Novel Simulations

Harpeth Lee

3/19/2022

```
library(ddsPLS2)
```

```
## Loading required package: foreach
## 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)
}
```

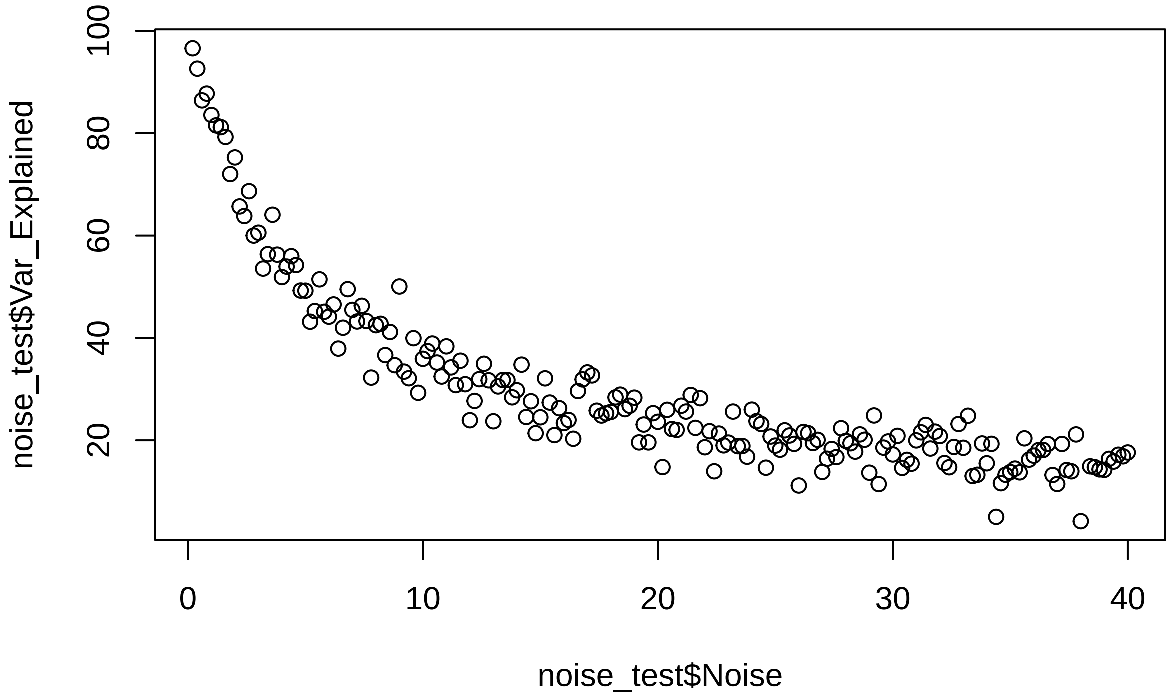
Noise Test

```
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  0.20000  0.30000  0.40000  0.50000  0.60000  0.7000  0.80000
## [2,] 98.42402 97.21253 95.12897 92.32721 90.29365 90.49785 88.7718 84.30148
##          [,9]    [,10]
## [1,]  0.90000  1.00000
## [2,] 81.55229 82.38865
```

```
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)
```



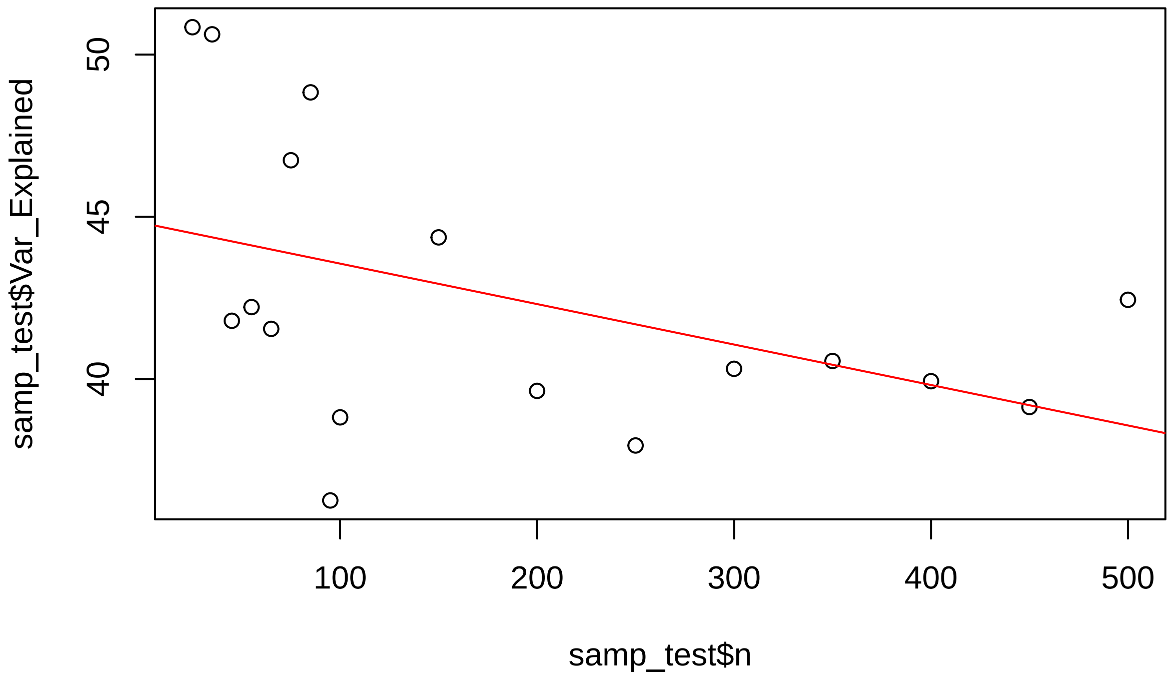
As we would predict, model performance decreases as the amount of noise increases. Initially, model performance decreases at a fairly rapid rate before becoming more gradual. Eventually, we would expect the percent variance explained to go to 0.

