PrivacyUnbiased

```
## Click here for overview of PrivacyUnbiased: bit.ly/PrivUexample
```

Introduction to PrivacyUnbiased

This package implements methods developed in:

- Evans, Georgina, and Gary King (2020): *"Statistically Valid Inferences from Differentially Private Data Releases"*. In: URL: https://gking.harvard.edu/dpd.

In a major development for research data sharing, data providers are beginning to supplement insecure privacy protection strategies, such as "de-identification" with a formal approach called "differential privacy". One version of differential privacy adds specially calibrated random noise to a dataset, which is then released to researchers. This offers mathematical guarantees for the privacy of research subjects while still making it possible to learn about aggregate patterns of interest. Unfortunately, adding random noise creates measurement error, which, if ignored, induces statistical bias — including in different situations attenuation, exaggeration, switched signs, and incorrect uncertainty estimates. The procedures implemented in `PrivacyUnbiased` account for these biases, producing statistically consistent point estimates from differentially private data.

`PrivacyUnbiased`, which corrects statistical problems with privacy protective procedures added to data, is designed to complement `UnbiasedPrivacy`, which corrects statistical problems with privacy protective procedures added to the results of statistical analyses [Evans et al., Working paper].

Installing PrivacyUnbiased

To install `PrivacyUnbiased`, run:

```r
devtools::install_github("georgieevans/PrivacyUnbiased")
library(PrivacyUnbiased)
```

Example

We demonstrate the capabilities of `PrivacyUnbiased` by simulating the scenario described above. We start with a hypothetical private data set (`private_data`). We then add random error to every cell of the data by drawing errors, ϵ_{ik}, from a mean 0 normal distribution, $\epsilon_{ik} \sim \mathcal{N}(0, \sigma_k^2)$. We set σ_k for each of the k columns of the data. This produces a differentially private data set (`dp_data`). In practice, the data analyst would not have access to `private_data` and would only be able to observe `dp_data`.

This example data can be loaded into the R environment (after loading the package) by running the following code:

```r
# Load the private data
data("private_data")

# Load the DP data
data('dp_data')
```

lmdp()

`lmdp()` is the primary function of the package. It returns estimates of bias corrected coefficients from differentially private data, alongside several other quantities. Users can interact with it in a similar way to lm(). There are only two required inputs, the formula and data. For instance:

```r
lmdp_test <- lmdp(Y ~ X1 + X2 + X3, data = dp_data)
```

You can read the documentation for lmdp() by running the code:

```r
?lmdp
```

An important distinction between lmdp() and lm() is that the first row of data must indicate the standard deviations of the DP error added to the rest of the data matrix. For instance, if we look at dp_data, we see by looking at row 1 that no noise was added to Y, the standard error of noise added to $X1$ was 0.7, and so on.

```r
head(dp_data)
```

```
##          Y       X1        X2       X3
## 1  0.00000 0.700000  1.200000 1.000000
## 2 75.91382 5.783422 13.789933 3.519179
## 3 86.24916 7.654997 14.050685 1.177196
## 4 73.44615 5.953728  8.070672 3.759313
## 5 42.39201 4.804003 16.633445 1.129892
## 6 39.04964 4.083823 12.605909 1.751404
```

An exception to this rule is if the argument noise is set to something other than it's default (= NULL). If noise = x (where x is any real number), then lmdp() will automatically set the error for every column to x. In this situation, the first row of the data matrix will be ignored.

