ls-vignette

```
library(ls)
```

1. Verision 0.1.0 Overview

- First version of the package
- To install, devtools::install_github("eleafeit/latent_strat")
- In future versions: upgrade code efficiency

2. Introduction

This package contains the necessary functions of the latent stratification model, which can be used in advertising experiments. This vignette will explain the usage of these functions and provide interpretations to the output.

If you are unfamiliar with the concept, Berman & Feit (2019) offers detailed descriptions for the latent stratification model.

3. Latent Stratification Model

In advertising experiments, noisy responses complicates the estimation of average treatment effect (ATE) and the assessment of the return of interest (ROI). Therefore, running normal t-tests will often yield estimates with high variance. The latent stratification model divides the customers into three strata: always buyer, who are positive under treatment and control (A); influenced buyer, who are positive only under treatment (B), and never buyer, who are zero under treatment and control (C). The model assumes that defiers, who are positive only under control, does not exist. It is able to improve the precision of the estimate by lowering

the variance of the estimate. At the same time, it is also able to improve the accuracy by yielding an ATE closer to the true value.

3.1 Data Preparation

In order to demonstrate the package, we first simulate the data via the `sim_latent_strat()` function. The number following the strata (A for always buyer, B for influenced buyer, and C for never buyer) indicates whether the subject receives treatment. In this case, if the number is 1, then it means that the subject has received treatment, and 0 otherwise. The function takes in the sample size (n), proportion of always buyer (piA), proportion of influenced buyer (piB), mean of always buyer who received treatment (muA1), mean of always buyers who are in control, mean of the influenced buyer who received treatment, and the variance of group A1, A0, and B1. We assume that the treatment proportion is 50%. The `sim_latent_strat()` function generates a list containing a data frame `data`, the ATE `ATE`, and a vector `par`.

```
set.seed(20030601)
sim <- sim_latent_strat(n=10000, piA=0.2, piB=0.1, muA1=5, muA0=4.5, muB1=3, sigma=0.3)
```

The input parameters are stored as the `par` vector. The data will be simulated from these true parameters. For example, 20% of all observations belongs to strata A, who are positive under treatment and control. Observations in strata A who received treatment have a mean outcome of 5, and those who are in control have a mean outcome of 4.5, both with a variance of 0.3.

```
sim$par
#>   piA   piB  muA1  muA0  muB1 sigma
#>   0.2   0.1   5.0   4.5   3.0   0.3
```

The true ATE can be calculated from these true parameters. In this example, the true ATE is 0.4.

```
sim$ATE
#> [1] 0.4
```

Let's now examine the simulated data.

```r
head(sim$data)
#>          y z s   sB   sC
#> 1 3.356879 1 B  1.1 -0.5
#> 2 0.000000 1 C  0.1  0.5
#> 3 0.000000 1 C  0.1  0.5
#> 4 0.000000 1 C  0.1  0.5
#> 5 3.167482 1 B  1.1 -0.5
#> 6 4.749002 1 A -0.9 -1.5
```

The data frame contains the 10000 randomly generated value (y) for each observation based on the strata (s; which can be A, always buyer; B, influenced buyer; or C, never buyer) and treatment (z; 1 for treatment, 0 for control). Since the treatment proportion is 50%, the first 5000 observations will have z = 1. For example, in the first entry, the subject belongs to strata B, who is an influenced buyer that only buys under treatment. Therefore, the subject in the first entry received treatment and has an outcome of 3.36. Following the same logic, the second entry is a subject in the never buyer strata. Therefore, even though he received treatment, his outcome is 0.

3.2 Estimating ATE Using t-Test

One of the most popular and straight-forward ways of estimating the ATE is done by the `t.test()` function in base R. In the code below, we separate the outcome into two groups based on whether the observation has received treatment (z=1, or not z=0), then carry out a t-test with the null hypothesis that the difference in means is zero.

```r
data = sim$data
ttest = t.test(data$y[data$z==1], data$y[data$z==0])
ttest
#>
#>  Welch Two Sample t-test
#>
#> data:  data$y[data$z == 1] and data$y[data$z == 0]
#> t = 11.375, df = 9789.5, p-value < 2.2e-16
#> alternative hypothesis: true difference in means is not equal to 0
```

```
#> 95 percent confidence interval:
#>  0.3639820 0.5155492
#> sample estimates:
#> mean of x mean of y
#> 1.3193750 0.8796094
```

The difference in mean of the two groups can be found by subtracting mean of x and mean of y, which is 1.319 - 0.880 = 0.439. The confidence interval is (0.364, 0.516). With the p-value being less than 2.2e-16, the null hypothesis can be rejected.