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)))
   }
}
```

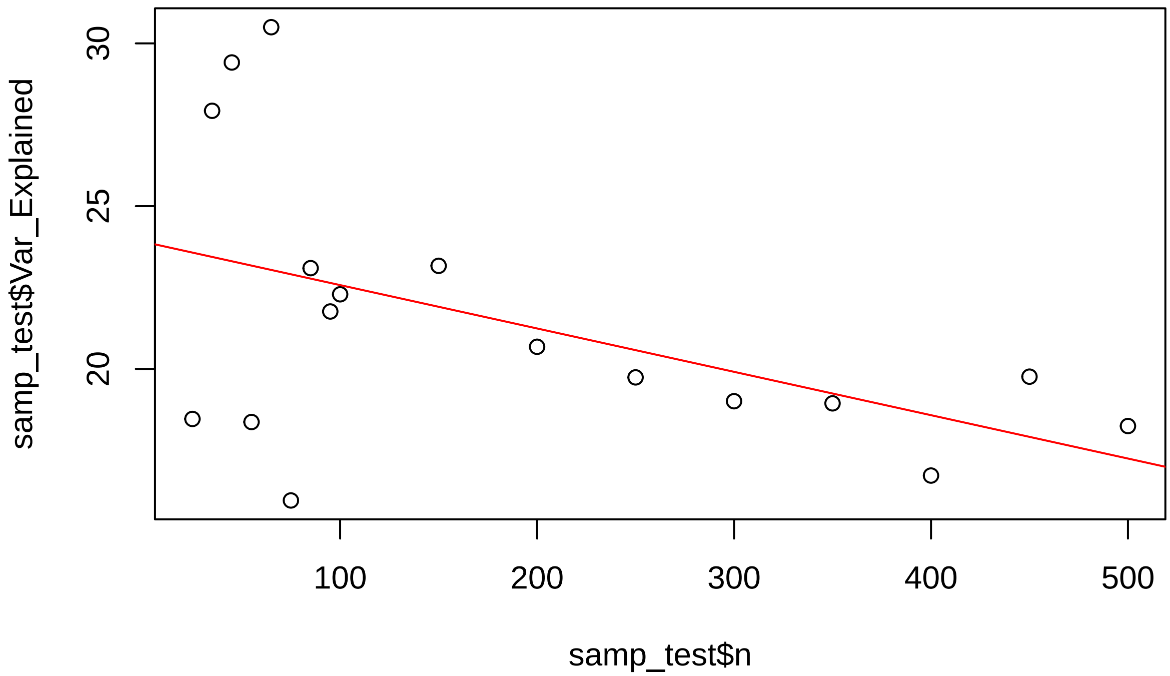
```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 10),
                            seq(from = 100, to = 500, by = 50)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 7)
```

```
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")
```



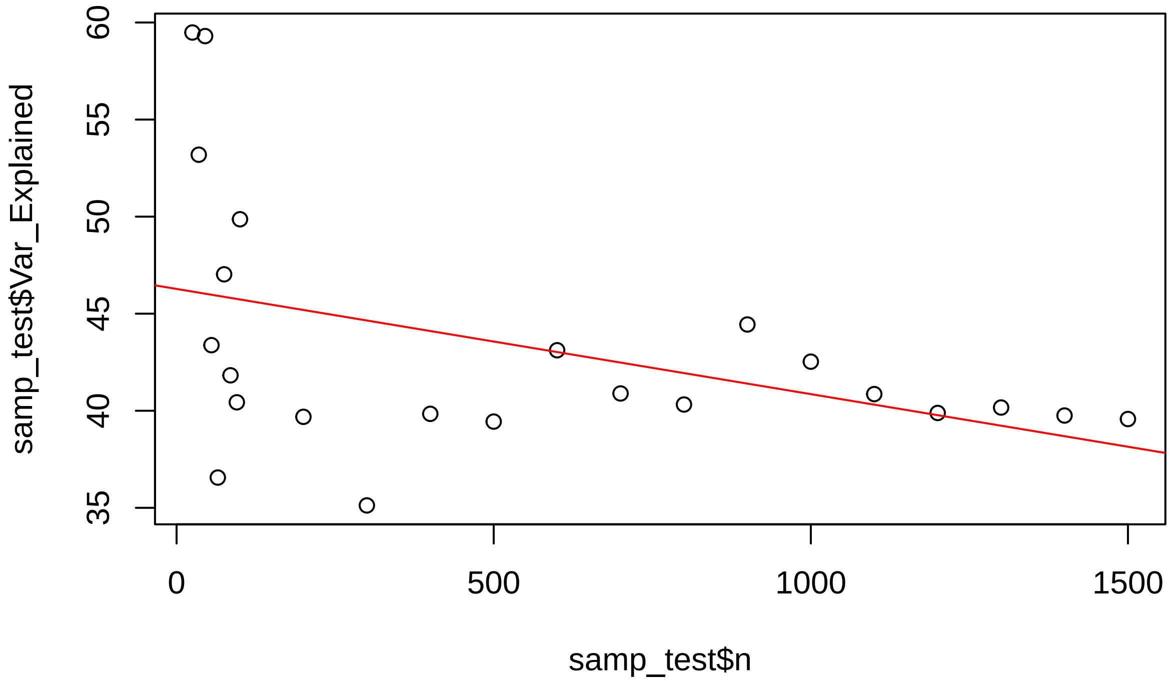

```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 10),
                            seq(from = 100, to = 500, by = 50)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 20)
```

```
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)
```

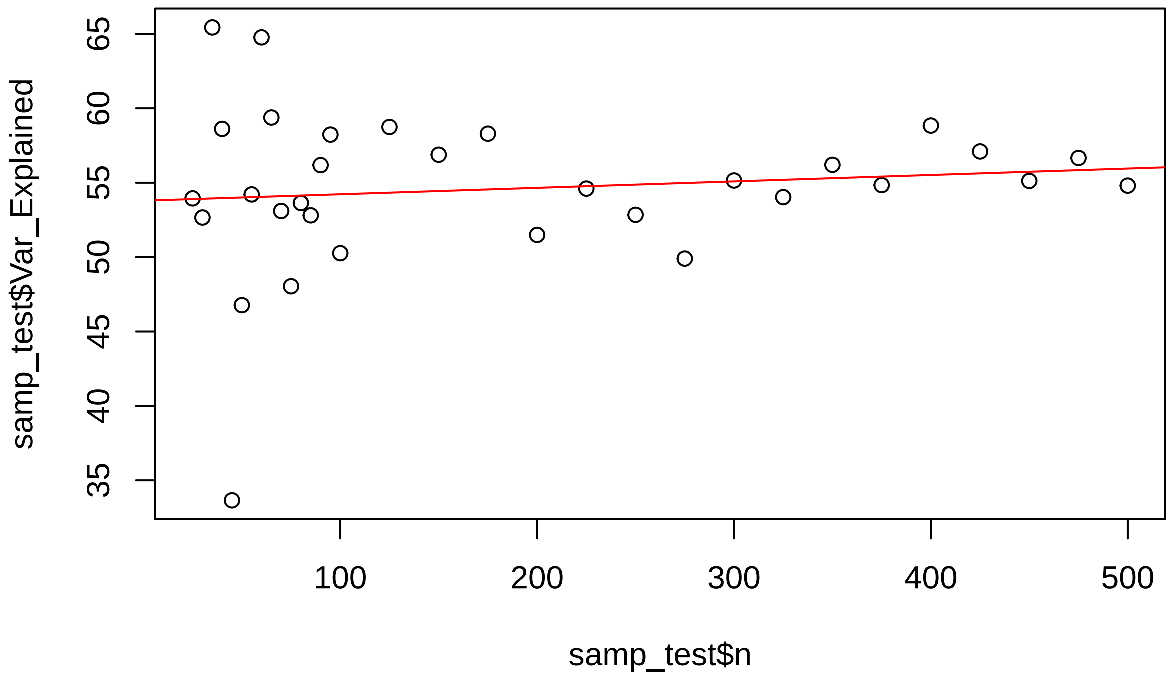
```
samp_test <- as.data.frame(t(samp_test))
colnames(samp_test) <- c("n", "Var_Explained")

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.

