SDR vs OLS Experiments When True DGP is Linear

Rajveer Jat

2024-10-23

Introduction

Sufficient Dimension Reduction (SDR) is a powerful method to identify a lower-dimensional subspace of the predictors without losing any predictive power. We use it in the context of instrumental variable regression to address endogeneity.

In this markdown document, we run some experiments to compare the SDR with OLS when the true DGP is linear. One naturally expect OLS to beat SDR, however as we move core assumptions of OLS, SDR gains in terms of mean squared error performance (MSE). We run 100 replications. We encourgae reader to try different experiments and share their insights.

Experiment 1: SDR Uses Linear Model in First-Stage

Our parameter of interest is β_1 which is the coefficient of x in a linear model of $y = \beta_0 + \beta_1 x + \varepsilon$.

We get two estimates of β_1 using OLS and SDR methods n_rep (say 100 replications) times and then compare the mean squared error. Followings are the steps:

1. *OLS*: i) We obtain first stage coefficients using OLS and

   ```
   ii) Use $\hat{x}$ in second stage to estimate parameter of interest $\beta_1$
   ```

2. *SDR*: i) For linear models, one SDR is enough. We fuse the given p number of instruments into one SDR index.

   ```
   ii) Use this Index to get predicted $\hat{x}$.

   iii) We replace true $x$ by this $\hat{x}$ in the equation of $y$ to obtain $\beta_1$.
   ```

DGP 1.1: IID Data Generating Process

$$
\begin{aligned}
&Z \sim \mathcal{N}(0, I_p), \quad Z \in \mathbb{R}^{n \times p} \\
&\begin{pmatrix} e \\ \epsilon \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right) \\
&x = Z\gamma + e, \quad \gamma = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} \\
&y = \beta_1 + \beta_1 x + \epsilon
\end{aligned}
$$

ρ measures endogeneity. We start with $\rho = 0$, the IID case.

Let's choose $p = 3, \beta_1 = 0, \beta_2 = 2, n = 500, \rho = 0$

```
library(MASS)  # For generating multivariate normal distribution
```

```
## Warning: package 'MASS' was built under R version 4.2.3
```

```
library(dr)    # For SDR method (SIR)
```

```
## Warning: package 'dr' was built under R version 4.2.3
```

```
OLS_SDR_2SLS = function(Z, rho) {

  n =dim(Z)[1]


  # Generate y = 0 + 2*x + epsilon where (e, epsilon) ~ Bivariate Normal
  Sigma = matrix(c(1, rho, rho, 1), 2, 2)
  errors = mvrnorm(n, mu = c(0, 0), Sigma = Sigma)
  e = errors[, 1]   # Error term for x
  epsilon = errors[, 2]  # Error term for y
  gamma=as.matrix(c(2,3,1), ncol=1)

  x=Z %*% gamma  +e
  y = 2 * x + epsilon  # True y with epsilon



  # 1. First Stage: Predict x using OLS
  data_x = data.frame(x, Z)
  colnames(data_x) = c("x", paste0("Z", 1:ncol(Z)))

  # OLS Model
  ols_model = lm(x ~ ., data = data_x)
  x_hat_ols = predict(ols_model, data_x)  # Predicted x using OLS

  # SDR Model (using SIR)
  sir_model = dr(x ~ ., data = data_x, method = "sir")
  sir_directions = sir_model$evectors  # SIR directions
  x_hat_sdr = as.matrix(Z) %*% sir_directions[, 1]  # Predicted x using SDR

  # 2. Second Stage: Use predicted x to estimate y

  # Second stage: regress y on predicted x (from OLS and SDR)
  # OLS-based prediction
  second_stage_ols = lm(y ~ x_hat_ols)
  y_hat_ols = predict(second_stage_ols)

  # SDR-based prediction
  second_stage_sdr = lm(y ~ x_hat_sdr)
  y_hat_sdr = predict(second_stage_sdr)
```

```
  # Return the results: OLS and SDR coefficient estimates for the second stage
  return(list(
    OLS_Coefficients_Stage2 = coef(second_stage_ols)[-1],  # Exclude intercept
    SDR_Coefficients_Stage2 = coef(second_stage_sdr)[-1],  # Exclude intercept
    OLS_MSE_Stage2 = mean((y - y_hat_ols)^2),
    SDR_MSE_Stage2 = mean((y - y_hat_sdr)^2)
  ))
}

# Repeat the experiment for 100 iterations
set.seed(42)
n_reps = 100
n = 500
p = 3
rho = 0  # Correlation between e and epsilon
ols_stage2_store = numeric(n_reps)
sdr_stage2_store = numeric(n_reps)

for (i in 1:n_reps) {
  # Generate p regressors (Z)
  Z = matrix(rnorm(n * p), ncol = p)
  result = OLS_SDR_2SLS(Z, rho)

  # Store MSE values for OLS and SDR in second stage
  ols_stage2_store[i] = result$OLS_MSE_Stage2
  sdr_stage2_store[i] = result$SDR_MSE_Stage2
}

# Print summary of results
cat("Summary of OLS Stage 2 MSE values:\n")
```

```
## Summary of OLS Stage 2 MSE values:
```

```
print(summary(ols_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.245   4.769   4.988   4.964   5.177   5.855
```

```
cat("\nSummary of SDR Stage 2 MSE values:\n")
```

```
##
## Summary of SDR Stage 2 MSE values:
```

```
print(summary(sdr_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.259   4.777   5.001   4.976   5.198   5.848
```

