Novel Simulations

Harpeth Lee

3/19/2022

```
library(ddsPLS2)
```

```
## Loading required package: foreach
## Warning: package 'foreach' was built under R version 4.1.2
## Loading required package: shiny
## Loading required package: doParallel
## Loading required package: iterators
## Loading required package: parallel
```

```
library(MASS)
library(spls)
```

```
## Sparse Partial Least Squares (SPLS) Regression and
## Classification (version 2.2-3)
```

```
library(pls)
```

```
##
## Attaching package: 'pls'
## The following object is masked from 'package:stats':
##
##     loadings
```

Sim Data Function

```
sim_data <- function(n = 5, p = 10, q = 2, R = 5, x = 3, noise_weight = 1, D_method = "new", noise_type

  # Ensures x<=R, if x>R the dimension of A is incompatible with phi
  if(x > R){
    x = R
  }

  # Creates A and D matrices
  A <- matrix(c(rep(rep(1,p),x), rep(rep(0,p),R-x)), ncol = p)

  if(D_method == "new") {
     D <- matrix(rep(1, R*q), nrow = R)
  } else {
    D <- diag(max(q, R))[1:R, 1:q]
  }
```

```
  d <- ncol(A)+nrow(A)+ncol(D)
  psi <- MASS::mvrnorm(n = n,mu = rep(0,d),Sigma = diag(d))
  phi <- psi[,1:nrow(A)]

  # If `rnorm` is used to generate noise a lower noise weight should be used as
  # the function is more sensitive since we directly weight results and not the
  # covariance matrix.

  if(noise_type == "mvrnorm") {
    epsilon_X <- mvrnorm(n = dim(phi)[1],
                         rep(0, dim(A)[2]),
                         Sigma = noise_weight*diag(dim(A)[2]))

    epsilon_Y <- mvrnorm(n = dim(phi)[1],
                         rep(0, dim(D)[2]),
                         Sigma = noise_weight*diag(dim(D)[2]))
  } else {
   epsilon_X <- matrix(noise_weight*rnorm(n = n*p), nrow = n)
   epsilon_Y <- matrix(noise_weight*rnorm(n = n*q), nrow = n)
  }

  X <- phi %*% A + epsilon_X
  Y <- phi %*% D + epsilon_Y


  list(X=X, Y=Y)
}
```

