Documentation: Federation Functions

```r
source("federated_base_func.R")
source("federated_pool_func.R")
```

In this document, we illustrate how we use the proposed federated inference methods that require only one-time sharing of aggregate data, to obtain the federated point estimates and variance. The data sources are generated with the data simulation scheme as below:

Data generation

In our data generating process, $\mathbf{X}_i = (X_{i1}, X_{i2}, X_{i3})^T \in \mathbb{R}^3$ are i.i.d. samples where each $X_{i,j} \sim \text{unif}(-1, 1)$ is a scalar for $j \in \{1, 2, 3\}$. W_i is a binary treatment variable that follows:

$$\frac{P(W_i = 1 \mid \mathbf{X}_i)}{P(W_i = 0 \mid \mathbf{X}_i)} = \exp(\gamma_c + \gamma_x^T \mathbf{X}_i)$$

where $\gamma_c = 0.1$ and $\gamma_x = [0.2, 0.3, 0.4]$. Y_i is a binary response variable that follows

$$\frac{P(Y_i = 1 \mid \mathbf{X}_i, W_i)}{P(Y_i = 0 \mid \mathbf{X}_i, W_i)} = \exp(\beta_c + \beta_w W_i + \beta_x^T \mathbf{X}_i)$$

where $\beta_c = -0.2$, $\beta_w = -0.3$, $\beta_x = [0.5, 0.7, -0.6]$.

We generate a total 20,000 observations and randomly split these observations into $D = 2$ equally-sized data sets. For the illustration purpose, we consider **ATE** as the estimand.

```r
# two equally-sized data sets
D = 2
# entire simulated sample size
N = 20000
estimand = "ATE"

gamma <- c(0.1, 0.2, 0.3, 0.4)
beta <- c(-0.2, -0.3, 0.5, 0.7, -0.6)
```

```r
expit <- function(x) {
    return(1/(1 + exp(-x)))
}

GenerateData <- function(N, D, seed=123) {
  set.seed(seed)
  subsample_lst <- list()
  for (ix in 1:D) {
    X1 <- runif(N/D, min = -1, max = 1)
    X2 <- runif(N/D, min = -1, max = 1)
```

```
    X3 <- runif(N/D, min = -1, max = 1)

    treat <- rbinom(N/D, 1, expit(gamma[1]+gamma[2]*X1+gamma[3]*X2+gamma[4]*X3))
    y0 <- rbinom(N/D, 1, expit(beta[1]+beta[2]*0+beta[3]*X1+beta[4]*X2+beta[5]*X3))
    y1 <- rbinom(N/D, 1, expit(beta[1]+beta[2]*1+beta[3]*X1+beta[4]*X2+beta[5]*X3))
    y <- y1*treat + y0*(1-treat)
    intercept = rep(1, N/D)
    dat <- data.frame(intercept, y, y1, y0, treat, X1, X2, X3)
    dat$subsample <- ix
    subsample_lst[[ix]] <- dat
  }
  pooled <- do.call("rbind", subsample_lst)
  true_ate <- mean(pooled$y1) - mean(pooled$y0)
  pooled$y <- pooled$y1*pooled$treat + pooled$y0*(1-pooled$treat)

  return(list(true_ate = true_ate, subsample_lst = subsample_lst, pooled = pooled))
}
```

MLE

data set-specific MLE estimation

```
temp <- GenerateData(N=20000, D=2, seed=123)
subsample_lst <- temp$subsample_lst
```

If we are interested in any data set-specific MLE estimations, we can use the `est_mle` function and speicfy the outcome (treatment) and covariates vectors, as well as the working candidate models. For example, we can obtain a MLE estimation for the **treatment model** (`treat.reg` set to `TRUE`) with robust variance estimator on the first dataset as follows:

```
# specify the first dataset
idx <- 1; subsample <- subsample_lst[[idx]]

# specify the working covariates
covariates <- c("X1", "X2", "X3")
# specify the working treatment model
treat.reg.formula <- as.formula(
      paste("treat ~ 0 + intercept + ", paste(covariates, collapse = " + ")))

# get the treatment assignment vector
treat <- subsample$treat
# get the design matrix
X <- model.matrix(treat.reg.formula, data = subsample)
# use robust variance estimator
robust <- TRUE

# MLE estimation for treatment model
result <- est_mle(this.y=treat, this.X=X, robust=TRUE, treat.reg=TRUE, treat.reg.formula = treat.reg.fo
```

