PrivacyUnbiased

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     0.1897   53.2429        0
## X1           11.9862     0.0234  511.8698        0
## X2           -2.9960     0.0155 -193.6089        0
## X3            9.0030     0.0295  305.7046        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 13
##  $ 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] 1.89e-02 -4.28e-05 -9.53e-04 -7.10e-04 -4.28e-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] 0.036 0.000279 -0.001839 -0.00218 0.000279 ...
##  $ var_sims       : num [1:500, 1:8] 13.5 13.8 13.7 13.8 14 ...
##  $ Sigma_sq_hat   : num [1, 1] 3.72
##  $ vc_pos_def     : logi TRUE
##  $ boot           : logi FALSE
##  $ est_vc         : num [1:14, 1:14] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Y              : num [1:100001] 0 75.9 86.2 73.4 42.4 ...
##  $ X              : num [1:100001, 1:4] 0 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:100001] "" "2" "3" "4" ...
##   .. ..$ : chr [1:4] "(Intercept)" "X1" "X2" "X3"
##  $ S              : num [1:4, 1:4] 0 0 0 0 0 0.49 0 0 0 0 ...
##  $ formula        :Class 'formula'  language Y ~ X1 + X2 + X3
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##  - attr(*, "class")= chr "lmdp"
```

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

```
##               [,1]          [,2]          [,3]          [,4]
## [1,]  0.0359997464  2.785722e-04 -1.839272e-03 -2.179992e-03
## [2,]  0.0002785722  5.483306e-04 -2.669257e-04  8.766347e-05
## [3,] -0.0018392716 -2.669257e-04  2.394643e-04 -6.559098e-05
## [4,] -0.0021799918  8.766347e-05 -6.559098e-05  8.673071e-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`:

```
  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
```

```
# 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     0.1544   89.9642        0
## X1           10.3655     0.0168  615.7564        0
## X2           -2.1032     0.0110 -190.7949        0
## X3            6.7456     0.0191  352.9559        0
```

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

```
  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
##   1.155   0.107   1.332
```

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

```
##    user  system elapsed
##  26.972   5.814  32.960
```

```
# 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
##      0.2006      0.0216      0.0153      0.0302
```

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

```
## (Intercept)          X1          X2          X3
##      0.1980      0.0226      0.0155      0.0289
```

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 accomodate two types of variable transformation: interaction variables, and squared variables. For example:

```
# 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     0.4190   22.0685   0.0000
## X1           12.1158     0.0603  200.8886   0.0000
## X2           -2.9425     0.0277 -106.1773   0.0000
## X3            9.0034     0.0302  297.7194   0.0000
## X1:X2        -0.0076     0.0033   -2.3480   0.0189
```

```
# 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.

Descriptive statistics

`PrivacyUnbiased` also includes a function to calculate estimates of descriptive statistics of the *private data*, using the moment estimation methods described in Evans and King [Working paper]. The primary function for descriptive statistics is `descriptiveDP()`, which takes two arguments, the variable name and the data frame.

```
# Estimate Descriptive statistics
  descriptiveDP(X3, dp_data)
```

```
## Loading required package: polynom
```

```
##                                Mean Std.Dev Skewness Kurtosis
## Before DP noise (estimated) 2.9867  1.7250   0.5924   3.4429
## After DP noise (observed)   2.9867  1.9939   0.3836   3.2481
```

```
# We can compare these estimates to the true values from private_data:
  true_descriptive <- round(c(mean(private_data$Z3),
          sd(private_data$Z3),
          moments::skewness(private_data$Z3),
          moments::kurtosis(private_data$Z3)), 4)

  names(true_descriptive) <- c('Mean', 'Std.Dev', 'Skewness', 'Kurtosis')

  true_descriptive
```

```
##     Mean  Std.Dev Skewness Kurtosis
##   2.9933   1.7256   0.5798   3.3534
```

We will look at simulated heteroskedastic data with zero-inflated covariates to demonstrate methods described in Evans and King [Working paper] for data of this nature. The simualted data can be loaded as follows.

```
# Load the private data
data('private_data2')

# Load the differntially private data
data('dp_data2')
```

Estimating private histograms

The function `distributionDP()` estimates private histograms from the differentialy private data. It takes three required arguments: `variable` should be the name of the column in the data, `data` is the DP data frame, and `distsribution`s is a vector of distributions to paramaterize. The remaining arguments are set as defaults but can be altered by the user.