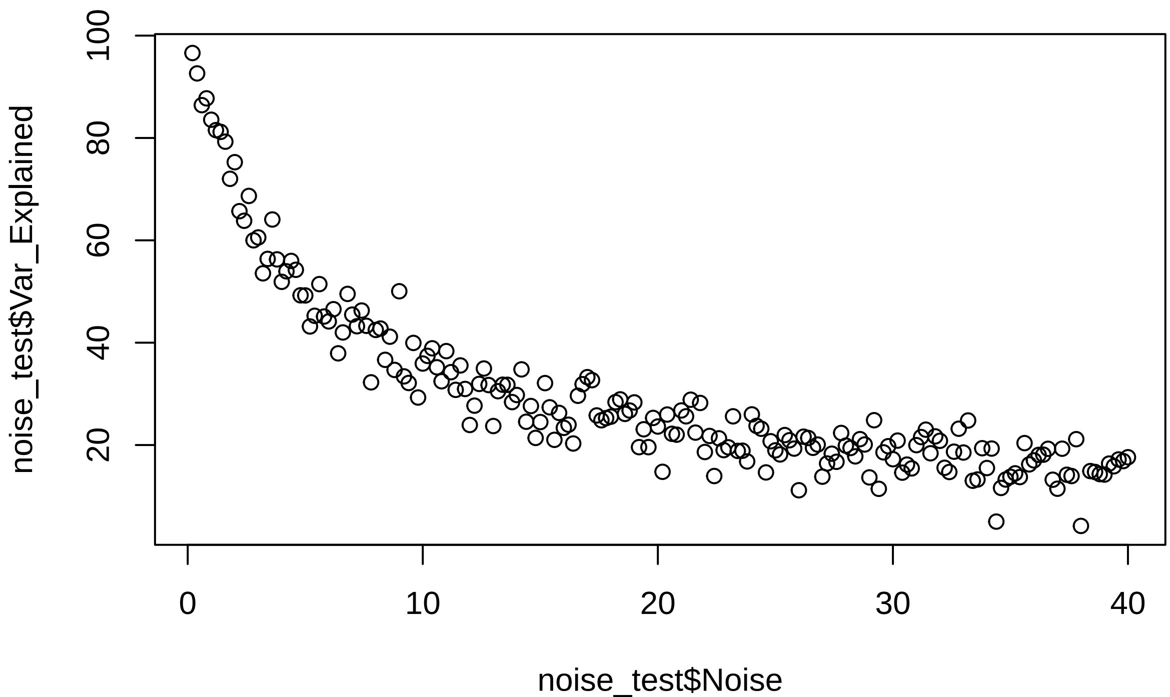
Noise Test

```
var_func <- function(noise_weight){
   sim <- sim_data(n = 100, p = 200, q = 5, noise_weight = noise_weight)
   mod <- ddsPLS(sim$X, sim$Y)
   if(!is.null(tail(mod$varExplained$Cumu, n=1))) {
     return(c(noise_weight, tail(mod$varExplained$Cumu, n=1)))
   }
}

apply(matrix(c(1:10/10), nrow = 1), MARGIN = 2, var_func)
```

```
##          [,1]    [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,]  0.10000  0.2000  0.30000  0.40000  0.50000  0.60000  0.70000  0.80000
## [2,] 98.55884 96.2226 95.23582 93.49471 91.66273 87.48346 89.34708 86.83083
##          [,9]    [,10]
## [1,]  0.90000  1.00000
## [2,] 85.25284 85.02745
```

```
noise_test <- apply(matrix(c(1:200/5), nrow = 1), MARGIN = 2, var_func)
```

```
noise_test <- as.data.frame(do.call(rbind, noise_test))
colnames(noise_test) <- c("Noise", "Var_Explained")

plot(noise_test$Noise, noise_test$Var_Explained)
```



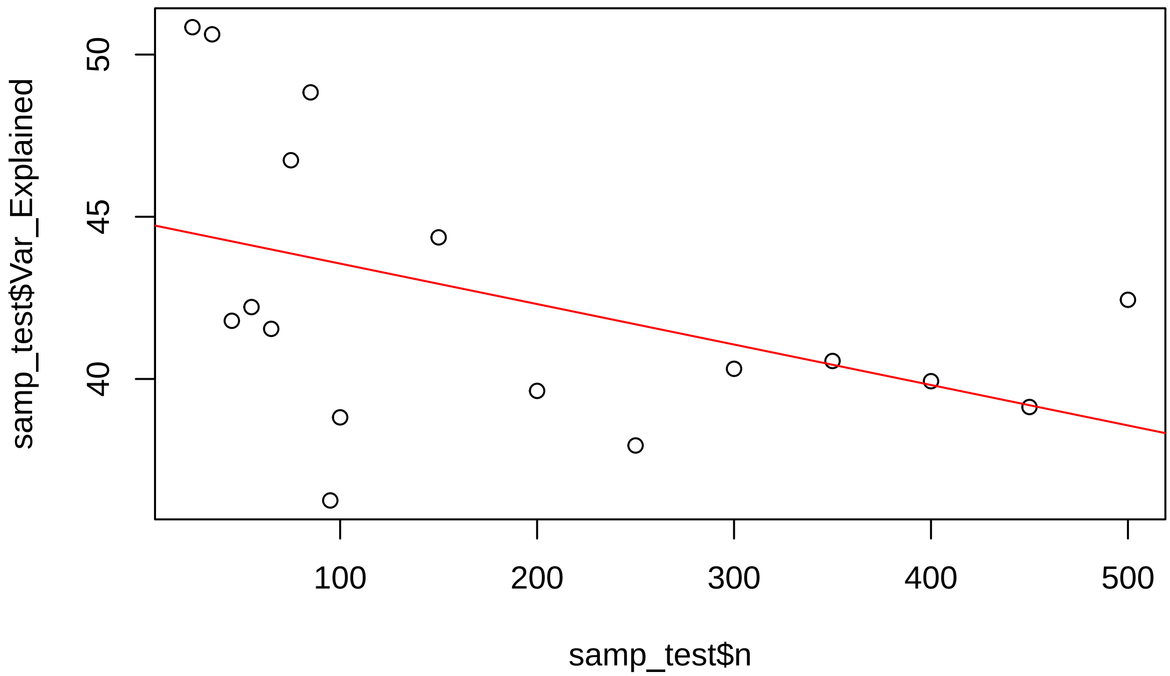
As we would predict, model performance decreases as the amount of noise increases. Initially, model performance decreases at a fairly rapid rate before becoming more gradual. Eventually, we would expect the percent variance explained to go to 0.