Although the standard output shows the accuracy of ATE, it doesn't have information about its precision. So call `stderr` of the t-test object. We can also compute the standard error from the confidence interval: divide half of the difference between the upper and the lower bound by 1.96. As shown below, the standard error of the mean is 0.0387.

```
ttest$stderr
#> [1] 0.03866101
```

3.3 Estimating ATE Using Latent Stratification

The alternative method described in Berman & Feit (2019) is to use the latent stratification model of this package. We simply call the `mle_ls()` function and input the data frame as a parameter.

```
test = mle_ls(sim$data)
#> Optimization runs took 0.1 secs
#> Greatest ll was achieved in run 1 at -7008.303 (worst was -7008.303 )
#> Computing variance-covariance matrix took 0.1  secs
test$pars
#>      par        est          se
#> 1    ATE  0.4032336 0.013019384
#> 2 expATE 13.6626614 0.401495726
#> 3    piA  0.2000032 0.004000203
#> 4    piB  0.1000800 0.004225939
#> 5   muA1  4.9938814 0.009284889
```

```
#> 6   muA0  4.4878030 0.009460486
#> 7   muB1  3.0177472 0.013298883
#> 8  sigma  0.2961601 0.004195862
```

The function returned the analysis result as a data frame. It includes the estimates and standard errors of ATE, strata proportions, strata outcome means, and variance. The latent stratification model provides a more accurate estimate of the ATE. This can be seen by ATE estimate of 0.403, compared to 0.439 in the t-test. In addition, the estimate is also more precise. The standard error of the estimate under the latent stratification model is 0.013, about 66% smaller than that under the t-test (0.039).

3.4 Estimating ATE Using the Oracle Model

By the way, if we actually know the strata, we can compute ATE with the `ATEo()` function, which will produce an even better response. As shown below, the ATE is 0.4030, even smaller than the 0.4032 ATE estimated using the latent stratification model.

```
ATEo(data)
#> [1] 0.4030454
```

4.1 Helper Function

This package is designed such that the only function that basic users need is the mle_ls() function. However, the `mle_ls()` function incorporates many helper functions that can be useful to high-end users. The most notable is the `ls_vcv()` function that is used to calculate the variance co-variance matrix.

4.2 Variance-Covariance Matrix

In the mle optimization, the `ls_vcv()` function is used in another function called `varATEdelta()`, which estimates the standard error of the output via the delta method as stated in White (1982). Aside from this use, you can also use the variance-covariance matrix from the `ls_vcv()` function to find the correlation between parameters. All you need to do is to plug the variance-covariance matrix to the `cov2cor()` function in base R.

```
vcv = ls_vcv(sim$par, sim$data, "hessian")

cov2cor(vcv)

#>                [,1]          [,2]          [,3]          [,4]          [,5]
#> [1,]  1.000000e+00 -0.1185490296 -0.0006384165  1.306327e-05 -0.0007784093
#> [2,] -1.185490e-01  1.0000000000  0.0011386816 -2.329970e-05  0.0013883732
#> [3,] -6.384165e-04  0.0011386816  1.0000000000  6.220982e-04  0.0027311290
#> [4,]  1.306327e-05 -0.0000232997  0.0006220982  1.000000e+00 -0.0014479902
#> [5,] -7.784093e-04  0.0013883732  0.0027311290 -1.447990e-03  1.0000000000
#> [6,]  3.553173e-04 -0.0006337451  0.0169208944  3.676509e-02 -0.0393849199
#>                [,6]
#> [1,]  0.0003553173
#> [2,] -0.0006337451
#> [3,]  0.0169208944
#> [4,]  0.0367650907
#> [5,] -0.0393849199
#> [6,]  1.0000000000
```

The correlation matrix follows the same order as that of the `par` vector. Let's see the order of `par` again.

```
sim$par

#>   piA   piB  muA1  muA0  muB1 sigma
#>   0.2   0.1   5.0   4.5   3.0   0.3
```

So for example, the [1,2] entry of the matrix means that the correlation between piA and piB is -0.119. Like the hessian matrix, the correlation matrix is also symmetrical. The diagonal of the correlation matrix will always be 1 since the correlation between a variable and itself is 1.