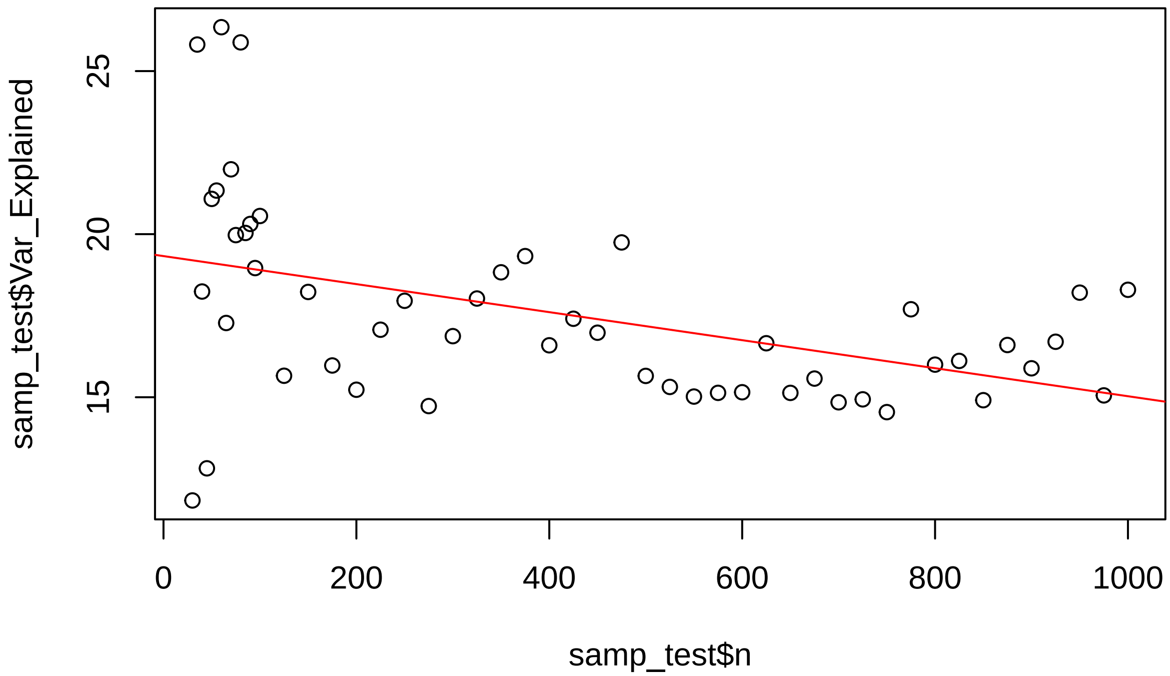
```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 5),
                            seq(from = 100, to = 500, by = 25)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 2,
                   noise_type = "rnorm")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

```
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")
```



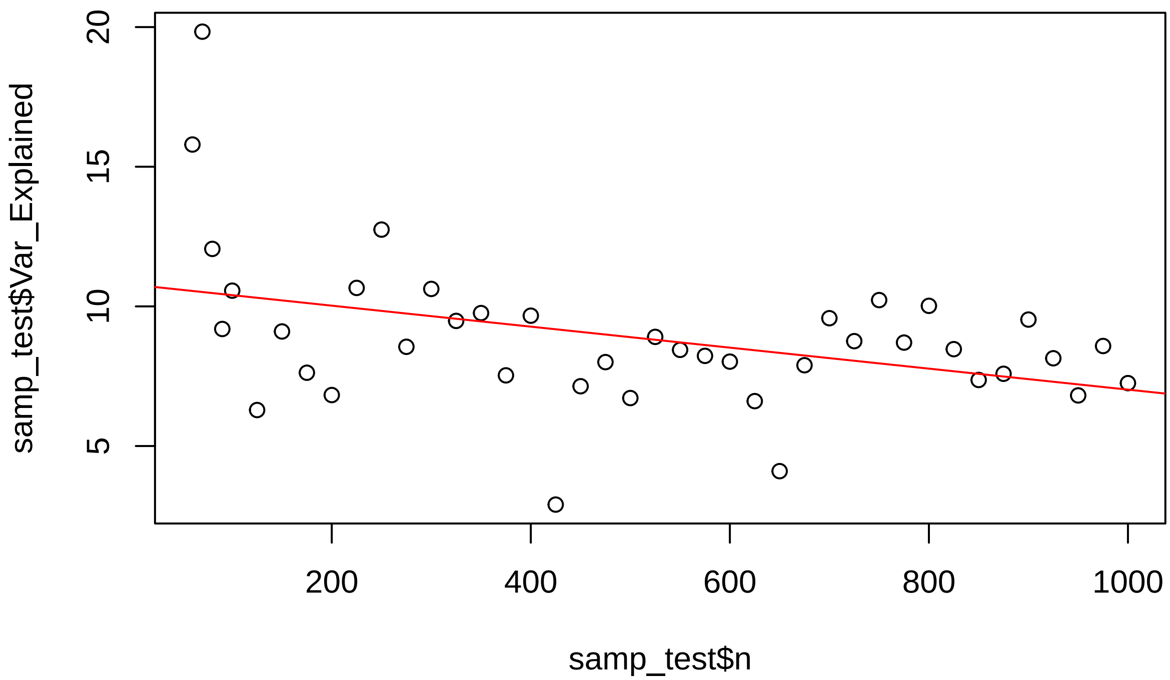

```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 5),
                            seq(from = 100, to = 1000, by = 25)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_func,
                   noise_weight = 5,
                   noise_type = "rnorm")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

```
# 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")
```