This result is a list of MLE estimates including estimated coefficients:

```r
result$est
```

```
## intercept        X1        X2        X3
## 0.1419502 0.2896832 0.2866983 0.4819945
```

The variance covariance matrix (robust sandwich form as we have specified, `result$V`):

```r
result$V
```

```
##              intercept           X1           X2           X3
## intercept 4.154601e-04 1.786187e-05 1.841158e-05 1.197858e-05
## X1        1.786187e-05 1.270287e-03 3.621446e-05 2.623310e-05
## X2        1.841158e-05 3.621446e-05 1.255755e-03 8.630906e-06
## X3        1.197858e-05 2.623310e-05 8.630906e-06 1.255393e-03
```

Besides, it returns the gradient (`result$grad`), outer product of the gradient (`result$outer.grad`), hessian (`result$hessian`) of the log likelihood function evaluated at the MLE estimates, and the estimated probabilities (`result$est.prob`) of being treated.

Similarly, we can obtain a MLE estimation for the **outcome model** (`treat.reg` set to `FALSE`(default)) with robust variance estimator on the first dataset as follows:

```r
# specify the first dataset
idx <- 1; subsample <- subsample_lst[[idx]]

# specify the working covariates
covariates <- c("X1", "X2", "X3")
# specify the working treatment model
reg.formula <- as.formula(
  paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))

# get the outcome vector
y <- subsample$y
# get the design matrix
X <- model.matrix(reg.formula, data = subsample)
# use robust variance estimator
robust <- TRUE

# MLE estimation for outcome model
result <- est_mle(this.y=y, this.X=X, robust=TRUE, treat.reg=FALSE, reg.formula = reg.formula)
```

federation using MLE

stable outcome models / treatment models

We provide an example of federating outcome models, the same approach applied to treatment models by changing the input list of data set-specific response variables as treatment assignments. In this example, we assume stable outcome models across data sets and we use a robust federated variance estimator (`model_misspec=TRUE`).

```r
# specify the working covariates
# for now we assume stable models, so same set of covariates across data sets
covariates <- c("X1", "X2", "X3")

# initialize inputs for federated MLE function
full_y <- vector(mode = "list", length = D)
full_X <- vector(mode = "list", length = D)
full_reg.formula <- vector(mode = "list", length = D)
model_misspec <- TRUE

for (ix in 1:D) {
  subsample <- subsample_lst[[ix]]
  full_y[[ix]] <- subsample$y
  formula <- as.formula(
    paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))

  full_y[[ix]] <- subsample$y
  full_X[[ix]] <- model.matrix(formula, data = subsample)
  full_reg.formula[[ix]] <- formula
}

# restricted federated MLE estimation for outcome model
result <- poolMLE(full_y=full_y, full_X=full_X, full_reg.formula=full_reg.formula, model_misspec=TRUE)
```

The results is a summary table of pooled estimates, standard errors, t statistics and corresponding p-values.

```r
result
```

```
##             Estimate Std. Error    t value Pr(>|z|)
## intercept -0.2105429 0.02178165  -9.666066        0
## treat     -0.2846099 0.03036730  -9.372250        0
## X1         0.5095513 0.02639508  19.304784        0
## X2         0.6811220 0.02632138  25.877140        0
## X3        -0.6025427 0.02654518 -22.698756        0
```

unstable outcome models / treatment models

In this example, we assume unstable outcome models across data sets (`X2` and `X3` are set as unstable covariates) and we use a robust federated variance estimator (`model_misspec=TRUE`).

```r
# specify the working covariates
# for now we assume stable models, so same set of covariates across data sets
covariates <- c("X1", "X2", "X3")
covariates_unstable <- c("X2", "X3")

# initialize inputs for federated MLE function
full_y <- vector(mode = "list", length = D)
full_X <- vector(mode = "list", length = D)
full_reg.formula <- vector(mode = "list", length = D)
model_misspec <- TRUE

for (ix in 1:D) {
```

```
  subsample <- subsample_lst[[ix]]
  full_y[[ix]] <- subsample$y
  formula <- as.formula(
    paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))

  full_y[[ix]] <- subsample$y
  full_X[[ix]] <- model.matrix(formula, data = subsample)
  full_reg.formula[[ix]] <- formula
}

# unrestricted federated MLE estimation for outcome model
result <- poolMLE(full_y=full_y, full_X=full_X,
                  unstable=TRUE, num_datasets=D, unstable_covars = covariates_unstable,
                  full_reg.formula=full_reg.formula, model_misspec=TRUE)
```

