More Simulations

Harpeth Lee

3/29/2022

```
library(ddsPLS2)
library(MASS)
library(pls)
library(mixOmics)
library(glmnet)
library(ggplot2)
```

New Data Simulation Method

```
sim_data <- function(n = 25, p = 50, q = 5, R = 5, noise_weight = 1, D_method = "newest", noise_type =

  # Creates A and D matrices

  if(D_method == "complex") {
    R = 13
  }

  Row1 <- c(rep(1, 5), rep(0, p - 5))
  Row2 <- c(rep(0, 5), rep(1, 10), rep(0, p - 15))

  reps1 <- round(3*R/5)
  reps2 <- R - reps1

  A1 <- do.call("rbind", replicate(reps1, Row1, simplify = FALSE))
  A2 <- do.call("rbind", replicate(reps2, Row2, simplify = FALSE))

  A <- rbind(A1, A2)


  if(struc == "complex") {
    R = 13

    Row1 <- c(rep(1, 5), rep(0, p - 5))
    Row2 <- c(rep(0, 5), rep(1, 10), rep(0, p - 15))
    Row3 <- c(rep(0, 15), rep(1, 5), rep(0, p - 20))
    Row4 <- c(rep(0, 20), 1, rep(0, p - 21))
    Row5 <- c(rep(0, 21), rep(1, 2), rep(0, p - 23))

    A1 <- do.call("rbind", replicate(3, Row1, simplify = FALSE))
    A2 <- do.call("rbind", replicate(2, Row2, simplify = FALSE))
    A3 <- do.call("rbind", replicate(3, Row3, simplify = FALSE))
    A4 <- do.call("rbind", replicate(2, Row4, simplify = FALSE))
```

```
  A5 <- do.call("rbind", replicate(3, Row5, simplify = FALSE))

  A <- rbind(A1, A2, A3, A4, A5)

}


if(D_method == "new") {

  D <- matrix(rep(1, R*q), nrow = R)

} else if(D_method == "diag") {

  D <- diag(max(q, R))[1:R, 1:q]

} else if(D_method == "simple") {

  D <- cbind(rep(1, R), matrix(rep(0, R*(q-1)), nrow = R))

} else if(D_method == "complex") {


  D1 <- c(rep(1, 3), rep(0, 10))
  D2 <- c(rep(0, 3), rep(1, 3), rep(0, 7))
  D3 <- c(rep(0, 6), rep(1, 3), rep(0, 4))
  D4 <- c(rep(0, 9), rep(1, 3), rep(0, 1))
  D5 <- matrix(rep(0, R*(q-4)), nrow = R)

  D <- cbind(D1, D2, D3, D4, D5)

} else {
   q_s <- round(q/4)

   if(q_s == 0) {
     q_s = 1
   }

   Row1D <- c(rep(1, q_s), rep(0, q - q_s))
   Row2D <- c(rep(0, q_s), rep(1, q_s), rep(0, q - 2*q_s))

   reps1D <- round(3*R/5)
   reps2D <- R - reps1D

   D1 <- do.call("rbind", replicate(reps1D, Row1D, simplify = FALSE))
   D2 <- do.call("rbind", replicate(reps2D, Row2D, simplify = FALSE))

   D <- rbind(D1, D2)
}

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

```
  phi <- matrix(rnorm(n*R), nrow = n)


  # 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

  #X <- scale(X)
  #Y <- scale(Y)