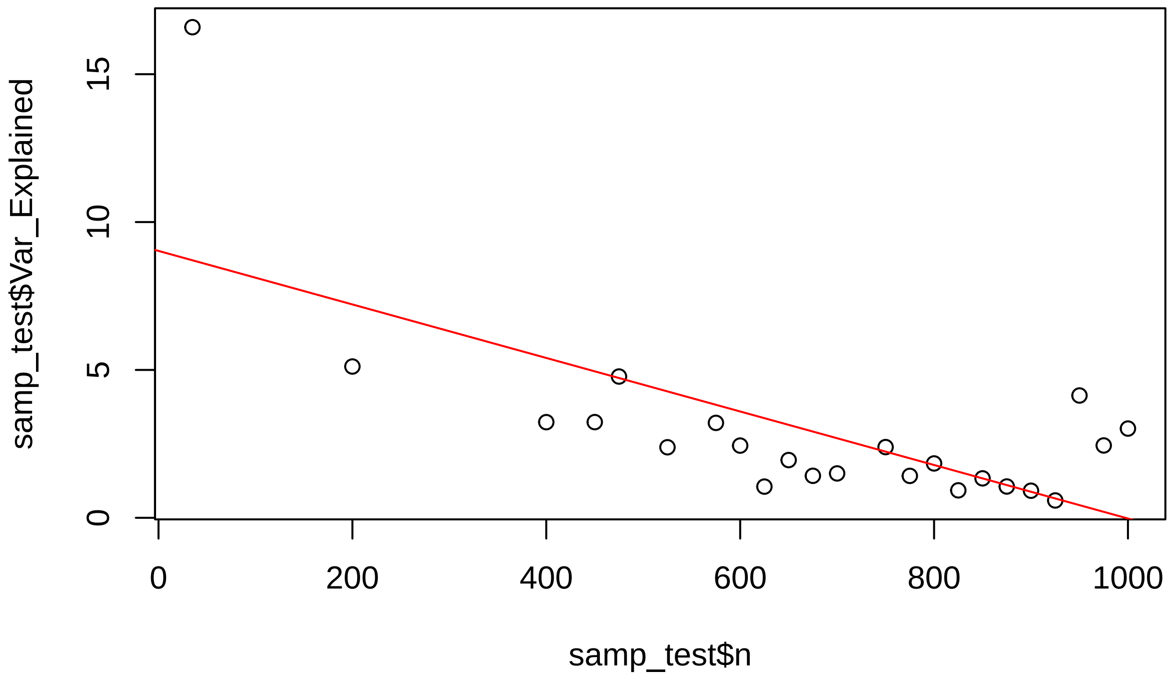
```
# Removes null values from list
samp_test <- Filter(Negate(is.null), samp_test)

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.

```
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")
```

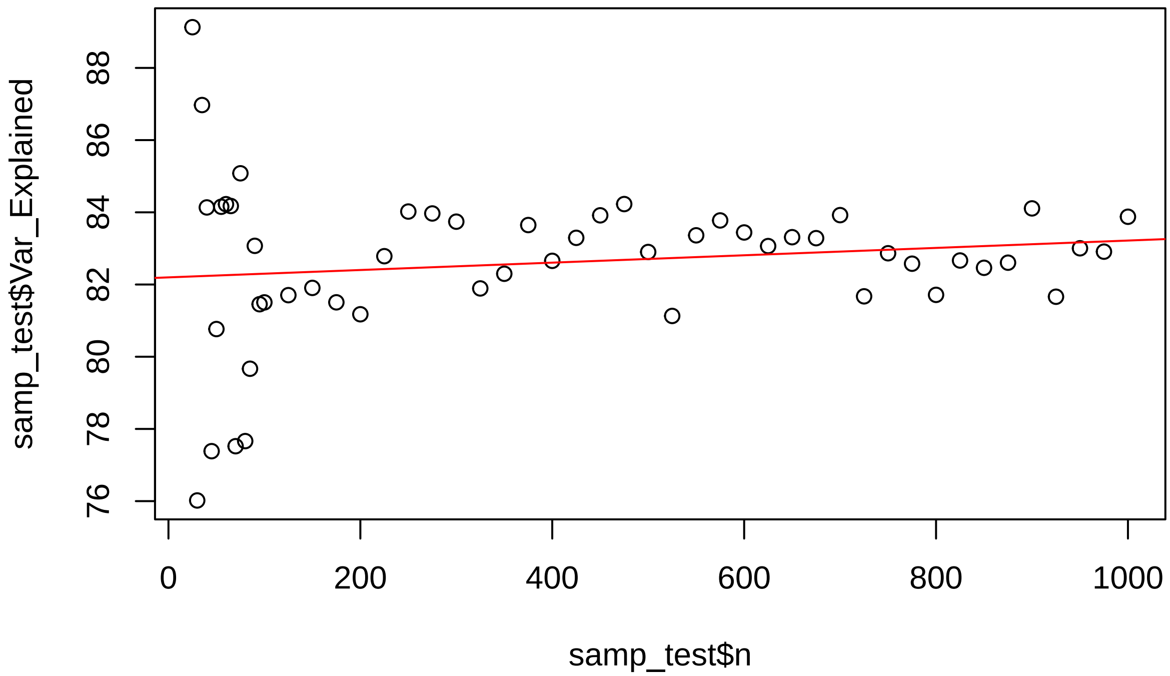
```
# Removes null values from list
samp_test <- Filter(Negate(is.null), samp_test)

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")
```



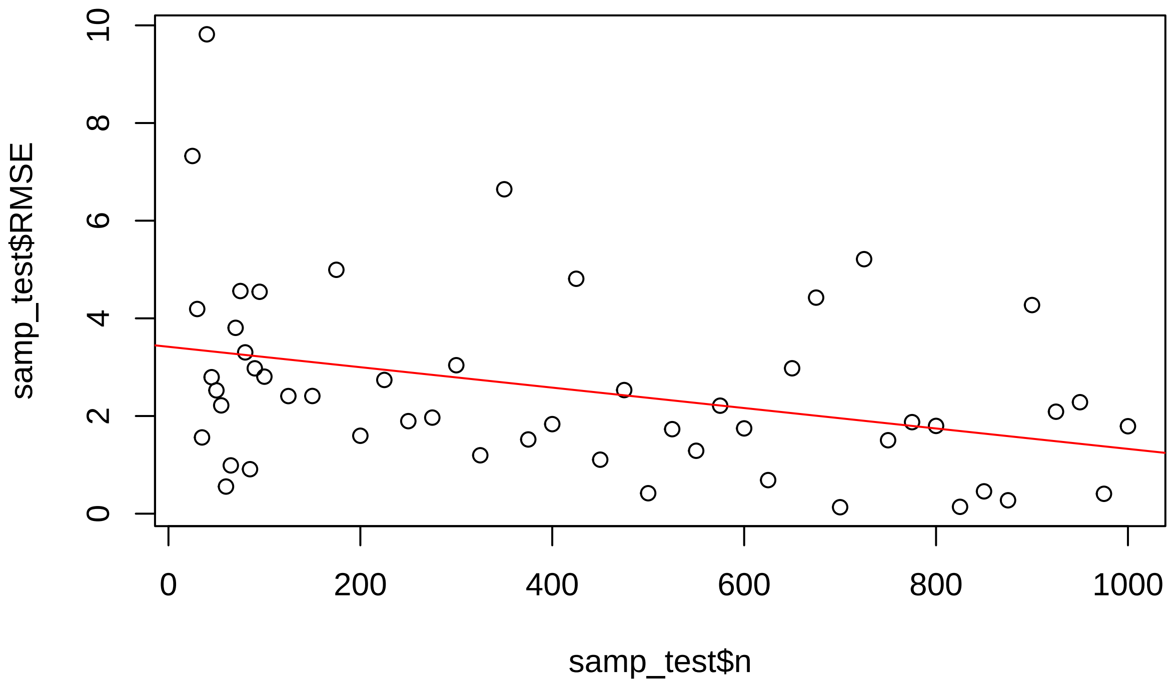
It might just be that there is more variance in the performance of models on smaller sized samples.

From the gathered samples there does not appear to be as clear a relationship between sample size and model performance as one would hope. It looks like model performance may decrease with sample size as noise increases. I will run a larger sample on this.

