SIMULATION STUDIES

Study the properties/performance of statistical methods by applying them repeatedly to computer-generated data.

Use what we know about design of experiments when implementing simulation studies.

Use what we know about quantifying uncertainty when summarizing simulation-study results.

Think about the conditions under which we want to study a given method (for instance, *where* in the parameter space).

Toy example

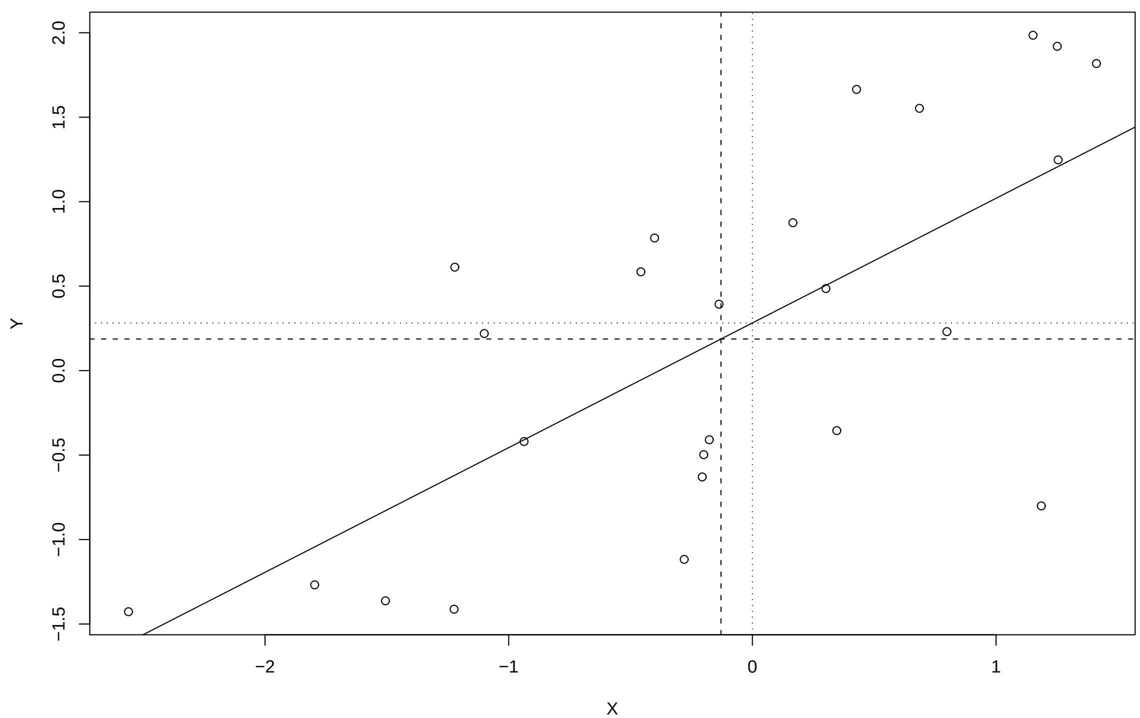
Compare two ways to estimate μ_Y, the population mean of Y

First estimator - the sample mean, $\bar{y}$, based on SRS of size n.

Does the availability of an 'auxillary' variable help?

Say can sample (X, Y) jointly, where the **population mean** μ_X of X is **known** (e.g., from census data).

Maybe X is income, Y is test score.



Second estimator: $\hat{\beta}_0 + \hat{\beta}_1 \mu_X$, where $(\hat{\beta}_0, \hat{\beta}_1)$ are coef. from LM fit of Y on X.

Look at mean-squared error, $MSE = E\left\{(\hat{\mu}_Y - \mu_Y)^2\right\}$, for both estimators.

BAD SIMULATION APPROACH

```
NSIM <- 100;  n <- 25
mu.x <- 0; sig.x <- 1; mu.y <- 0; sig.y <- 1; rho <- 0.6

set.seed(13)
for (i in 1:NSIM) {
  x <- rnorm(n); y <- rho*x+sqrt(1-rho^2)*rnorm(n)
  x <- mu.x+sig.x*x; y <- mu.y+sig.y*y
  est1[i] <- mean(y) }

for (i in 1:NSIM) {
  x <- rnorm(n); y <- rho*x+sqrt(1-rho^2)*rnorm(n)
  x <- mu.x+sig.x*x; y <- mu.y+sig.y*y
  tmp <- lm(y~x)
  est2[i] <- coef(tmp)[1]+coef(tmp)[2]*mu.x }
```

```
# estimated MSE's
sqerr1 <- (est1 - mu.y)^2
sqerr2 <- (est2 - mu.y)^2
print(round(c(mean(sqerr1), mean(sqerr2),
      mean(sqerr1)-mean(sqerr2)),4))

[1] 0.0383 0.0249 0.0134


# simulation SE's of estimated MSE's
print(round(c(
  sqrt( var(sqerr1) / NSIM ),
  sqrt( var(sqerr2) / NSIM ),
  sqrt( (var(sqerr1)+var(sqerr2)) / NSIM ) ) ,4))

[1] 0.0063 0.0035 0.0072
```