The results is a summary table of pooled estimates, standard errors, t statistics and corresponding p-values.

```
result
```

```
##                Estimate Std. Error    t value     Pr(>|z|)
## treat        -0.2846783 0.03036726  -9.374511 0.000000e+00
## X1            0.5096438 0.02639534  19.308099 0.000000e+00
## I1_intercept -0.2124535 0.02645318  -8.031301 8.881784e-16
## I1_X2         0.6777567 0.03712294  18.257084 0.000000e+00
## I1_X3        -0.5791784 0.03710985 -15.607131 0.000000e+00
## I2_intercept -0.2084741 0.02643308  -7.886866 3.108624e-15
## I2_X2         0.6841610 0.03709005  18.445945 0.000000e+00
## I2_X3        -0.6263463 0.03774205 -16.595447 0.000000e+00
```

Note that we can also specify different working models and specify the non-overlapped covariates of the working models as the unstable covariates as the following example is showing:

```
# specify the working covariates
# for now we assume stable models, so same set of covariates across data sets
covariates <- list(c("X1", "X2", "X3"), c("X1", "X3"))
covariates_unstable <- c("X2")

# initialize inputs for federated MLE function
full_y <- vector(mode = "list", length = D)
full_X <- vector(mode = "list", length = D)
full_reg.formula <- vector(mode = "list", length = D)
model_misspec <- TRUE

for (ix in 1:D) {
  subsample <- subsample_lst[[ix]]
  full_y[[ix]] <- subsample$y
  formula <- as.formula(
    paste("y ~ 0 + intercept + treat + ", paste(covariates[[ix]], collapse = " + ")))

  full_y[[ix]] <- subsample$y
  full_X[[ix]] <- model.matrix(formula, data = subsample)
  full_reg.formula[[ix]] <- formula
```

```
}

# unrestricted federated MLE estimation for outcome model
result <- poolMLE(full_y=full_y, full_X=full_X,
                  unstable=TRUE, num_datasets=D, unstable_covars = covariates_unstable,
                  full_reg.formula=full_reg.formula, model_misspec=TRUE)
```

```
result
```

```
##                Estimate Std. Error    t value Pr(>|z|)
## treat        -0.2499984 0.02999831  -8.333748        0
## X1            0.4971398 0.02609173  19.053537        0
## X3           -0.5968037 0.02627352 -22.715024        0
## I1_intercept -0.2307550 0.02635871  -8.754411        0
## I1_X2         0.6747739 0.03706040  18.207410        0
## I2_intercept -0.2165478 0.02602194  -8.321738        0
```

IPW-MLE

data set-specific IPW-MLE estimation

If we are interested in any data set-specific IPW-MLE estimations, we can use the `est_ht` function and speicfy the outcome, treatment, covariates vectors, as well as the working candidate models. For example, we can obtain a IPW-MLE estimation for ATE (`estimand="ATE"`) with estimated propensity scores on the first dataset as follows:

```
# specify the first dataset
idx <- 1; subsample <- subsample_lst[[idx]]

# specify the working covariates
covariates <- c("X1", "X2", "X3")
# specify the working treatment model
treat.reg.formula <- as.formula(
      paste("treat ~ 0 + intercept + ", paste(covariates, collapse = " + ")))
# specify the working outcome model
reg.formula <- as.formula(
      paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))

# get the outcome vector
y <- subsample$y
# get the treatment assignment vector
treat <- subsample$treat
# get the design matrix (with treatment column)
X.tilde <- model.matrix(reg.formula, data = subsample)
# get the design matrix (no treatment column)
X <- model.matrix(treat.reg.formula, data = subsample)

# IPW-MLE estimation for treatment model
result <- est_ht(this.y=y, this.X.tilde=X.tilde, this.X=X, this.treat=treat,
                 reg.formula=reg.formula, treat.reg.formula = treat.reg.formula,
                 estimand="ATE", estimated_propensity=TRUE)
```