Results: OLS has 0.24% lesser MSE.

DGP 1.2: Data Generating Process with Endogeneity

$$
\begin{aligned}
&Z \sim \mathcal{N}(0, I_p), \quad Z \in \mathbb{R}^{n \times p} \\
&\begin{pmatrix} e \\ \epsilon \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right) \\
&x = Z\gamma + e, \quad \gamma = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} \\
&y = \beta_1 + \beta_1 x + \epsilon
\end{aligned}
$$

$\rho \neq 0$ in this case, we check for $\rho = \{0.2, 0.5, 0.9\}$.

Let's choose $p = 3, \beta_1 = 0, \beta_2 = 2, n = 500$

```
# Repeat the experiment for 100 iterations
set.seed(42)
n_reps = 100
n = 500
p = 3
rho = 0.5  # Correlation between e and epsilon
ols_stage2_store = numeric(n_reps)
sdr_stage2_store = numeric(n_reps)

for (i in 1:n_reps) {
  # Generate p regressors (Z)
  Z = matrix(rnorm(n * p), ncol = p)
  result = OLS_SDR_2SLS(Z, rho)

  # Store MSE values for OLS and SDR in second stage
  ols_stage2_store[i] = result$OLS_MSE_Stage2
  sdr_stage2_store[i] = result$SDR_MSE_Stage2
}

# Print summary of results
cat("Summary of OLS Stage 2 MSE values:\n")
```

```
## Summary of OLS Stage 2 MSE values:
```

```
print(summary(ols_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   6.110   6.647   6.968   6.948   7.250   8.040
```

```
cat("\nSummary of SDR Stage 2 MSE values:\n")
```

```
##
## Summary of SDR Stage 2 MSE values:
```

```
print(summary(sdr_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   6.112   6.666   6.996   6.964   7.255   8.075
```

Results:

1. OLS has 0.55% lesser MSE than SDR when $\rho = 0.2$.
2. OLS has 0.23% lesser MSE than SDR when $\rho = 0.5$.
3. OLS has 0.15% lesser MSE than SDR when $\rho = 0.9$.

DGP 1.3: Data Generating Process with Endogeneity and Errors are AR(1)

$$
\begin{aligned}
&Z \sim \mathcal{N}(0, I_p), \quad Z \in \mathbb{R}^{n \times p} \\
&\begin{pmatrix} \eta_t \\ \zeta_t \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right) \\
&\epsilon_t = \alpha_1 \cdot \epsilon_{t-1} + \eta_t \qquad e_t = \alpha_2 \cdot e_{t-1} + \zeta_t, \qquad \text{(AR(1) process)} \\
&x = Z\gamma + e, \quad \gamma \text{ is a vector of length p generated from } Uniform[1, 2] \\
&y = 2x + \epsilon
\end{aligned}
$$