Sample Size Test

```
samp_func <- function(n, noise_weight, noise_type = "mvrnorm"){
   sim <- sim_data(n = n, p = 100, q = 5, noise_weight = noise_weight, noise_type = noise_type)
   mod <- ddsPLS(sim$X, sim$Y)
   if(!is.null(tail(mod$varExplained$Cumu, n=1))) {
     return(c(n, tail(mod$varExplained$Cumu, n=1)))
   }
}
```

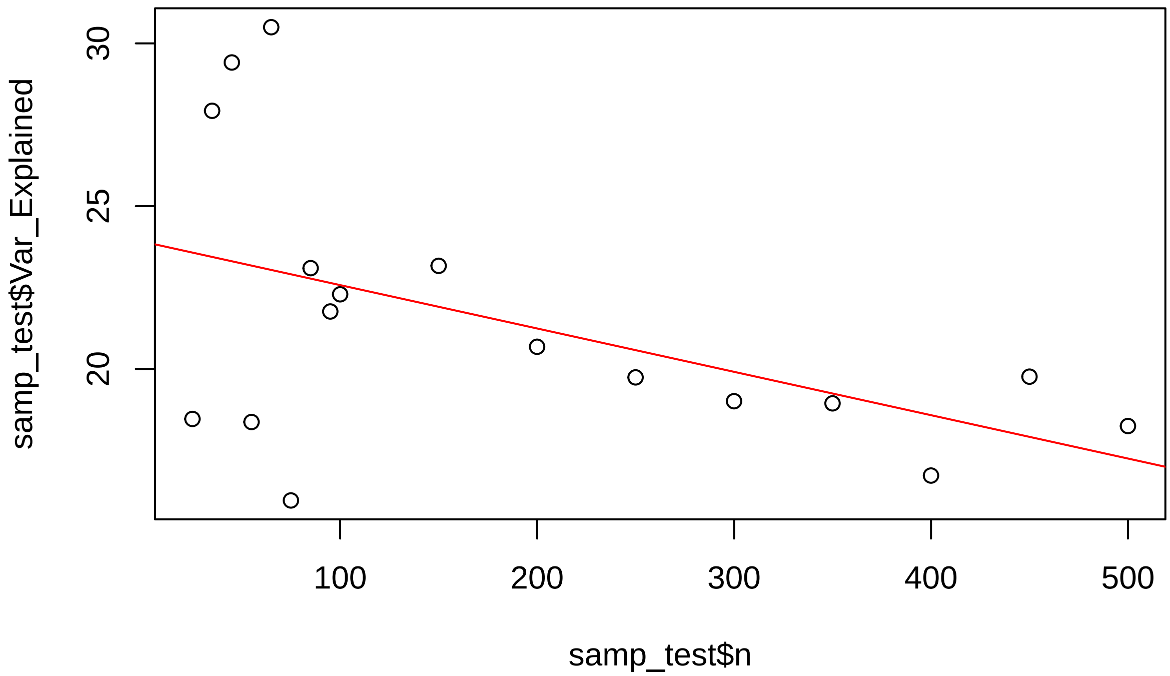
```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 10),
                            seq(from = 100, to = 500, by = 50)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 7)
```

```
samp_test <- as.data.frame(t(samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

reg <- lm(Var_Explained ~ n, data = samp_test)

plot(samp_test$n, samp_test$Var_Explained)
abline(reg, col = "red")
```



