Preregistration

July 8, 2022

1 Preregistration

1.1 Study Information

1.1.1 Title

Benchmarking the Visual Utility of Private Scatterplots

1.1.2 Authors

Anonymous for peer review. Online form on osf.io will list authors upon publication or embargo expiration.

1.1.3 Hypotheses

1. Different algorithms' visual utility will vary depending on the shape of the data. This will be determined by the median utility rating being different for at least two of the algorithms as determined by the Friedman test.
2. Different algorithms' visual utility will vary depending on the task associated with the data. This will be determined by the median utility rating being different for at least two of the algorithms as determined by the Friedman test.
3. Different algorithms' visual utility will vary depending on the privacy level chosen. This will be determined by the median utility rating being different for at least two of the algorithms as determined by the Friedman test.
4. Different algorithms' visual utility will vary depending on the bin size chosen. This will be determined by the median utility rating being different for at least two of the algorithms as determined by the Friedman test.
5. At least two common metrics of utility will have different strengths of association with the visual utility scores generated.

1.1.4 Contributions

1. A comparison of differential privacy algorithms' visual utility across a range of tasks, bin sizes, privacy levels, and scatterplot shapes.
2. Guidance on how to display 2D histrograms privately using differential privacy. Discussion on how to improve the visual quality of the output of the data.
3. An examination of how utility metrics correspond to human-percieved utilty.

1.2 Design Plan

1.2.1 Study type

Observational Study/Data study

1.2.2 Blinding

Experts will not be able to see either the privacy parameter nor the algorithm used. They will only see the two plots side by side.

1.2.3 Study Design

The questions are blocked by task. The experts will therefore first go through all questions related to their ability to identify the same correlation in the private plot as the original plot. Then they will proceed through clusters and distribution questions. Inside of these blocks the questions are randomly ordered. Because this is a data study we are not concerned with ordering effects.

3 raters will see all the stimuli and give a rating for each one.

Five factors:

- Algorithm (DAWA, AHP, AGrid, Laplace, Geometric Truncated)
- Epsilon, ϵ (.5, .1, .05, .01)
- Task (Clusters, Distributions, Correlation)
- Scatterplot shape (20 different shapes)
- Bin Size (32x32, 64x64)

Utility Metrics tested:

- Average per Query Error - Paper, Implementation
- MS-SSIM - Paper, Implementation
- Scagnostics - Paper, Implementation
- Earth Movers Distance - Paper, Implementation
- 1- Paper
- SDMetrics - Paper, Implementation
  - KSTest
  - BNLikelihood

1.3 Sampling Plan

1.3.1 Existing data

Registration prior to creation of data: As of the date of submission of this research plan for pre-registration, the data have not yet been collected, created, or realized.

1.3.2 Data collection procedures

Population: Visualization Experts

Recruitment efforts: Project collaborators

Payment: None

Eligibility: Data Visualization Professor

Timeline: Data collected in July

Data is collected through a study website. There each expert will be able to rate the utility of each plot they see. The data for each expert is exported in a json format indicating the underlying parameters that generated the private plot and the expert rating.

1.3.3 Sample size

1,200 judgements from three expert reviewers

1.3.4 Sample size rationale

Our study is a qualitative data study. We are interested in the way the manipulated variables affected the utility of the data and the not differences of people's perception of utility. Our goal is to test as many parameters as possible and this can be done with a smaller amount of dedicated coders. Previous work done by Seldmair et al. uses two coders when they propose the concept of a qualitative data study. We use three coders since the definition of utility can be hard to pinpoint and having another perspective ensures that our consensus on the private plots utility stems from a broad background. The three coders make 1,200 judgements (5 algorithms x 2 bins sizes x 4 privacy levels x 30 scaterplot/tasks). Not every shape is conducive to all three tasks so each plot was labeled with the tasks an analyst would likely use it for.

