R Code and Output

July 28, 2025

```
# We will begin by loading ggplot2.
require(ggplot2)

####################################################
#################### Figure 1 ######################
####################################################

# We will start with the panel design.

# Set the seed.
set.seed(0)

# Create a vector for sample sizes ({1000, 2000, ..., 10000}).
ns<-seq(1000,10000,by=1000)

# Create a vector to store the average estimate for each sample size.
ests<-rep(NA,length(ns))

# Create a vector to store the standard error for each sample size.
ses<-rep(NA,length(ns))

# Create a vector to store the average t-test p-value for each sample size.
ps<-rep(NA,length(ns))

# Start a for loop to run through each sample size.
for(j in 1:length(ns)){

        # Set n as the sample size for this iteration.
        n<-ns[j]

        # Set the number of Monte Carlo runs at 10,000.
        r<-10000

        # Set the percentage of respondents who will repeat their Wave 1 answer in Wave 2
        # at 20%.
        p_same<-0.2

        # Create a vector to store the difference between the estimate and the ATE each
        # run.
        diff<-rep(NA,r)

        # Create a vector to store the estimate for each run.
        est<-rep(NA,r)

        # Create a vector to store the t-test p-value for each run.
        p<-rep(NA,r)
```

*This report is automatically generated with the R package **knitr** (version 1.49). It was then modified by Andrew Bertoli.

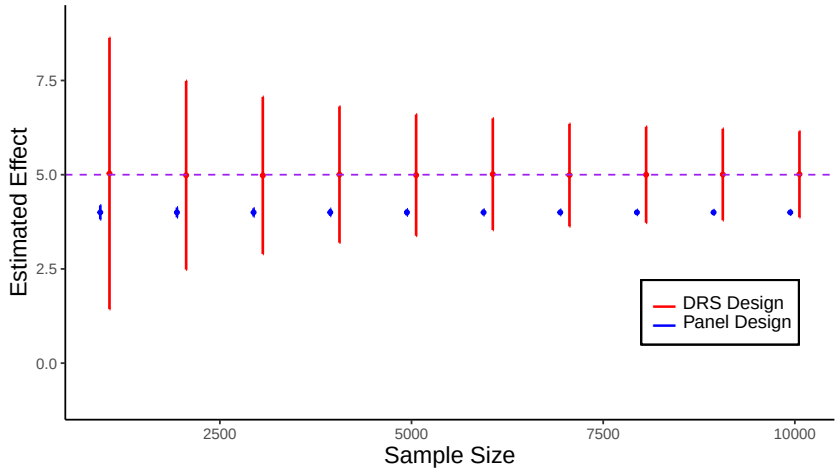
```r
# Start the for loop to do the r simulations for this specific sample size.
for(i in 1:r){

        # The potential outcomes under control will be randomly drawn from {1, 2, ...,
        # 100}. Note that these potential outcomes are defined as the true and honest
        # Wave 2 answers of the respondents in the counterfactual world where the
        # event "did not occur." As described in the article, the meaning of "did not
        # occur" depends on the counterfactual that researchers have in mind.
        y_ikc<-sample(1:100,n,replace=T)

        # The potential outcomes under treatment will be the potential outcomes under
        # control plus an individual-level treatment effect that will be randomly
        # drawn from {-2, -1, ..., 12}. Note that these potential outcomes are defined
        # as the true and honest Wave 2 answers of the respondents in the world where
        # the event did occur but where the respondents were not surveyed in Wave 1.
        y_ikt<-y_ikc+sample(-2:12, n, replace=T)

        # The level of temporal variation for each respondent will be randomly drawn
        # from {-3, -2, ..., 3}.
        temporal_variation<-sample(-3:3, n, replace=T)