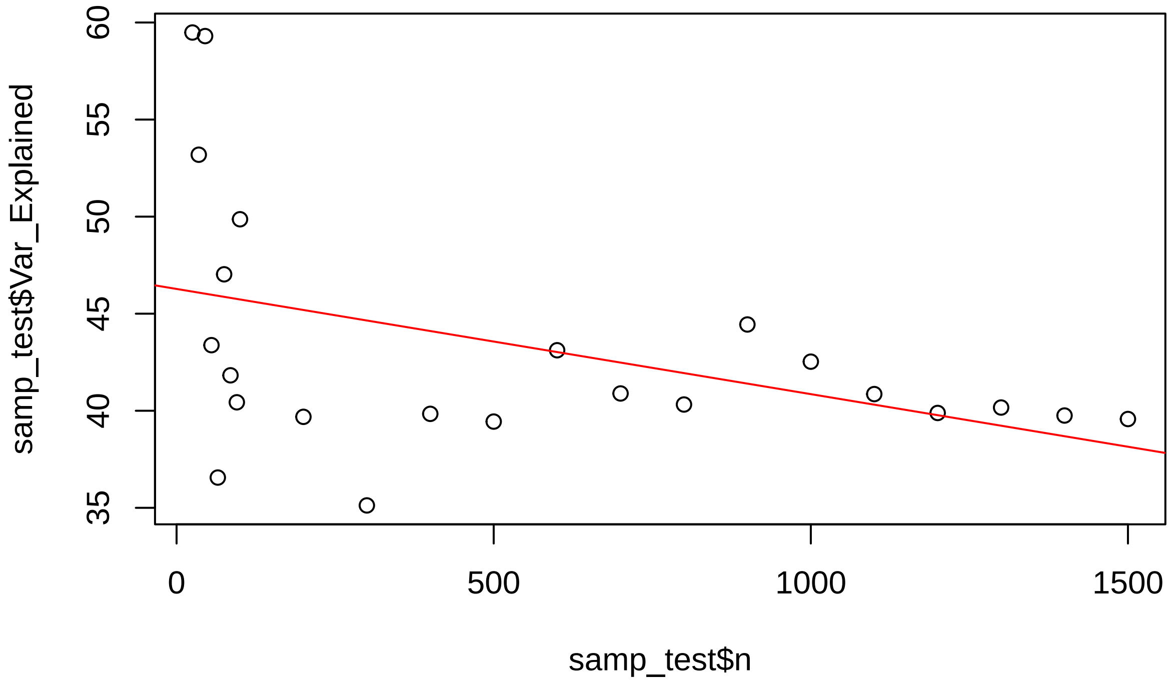

```r
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 10),
                            seq(from = 100, to = 500, by = 50)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 20)
```

```r
samp_test <- as.data.frame(t(samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

reg <- lm(Var_Explained ~ n, data = samp_test)

plot(samp_test$n, samp_test$Var_Explained)
abline(reg, col = "red")
```




```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 10),
                            seq(from = 100, to = 1500, by = 100)),
                          nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 7)
```