Let's choose $p = 3, \beta_1 = 0, \beta_2 = 2, n = 500$

```
library(MASS)  # For generating multivariate normal distribution
library(dr)    # For SDR method (SIR)

# Function to generate AR(1) errors
generate_AR1_errors = function(n, rho, alpha=0.5) { #if no AR(1) coefficient supplied, we set it to 0.5
  # Variance-covariance matrix
  Sigma = matrix(c(1, rho, rho, 1), 2, 2)

  # Generate white noise errors
  errors = mvrnorm(n, mu = c(0, 0), Sigma = Sigma)

  # Initialize AR(1) errors
  e = numeric(n)
  epsilon = numeric(n)

  # AR(1) process with coefficient 0.5
  for (t in 2:n) {
    e[t] = alpha * e[t-1] + errors[t, 1]
    epsilon[t] = alpha * epsilon[t-1] + errors[t, 2]
  }

  return(list(e = e, epsilon = epsilon))
}
```

```r
OLS_SDR_2SLS = function(Z, rho, alpha=0.5) {

  n =dim(Z)[1]
  p=dim(Z)[2]

  # Generate AR(1) errors
  errors = generate_AR1_errors(n, rho, alpha)
  e = errors$e   # Error term for x
  epsilon = errors$epsilon  # Error term for y

  gamma=as.matrix(runif(p,min=1,max=2), ncol=1)

  x=Z %*% gamma  +e
  y = 2 * x + epsilon  # True y with epsilon

  # 1. First Stage: Predict x using OLS
  data_x = data.frame(x, Z)
  colnames(data_x) = c("x", paste0("Z", 1:ncol(Z)))

  # OLS Model
  ols_model = lm(x ~ ., data = data_x)
  x_hat_ols = predict(ols_model, data_x)  # Predicted x using OLS

  # SDR Model (using SIR)
  sir_model = dr(x ~ ., data = data_x, method = "sir")
  sir_directions = sir_model$evectors  # SIR directions
  x_hat_sdr = as.matrix(Z) %*% sir_directions[, 1]  # Predicted x using SDR

  # 2. Second Stage: Use predicted x to estimate y

  # Second stage: regress y on predicted x (from OLS and SDR)
  # OLS-based prediction
  second_stage_ols = lm(y ~ x_hat_ols)
  y_hat_ols = predict(second_stage_ols)

  # SDR-based prediction
  second_stage_sdr = lm(y ~ x_hat_sdr)
  y_hat_sdr = predict(second_stage_sdr)

  # Return the results: OLS and SDR coefficient estimates for the second stage
  return(list(
    OLS_Coefficients_Stage2 = coef(second_stage_ols)[-1],  # Exclude intercept
    SDR_Coefficients_Stage2 = coef(second_stage_sdr)[-1],  # Exclude intercept
    OLS_MSE_Stage2 = mean((y - y_hat_ols)^2),
    SDR_MSE_Stage2 = mean((y - y_hat_sdr)^2)
  ))
}

# Repeat the experiment for 100 iterations
set.seed(42)
n_reps = 100
n = 500
p = 3
```

```r
alpha=0.5
rho = 0.5  # Correlation between e and epsilon
ols_stage2_store = numeric(n_reps)
sdr_stage2_store = numeric(n_reps)

for (i in 1:n_reps) {
  # Generate p regressors (Z)
  Z = matrix(rnorm(n * p), ncol = p)
  result = OLS_SDR_2SLS(Z, rho, alpha)

  # Store MSE values for OLS and SDR in second stage
  ols_stage2_store[i] = result$OLS_MSE_Stage2
  sdr_stage2_store[i] = result$SDR_MSE_Stage2
}

# Print summary of results
cat("Summary of OLS Stage 2 MSE values:\n")
```

```
## Summary of OLS Stage 2 MSE values:
```

```r
print(summary(ols_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.095   8.695   9.060   9.081   9.478  10.855
```

```r
cat("\nSummary of SDR Stage 2 MSE values:\n")
```

```
##
## Summary of SDR Stage 2 MSE values:
```

```r
print(summary(sdr_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.095   8.709   9.071   9.086   9.491  10.850
```

Result

AR(1)'S effect (with no endogeneity)

1. For $(\rho = 0, \alpha = 0.5)$, OLS has 0.14% lesser MSE than SDR.
2. For $(\rho = 0, \alpha = 0.9)$, OLS has 0.041% lesser MSE than SDR.

AR(1) errors with low endogeneity 1. For $(\rho = 0.2, \alpha = 0)$, OLS has 0.087% lesser MSE than SDR.

2. For $(\rho = 0.2, \alpha = 0.5)$, OLS has 0.067% lesser MSE than SDR.
3. For $(\rho = 0.2, \alpha = 0.9)$, OLS has 0.035% lesser MSE than SDR.

AR(1) errors with medium endogeneity 1. For $(\rho = 0.5, \alpha = 0)$, OLS has 0.087% lesser MSE than SDR.