BETTER SIMULATION APPROACH

```
set.seed(13)
for (i in 1:NSIM) {
  x <- rnorm(n); y <- rho*x+sqrt(1-rho^2)*rnorm(n)
  x <- mu.x+sig.x*x; y <- mu.y+sig.y*y
  tmp <- lm(y~x)
  est1[i] <- mean(y)
  est2[i] <- coef(tmp)[1]+coef(tmp)[2]*mu.x }

# estimated MSE's
sqerr1 <- (est1 - mu.y)^2
sqerr2 <- (est2 - mu.y)^2
print(round(c(mean(sqerr1), mean(sqerr2),
      mean(sqerr1)-mean(sqerr2)),4))

[1] 0.0383 0.0278 0.0106
```

```
# simulation SE's
print(round(c(
  sqrt( var(sqerr1) / NSIM ),
  sqrt( var(sqerr2) / NSIM ),
  sqrt( (var(sqerr1-sqerr2) / NSIM ) ) ), 4))

[1] 0.0063 0.0037 0.0042
```

SIMPLE DOE IDEA - aside from the difference you are interested in, make everything as similar as possible

In general say have parameter **vector** θ, scalar parameter of interest $\psi = g(\theta)$, and one or more estimators $\hat{\psi}(D)$.

In general mean-squared error (and other performance criteria) can/will vary across parameter space, e.g.,
$MSE(\theta) = E_\theta\left\{(\hat{\psi}(D) - \psi)^2\right\}$

Also, performance may vary with other aspects of the "scenario," e.g., shape of X distribution in a regression situation.

Moreover, in a problem where simulation studies are really needed, little may be known about the nature of this variation in advance.

Say have a total simulation 'budget' of N datasets. How to allocate???

Pick k points in parameter (and scenario) space, and simulate N/k datasets at each point?

Make the k points 'representative' of practical scenarios?

Say there are p parameters (and other scenario values to set). Might try full-factorial design - all combinations of d_j levels for j-th parameter, so $k = d_1 \times \ldots \times d_p$.

Of course, have usual curse-of-dimensionality - N/k gets small very quickly as p increases.

Variant on the full-factorial idea. Try a *random-parameter* or *Bayesian* simulation study.

for (i in 1:N) {
- sample $\theta_i \sim \Pi$,
- sample $D_i|\theta_i$,
- compute $Err_i = \hat{\psi}_i - \psi_i$

}.