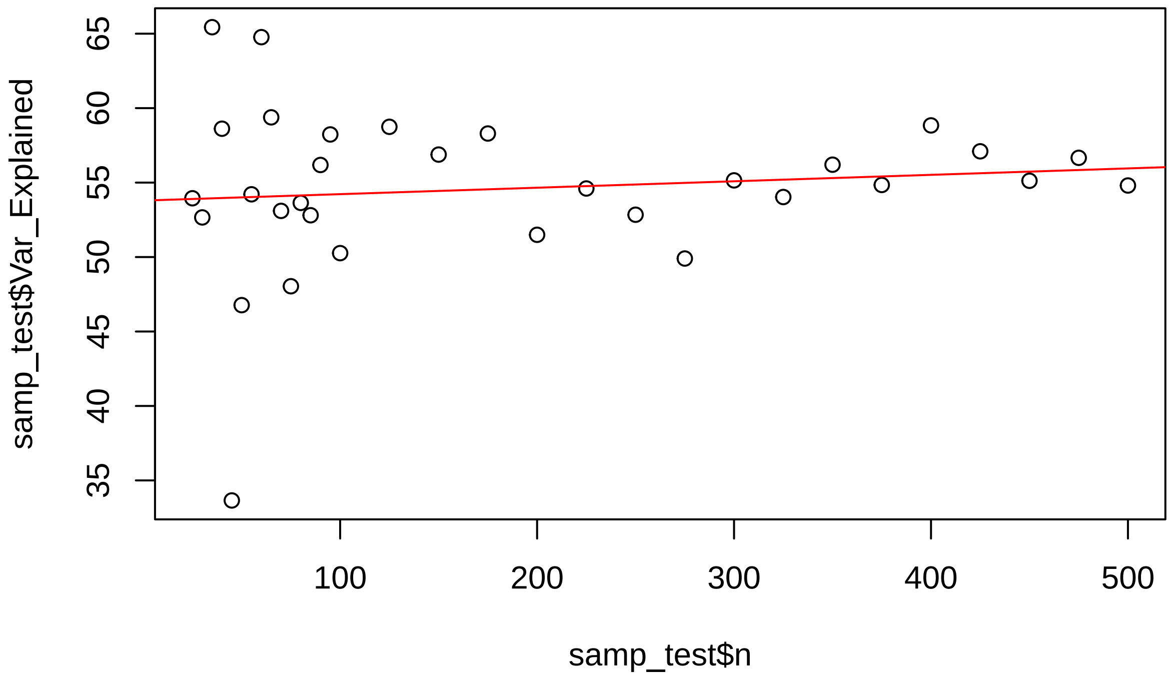
```
samp_test <- as.data.frame(t(samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

reg <- lm(Var_Explained ~ n, data = samp_test)

plot(samp_test$n, samp_test$Var_Explained)
abline(reg, col = "red")
```



Model performance seems to be much more variable at a low sample size before stabilizing. It looks like there may be a slight improvement as model size increases however this would need more inquiry. I am curious as to why models with small sample size can perform much better than those based on a larger sample size.