        # Note that the expected value of the temporal variation is 0.

        # We now compute the potential outcomes for each respondent before the event
        # under control, meaning their Wave 1 true and honest answers in the world
        # where the event did not occur.
        y_ikbc<-y_ikc-temporal_variation

        # The level of variation from anticipation of the event for each respondent
        # will be randomly drawn from {-1, 0, 1}.
        anticipation_variation<-sample(-1:1, n, replace=T)

        # The true and honest Wave 1 answers for each respondent in the real world
        # are then just their y_ikbc potential outcomes plus the variation from
        # their anticipation of the event.
        y_ikb<-y_ikbc+anticipation_variation

        # For now, we will set the respondents true and honest Wave 2 answers to be
        # the same as their Wave 2 true and honest answers in the world where they
        # were not surveyed in Wave 1. In other words, we will assume 0% panel
        # conditioning, which we will change later.
        y_ika<-y_ikt

        # We will draw measurement error before the event from a Poisson distribution
        # with rate 1.
        e_kb<-rpois(n,1)

        # We will draw measurement error after the event from another Poisson
        # distribution with rate 1.
        e_ka<-rpois(n,1)

        # The observed answers in Wave 2 are the respondents true and honest Wave 2
        # answers after completing the survey in Wave 1, plus the measurement error.
        y_ikao<-y_ika+e_ka

        # The observed answers in Wave 1 are the respondents true and honest Wave 1
        # answers plus the measurement error.
        y_ikbo<-y_ikb+e_kb
```

```
        # To account for panel conditioning, we will randomly set some respondents to
        # repeat their Wave 1 answers in Wave 2. We will start by making a vector with
        # 1's representing respondents who will repeat their answers. Note that we
        # previously set p_same at 20%.
        same<-sample(c(rep(0,n*(1-p_same)),rep(1,n*p_same)))

        # For the respondents represented by 1 in the vector same, we will set their
        # observed Wave 2 answers to be the same as their observed Wave 1 answers.
        y_ikao[same==1]<-y_ikbo[same==1]

        # We can now compute and store the difference between the estimate and ATE
        # for this sample size on this particular run of the for loop.
        diff[i]<-mean((y_ikao-y_ikbo)-(y_ikt-y_ikc))

        # We can also compute and store the estimated treatment effect for this sample
        # size on this particular run of the for loop.
        est[i]<-mean(y_ikao-y_ikbo)

        # We will also compute and store the t-test p-value for this particular run of
        # the for loop.
        p[i]<-t.test(y_ikao,y_ikbo,paired=T)$p.value

        } # End the for loop for the r runs for this specific sample size.

    ests[j]<-mean(est) # Store the average of the estimates for this sample size.

    ses[j]<-sd(diff)  # Store the standard error for this sample size. We use diff
                      # here instead of est because doing so adjusts for variation
                      # in the data generating process across the simulation runs.
                      # The ATE is about 5 each run, but this value moves around
                      # slightly because of randomness in the data generating
                      # process used to generate the potential outcomes. If we used
                      # the standard deviation of est, this variation would spill
                      # over and inflate our estimate of the true standard error of
                      # the estimator. On the other hand, diff is the difference
                      # between the estimate and ATE each run, so taking the standard
                      # deviation of this difference each iteration removes variation
                      # from small fluctuations in the ATE.

    ps[j]<-mean(p) # Store the average of the t-test p-values for this sample size.

    } # End the for loop that runs through each sample size.



# We will now do the DRS design.

# Create a vector for sample sizes ({1000, 2000, ..., 10000}).
ns<-seq(1000,10000,by=1000)

# Create a vector to store the average estimate for each sample size.
ests2<-rep(NA,length(ns))

# Create a vector to store the standard error for each sample size.
ses2<-rep(NA,length(ns))
```

```r
# Create a vector to store the average t-test p-value for each sample size.
ps2<-rep(NA,length(ns))

# Start a for loop to run through each sample size.
for(j in 1:length(ns)){

        # Set n as the sample size for this iteration.
        n<-ns[j]

        # Set the number of Monte Carlo runs at 10,000.
        r<-10000

        # Create a vector to store the difference between the estimate and the
        # ATE each run.
        diff<-rep(NA,r)