This result is a list of IPW-MLE estimates including estimated coefficients for outcome model (`result$est`):

```
result$est
```

```
##   intercept      treat         X1         X2         X3
## -0.1864882 -0.3480863  0.5480803  0.6835608 -0.5719763
```

The variance covariance matrix (`result$V`):

```
result$V
```

```
##               intercept         treat            X1            X2            X3
## intercept  9.734854e-04 -0.0009733726  0.0000785715  6.605915e-05  1.479886e-04
## treat     -9.733726e-04  0.0018669168 -0.0001925042 -1.821724e-04 -1.735410e-04
## X1         7.857150e-05 -0.0001925042  0.0014691801  1.193809e-04 -4.204430e-05
## X2         6.605915e-05 -0.0001821724  0.0001193809  1.457098e-03 -6.940819e-05
## X3         1.479886e-04 -0.0001735410 -0.0000420443 -6.940819e-05  1.444768e-03
```

And the estimated probabilities for outcomes (`result$est.prob`).

federation using IPW-MLE

stable models

In this example, we assume stable outcome models and stable treatment models across data sets. We assume the propensity scores are estimated (`estimated_propensity=TRUE`) and the target estimand is ATE (`estimand="ATE"`).

```
# specify the working covariates
# for now we assume stable models, so same set of covariates across data sets
covariates <- c("X1", "X2", "X3")

# initialize inputs for federated IPW-MLE function
full_y <- vector(mode = "list", length = D)
full_X.tilde <- full_X <- vector(mode = "list", length = D)
full_treat <- vector(mode = "list", length = D)
full_reg.formula <- full_treat.reg.formula <- vector(mode = "list", length = D)

for (ix in 1:D) {
  subsample <- subsample_lst[[ix]]

  reg.formula <- as.formula(
    paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))
  treat.reg.formula <- as.formula(
    paste("treat ~ 0 + intercept + ", paste(covariates, collapse = " + ")))

  full_y[[ix]] <- subsample$y
  full_treat[[ix]] <- subsample$treat
  full_X[[ix]] <- model.matrix(treat.reg.formula, data = subsample)
  full_X.tilde[[ix]] <- model.matrix(reg.formula, data = subsample)
  full_reg.formula[[ix]] <- reg.formula
```

```
  full_treat.reg.formula[[ix]] <- treat.reg.formula
}

# restricted federated IPW-MLE estimation
result <- poolHT(full_y=full_y, full_X.tilde=full_X.tilde, full_X=full_X, full_treat=full_treat,
                 unstable=FALSE,
                 full_reg.formula=full_reg.formula, full_treat.reg.formula=full_treat.reg.formula,
                 estimand="ATE", estimated_propensity=TRUE)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

The results is a summary table of pooled estimates, standard errors, t statistics and corresponding p-values.

```
result
```

```
##             Estimate Std. Error    t value Pr(>|z|)
## intercept -0.2122661 0.02198841  -9.653541        0
## treat     -0.2847750 0.03041323  -9.363522        0
## X1         0.5152052 0.02682732  19.204498        0
## X2         0.6786824 0.02673363  25.386838        0
## X3        -0.6029444 0.02693749 -22.383094        0
```

unstable outcome models / treatment models

In this example, we assume unstable outcome models and unstable treatment models across data sets. We assume the propensity scores are estimated (`estimated_propensity=TRUE`) and the target estimand is ATE (`estimand="ATE"`).

```
# specify the working covariates
# for now we assume stable models, sosame set of covariates across data sets
covariates <- c("X1", "X2", "X3")
covariates_unstable <- c("X2", "X3")

# initialize inputs for federated IPW-MLE function
full_y <- vector(mode = "list", length = D)
full_X.tilde <- full_X <- vector(mode = "list", length = D)
full_treat <- vector(mode = "list", length = D)
full_reg.formula <- full_treat.reg.formula <- vector(mode = "list", length = D)

for (ix in 1:D) {
  subsample <- subsample_lst[[ix]]

  reg.formula <- as.formula(
    paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))
  treat.reg.formula <- as.formula(
    paste("treat ~ 0 + intercept + ", paste(covariates, collapse = " + ")))

  full_y[[ix]] <- subsample$y
  full_treat[[ix]] <- subsample$treat
  full_X[[ix]] <- model.matrix(treat.reg.formula, data = subsample)
```