```
dist_test <- distributionDP(variable = X1, data = dp_data2,
                            distributions = c('Normal', 'Poisson', 'ZIP', 'NB', 'ZINB'),
                            moments_fit = 6, plot = TRUE, plot_dp = TRUE)
```

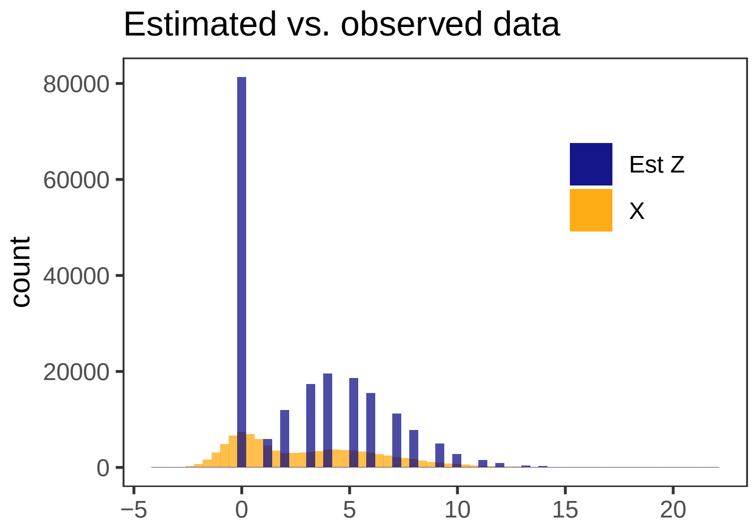
```
## Warning in paramsZIP(X = X, S = S, R = moments_fit, moments_df =
## moments_df, : ZIP estimates outside logical bounds

##
##  Estimated Raw Moments/Implied Distribution Moments:
##
##         mu1  mu2  mu3  mu4  mu5  mu6
## Normal    1 0.99 1.19 1.27 1.46 1.64
## Poisson   1 1.57 2.43 3.74 5.76 8.89
## NB        1 1.01 0.82 0.58 0.36 0.21
## ZINB      1 1.00 1.00 0.99 0.98 0.97
```

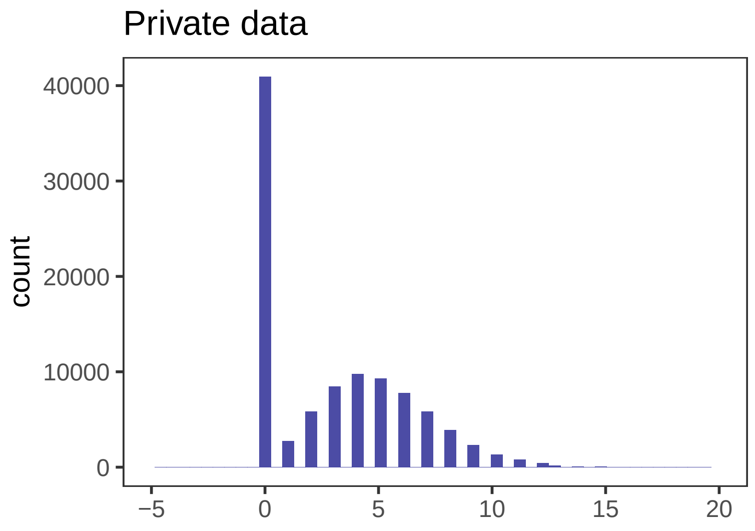
```
# Check that moments were estimated with sufficient precision
dist_test$Normal$moment_precision
```

```
## [1] 332049.11    356.25    221.65    140.64     89.74     57.28
```

```
# So it looks like ZINB fits the data well
# Here we can check this is right by comparing paramaterized distribution to private data
```

```r
# Normally we would not have access to the private data and would rely on the moment fit
dist_test$ZINB$plot
```





Regression diagnostics

The function `diagnosticsDP()` runs regression diagnostics of the kind that would be run after running OLS on private data. It takes a single argument which must be an `lmdp()` object.

```r
# First we run a bias corrected regression
lmdp_obj <- lmdp(Y ~ X1, data = dp_data2)

# Now we can run regression diagnostics
diagnostics <- diagnosticsDP(lmdp_obj)
```

```
##
##  Heteroskedsaticity test via Variance Function Regression:
##
```

```
##              Estimate  Std. Error  t value  Pr(>|t|)
## (Intercept)    9.5007      0.4486  21.1765         0
## X1             2.1201      0.1062  19.9721         0
##
##  Error Normality Test:
##
##  Warning: Error moment estimates are imprecise - unable to accurately test normality of errors
## Skewness 0.07748076           0
## Kurtosis 3.79233169           3
```

```
# We see that there is no strong evidence of non-normality
# We also detect heteroskedasticity since X1 coefficient deviates significantly from 0

# To validate, we can check these results against the private data

# Calculate the true errors
true_coef <- c(2, 8)
true_errors <- private_data2$Y - cbind(1, private_data2$Z1)%*%true_coef

# Validate they are ~ normally distributed

# Skewness of a normal is  0
moments::skewness(true_errors)
```

```
## [1] 0.001887849
```

```
# Kurtosis of a normal is 3
moments::kurtosis(true_errors)
```

```
## [1] 3.453978
```

```
# Test for heteroskedasticity (we obtain similar coefficients to above showing heteroskedasticity)
summary(lm(true_errors^2 ~ private_data2$Z1))$coefficients
```

```
##                  Estimate Std. Error  t value Pr(>|t|)
## (Intercept)      9.922274 0.10570836 93.86462        0
## private_data2$Z1 2.015053 0.02445956 82.38307        0
```

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.