We did not do the typical user study with many participants and a power analysis for several reasons. Asking participants to answer over a thousand questions requires an unreasonable amount of time and resources. Additionally, participants recruited from a general pool have likely not done extensive data analysis. All three of our experts are experienced in analyzing scatterplot data for the specified tasks. Therefore, they were able to discuss what utility means for a data analyst and come to an agreed upon ranking of utility related to their experience that would have been impossible if a random population sample had been chosen as the study participants. Finally, the majority variation in the ratings stems from the data and not from the reviewers perception of the data.

Since we are not comparing across individuals but instead comparing across data variables, we first ran a pilot study to ensure the trained coders agreed upon the evaluation of the data. We use an inter-rater reliability (IRR) metric to ensure the coders consistently agree upon their evaluation of utility. A high IRR ensures that the variation in the utility of the private plots arises from the differences in the plot parameters and not from the differences in coder perception. IRR scores range from a score of 0 to 1. As an example, a score of .8 would mean that 80% of the variance is due to the true variance of the underlying data and 20% of the variance is due to the differences in coder ratings. A common practice is to set an a-priori value of inter-rater reliability to ensure there is an appropriate amount of agreement amongst the coders before they are presented with the real data. Our a-priori was set at .6 which is classified as substantial aggreement by Kevin Hallgren. This allows us to attribute the differences in the scores to the data and not to the users.

For our pilot study we found an IRR of .7311 with a 95% confidence interval of [0.68, 0.78]. This was an acceptable score based of Kevin Hallgren's recommendations. The statistical analysis can be found in the inter-coder reliability section.

1.4 Variables

1.4.1 Manipulated variables

We manipulated 5 variables: task type, scatterplot type, privacy level, bin number, algorithm used.

There are 3 tasks that we wanted to ensure the user of the private scatterplot could still complete. They are listed below with the wording of the question from the study.

- Distribution: The distribution of points in space for the graph on the right is comparable to the graph on the left, including the visibility of manifolds and the relative density of each region.
- Correlation: The graph on the right preserves the level of dependence between the two attributes—including non-linear dependence.
- Clusters: The clusters visible in the graph on the left—and no other clusters—are visible on the graph on the right and occur in the same places.

For our scatterplot types we chose scatterplots from each of the 20 categories generated by Pandey et al.. The scatterplots had points added until all scatterplots had 5000 points. The 20 scatterplots can all be seen in `scatterplotStimuli.pdf` and the code to generate them can be found at `scatterplotDensification.ipynb`. The website that the study was conducted with can be found under `pilotPrivatePlotsStudyWebsite.zip` with the pilot data added.

The privacy level was adjusted by epsilon. The four different levels of epsilon chosen were .5, .1, .05, .01. The two different bin numbers used to generate the binned scatterplots were 32 and 64 evenly-sized bins. The five algorithms selected were DAWA, AHP, AGrid, Laplace, and Geometric Truncated. The implementation for DAWA, AHP, and AGrid can be found here and the implementation for Laplace and Geometric Truncated can be found here. The full code to generate the plots can be found at `pilotPrivatePlotGenerationAndStatsErrors.ipynb`.

1.4.2 Measured variables

The ability of the private plot to retain the task utility is evaluated on a 4 point Likert scale. The categories were limited to four for two reasons. The nuances between different ratings of a more fine grained scale would between difficult to discern and our goal was to keep inter rater reliability high to validate our approach of many datasets tested with a small group of people. Our abbreviated scale can be described as: the user [doesn't, suggests, somewhat, does] retain the ability to complete the task using the private scatterplot when compared to the original binned plot.

- 0: Doesn't preserve the feature
- 1: Suggests the feature could exist
- 2: Somewhat preserves the feature
- 3: Preserves the feature very well

To better explain each ratings the reviewers agreed upon the following definitions for each rating:

- Doesn't preserve the feature: I have no confidence that the feature exists.
- Suggests the feature could exist: It looks like the feature might exist but I have low confidence and/or the feature is shown with little clarity.
- Somewhat preserves the feature: I'm confident this feature exists but its fidelity is meaningfully lower than the original plot.
- Preserves the feature very well: I'm confident the feature exists and it is shown with high fidelity relative to the original plot.