The output from lmdp()can be summarized using summary(), just like a standard lm object.

```r
summary(lmdp_test)
```

```
##             Estimate Std. Error   t value Pr(>|t|)
## (Intercept)  10.1021     0.1907   52.9608        0
## X1           11.9862     0.0211  569.2986        0
## X2           -2.9960     0.0142 -210.7348        0
## X3            9.0030     0.0271  332.3592        0
```

The additional output from lmdp() is stored in a list that can be accessed as follows:

```r
# This summarizes the output of an lmdp object
str(lmdp_test)
```

```
## List of 8
##  $ b              : Named num [1:4] 13.9 10.37 -2.1 6.75
##   ..- attr(*, "names")= chr [1:4] "(Intercept)" "X1" "X2" "X3"
##  $ b_vcov         : num [1:4, 1:4] 1.92e-02 -1.41e-05 -9.84e-04 -7.29e-04 -1.41e-05 ...
##  $ beta_tilde     : Named num [1:4] 10.1 12 -3 9
##   ..- attr(*, "names")= chr [1:4] "(Intercept)" "X1" "X2" "X3"
##  $ beta_tilde_vcov: num [1:4, 1:4] 0.036384 0.000307 -0.00189 -0.002161 0.000307 ...
##  $ var_sims       : num [1:500, 1:8] 13.9 13.8 14.1 13.8 13.8 ...
##  $ Sigma_sq_hat   : num [1, 1] 3.71
##  $ vc_pos_def     : logi TRUE
##  $ boot           : logi FALSE
##  - attr(*, "class")= chr "lmdp"
```

```r
# For instance we can access the variance covariance matrix as follows
lmdp_test$beta_tilde_vcov
```

```
##                [,1]          [,2]          [,3]          [,4]
## [1,]  0.0363842837  3.066594e-04 -0.0018900522 -2.161048e-03
## [2,]  0.0003066594  4.432831e-04 -0.0002026790 -4.921453e-05
## [3,] -0.0018900522 -2.026790e-04  0.0002021244  1.990030e-05
## [4,] -0.0021610479 -4.921453e-05  0.0000199003  7.337729e-04
```

The impact of bias correction

It is informative to compare `lmdp()` estimates to the estimates produced from `lm()` that do not adjust for the random error in `dp_data`:

```r
lm_test <- lm(Y ~ X1 + X2 + X3, data = dp_data)

# Biased  OLS estimates
round(summary(lm_test)$coef, 4)
```

```
##             Estimate Std. Error   t value Pr(>|t|)
## (Intercept)  13.8938     0.1552   89.4991        0
## X1           10.3655     0.0170  611.4476        0
## X2           -2.1032     0.0111 -189.5703        0
## X3            6.7456     0.0184  366.8663        0
```

```r
# Notice that if we set noise = 0, lmdp gives the same point estimates as lm()
# Standard errors differ since we use a different estimate procedure

lmdp_test_0 <- lmdp(Y ~ X1 + X2 + X3, data = dp_data, noise = 0)

summary(lmdp_test_0)
```

```
##             Estimate Std. Error   t value Pr(>|t|)
## (Intercept)  13.8938     0.1584   87.7351        0
## X1           10.3655     0.0169  612.8064        0
## X2           -2.1032     0.0116 -181.4450        0
## X3            6.7456     0.0180  374.3985        0
```

We can compare the `lmdp()` estimates and `lm()` estimates to the unbiased estimates on private data.

```r
lm_true <- lm(Y ~ Z1 + Z2 + Z3, data = private_data)

# We see that the lmdp estimates are very close to the lm estimates on private data
# In contrast, the lm estimates appear biased
round(summary(lm_true)$coef, 4)
```

```
##             Estimate Std. Error    t value Pr(>|t|)
## (Intercept)  10.0071     0.0283   353.6061        0
## Z1           11.9973     0.0032  3757.4418        0
## Z2           -2.9997     0.0021 -1421.4332        0
## Z3            9.0020     0.0037  2454.6365        0
```