        # Create a vector to store the estimate for each run.
        est<-rep(NA,r)

        # Create a vector to store the t-test p-value for each run.
        p<-rep(NA,r)

        # Start the for loop to do the r simulations for this specific sample size.
        for(i in 1:r){

                # The potential outcomes under control will be randomly drawn from {1, 2, ...,
                # 100}.
                y_ikc<-sample(1:100,n,replace=T)

                # The potential outcomes under treatment will be the potential outcomes under
                # control plus an individual-level treatment effect that will be randomly
                # drawn from {-2, -1, ..., 12}.
                y_ikt<-y_ikc+sample(-2:12, n, replace=T)

                # The level of temporal variation for each respondent will be randomly drawn
                # from {-3, -2, ..., 3}.
                temporal_variation<-sample(-3:3, n, replace=T)

                # We now compute the potential outcomes for each respondent before the event
                # under control, meaning their Wave 1 true and honest answers in the world
                # where the event did not occur.
                y_ikbc<-y_ikc-temporal_variation

                # The level of variation from anticipation of the event for each respondent
                # will be randomly drawn from {-1, 0, 1}.
                anticipation_variation<-sample(-1:1, n, replace=T)

                # The true and honest Wave 1 answers for each respondent in the real world
                # are then just their y_ikbc potential outcomes plus the variation from
                # their anticipation of the event.
                y_ikb<-y_ikbc+anticipation_variation

                # We are assuming no priming bias for DRS, so we will set the y_ika values as
                # equivalent to the y_ikt values.
                y_ika<-y_ikt

                # We will draw measurement error before the event from a Poisson distribution
                # with rate 1.
```

```
            e_kb<-rpois(n,1)

            # We will draw measurement error after the event from another Poisson
            # distribution with rate 1.
            e_ka<-rpois(n,1)

            # The observed answers in Wave 2 are the respondents true and honest Wave 2
            # answers after completing the survey in Wave 1, plus the measurement error.
            y_ikao<-y_ika+e_ka

            # The observed answers in Wave 1 are the respondents true and honest Wave 1
            # answers plus the measurement error.
            y_ikbo<-y_ikb+e_kb

            # Because DRS involves splitting the sample into one group that does Survey A
            # before the event and another group that does Survey A after the event, we
            # only see y_ikbo or y_ikao for each respondent (with respect to Survey A).
            # To model this in the simulation, we will use the code below to choose about
            # half the respondents to give Survey A to in Wave 2:
            rand<-sample(1:n, round(n/2))

            # We will get the Wave 2 answers from these respondents.
            y_ikao<-y_ikao[rand]

            # We will get the Wave 1 answers from the other respondents.
            y_ikbo<-y_ikbo[-rand]

            # We can now compute and store the difference between the estimate and ATE
            # for this sample size on this particular run of the for loop.
            diff[i]<-mean((y_ikao-y_ikbo)-(y_ikt-y_ikc))

            # We will also calculate and store the estimate for this round.
            est[i]<-mean(y_ikao)-mean(y_ikbo)

            # We can also store the t-test p-value for this round.
            p[i]<-t.test(y_ikao,y_ikbo)$p.value

            } # End the for loop for the r runs for this specific sample size.

        ests2[j]<-mean(est) # Store the average of the estimates for this sample size.

        ses2[j]<-sd(diff) # Store the standard error for this sample size. We use diff
                          # here instead of est for the reason explained in lines 133-144.

        ps2[j]<-mean(p) # Store the average of the t-test p-values for this sample size.

        } # End the for loop that runs through each sample size.