2. For $(\rho = 0.5, \alpha = 0.5)$, OLS has 0.055% lesser MSE than SDR.
3. For $(\rho = 0.5, \alpha = 0.9)$, OLS has 0.059% lesser MSE than SDR.

AR(1) errors with high endogeneity 1. For $(\rho = 0.9, \alpha = 0)$, OLS has 0.082% lesser MSE than SDR.

2. For $(\rho = 0.9, \alpha = 0.5)$, OLS has 0.045% lesser MSE than SDR.
3. For $(\rho = 0.9, \alpha = 0.9)$, OLS has 0.024% lesser MSE than SDR.

Experiment 2: First-stage SDR uses Non-parametric Function

Our parameter of interest is β_1 which is the coefficient of x in a linear model of $y = \beta_0 + \beta_1 x + \varepsilon$.

We get two estimates of β_1 using OLS and SDR methods n_rep (say 100 replications) times and then compare the mean squared error. Followings are the steps:

1. *OLS*: i) We obtain first stage coefficients using OLS and

   `ii) Use $\hat{x}$ in second stage to estimate parameter of interest $\beta_1$`

2. *SDR*:

   i) For linear models, one SDR is enough. We fuse the given p number of instruments into one SDR index.

   ii) `Then we use non-parametric method to get predicted $\hat{x}$`.

   iii) Then we replace true x by this $\hat{x}$ in the equation of y to obtain β_1.

DGP: DGP remains the same as 1.3

$$
\begin{aligned}
&Z \sim \mathcal{N}(0, I_p), \quad Z \in \mathbb{R}^{n \times p} \\
&\begin{pmatrix} \eta_t \\ \zeta_t \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right) \\
&\epsilon_t = \alpha_1 \cdot \epsilon_{t-1} + \eta_t \qquad e_t = \alpha_2 \cdot e_{t-1} + \zeta_t, \qquad \text{(AR(1) process)} \\
&x = Z\gamma + e, \quad \gamma \text{ is a vector of length p generated from } Uniform[1, 2] \\
&y = 2x + \epsilon
\end{aligned}
$$

True underlying model is linear but we estimate it using non-parametric function.

Let's choose $p = 3, \beta_1 = 0, \beta_2 = 2, n = 500$

```
library(MASS)  # For generating multivariate normal distribution
library(dr)    # For SDR method (SIR)
library(np)    # For nonparametric regression
```

```
## Warning: package 'np' was built under R version 4.2.3
```

```
## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-17)
## [vignette("np_faq",package="np") provides answers to frequently asked questions]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

```r
OLS_SDR_NP = function(Z, rho, alpha=0.5) {

  n = dim(Z)[1]

  # Generate AR(1) errors
  errors = generate_AR1_errors(n, rho, alpha)
  e = errors$e   # Error term for x
  epsilon = errors$epsilon  # Error term for y

  gamma=as.matrix(runif(p,min=1,max=2), ncol=1)

  x = Z %*% gamma + e
  y = 2 * x + epsilon  # True y with epsilon

  # 1. First Stage: Predict x using OLS
  data_x = data.frame(x, Z)
  colnames(data_x) = c("x", paste0("Z", 1:ncol(Z)))

  # OLS Model
  ols_model = lm(x ~ ., data = data_x)
  x_hat_ols = predict(ols_model, data_x)  # Predicted x using OLS

  # SDR Model (using SIR)
  sir_model = dr(x ~ ., data = data_x, method = "sir")
  sir_directions = sir_model$evectors  # SIR directions

  # Project Z onto the first SDR direction
  sdr_projection = as.matrix(Z) %*% sir_directions[, 1]

  # Ensure sdr_projection and x are numeric vectors
  if (!is.numeric(sdr_projection)) sdr_projection = as.numeric(sdr_projection)
  if (!is.numeric(x)) x = as.numeric(x)

  # Simplify bandwidth selection (set manually instead of cross-validation)
  bw_value = 0.5  # Set an appropriate bandwidth value
  tryCatch({
    np_model = npreg(bws = bw_value, txdat = sdr_projection, tydat = as.vector(x))  # Kernel regression
    x_hat_sdr = fitted(np_model)  # Predicted x using nonparametric regression on SDR index
  }, error = function(e) {
    cat("Error in nonparametric regression: ", e$message, "\n")
    stop(e)  # Stop execution if there's an error
  })

