X) data collection you can realize in less than two weeks where privacy is key? Remember, the idea is that you *can* collect the data, but you do *not* wanna share them as they are (with me in particular) for further statistical analyses.
All right, after the brain-storm, collect the data, privatize them with the *perturbed histogram* approach, and report your analyses to me by sharing a *private* dataset $\mathcal{Z}_k = \{Z_1, \ldots, Z_k\}$ together with some context (e.g. what was the main goal of the analysis, how you got the data, how did you choose m, how did you choose k and ϵ, what happened upon privatization, what are the relevant statistics/summaries I must reproduce from the privatized data, etc).

4. (**Bonus**) Provide *some* evidence to support the claim that the *perturbed histogram* $\widehat{q}_{\epsilon,m}(\cdot)$ in Equation 3 is indeed ϵ-private as defined in Equation 1.