```
samp_rmse <- function(n, noise_weight, noise_type = "mvrnorm"){
   sim <- sim_data(n = n+100, p = 100, q = 5, noise_weight = noise_weight, noise_type = noise_type)

   split <- sample(c(rep(0, n), rep(1, 100)))

   sim_train_X <- sim$X[split == 0, ]
   sim_train_Y <- sim$Y[split == 0, ]

   sim_test_X <- sim$X[split == 1, ]
   sim_test_Y <- sim$Y[split == 1, ]

   mod <- ddsPLS(sim_train_X, sim_train_Y)

   preds <- predict(mod, sim_test_X)

   rmse <- sqrt(sum(preds$y_est - sim_test_Y)^2/nrow(sim_test_Y))

   return(c(n, rmse))
}
```

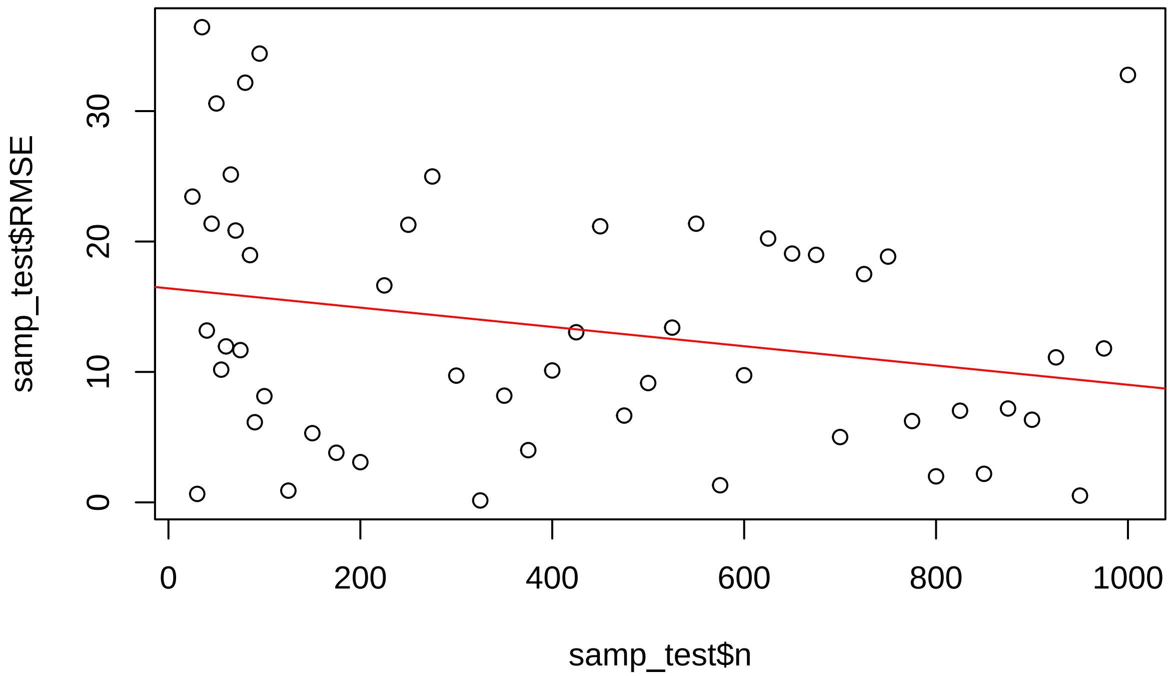
```
samp_test <- apply(matrix(c(seq(from = 25, to = 95, by = 5),
                            seq(from = 100, to = 1000, by = 25)),
                            nrow = 1),
                   MARGIN = 2,
                   samp_rmse,
                   noise_weight = 1,
                   noise_type = "rnorm")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

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

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

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




```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_rmse,
                   noise_weight = 5,
                   noise_type = "rnorm")
```

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

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

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



It looks the mean and variance of the RMSE both decrease as sample size increases.

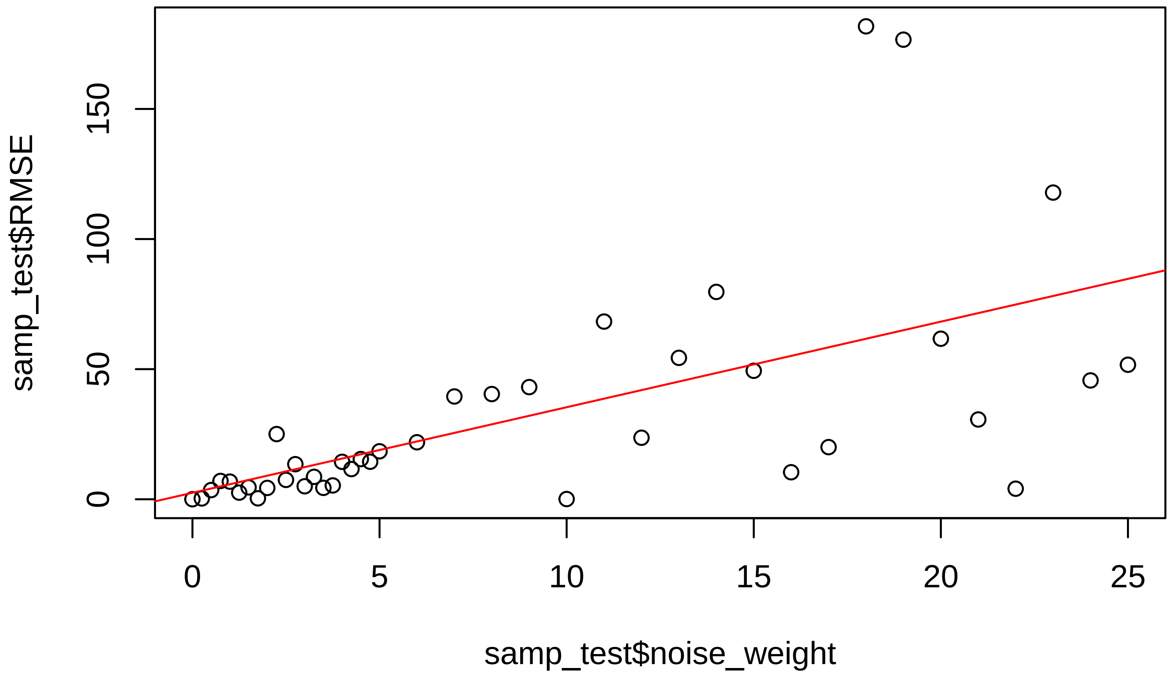
Noise and Test RMSE

```
noise_rmse <- function(noise_weight){
   sim <- sim_data(n = 150, p = 100, q = 5, noise_weight = noise_weight, noise_type = "rnorm")

   split <- sample(c(rep(0, 50), rep(1, 100)))

   sim_train_X <- sim$X[split == 0, ]
   sim_train_Y <- sim$Y[split == 0, ]

   sim_test_X <- sim$X[split == 1, ]
   sim_test_Y <- sim$Y[split == 1, ]

   mod <- ddsPLS(sim_train_X, sim_train_Y)

   preds <- predict(mod, sim_test_X)

   rmse <- sqrt(sum(preds$y_est - sim_test_Y)^2/nrow(sim_test_Y))

   return(c(noise_weight, rmse))
}
```

```
samp_test <- apply(matrix(c(seq(from = 0, to = 5, by = 0.25),
                            seq(from = 6, to = 25, by = 1)),
                            nrow = 1),
                   MARGIN = 2,
                   noise_rmse)
```

```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("noise_weight", "RMSE")
```

```r
reg <- lm(RMSE ~ noise_weight, data = samp_test)

plot(samp_test$noise_weight, samp_test$RMSE)
abline(reg, col = "red")
```