# Create data frames to store the estimates, confidence intervals, and sample sizes.
Panel<-data.frame(ns,ests,upper=ests+ses*1.96,lower=ests-ses*1.96) # Panel design
DRS<-data.frame(ns,ests2=ests2,upper2=ests2+ses2*1.96,lower2=ests2-ses2*1.96) # DRS

# Make the ggplot with the desired customizations.
plot1<-ggplot(Panel,aes(x=ests, y=ns-60), color=blue) + # Start the ggplot, specify
                                                        # the data frame, set x as
                                                        # the sample size and y as the
                                                        # estimate, and set the color.
```

```
# Plot the estimates for the panel design.
geom_point(stat="identity",size=1,fill="white",color="blue") +
# Plot the standard errors for the panel design.
geom_errorbarh(aes(xmax=upper,xmin=lower),
               linewidth=0.75, height=0,color="blue") +
# Plot the estimates for the DRS design.
geom_point(y=DRS$ns+60,x=DRS$ests2,color="red",size=1) +
# Plot the standard errors for the DRS design.
geom_errorbarh(xmax=DRS$upper2,xmin=DRS$lower2,aes(y=ns+60),
               linewidth=0.75, height=0,color="red") +
xlab("Estimated Effect") + # Set the x-axis title.
ylab("Sample Size") + # Set the y-axis title.
theme_classic()+ # Set the theme.
theme(legend.position="none", # Omit the legend.
      axis.text=element_text(size=9.3), # Set the axis text size.
      axis.title=element_text(size=14)) + # Set the axis title size.
xlim(-1,9) + # Set the x-axis limits.
coord_flip() + # Flip the x and y axes.
# Add a legend.
geom_rect(aes(xmin = 0.5, xmax = 2.2, ymin = 8000, ymax = 10050),
                 fill="white",colour="black") +
annotate("text", x = c(1.535, 2.04, 1.103, 1.6135), y = c(8300, 8300, 9250, 9200),
label = c("_", "_", "Panel Design", "DRS Design"),
colour=c("Blue", "Red", "black", "black"), size=c(8, 8, 4, 4)) +
# Add a dashed line at y=5 to show the ATE.
geom_vline(xintercept=5,linetype=2,colour="purple")


# View the figure.
plot1
```




```
# Save the figure.
ggsave("Figure_1.pdf",width=7,height=4)


# Check the size of the treatment effect

5/sd(y_ikao)
```

```
## [1] 0.1735005
```

```
#####################################################
#################### Figure 2 ######################
#####################################################

# We will begin by estimating the MSE for the panel design for different
# sample sizes and levels of panel conditioning

# Specify the number of runs for each round of the Monte Carlo simulation.
r<-50000

# Create a vector containing different values for the proportion of
# respondents who repeat their Wave 1 answers in Wave 2.
ps_same<-seq(0,0.5,by=0.05)

# Create a vector to store the average estimates for each round.
ests<-rep(NA,length(ps_same))

# Create a vector to store the standard errors each round.
ses<-rep(NA,length(ps_same))


# Create a vector to store the sample sizes.
ns<-c(seq(700,900,by=100),seq(1000,10000,by=1000))

# Create a vector to store the locations where the MSE of the DRS
# design is (approximately) equal to the MSE of the panel design.
intersects<-rep(NA, length(ns))

# Create a for loop to run through each of the sample sizes
# ({700, 800, 900, 1000, 2000, 3000, ..., 10000}).
for(k in 1:length(ns)){

        # Set n as a particular sample size (the k-th element of ns).
        n<-ns[k]

        # Print the sample size. Since the code will take some time to run,
        # this line will make it easy to tell about how far along we are in
        # the for loop while it is running.
        print(n)

        # We will first estimate the MSE for the panel design at different levels of
        # panel conditioning. To do this, we will start a new for loop that will run
        # through the different levels of panel conditioning that we stored in the
        # vector ps.
        for(j in 1:length(ps_same)){

                # Create a vector to store the difference between the estimate and the
                # ATE each run. (We will do r total runs for each sample size and level
                # of panel conditioning.)
                diff<-rep(NA,r)

                # Create a vector to store the estimated effect for each run.
                est<-rep(NA,r)