  # 2. Second Stage: Use predicted x to estimate y

  # OLS-based prediction
  second_stage_ols = lm(y ~ x_hat_ols)
  y_hat_ols = predict(second_stage_ols)
```

```
  # SDR-based prediction (using nonparametric SDR prediction)
  second_stage_sdr = lm(y ~ x_hat_sdr)
  y_hat_sdr = predict(second_stage_sdr)

  # Return the results: OLS and SDR coefficient estimates for the second stage
  return(list(
    OLS_Coefficients_Stage2 = coef(second_stage_ols)[-1],  # Exclude intercept
    SDR_Coefficients_Stage2 = coef(second_stage_sdr)[-1],  # Exclude intercept
    OLS_MSE_Stage2 = mean((y - y_hat_ols)^2),
    SDR_MSE_Stage2 = mean((y - y_hat_sdr)^2)
  ))
}

# Repeat the experiment for 100 iterations
set.seed(42)
n_reps = 100
n = 500
p = 3
alpha=0.9
rho = 0.9  # Correlation between e and epsilon
ols_stage2_store = numeric(n_reps)
sdr_stage2_store = numeric(n_reps)

for (i in 1:n_reps) {
  # Generate p regressors (Z)
  Z = matrix(rnorm(n * p), ncol = p)
  result = OLS_SDR_NP(Z, rho, alpha)

  # Store MSE values for OLS and SDR in second stage
  ols_stage2_store[i] = result$OLS_MSE_Stage2
  sdr_stage2_store[i] = result$SDR_MSE_Stage2
}

# Print summary of results
cat("Summary of OLS Stage 2 MSE values:\n")
```

```
## Summary of OLS Stage 2 MSE values:
```

```
print(summary(ols_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.30   38.02   42.57   42.84   46.90   65.08
```

```
cat("\nSummary of SDR Stage 2 MSE values:\n")
```

```
##
## Summary of SDR Stage 2 MSE values:
```

```
print(summary(sdr_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.04   37.86   42.31   42.54   46.46   64.54
```

Effect of Non-parametric step (with No AR(1) errors and No endogeneity)

1. For $(\rho = 0, \alpha = 0)$, SDR has 0.08% lesser MSE than OLS.

Non-parametric step with AR(1) errors But No endogeneity

1. For $(\rho = 0, \alpha = 0.5)$, SDR has 0.201% lesser MSE than OLS.
2. For $(\rho = 0, \alpha = 0.9)$, SDR has 0.532% lesser MSE than OLS.

Non-parametric step with AR(1) errors with low endogeneity 1. For $(\rho = 0.2, \alpha = 0)$, SDR has 0.035% lesser MSE than OLS.

2. For $(\rho = 0.2, \alpha = 0.5)$, SDR has 0.225% lesser MSE than OLS.
3. For $(\rho = 0.2, \alpha = 0.9)$, SDR has 0.625% lesser MSE than OLS.

Non-parametric step with AR(1) errors with medium endogeneity 1. For $(\rho = 0.5, \alpha = 0)$, SDR has 0.144% lesser MSE than OLS.

2. For $(\rho = 0.5, \alpha = 0.5)$, SDR has 0.296% lesser MSE than OLS.
3. For $(\rho = 0.5, \alpha = 0.9)$, SDR has 0.661% lesser MSE than OLS.

Non-parametric step with AR(1) errors with high endogeneity 1. For $(\rho = 0.9, \alpha = 0)$, SDR has 0.223% lesser MSE than OLS.

2. For $(\rho = 0.9, \alpha = 0.5)$, SDR has 0.374% lesser MSE than OLS.
3. For $(\rho = 0.9, \alpha = 0.9)$, SDR has 0.611% lesser MSE than OLS.

Experiment 3 Moving to High Dimensions: Many instruments Generated from Factors

In this experimentation, we allow number of instruments grow large. We use factor structure to generate many instruments. we aim to demonstrate the performance of our SIF(SDR based approach) relative to FIV (OLS based approach)

$$
\begin{aligned}
&f_{jt} \sim \mathcal{N}(0,1) \quad \text{for all } j = 1, 2, .., r. \\
&Z_{it} = \mathbf{b}_i \mathbf{f}_t + u_{it} \qquad u_{it} \sim \mathcal{N}(0,1) \qquad i = 1, 2, .., N. \\
&\begin{pmatrix} \eta_t \\ \zeta_t \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right) \\
&\epsilon_t = \alpha_1 \cdot \epsilon_{t-1} + \eta_t \qquad e_t = \alpha_2 \cdot e_{t-1} + \zeta_t, \qquad \text{(AR(1) process)} \\
&x_t = \phi' \mathbf{f}_t + e_t, \quad \phi \text{ is a vector of length r generated from } Uniform[1, 2] \\
&y_t = 2x_t + \epsilon_t
\end{aligned}
$$

N is the number of instrument and T is the sample size, but in R `T` is a reserved symbol so we use n for sample size in code-block. r is the number of factors.

Let's choose $r = 3, \beta_1 = 0, \beta_2 = 2, n = 500$