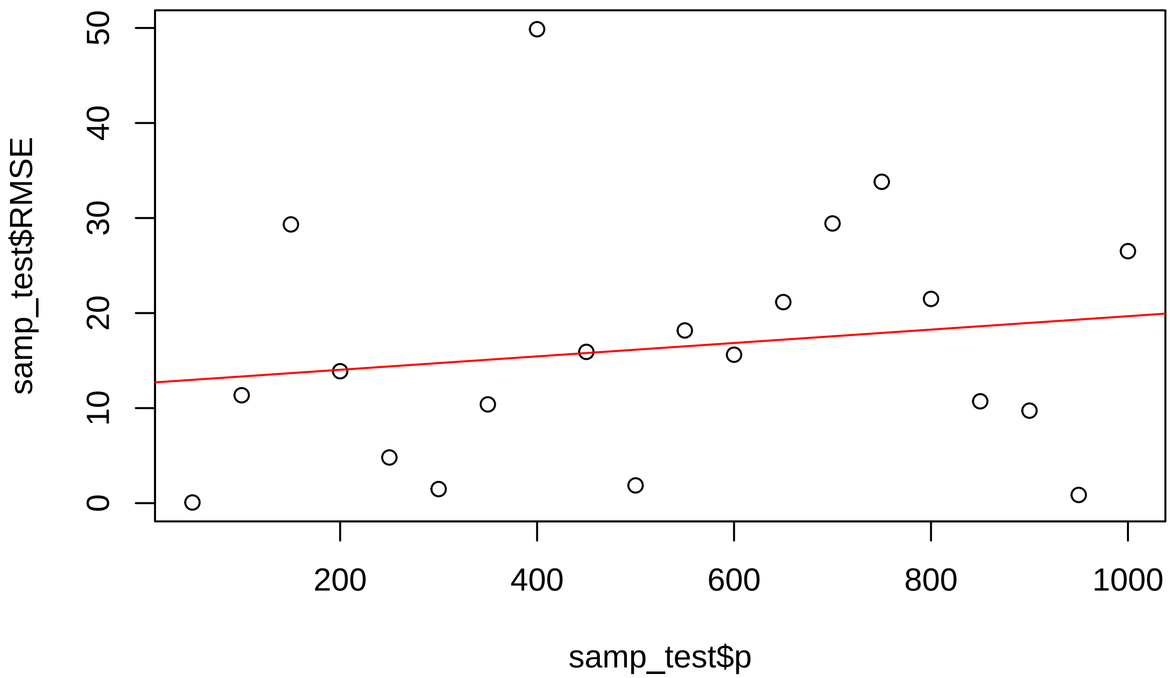
Predictors and RMSE

```r
p_rmse <- function(p, noise_weight = 5, n = 150){

   # Randomly simulates data
   sim <- sim_data(n = n, p = p, q = 5, noise_weight = noise_weight, noise_type = "rnorm")

   # Splits into training and test
   in_train <- round(n/3)
   in_test <- round(2*n/3)

   split <- sample(c(rep(0, in_train), rep(1, in_test)))

   sim_train_X <- sim$X[split == 0, ]
   sim_train_Y <- sim$Y[split == 0, ]

   sim_test_X <- sim$X[split == 1, ]
   sim_test_Y <- sim$Y[split == 1, ]

   # Generates model using the training set
   mod <- ddsPLS(sim_train_X, sim_train_Y)

   # Makes prediction and calculates the RMSE
   preds <- predict(mod, sim_test_X)

   rmse <- sqrt(sum(preds$y_est - sim_test_Y)^2/nrow(sim_test_Y))
```

```
  return(c(p, rmse))
}
```

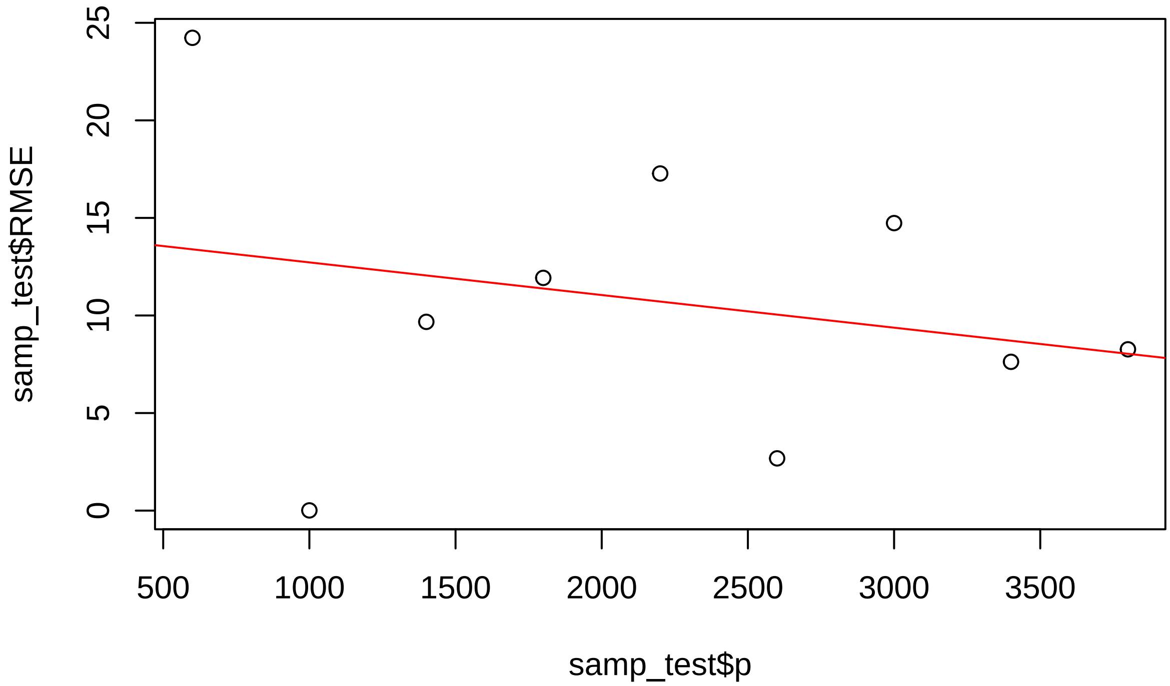
```
samp_test <- apply(matrix(seq(from = 50, to = 1000, by = 50),
                          nrow = 1),
                   MARGIN = 2,
                   p_rmse)
```

```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("p", "RMSE")

reg <- lm(RMSE ~ p, data = samp_test)

plot(samp_test$p, samp_test$RMSE)
abline(reg, col = "red")
```




```
samp_test <- apply(matrix(seq(from = 600, to = 4000, by = 400),
                          nrow = 1),
                   MARGIN = 2,
                   p_rmse)
```