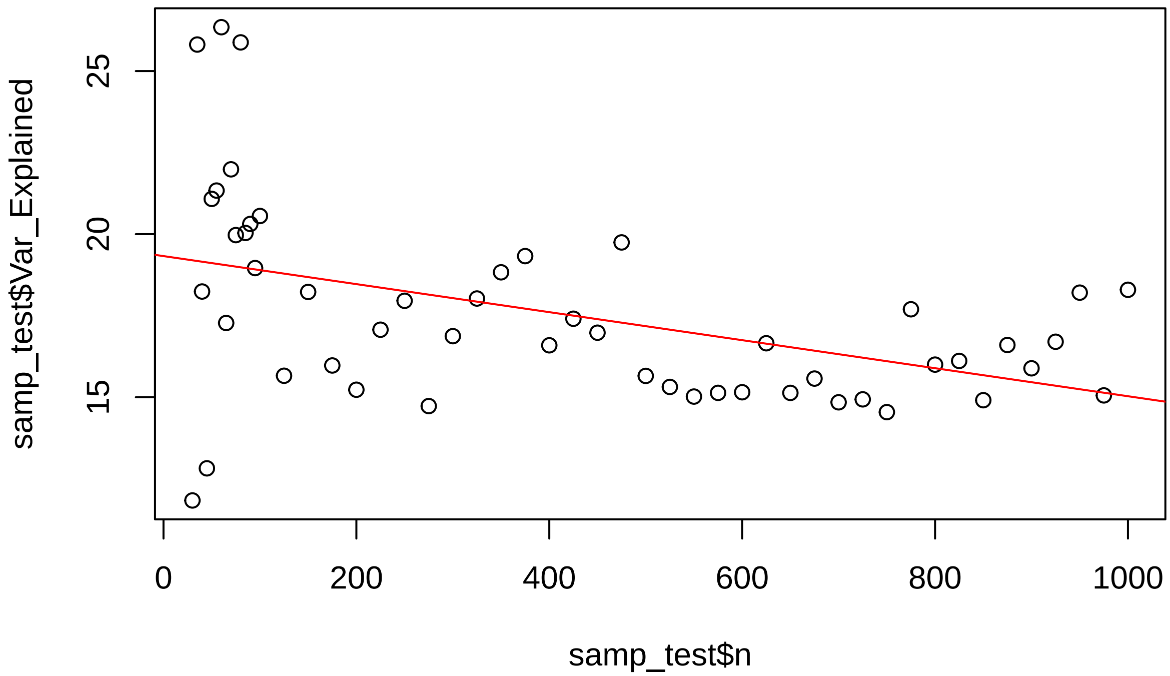
```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 5),
                            seq(from = 100, to = 500, by = 25)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 2,
                   noise_type = "rnorm")
```

```
samp_test <- as.data.frame(t(samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

reg <- lm(Var_Explained ~ n, data = samp_test)

plot(samp_test$n, samp_test$Var_Explained)
abline(reg, col = "red")
```



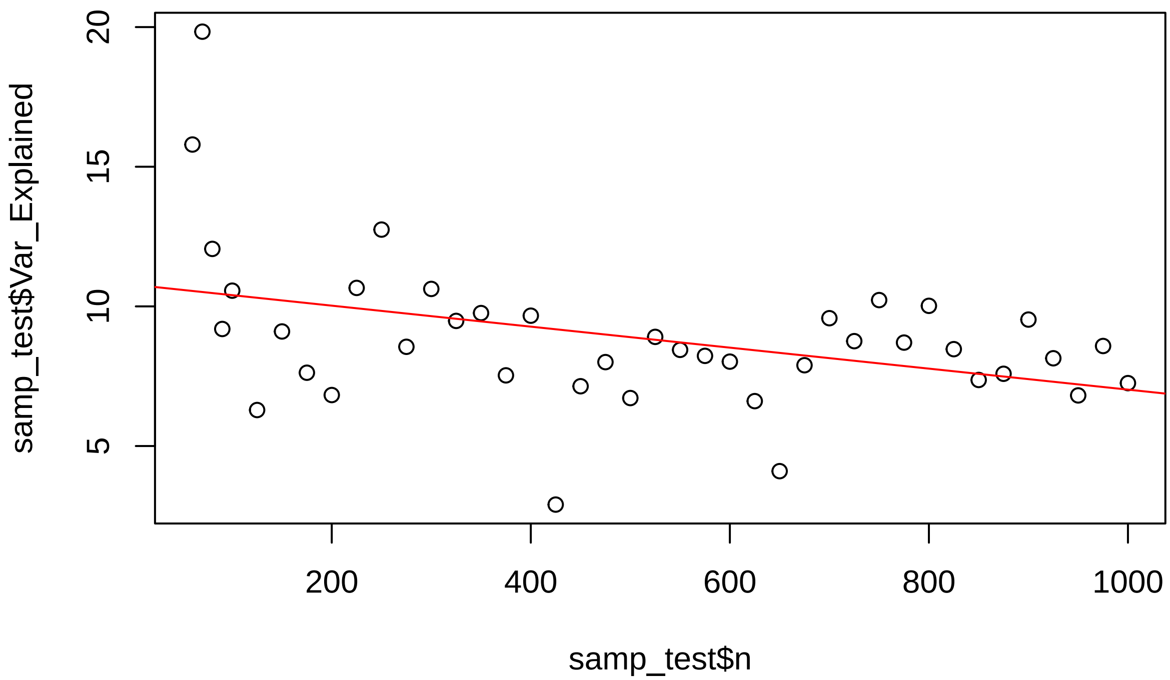

```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 5),
                            seq(from = 100, to = 1000, by = 25)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 5,
                   noise_type = "rnorm")
```

```
# Removes null values from list
samp_test <- Filter(Negate(is.null), samp_test)

samp_test <-as.data.frame(do.call(rbind, samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

reg <- lm(Var_Explained ~ n, data = samp_test)

plot(samp_test$n, samp_test$Var_Explained)
abline(reg, col = "red")
```




```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 5),
                            seq(from = 100, to = 1000, by = 25)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 7,
                   noise_type = "rnorm")
```