                # We will now start a third for loop to do the Monte Carlo simulations
                # (repeating the process r times for each sample size and level of
                # panel conditioning).
                for(i in 1:r){
```

```r
# The potential outcomes under control will be randomly drawn from {1,
# 2, ..., 100}. Remember that these potential outcomes are defined as
# the true and honest Wave 2 answers of the respondents in the
# counterfactual world where the event "did not occur." As described
# in the article, the meaning of "did not occur" depends on the
# counterfactual that researchers have in mind.
y_ikc<-sample(1:100,n,replace=T)

# The potential outcomes under treatment will be the potential outcomes
# under control plus an individual-level treatment effect that will be
# randomly drawn from {-2, -1, ..., 12}. Note that these potential
# outcomes are defined as the true and honest Wave 2 answers of the
# respondents in the world where the event did occur but where the
# respondents were not surveyed in Wave 1.
y_ikt<-y_ikc+sample(-2:12, n, replace=T)

# The level of temporal variation for each respondent will be randomly
# drawn from {-3, -2, ..., 3}.
temporal_variation<-sample(-3:3, n, replace=T)

# We now compute the potential outcomes for each respondent before the
# event under control, meaning their Wave 1 true and honest answers in
# the world where the event did not occur.
y_ikbc<-y_ikc-temporal_variation

# The level of variation from anticipation of the event for each
# respondent will be randomly drawn from {-1, 0, 1}.
anticipation_variation<-sample(-1:1, n, replace=T)

# The true and honest Wave 1 answers for each respondent in the real
# world are then just their y_ikbc potential outcomes plus the variation
# from their anticipation of the event.
y_ikb<-y_ikbc+anticipation_variation

# For now, we will set the respondents true and honest Wave 2 answers to
# be the same as their Wave 2 true and honest answers in the world where
# they were not surveyed in Wave 1. In other words, we will assume 0%
# panel conditioning, which we will change later.
y_ika<-y_ikt

# We will draw measurement error before the event from a Poisson
# distribution with rate 1.
e_kb<-rpois(n,1)

# We will draw measurement error after the event from another Poisson
# distribution with rate 1.
e_ka<-rpois(n,1)

# The observed answers in Wave 2 are the respondents' true and honest
# Wave 2 answers after completing the survey in Wave 1, plus the
# measurement error.
y_ikao<-y_ika+e_ka

# The observed answers in Wave 1 are the respondents' true and honest
# Wave 1 answers plus the measurement error.
y_ikbo<-y_ikb+e_kb

# To account for panel conditioning, we will randomly set some respondents
```

```
                # to repeat their Wave 1 answers in Wave 2. We will start by making a
                # vector with 1's representing respondents who will repeat their answers.
                same<-sample(c(rep(0,n*(1-ps_same[j])),rep(1,n*ps_same[j])))

                # For the respondents represented by 1 in the vector same, we will set
                # their observed Wave 2 answers to be the same as their observed Wave 1
                # answers.
                y_ikao[same==1]<-y_ikbo[same==1]

                # We can now compute and store the difference between the estimate and
                # ATE for this sample size on this particular run of the for loop.
                diff[i]<-mean((y_ikao-y_ikbo)-(y_ikt-y_ikc))

                # We can also compute and store the estimated treatment effect for this
                # sample size and level of panel conditioning on this particular run of
                # the for loop.
                est[i]<-mean(y_ikao-y_ikbo)

                }


        ests[j]<-mean(est) # We will store the average of the estimates for the r
                           # runs of the for loop at this sample size and level of
                           # panel conditioning.

        ses[j]<-sd(diff)  # Store the standard error for this sample size. We use
                          # diff here instead of est for the reason explained in
                          # lines 133-144.

        }

# We will calculate the mean squared error for this estimator for this
# particular sample size.
mse1<-data.frame(ps_same,mse=ses^2+(ests-mean(y_ikt-y_ikc))^2)