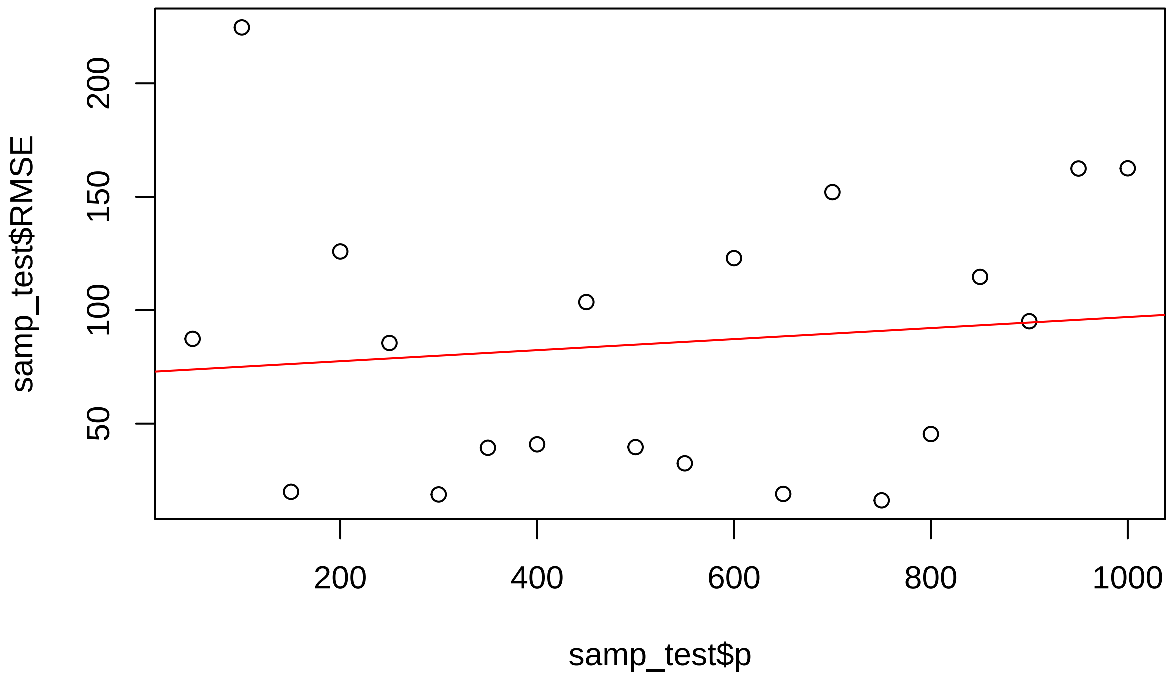
```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("p", "RMSE")

reg <- lm(RMSE ~ p, data = samp_test)

plot(samp_test$p, samp_test$RMSE)
abline(reg, col = "red")
```



Should look more into how changing the number of predictors effects model performance. Initial results look like model performance is able to still perform fairly well even as the number of predictors increases a fair amount.

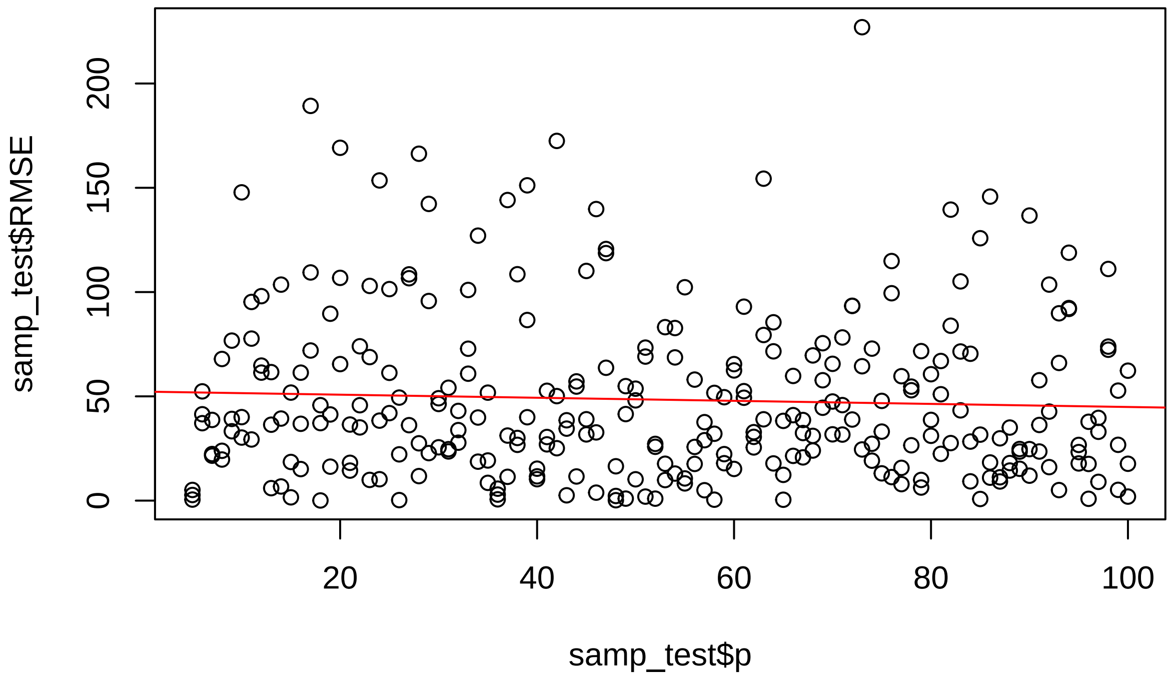
```
samp_test <- apply(matrix(seq(from = 50, to = 1000, by = 50),
                          nrow = 1),
                   MARGIN = 2,
                   p_rmse,
                   noise_weight = 25)
```

```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("p", "RMSE")

reg <- lm(RMSE ~ p, data = samp_test)

plot(samp_test$p, samp_test$RMSE)
abline(reg, col = "red")
```




```r
samp_test <- apply(matrix(rep(seq(from = 5, to = 100, by = 1), 3),
                          nrow = 1),
                   MARGIN = 2,
                   p_rmse,
                   noise_weight = 15)
```