Variance estimation

The default setting of `lmdp()` is to estimate the standard errors using the simulation method developed in Evans and King [Working paper]. We also offer the option to bootstrap the standard errors by setting the argument `bootstrap_var` to `TRUE`. In general the two methods will produce very similar estimates. The advantage of the simulation method is computational. For large datasets, the bootstrap is essentially infeasible without access to large amounts of computing power. In contrast, the computational time of our simulation procedure scales only slowly in dataset size.

```
# Timing simulation variance estimation
system.time(simulation <- lmdp(Y ~ X1 + X2 + X3, data = dp_data))
```

```
##    user  system elapsed
##   1.144   0.086   1.261
```

```
# Timing bootstrap variance estimation
system.time(bootstrap <- lmdp(Y ~ X1 + X2 + X3, data = dp_data, bootstrap_var = TRUE))
```

```
##    user  system elapsed
##  28.211   6.260  38.702
```

```
# Bootstrap takes ~30 times longer than simulation for dataset of size N = 100000

# The standard error estimates are similar between the two methods:

  # Bootstrap Std. Error
summary(bootstrap)[, "Std. Error"]
```

```
## (Intercept)          X1          X2          X3
##      0.1916      0.0222      0.0147      0.0277
```

```
  # Simulation Std. Error
summary(simulation)[, "Std. Error"]
```

```
## (Intercept)          X1          X2          X3
##      0.1858      0.0218      0.0140      0.0272
```

On small datasets with a relatively large amount of DP error, the variance-covariance matrix we estimate as a paramater to draw random variables may not be positive definite. If this happens, then we use the function `nearPD()` from the package `Matrix`, which finds a close positive definite matrix [Bates and Maechler, 2019]. `lmdp()` will poduce the warning message `VC matrix not positive definite` to alert users to this. The function also returns an indicator variable which records whether the matrix was positive definite which can be accessed as follows:

```
lmdp_test$vc_pos_def
```

```
## [1] TRUE
```

Variable transformation

As discussed in Evans and King [Working paper], transforming variables with random error poses additional complications for estimation. `PrivacyUnbiased` can currently accomdate two types of variable transformation: interaction variables, and squared variables. For example:

```r
# Interaction variable
lmdp_interaction <- lmdp(Y ~ X1 + X2 + X3 + X1*X2, data = dp_data)
summary(lmdp_interaction)
```

```
##             Estimate Std. Error   t value Pr(>|t|)
## (Intercept)   9.2477     0.4155   22.2548   0.0000
## X1           12.1158     0.0612  198.0767   0.0000
## X2           -2.9425     0.0276 -106.6993   0.0000
## X3            9.0034     0.0294  306.1911   0.0000
## X1:X2        -0.0076     0.0033   -2.2985   0.0215
```

```r
# lmdp with interactions produces similar estimates to lm on private data
# Standard errors are lower since Z's do not contain random noise
lm_interaction <- lm(Y ~ Z1 + Z2 + Z3 + Z1*Z2, data = private_data)
round(summary(lm_interaction)$coefficients, 4)
```

```
##             Estimate Std. Error   t value Pr(>|t|)
## (Intercept)   9.9577     0.0622  159.9740   0.0000
## Z1           12.0047     0.0090 1334.0903   0.0000
## Z2           -2.9966     0.0041 -736.7444   0.0000
## Z3            9.0020     0.0037 2454.6331   0.0000
## Z1:Z2        -0.0004     0.0005   -0.8903   0.3733
```

Other variable transformations, or multiple variable transformations, are not allowed in this version of the package and their inclusion will induce an error message. Note also that `lmdp()` currently only supports bootstrap estimation when the model includes transformed variables. For future releasse, we are working on expanding the set of admisssible variable transformations and introducing the simulation approach to variance estimation for these cases.

References

Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2019. URL https://CRAN.R-project.org/package=Matrix. R package version 1.2-18.

Georgina Evans and Gary King. Statistically valid inferences from differentially private data releases, Working paper. URL https://gking.harvard.edu/dpd.

Georgina Evans, Gary King, Margaret Schwenzfeier, and Abhradeep Thakurta. Statistically valid inferences from privacy protected data, Working paper. URL https://gking.harvard.edu/dp.