```
library(MASS)  # For generating multivariate normal distribution
library(dr)    # For SDR method (SIR)
library(np)    # For nonparametric regression

# Function to estimate the number of factors
No_of_Factors_to_take = function(Z) {
  TT = nrow(Z)
  N = ncol(Z)
  ZZt = t(Z) %*% Z / (TT * N)
  eigen_decomp = eigen(ZZt)
  eigenvalues = eigen_decomp$values
  ratio_vector = numeric(length(eigenvalues))
  for (k in 2:floor(length(eigenvalues)/2)) {
    ratio_vector[k] = eigenvalues[k] / eigenvalues[k + 1]
  }
  max_index = which.max(ratio_vector)
  return(max_index)
}

# Main function to implement the SDR and OLS methods
OLS_SDR_NP_Factors = function(Z, rho, phi, B, factors, alpha=0.5) {

  n = dim(Z)[1]

  # Generate AR(1) errors
  errors = generate_AR1_errors(n, rho,alpha)
  e = errors$e   # Error term for x
  epsilon = errors$epsilon  # Error term for y

  # Generate x using factors and error
  x = factors %*% phi + e
  y = 2 * x + epsilon  # True y with epsilon

  # 1. PCA to estimate factors from instruments Z
  pca_result = prcomp(Z, scale. = TRUE)
  r=No_of_Factors_to_take(Z)
  estimated_factors = pca_result$x[, 1:r]  # First K factors based on PCA

  # --- SDR Method ---
  # Step-1: Use estimated factors for SDR
  data_x = data.frame(x, estimated_factors)
  colnames(data_x) = c("x", paste0("Factor", 1:r))

  # SDR Model (using SIR)
  sir_model = dr(x ~ ., data = data_x, method = "sir")
  sir_directions = sir_model$evectors

  # Project estimated factors onto the first SDR direction
  sdr_projection = as.matrix(estimated_factors) %*% sir_directions[, 1]
```

```
  # Nonparametric regression of x on SDR projections
  np_model = npreg(bws = 0.5, txdat = sdr_projection, tydat = as.vector(x))
  x_hat_sdr = fitted(np_model)  # Predicted x using nonparametric regression on SDR index

  # Second stage: regress y on predicted x_hat_sdr
  second_stage_sdr = lm(y ~ x_hat_sdr)
  y_hat_sdr = predict(second_stage_sdr)

  # --- OLS Method ---
  # Step-1: OLS using estimated factors
  ols_model = lm(x ~ ., data = data_x)
  x_hat_ols = predict(ols_model, data_x)  # Predicted x using OLS

  # Second stage: regress y on predicted x_hat_ols
  second_stage_ols = lm(y ~ x_hat_ols)
  y_hat_ols = predict(second_stage_ols)

  # Return the results: OLS and SDR coefficient estimates for the second stage
  return(list(
    OLS_Coefficients_Stage2 = coef(second_stage_ols)[-1],  # Exclude intercept
    SDR_Coefficients_Stage2 = coef(second_stage_sdr)[-1],  # Exclude intercept
    OLS_MSE_Stage2 = mean((y - y_hat_ols)^2),
    SDR_MSE_Stage2 = mean((y - y_hat_sdr)^2)
  ))
}

# Repeat the experiment for 100 iterations
set.seed(42)
n_reps = 100
n = 500
N = 400  # Number of instruments
r = 3 # Number of factors

# Set up the loadings matrix B and the factors
B = matrix(rnorm(N * r), ncol = r)
factors = matrix(rnorm(n * r), ncol = r)
phi = matrix(runif(r, min = 1, max = 2), ncol = 1)