1.4.3 Indices

- Median Utility Rating: To create one rating for each stimulus to use for our analysis from the three ratings that are provided by the coders we use the median of the ratings. Since the data is ordinal and does not have an equal defined spacing between the four ratings we do not use the mean. The median provides us with one homogenous rating from the three raters.

- Kendall's τ_b: For each automated utility metric we will calculate Kendall's τ_b for the metric against the median utility rating. This indice will be used to test hypothesis 5.

1.5 Analysis Plan

The current analysis code is run with pilot data. Once the study data is collected the exact same code will be run on the full data.

1.5.1 Inter-Rater Reliability

To ensure that our task questions and structure were clear across all raters, we ran pilot tests to ensure inter-rater reliability was acceptable. To test the interrater reliability we follow the guidelines for choosing intraclass correlation coefficient (ICC) selection as set out by Koo and Li. ICC's are a common way of testing interrater reliatiblity.

There are 10 different forms of ICC that rely on different assumptions. To start we are testing for interrater reliability which should not be mistaken for test-retest reliability or intrarater reliability. Interrater reliability measures the variance of across raters on a set of stimuli (stimuli in our case are the private data plots). The selection of the correct ICC comes by defining the model, type, and definition of relationship. Our model is a two-way random effects model. We choose this model because we have the same set of raters evaluate all the data and we generalize our results to any raters who have the same characteristics as our raters. Type asks how the rating will be used in the application. Multiple raters means the average of the ratings will be used while single rater means we will treat the rating as if it came from a single rater though multiple raters made the judgement. Our study involves ordinal data and therefore the average of the three raters does not work. Therefore, we use the single rater type. Finally, for the relationship of the evaluations we can choose between absolute agreement and consistency. For our ratings we are looking for absolute agreement since we want a consensus on utility.

Therefore our final ICC selection corresponds to a Two-way mixed effects, absolute agreement, single rater/measurement - ICC(3,1).

$\frac{MS_R - MS_E}{MS_R + (k-1)MS_E}$

where MS_R = mean square for rows; MS_E = mean square for error; and k = number of raters/measurements.

We used .6 as our minimum cutoff for the kappa to proceed with the actual study. This corresponds with good and substantial agreement from the papers presented by Cicchetti and Landis and Koch.

[ ]:

```
%%capture

!pip install pingouin
```

[ ]:

```
import pandas as pd
import os
import json
# assign directory
directory = './pilotResults'

# iterate over files in
# that directory
answerDict = {}
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # checking if it is a file
    if os.path.isfile(f):
      # print(f)
        file=open(f)
        data=json.load(file)
        for d in data['answers']:
          if d[0] in answerDict:
            answerDict[d[0]].append(d[1])
          else:
            answerDict[d[0]] = [d[1]]

answerDataFrame = pd.DataFrame.from_dict(answerDict)
answerDataFrame = answerDataFrame.T
answerDataFrame
```

[ ]:

```
import numpy as np

question = answerDataFrame.index.to_numpy()

score0 = answerDataFrame[0].to_numpy()
score1 = answerDataFrame[1].to_numpy()
score2 = answerDataFrame[2].to_numpy()
name0 = np.full(len(score0), 0)
name1 = np.full(len(score1),1)
name2 = np.full(len(score2),2)

questions = np.concatenate((question, question, question))
raters = np.concatenate((name0, name1, name2))
```

```
ratings = np.concatenate((score0, score1, score2))

df = pd.DataFrame({'question': questions, 'rater': raters, 'ratings': ratings})
```

```
[ ]: import pingouin as pg
     icc = pg.intraclass_corr(data=df, targets='question', raters='rater',
                              ratings='ratings')
     icc
```

1.5.2 Algorithm Comparison

The selection of best algorithm for all the subsequent analysis follows a three step process:

1. Using the Friedman test, we ensure that there is a difference between the 5 different algorithms. If the Friedman test returns a p value < .05 we continue on step 2 - the post-hoc analysis.
2. We conduct a post-hoc conover test to see which algorithms significantly differ from each other. This gives us more specific insight into which algorithms specifically are different from one another. The post hoc test is corrected for multiple hypothesis testing using the Benjamini/Hochberg method.
3. We visualize the data to examine the results from the post-hoc conover test to see which of the algorithms that have statistically significant differences in their results yield higher visual utility.