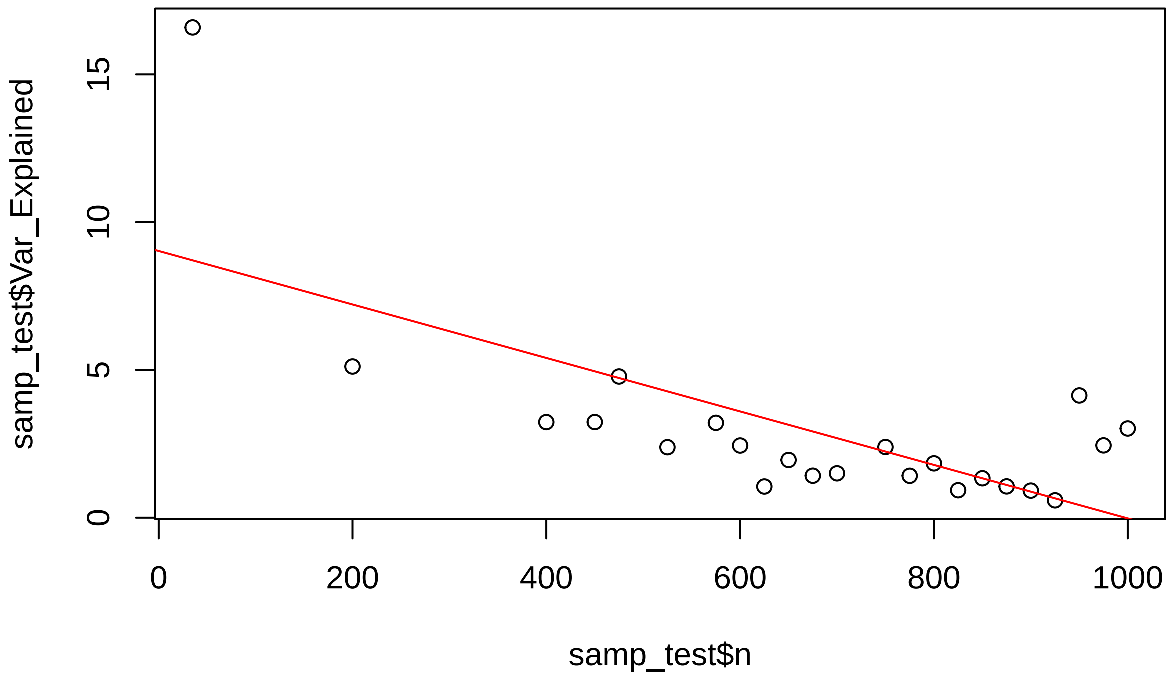
```
# Removes null values from list
samp_test <- Filter(Negate(is.null), samp_test)

samp_test <-as.data.frame(do.call(rbind, samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

reg <- lm(Var_Explained ~ n, data = samp_test)

plot(samp_test$n, samp_test$Var_Explained)
abline(reg, col = "red")
```