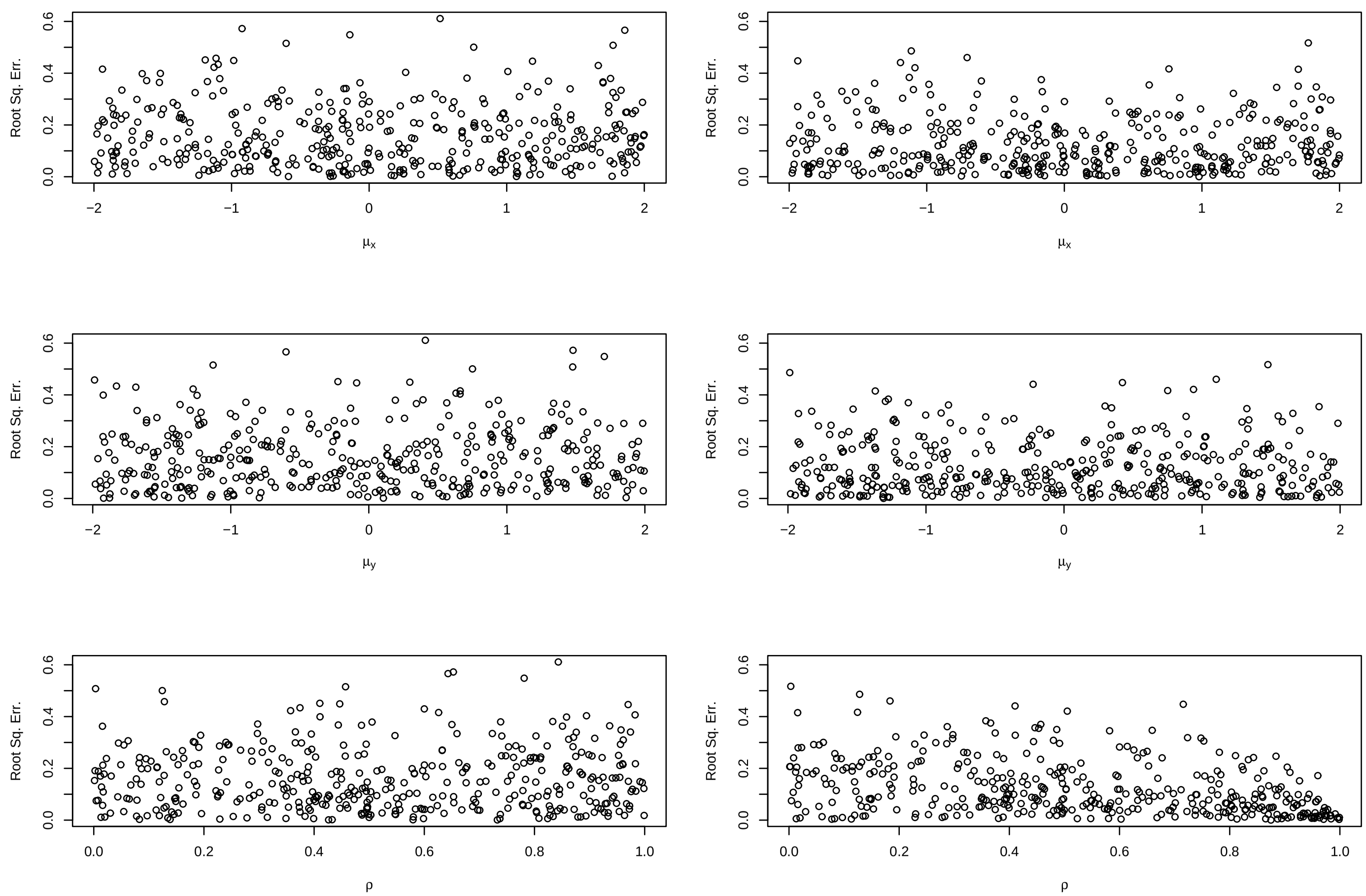
Then summarize $Err_1, \ldots, Err_N$.

For instance, averaging Err_i^2 gives an averaged-mean-squared-error (decision theory connection).

Also, can look for trends - does $MSE(\theta)$ seem to vary particularly with one component of θ?

```
for (i in 1:NSIM) {

  ### generate parameters
  sig.x <- sig.y <- 1
  mu.x <- runif(1,-2,2); mu.y <- runif(1,-2,2)
  rho <- runif(1)
  ### save parameters and estimand!
  theta[i,] <- c(mu.x, mu.y, sig.x, sig.y, rho)
  ans[i] <- mu.y

  ### generate data given parameters, compute estimates
  x <- rnorm(n); y <- rho*x+sqrt(1-rho^2)*rnorm(n)
  x <- mu.x+sig.x*x; y <- mu.y+sig.y*y
  tmp <- lm(y~x)
  est1[i] <- mean(y)
  est2[i] <- coef(tmp)[1]+coef(tmp)[2]*mu.x }
```

```r
# estimated MSE's
sqerr1 <- (est1 - ans)^2
sqerr2 <- (est2 - ans)^2
print(round(c(mean(sqerr1), mean(sqerr2),
            mean(sqerr1)-mean(sqerr2)),4))

[1] 0.0395 0.0258 0.0137


# simulation SE's for estimated MSE's
print(round(c(
  sqrt( var(sqerr1) / NSIM ),
  sqrt( var(sqerr2) / NSIM ),
  sqrt( (var(sqerr1-sqerr2) / NSIM ) ) ), 4))

[1] 0.0028 0.0020 0.0022
```



Root Sq. Err.
0.0
0.2
0.4
0.6
−2
−1
0
1
2
μ_x
μ_y
ρ
0.0
0.2
0.4
0.6
0.8
1.0