1.5.3 Automated Utility Benchmarking

We want to measure the strength of association between the visual utility ratings generated by our coders and metrics of utility that can be generated computationaly. Our coder rankings are ordinal and the metrics are continuous. When obtaining the association of ordinal-continuous variables Harry Khamis recommends using Kendall's coeficient of rank correlation $_b$. $_b$ can range from -1 (perfect negative association) to 1 (perfect positive assocation).

We calculate every metric (Average per query error, MS-SSIM, Scagnostics, 1/epsilon, Earth Movers Distance, KSTest, BNLikelihood) on every stimulus comparing the binned non private with the binned private plot. We match these plots based on all the parameters (algorithm, task, bin size, shape, epsilon). For each metric we find Kendall's τ_b. We use the implementation provided by scipy.stats.

We are submitting the preregistration without the implementation for all the metrics in our code. Links to the respective libraries we plan to use are provided.

1.5.4 Inference Criteria

To test our hypotheses we will be using a holistic approach. We will look at both the p-value the Friedman test returns as well as examining the distributions found in the data to see if there is a meaningful difference between the algorithms. We will also this approach for the post-hoc analysis.

While we have no specific hypothesis for the statistical utility correlations with visual utility, we will determine which metric is best used by finding the one with the highest positive correlation that has a p-value of less than .05. To test hypothesis 5, we will use the fisher z-transformation to test the different metrics corrrelation coefficients.

P-Value Interpretation This p-value is the conditional probability of the data (or any result even more extreme) assuming:

1. the null hypothesis H0 is exactly true,
2. the study is repeated an infinite number times by drawing random samples from the same populations(s),
3. all distributional requirements are met, and
4. there is no source of error besides sampling or measurement error.

p-value interpretation based on (Kline, 2013]).

Code for analysis

[ ]:

```
%%capture
import statistics
from scipy.stats import friedmanchisquare


!pip install scikit-posthocs
import scikit_posthocs as sp
import matplotlib.pyplot as plt
```

[ ]:

```
medianArray = []
for index, row in answerDataFrame.iterrows():
    medianArray.append( statistics.median([row[0], row[1], row[2]]))

answerDataFrame['totalRating'] = medianArray
```

[ ]:

```
tasks = ['distribution', 'clusters', 'correlation']
epsilons = ['0.5', '0.1', '0.05', '0.01']
algorithms = ['DAWA', 'AHP', "AGrid", 'Geometric', 'Laplace']
bins = ['32','64']
chart = ['0','2','3','9']
```

[ ]:

```
def getFilteredArray(df, name):
  dataFiltered = df.filter(like=name, axis=0)
  DAWA = []
  AHP = []
  AGrid = []
  Geometric = []
  Laplace = []
  exDF = dataFiltered
  while len(exDF.index)>0:
    count = 0
    for e in epsilons:
      for b in bins:
        for t in tasks:
          for c in chart:
            for index, row in exDF.iterrows():
              if count !=5:

                if c+'_' in index and b in index and t in index and e in index:


                  if 'Laplace' in index:

                      Laplace.append(row['totalRating'])
                      exDF = exDF.drop(index)
                      count = count+1
                  if 'AHP' in index:


                    AHP.append(row['totalRating'])
                    exDF = exDF.drop(index)
                    count = count+1
                  if 'AGrid' in index:


                    AGrid.append(row['totalRating'])
                    exDF = exDF.drop(index)
                    count = count+1
                  if 'Geometric' in index:


                    Geometric.append(row['totalRating'])
                    exDF = exDF.drop(index)
                    count = count+1
                  if 'DAWA' in index:


                    DAWA.append(row['totalRating'])
                    exDF = exDF.drop(index)
                    count = count+1

  dfAlgorithms =  pd.DataFrame({'DAWA': DAWA, 'AHP': AHP, 'AGrid': AGrid,␣
 ↪'Geometric': Geometric, 'Laplace': Laplace})
  return dfAlgorithms
```