# We will now estimate the MSE for the DRS design for the given n.

# Specify the number of runs for each round of the Monte Carlo simulation.
r2<-50000

# Create a vector to store the estimates for each run.
est<-rep(NA,r2)

# Create a vector to store the difference between the estimate and the ATE
# each run.
diff<-rep(NA,r)

# Start the for loop.
for(i in 1:r2){

        # The potential outcomes under control will be randomly drawn from {1,
        # 2, ..., 100}.
        y_ikc<-sample(1:100,n,replace=T)

        # The potential outcomes under treatment will be the potential outcomes under
        # control plus an individual-level treatment effect that will be randomly
        # drawn from {-2, -1, ..., 12}.
```

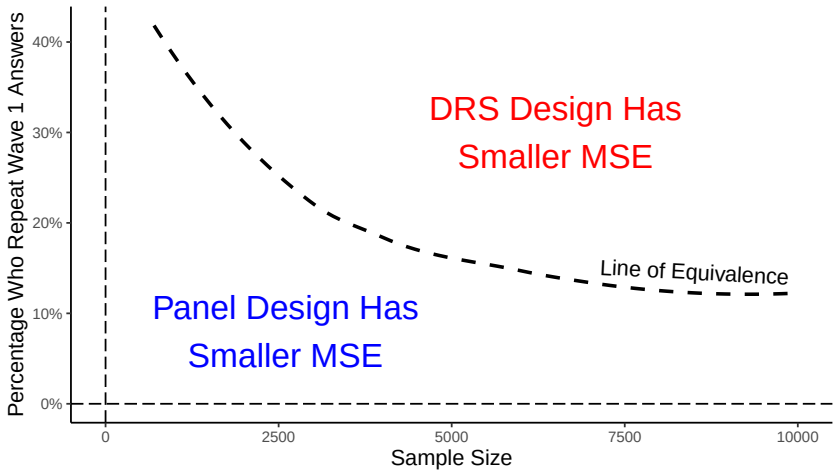
```
y_ikt<-y_ikc+sample(-2:12, n, replace=T)

# The level of temporal bias for each respondent will be randomly drawn from
# {-3, -2, ..., 3}.
temporal_bias<-sample(-3:3, n, replace=T)

# We now compute the potential outcomes for each respondent before the event
# under control, meaning their Wave 1 true and honest answers in the world
# where the event did not occur.
y_ikbc<-y_ikc-temporal_bias

# The level of variation from anticipation of the event for each respondent
# will be randomly drawn from {-1, 0, 1}.
anticipation_variation<-sample(-1:1, n, replace=T)

# The true and honest Wave 1 answers for each respondent in the real world are
# then just their y_ikbc potential outcomes plus the variation from their
# anticipation of the event.
y_ikb<-y_ikbc+anticipation_variation

# We are assuming no priming bias for DRS, so we will set the y_ika values
# as equivalent to the y_ikt values.
y_ika<-y_ikt

# We will draw measurement error before the event from a Poisson distribution
# with rate 1.
e_kb<-rpois(n,1)

# We will draw measurement error after the event from another Poisson
# distribution with rate 1.
e_ka<-rpois(n,1)

# The observed answers in Wave 2 are the respondents true and honest Wave 2
# answers after completing the survey in Wave 1, plus the measurement error.
y_ikao<-y_ika+e_ka

# The observed answers in Wave 1 are the respondents true and honest Wave 1
# answers plus the measurement error.
y_ikbo<-y_ikb+e_kb

# Because DRS involves splitting the sample into one group that does Survey A
# before the event and another group that does Survey A after the event, we
# only see y_ikbo or y_ikao for each respondent (with respect to Survey A).
# To model this in the simulation, we will use the code below to choose about
# half the respondents to give Survey A to in Wave 2:
rand<-sample(1:n, round(n/2))

# We will get the Wave 2 answers to Survey A from these respondents.
y_ikao<-y_ikao[rand]

# We will get the Wave 1 answers to Survey A from the other respondents.
y_ikbo<-y_ikbo[-rand]

# We will calculate and store the estimate for this run.
est[i]<-mean(y_ikao)-mean(y_ikbo)

# We can now compute and store the difference between the estimate and
# ATE for this sample size on this particular run of the for loop.
```

```
            diff[i]<-mean((y_ikao-y_ikbo)-(y_ikt-y_ikc))

            } # End the for loop.