It looks like model performance decreases as sample size increases. I am quite confused by this result and should look at it across levels of noise.

```r
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 5),
                            seq(from = 100, to = 1000, by = 25)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 10,
                   noise_type = "rnorm")
```

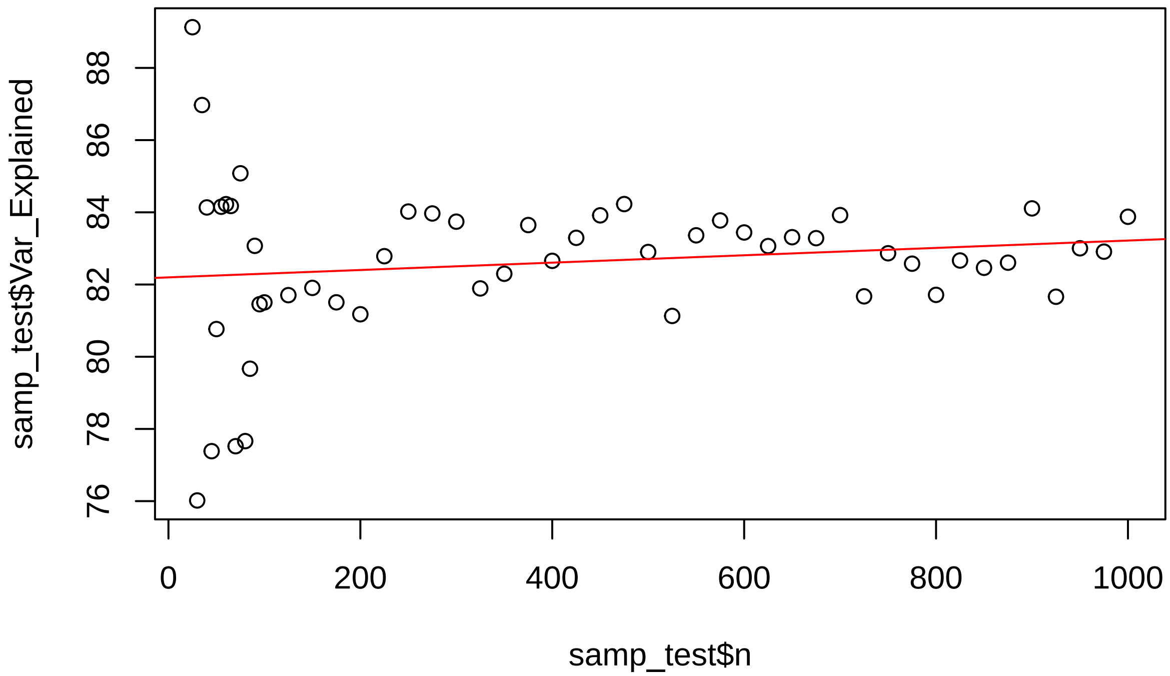
```r
# Removes null values from list
samp_test <- Filter(Negate(is.null), samp_test)

samp_test <-as.data.frame(do.call(rbind, samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

reg <- lm(Var_Explained ~ n, data = samp_test)

plot(samp_test$n, samp_test$Var_Explained)
abline(reg, col = "red")
```



When a large amount of noise is added, it requires a larger sample size in order to

```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 5),
                            seq(from = 100, to = 1000, by = 25)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 1,
                   noise_type = "rnorm")
```

```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

reg <- lm(Var_Explained ~ n, data = samp_test)

plot(samp_test$n, samp_test$Var_Explained)
abline(reg, col = "red")
```



It might just be that there is more variance in the performance of models on smaller sized samples.