Step 1: Friedman Test

[ ]:

```
def getFriedmanResult(df):
  stat, p = friedmanchisquare(df['DAWA'], df['AHP'],df['AGrid'],␣
 ↪df['Geometric'], df['Laplace'])
```

```python
    print('Statistics=%.3f, p=%.3f' % (stat, p))
```

Step 2: Post Hoc Conover

[ ]:
```python
def getConoverPostHoc(df):
    newLook = sp.__convert_to_block_df(df)
    pc = sp.posthoc_conover(newLook[0], val_col='y', group_col='groups', p_adjust␣
 ↪= 'fdr_bh')
    pd.set_option('display.float_format', lambda x: '%.5f' % x)
    return pc
```

Step 3: Visualization

[ ]:
```python
def getVisual(df):
    frequencies = {}
    for i in df.columns:
        frequencies[i] = df[i].value_counts()
    plotdata = pd.DataFrame(frequencies)
    plotdata = plotdata.transpose()
    plotdata2 = plotdata.div(plotdata.sum(axis=1), axis=0)*100
    plotdata2.plot(kind="barh", stacked=True)
    plt.legend(bbox_to_anchor=(1.05, 1))
    plt.ylabel('Algorithm')
    plt.xlabel('Cumulative Percent')
    # return plotdata2
```

Best Overall Algorithm

[ ]:
```python
filteredDf = getFilteredArray(answerDataFrame, '0')
```

[ ]:
```python
getFriedmanResult(filteredDf)
```

[ ]:
```python
getConoverPostHoc(filteredDf)
```

[ ]:
```python
getVisual(filteredDf)
```

Different Privacy Levels

Epsilon = .5

[ ]:
```python
filteredDf = getFilteredArray(answerDataFrame, '_0.5_')
```

[ ]:
```python
getFriedmanResult(filteredDf)
```

[ ]:
```python
getConoverPostHoc(filteredDf)
```

[ ]:
```python
getVisual(filteredDf)
```

Epsilon .1

```
[ ]: filteredDf = getFilteredArray(answerDataFrame, '_0.1_')
```

```
[ ]: getFriedmanResult(filteredDf)
```

```
[ ]: getConoverPostHoc(filteredDf)
```

```
[ ]: getVisual(filteredDf)
```

Epsilon .05

```
[ ]: filteredDf = getFilteredArray(answerDataFrame, '_0.05_')
```

```
[ ]: getFriedmanResult(filteredDf)
```

```
[ ]: getConoverPostHoc(filteredDf)
```

```
[ ]: getVisual(filteredDf)
```

Epsilon .01

```
[ ]: filteredDf = getFilteredArray(answerDataFrame, '_0.01_')
```

```
[ ]: getFriedmanResult(filteredDf)
```

```
[ ]: getConoverPostHoc(filteredDf)
```

```
[ ]: getVisual(filteredDf)
```

Different Tasks

Distribution

```
[ ]: filteredDf = getFilteredArray(answerDataFrame, 'distribution')
```

```
[ ]: getFriedmanResult(filteredDf)
```

```
[ ]: getConoverPostHoc(filteredDf)
```

```
[ ]: getVisual(filteredDf)
```

Correlation

```
[ ]: filteredDf = getFilteredArray(answerDataFrame, 'correlation')
```

```
[ ]: getFriedmanResult(filteredDf)
```

```
[ ]: getConoverPostHoc(filteredDf)
```

```
[ ]: getVisual(filteredDf)
```