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

reg <- lm(RMSE ~ p, data = samp_test)

plot(samp_test$p, samp_test$RMSE)
abline(reg, col = "red")
```



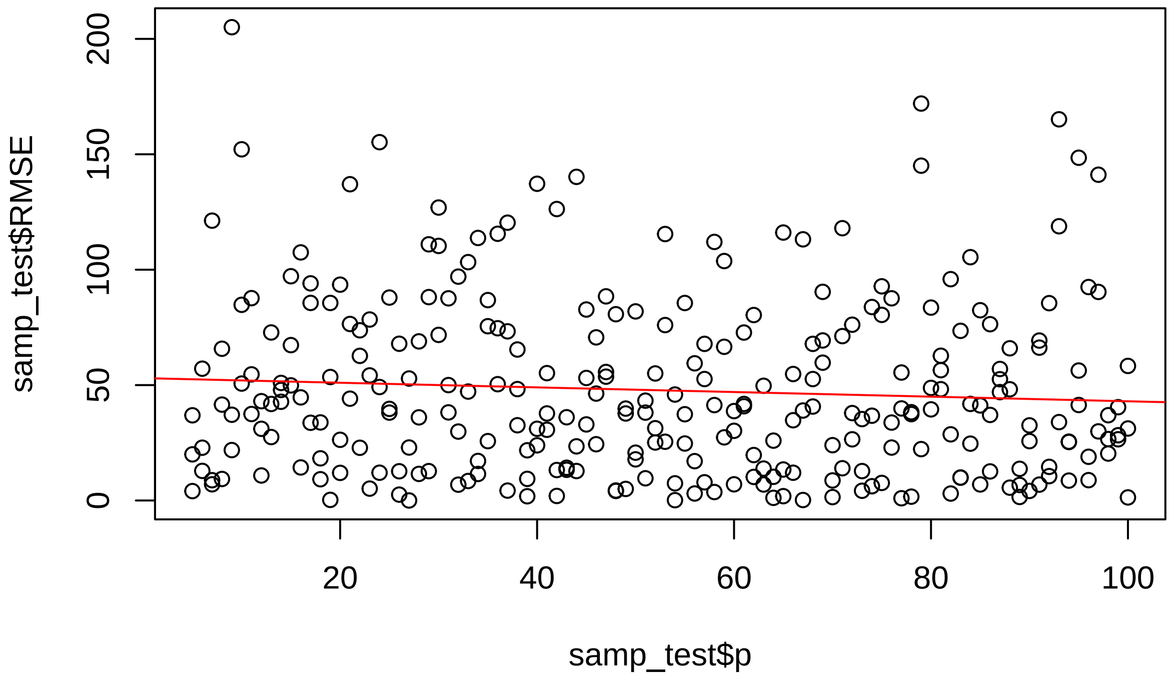

```
reg$coefficients
```

```
## (Intercept)           p
## 52.27019515 -0.07362973
```

At this low level, increasing p has little effect on the model performance. It even looks like the RMSE decreases as p increases. We may want to give the data a slightly more complex structure and see how adding more noise and other changes effect the output.

```
samp_test <- apply(matrix(rep(seq(from = 5, to = 100, by = 1), 3),
                          nrow = 1),
                   MARGIN = 2,
                   p_rmse,
                   noise_weight = 15,
                   n = 60)
```

```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("p", "RMSE")

reg <- lm(RMSE ~ p, data = samp_test)

plot(samp_test$p, samp_test$RMSE)
abline(reg, col = "red")
```

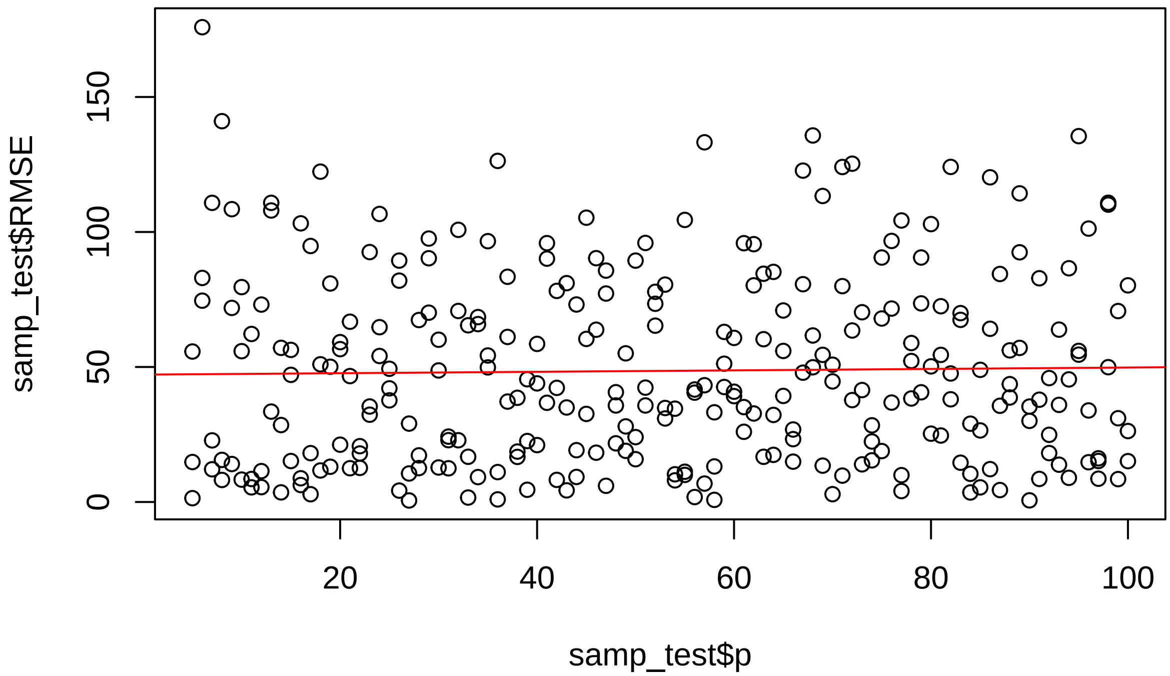



```
reg$coefficients
```

```
## (Intercept)           p
##  53.0219520  -0.1004443
```

```
samp_test <- apply(matrix(rep(seq(from = 5, to = 100, by = 1), 3),
                          nrow = 1),
                   MARGIN = 2,
                   p_rmse,
                   noise_weight = 15,
                   n = 45)
```

```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("p", "RMSE")

reg <- lm(RMSE ~ p, data = samp_test)

plot(samp_test$p, samp_test$RMSE)
abline(reg, col = "red")
```




```r
reg$coefficients
```

```
## (Intercept)           p
## 47.17100736  0.02625519
```