# rho for AR(1) errors
rho = 0.5
alpha=0.5

ols_stage2_store = numeric(n_reps)
sdr_stage2_store = numeric(n_reps)

for (i in 1:n_reps) {
  # Generate instruments Z from the factors
  Z = matrix(0, nrow = n, ncol = N)
  for (j in 1:N) {
    Z[, j] = factors %*% B[j, ] + rnorm(n)  # Instruments as linear combinations of factors plus noise
  }

  # Call the OLS_SDR_2SLS function
```

```
  result = OLS_SDR_NP_Factors(Z, rho, phi, B, factors, alpha)

  # Store MSE values for OLS and SDR in second stage
  ols_stage2_store[i] = result$OLS_MSE_Stage2
  sdr_stage2_store[i] = result$SDR_MSE_Stage2
}

# Print summary of results
cat("Summary of OLS Stage 2 MSE values:\n")
```

```
## Summary of OLS Stage 2 MSE values:
```

```
print(summary(ols_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.939   8.818   9.080   9.316   9.792  11.319
```

```
cat("\nSummary of SDR Stage 2 MSE values:\n")
```

```
##
## Summary of SDR Stage 2 MSE values:
```

```
print(summary(sdr_stage2_store))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.329   8.158   8.422   8.627   9.190  10.374
```

Result We are looking at the performance difference as N increases.

Low Endogenity + Low Serial Correlation

1. For $(N = 5, \rho = 0.2, \alpha = 0.2)$, SDR has 0.938% lesser MSE than OLS.
2. For $(N = 10, \rho = 0.2, \alpha = 0.2)$, SDR has 0.946% lesser MSE than OLS.
3. For $(N = 50, \rho = 0.2, \alpha = 0.2)$, SDR has 2.46% lesser MSE than OLS.
4. For $(N = 200, \rho = 0.2, \alpha = 0.2)$, SDR has 5.61% lesser MSE than OLS.
5. For $(N = 400, \rho = 0.2, \alpha = 0.2)$, SDR has 7.43% lesser MSE than OLS.
6. For $(N = 600, \rho = 0.2, \alpha = 0.2)$, SDR has 9.03% lesser MSE than OLS.

Medium Endogenity + Medium Serial Correlation

1. For $(N = 5, \rho = 0.5, \alpha = 0.5)$, SDR has 0.468% lesser MSE than OLS.
2. For $(N = 10, \rho = 0.5, \alpha = 0.5)$, SDR has 1.10% lesser MSE than OLS.
3. For $(N = 100, \rho = 0.5, \alpha = 0.5)$, SDR has 3.46% lesser MSE than OLS.
4. For $(N = 200, \rho = 0.5, \alpha = 0.5)$, SDR has 5.521% lesser MSE than OLS.

5. For $(N = 400, \rho = 0.5, \alpha = 0.5)$, SDR has 9.15% lesser MSE than OLS.
6. For $(N = 600, \rho = 0.5, \alpha = 0.5)$, SDR has 8.891% lesser MSE than OLS.

High Endogenity + High Serial Correlation

1. For $(N = 5, \rho = 0.9, \alpha = 0.9)$, SDR has 0.748% lesser MSE than OLS.
2. For $(N = 10, \rho = 0.9, \alpha = 0.9)$, SDR has 1.07% lesser MSE than OLS.
3. For $(N = 100, \rho = 0.9, \alpha = 0.9)$, SDR has 3.84% lesser MSE than OLS.
4. For $(N = 200, \rho = 0.9, \alpha = 0.9)$, SDR has 5.24% lesser MSE than OLS.
5. For $(N = 400, \rho = 0.9, \alpha = 0.9)$, SDR has 7.43% lesser MSE than OLS.
6. For $(N = 600, \rho = 0.9, \alpha = 0.9)$, SDR has 8.86% lesser MSE than OLS.

Conclusions

We note the following major points:

1. From Experiment-1, it is clear that there is both SDR and OLS are neck-to-neck in terms of MSE. The difference is less than 1%. When we introduce the endogenity and serial correlation in errors, OLS's relative performance weakens.
2. From Experiment-2, we see that having non-parametric step makes SDR perform better than the OLS. However the performance difference is not large.
3. From Experiment-3, SDR based method clearly outperform the OLS based method under many specifications.
4. If we further complicate the DGP e.g. making factors AR(p) and so on, SDR based method works better.

Therefore it is clear that the fusion of estimated factors into one SDR Index gives non-trivial advantage.