Clusters

[ ]:
```
filteredDf = getFilteredArray(answerDataFrame, 'clusters')
```

[ ]:
```
getFriedmanResult(filteredDf)
```

[ ]:
```
getConoverPostHoc(filteredDf)
```

[ ]:
```
getVisual(filteredDf)
```

Different Bin Sizes

32 Bins

[ ]:
```
filteredDf = getFilteredArray(answerDataFrame, '32')
```

[ ]:
```
getFriedmanResult(filteredDf)
```

[ ]:
```
getConoverPostHoc(filteredDf)
```

[ ]:
```
getVisual(filteredDf)
```

64 Bins

[ ]:
```
filteredDf = getFilteredArray(answerDataFrame, '64')
```

[ ]:
```
getFriedmanResult(filteredDf)
```

[ ]:
```
getConoverPostHoc(filteredDf)
```

[ ]:
```
getVisual(filteredDf)
```

Statistical Metrics vs Visual Utility

[ ]:
```
import scipy.stats as stats
```

The code for generating the pilot plots and utility metrics csv can be found at pilotPrivatePlotGenerationAndStatsErrors.ipynb

[ ]:
```
dfMetrics = pd.read_csv('./statisticalUtilityMetrics.csv')
```

[ ]:
```
from re import M
import numpy as np
len(answerDataFrame['totalRating'])

metrics = ['Random Query', 'MSSIM']

associationDf = pd.DataFrame(columns = ['tau', 'pValue'])
for m in metrics:
  tau, p_value = stats.kendalltau(dfMetrics['totalRating'], dfMetrics[m])
  row = pd.Series({'tau':tau,'pValue':p_value},name= m)
  associationDf = associationDf.append(row)
```

```
associationDf
```

Correlation Coefficient Comparison We use Fisher's z-transformation to test all the different correlations for any difference. If the p-value is less than .05 for at least one of the comparisons we accept hypothesis 5.

[ ]:
```
# Code copied from https://github.com/psinger/CorrelationStats/blob/master/
 ↪corrstats.py

from scipy.stats import t, norm
from math import atanh, pow
from numpy import tanh

def independent_corr(xy, ab, n, n2 = None, twotailed=True, conf_level=0.95,␣
 ↪method='fisher'):
    """
    Calculates the statistic significance between two independent correlation␣
 ↪coefficients
    @param xy: correlation coefficient between x and y
    @param xz: correlation coefficient between a and b
    @param n: number of elements in xy
    @param n2: number of elements in ab (if distinct from n)
    @param twotailed: whether to calculate a one or two tailed test, only works␣
 ↪for 'fisher' method
    @param conf_level: confidence level, only works for 'zou' method
    @param method: defines the method uses, 'fisher' or 'zou'
    @return: z and p-val
    """

    if method == 'fisher':
        xy_z = 0.5 * np.log((1 + xy)/(1 - xy))
        ab_z = 0.5 * np.log((1 + ab)/(1 - ab))
        if n2 is None:
            n2 = n

        se_diff_r = np.sqrt(1/(n - 3) + 1/(n2 - 3))
        diff = xy_z - ab_z
        z = abs(diff / se_diff_r)
        p = (1 - norm.cdf(z))
        if twotailed:
            p *= 2

        return z, p
    elif method == 'zou':
        L1 = rz_ci(xy, n, conf_level=conf_level)[0]
        U1 = rz_ci(xy, n, conf_level=conf_level)[1]
        L2 = rz_ci(ab, n2, conf_level=conf_level)[0]
        U2 = rz_ci(ab, n2, conf_level=conf_level)[1]
        lower = xy - ab - pow((pow((xy - L1), 2) + pow((U2 - ab), 2)), 0.5)
        upper = xy - ab + pow((pow((U1 - xy), 2) + pow((ab - L2), 2)), 0.5)
        return lower, upper
    else:
        raise Exception('Wrong method!')
```

[ ]:

```
independent_corr(associationDf['tau'][0], associationDf['tau'][1], 1200, 1200)
```