```r
  full_X.tilde[[ix]] <- model.matrix(reg.formula, data = subsample)
  full_reg.formula[[ix]] <- reg.formula
  full_treat.reg.formula[[ix]] <- treat.reg.formula
}

# restricted federated IPW-MLE estimation
result <- poolHT(full_y=full_y, full_X.tilde=full_X.tilde, full_X=full_X, full_treat=full_treat,
                 unstable=TRUE, unstable_covars = covariates_unstable, num_datasets = D,
                 full_reg.formula=full_reg.formula, full_treat.reg.formula=full_treat.reg.formula,
                 estimand="ATE", estimated_propensity=TRUE)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

The results is a summary table of pooled estimates, standard errors, t statistics and corresponding p-values.

```r
result
```

```
##                Estimate Std. Error    t value     Pr(>|z|)
## treat        -0.2850349 0.03041431  -9.371734 0.000000e+00
## X1            0.5156843 0.02682576  19.223475 0.000000e+00
## I1_intercept -0.2160791 0.02679283  -8.064811 6.661338e-16
## I1_X2         0.6796142 0.03778805  17.984895 0.000000e+00
## I1_X3        -0.5703963 0.03784860 -15.070472 0.000000e+00
## I2_intercept -0.2084478 0.02672656  -7.799275 6.217249e-15
## I2_X2         0.6781071 0.03763991  18.015640 0.000000e+00
## I2_X3        -0.6352524 0.03843071 -16.529814 0.000000e+00
```

AIPW

data set-specific AIPW estimation

If we are interested in any data set-specific AIPW estimations, we can use the `est_aipw` function and speicfy the outcome, treatment, covariates vectors, as well as the working candidate models. For example, we can obtain a AIPW estimation for ATE (`estimand="ATE"`) on the first dataset as follows:

```r
# specify the first dataset
idx <- 1; subsample <- subsample_lst[[idx]]

# specify the working covariates
covariates <- c("X1", "X2", "X3")
# specify the working treatment model
treat.reg.formula <- as.formula(
      paste("treat ~ 0 + intercept + ", paste(covariates, collapse = " + ")))
# specify the working outcome model
reg.formula <- as.formula(
      paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))

# get the outcome vector
```

```r
y <- subsample$y
# get the treatment assignment vector
treat <- subsample$treat
# get the design matrix (with treatment column)
X.tilde <- model.matrix(reg.formula, data = subsample)
# get the design matrix (no treatment column)
X <- model.matrix(treat.reg.formula, data = subsample)

result <- est_aipw(this.y=y, this.X.tilde=X.tilde, this.X=X, this.treat=treat, estimand=estimand,
                   treat.reg.formula=treat.reg.formula, reg.formula=reg.formula)
```

The results is a list of estimation outputs including the ATE estimate (`result$est`), the corresponding variance (`result$V`) and the influence functions (`result$influence.function`).

federation using AIPW

stable models

In this example, we assume stable models across data sets. Our target estimand is ATE (`estimand="ATE"`).

```r
# specify the working covariates
# for now we assume stable models, so same set of covariates across data sets
covariates <- c("X1", "X2", "X3")

# initialize inputs for federated AIPW function
full_y <- vector(mode = "list", length = D)
full_X.tilde <- full_X <- vector(mode = "list", length = D)
full_treat <- vector(mode = "list", length = D)
full_reg.formula <- full_treat.reg.formula <- vector(mode = "list", length = D)

for (ix in 1:D) {
  subsample <- subsample_lst[[ix]]

  reg.formula <- as.formula(
    paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))
  treat.reg.formula <- as.formula(
    paste("treat ~ 0 + intercept + ", paste(covariates, collapse = " + ")))

  full_y[[ix]] <- subsample$y
  full_treat[[ix]] <- subsample$treat
  full_X[[ix]] <- model.matrix(treat.reg.formula, data = subsample)
  full_X.tilde[[ix]] <- model.matrix(reg.formula, data = subsample)
  full_reg.formula[[ix]] <- reg.formula
  full_treat.reg.formula[[ix]] <- treat.reg.formula
}

# restricted federated AIPW estimation
result <- poolAIPW(full_y=full_y, full_X.tilde=full_X.tilde, full_X=full_X, full_treat=full_treat,
                   unstable=FALSE,
                   full_reg.formula=full_reg.formula, full_treat.reg.formula=full_treat.reg.formula,
                   estimand="ATE")
```