        # Store the average of these estimates for this sample size.
        ests2<-mean(est)

        # Store the standard error for this sample size. We use diff here instead of
        # est for the reason explained in lines 133-144.
        ses2<-sd(diff)

        # Store the MSE for the DRS estimator for this sample size.
        drs_mse<-ses2^2+(ests2-mean(y_ikt-y_ikc))^2

        # Estimate an equation for the MSE values for the panel estimator given
        # different rates of panel conditioning, for this fixed sample size.
        panel_mse<-ggplot(mse1,aes(x=ps_same,y=mse))+geom_smooth(method="loess",
                                        formula=y~x,span=0.5,se=F)
        ggplot_build(panel_mse)$data[[1]]$y

        # Compute the difference between these estimated MSEs for the panel design and the
        # estimated MSE for the DRS design.
        differences<-ggplot_build(panel_mse)$data[[1]]$y-drs_mse

        # Find the two of these values closest to each side of 0 (meaning closest to where
        # the panel design and DRS design have the same MSE).
        y_lower<-max(differences[differences<=0])
        y_upper<-min(differences[differences>=0])

        # Calculate where 0 is in relation to these two points (on a scale from 0 to 1).
        percent_between <- 1-y_upper/(y_upper-y_lower)

        # Find the x-values for these two points (meaning the level of panel conditioning
        # where the MSE for the panel and DRS designs are close to equivalent).
        x_lower<-ggplot_build(panel_mse
                        )$data[[1]]$x[differences<=0][which.max(differences[differences<=0])]
        x_upper<-ggplot_build(panel_mse
                        )$data[[1]]$x[differences>=0][which.min(differences[differences>=0])]

        # Find the level of panel conditioning for the point where the panel and DRS
        # designs have approximately equivalent MSEs (given this sample size).
        intersects[k]<-percent_between*x_upper+(1-percent_between)*x_lower

        # End the for loop that runs through the sample sizes.
        }
```

```
## [1] 700
## [1] 800
## [1] 900
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] 8000
## [1] 9000
## [1] 10000
```

```
# Store the values along the line of equivalence as a data frame. Note that this line
# is an approximation.
dat<-data.frame(n=ns, ps_same=intersects)

mse_plot<-ggplot(dat,aes(x=n,y=ps_same),color=black) + # Start the ggplot, specify the
                                                       # data frame, set x as the
                                                       # sample size and y as the
                                                       # percent of respondents who
                                                       # repeat their answers,
                                                       # and set the color.
# Plot the line of equivalence. Again, this line is an approximation.
geom_smooth(method="loess",span=0.9,se=F,color="black",linetype="dashed") +
geom_vline(xintercept=0,linetype="longdash") + # Draw a vertical dashed line at x=0.
geom_hline(yintercept=0,linetype="longdash") + # Draw a horizontal dashed line at y=0.
xlab("Sample Size") + # Set the x-axis title.
ylab("Percentage Who Repeat Wave 1 Answers") + # Set the y-axis title.
# Set the y-axis tick marks.
scale_y_continuous(breaks=seq(0,0.5,by=0.1),
                   labels=c("0%","10%","20%","30%","40%","50%")) +
  theme_classic() + # Set the theme.
  theme(legend.position="none", # Omit the legend.
        axis.text=element_text(size=9.3), # Set the axis text size.
        axis.title=element_text(size=13)) + # Set the axis title size.
  # Add text at these positions.
  annotate("text", x = c(2600, 6500, 8510), y = c(0.08, 0.3, 0.146),
label = c("Panel Design Has\nSmaller MSE",
                  "DRS Design Has\nSmaller MSE",
                  "Line of Equivalence"), # Specify the text to be added.
colour=c("Blue","Red","Black"), size=c(7,7,4.5),angle=c(0,0,-3)) # Customize the text.

# View the figure.
mse_plot
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Save the figure.
ggsave("Figure_2.pdf",width=7,height=4)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```