The returned summary table reports the pooled estimates, standard errors, t statistics and corresponding p-values of the ATE estimation.

```
result
```

```
##         Estimate  Std. Error   t value Pr(>|z|)
## [1,] -0.06401707 0.006803952 -9.408807        0
```

unstable models

In this example, we assume unstable models across data sets. Assume that we still use ATE as our target estimand (`estimand="ATE"`).

```
# specify the working covariates
# we assume unstable models
covariates <- c("X1", "X2", "X3")

# initialize inputs for federated AIPW function
full_y <- vector(mode = "list", length = D)
full_X.tilde <- full_X <- vector(mode = "list", length = D)
full_treat <- vector(mode = "list", length = D)
full_reg.formula <- full_treat.reg.formula <- vector(mode = "list", length = D)

for (ix in 1:D) {
  subsample <- subsample_lst[[ix]]

  reg.formula <- as.formula(
    paste("y ~ 0 + intercept + treat + ", paste(covariates, collapse = " + ")))
  treat.reg.formula <- as.formula(
    paste("treat ~ 0 + intercept + ", paste(covariates, collapse = " + ")))

  full_y[[ix]] <- subsample$y
  full_treat[[ix]] <- subsample$treat
  full_X[[ix]] <- model.matrix(treat.reg.formula, data = subsample)
  full_X.tilde[[ix]] <- model.matrix(reg.formula, data = subsample)
  full_reg.formula[[ix]] <- reg.formula
  full_treat.reg.formula[[ix]] <- treat.reg.formula
}

# unrestricted federated AIPW estimation
result <- poolAIPW(full_y=full_y, full_X.tilde=full_X.tilde, full_X=full_X, full_treat=full_treat,
                   unstable=TRUE,
                   full_reg.formula=full_reg.formula, full_treat.reg.formula=full_treat.reg.formula,
                   estimand="ATE")
```

The returned summary table reports the pooled estimates, standard errors, t statistics and corresponding p-values of the ATE estimation.

```
result
```

```
##         Estimate  Std. Error   t value Pr(>|z|)
## [1,] -0.06415986 0.006805098 -9.428205        0
```

We can specify different working models across data sets and then use unrestricted federated AIPW estimator:

```
# specify the working covariates
# we assume unstable models which include different sets of covariates
covariates <- list(c("X1", "X2", "X3"), c("X1", "X3"))

# initialize inputs for federated AIPW function
full_y <- vector(mode = "list", length = D)
full_X.tilde <- full_X <- vector(mode = "list", length = D)
full_treat <- vector(mode = "list", length = D)
full_reg.formula <- full_treat.reg.formula <- vector(mode = "list", length = D)

for (ix in 1:D) {
  subsample <- subsample_lst[[ix]]

  reg.formula <- as.formula(
    paste("y ~ 0 + intercept + treat + ", paste(covariates[[ix]], collapse = " + ")))
  treat.reg.formula <- as.formula(
    paste("treat ~ 0 + intercept + ", paste(covariates[[ix]], collapse = " + ")))

  full_y[[ix]] <- subsample$y
  full_treat[[ix]] <- subsample$treat
  full_X[[ix]] <- model.matrix(treat.reg.formula, data = subsample)
  full_X.tilde[[ix]] <- model.matrix(reg.formula, data = subsample)
  full_reg.formula[[ix]] <- reg.formula
  full_treat.reg.formula[[ix]] <- treat.reg.formula
}

# unrestricted federated AIPW estimation
result <- poolAIPW(full_y=full_y, full_X.tilde=full_X.tilde, full_X=full_X, full_treat=full_treat,
                   unstable=TRUE,
                   full_reg.formula=full_reg.formula, full_treat.reg.formula=full_treat.reg.formula,
                   estimand="ATE")
```

And here is the returned summary table of the federated ATE estimation:

```
result
```

```
##         Estimate  Std. Error   t value Pr(>|z|)
## [1,] -0.05761795 0.006853241 -8.407402        0
```