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

```
pls_test <- function(n, sim, func, passed_arg) {
   # 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 and predicts RMSE
   if(func == "ddsPLS") {
     mod <- ddsPLS(sim_train_X, sim_train_Y)

     preds <- predict(mod, sim_test_X)
     preds_ib <- predict(mod, sim_train_X)

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

     pred_mean_tr <- t(replicate(nrow(sim_train_Y), colMeans(sim_train_Y)))
     R2 <- 1 - (sum((preds_ib$y_est - sim_train_Y)^2)/sum((sim_train_Y - pred_mean_tr)^2))
```

```
    pred_mean_ts <- t(replicate(nrow(sim_test_Y), colMeans(sim_test_Y)))
    Q2 <- 1 - (sum((preds$y_est - sim_test_Y)^2)/sum((sim_test_Y - pred_mean_ts)^2))

    ncomp <- mod$R
    } else if(func == "pls") {

    df <- data.frame(X = I(sim_train_X), Y = I(sim_train_Y))
    mod <- plsr(Y~X, data = df, ncomp = 10, method = "oscorespls", validation = "CV", scale = TRUE)

    R2 <- R2(mod)
    Q2 <- colSums(R2$val, dims = 2)

    ncomp <- which(Q2 == max(Q2)) - 1

    ncomp <- unname(ncomp)

    if(ncomp == 0){
      preds <- t(replicate(nrow(sim_test_Y), colMeans(sim_train_Y)))
      preds_ib <- t(replicate(nrow(sim_train_Y), colMeans(sim_train_Y)))
    } else {
      preds <- predict(mod, sim_test_X)[,,ncomp]
      preds_ib <- predict(mod, sim_train_X)[,,ncomp]
       }

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

    pred_mean_tr <- t(replicate(nrow(sim_train_Y), colMeans(sim_train_Y)))
    R2 <- 1 - (sum((preds_ib - sim_train_Y)^2)/sum((sim_train_Y - pred_mean_tr)^2))

    pred_mean_ts <- t(replicate(nrow(sim_test_Y), colMeans(sim_test_Y)))
    Q2 <- 1 - (sum((preds - sim_test_Y)^2)/sum((sim_test_Y - pred_mean_ts)^2))

  } else if(func == "lasso") {
    mod <- cv.glmnet(sim_train_X, sim_train_Y, family = "mgaussian")

    preds <- predict(mod, sim_test_X)[,,1]
    preds_ib <- predict(mod, sim_train_X)[,,1]

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

    pred_mean_tr <- t(replicate(nrow(sim_train_Y), colMeans(sim_train_Y)))
    R2 <- 1 - (sum((preds_ib - sim_train_Y)^2)/sum((sim_train_Y - pred_mean_tr)^2))

    pred_mean_ts <- t(replicate(nrow(sim_test_Y), colMeans(sim_test_Y)))
    Q2 <- 1 - (sum((preds - sim_test_Y)^2)/sum((sim_test_Y - pred_mean_ts)^2))

    ncomp <- NA

  } else {
    colnames(sim_train_X) <- c(1:ncol(sim_train_X))
    colnames(sim_test_X) <- c(1:ncol(sim_train_X))
    colnames(sim_test_Y) <- c(1:ncol(sim_test_Y))
    colnames(sim_train_Y) <- c(1:ncol(sim_test_Y))
```

```
    tune <- tune.spls(sim_train_X, sim_train_Y,
                      validation = "Mfold",
                      folds = 10,
                      ncomp = 10,
                      mode = "regression")
    ncomp <- tune$choice.ncomp

    mod <- spls(sim_train_X, sim_train_Y, ncomp = ncomp, mode = "regression")


    if(ncomp == 0){
      preds <- t(replicate(nrow(sim_test_Y), colMeans(sim_train_Y)))
      preds_ib <- t(replicate(nrow(sim_train_Y), colMeans(sim_train_Y)))
      } else {
        preds <- predict(mod, sim_test_X)
        preds <- preds$predict[,,ncomp]

        preds_ib <- predict(mod, sim_train_X)
        preds_ib <- preds_ib$predict[,,ncomp]
       }

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

    pred_mean_tr <- t(replicate(nrow(sim_train_Y), colMeans(sim_train_Y)))
    R2 <- 1 - (sum((preds_ib - sim_train_Y)^2)/sum((sim_train_Y - pred_mean_tr)^2))

    pred_mean_ts <- t(replicate(nrow(sim_test_Y), colMeans(sim_test_Y)))
    Q2 <- 1 - (sum((preds - sim_test_Y)^2)/sum((sim_test_Y - pred_mean_ts)^2))
  }

  out <- c(passed_arg, ncomp, rmse, R2, Q2, R2-Q2)

  return(out)
}
```

```
noise_eval <- function(noise_weight, func = "ddsPLS", n = 300, p = 100, q = 5){
  sim <- sim_data(n = n, p = p, q = q, noise_weight = noise_weight, noise_type = "rnorm", struc = "com

  pls_test(n = n, sim = sim, func = func, passed_arg = noise_weight)

}
```

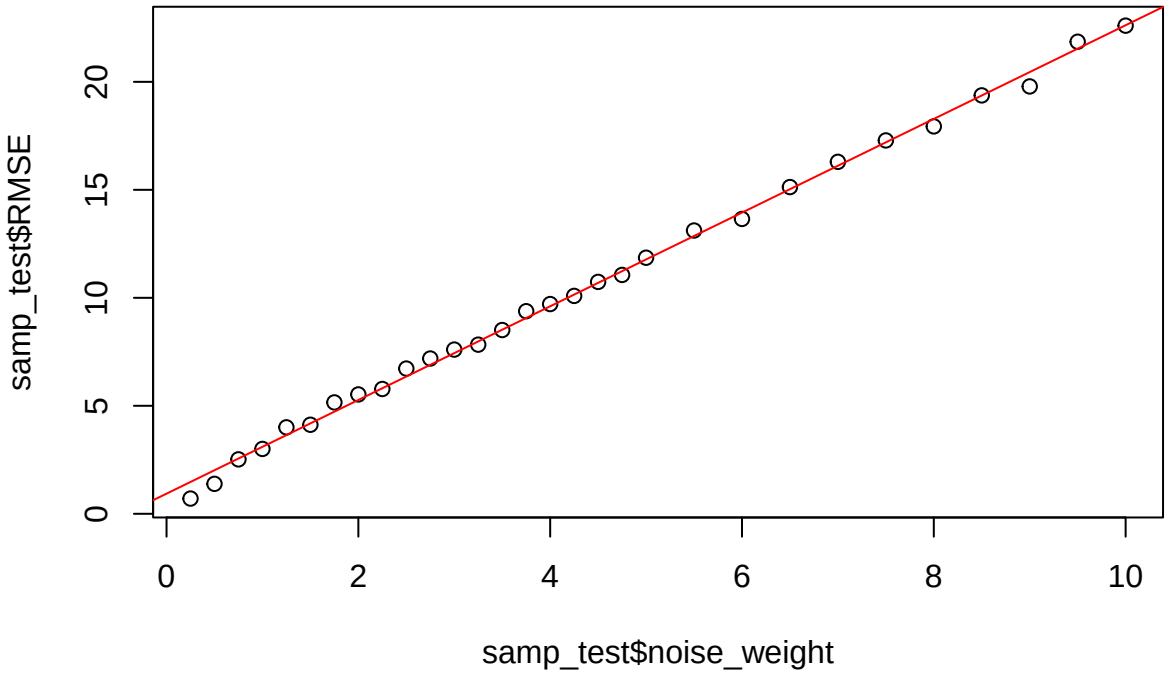
```
samp_test <- apply(matrix(c(seq(from = 0.25, to = 5, by = 0.25),
                            seq(from = 5.5, to = 10, by = 0.5)),
                            nrow = 1),
                   MARGIN = 2,
                   noise_eval)
```

```
samp_test <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_noise_weight_1.csv")
```

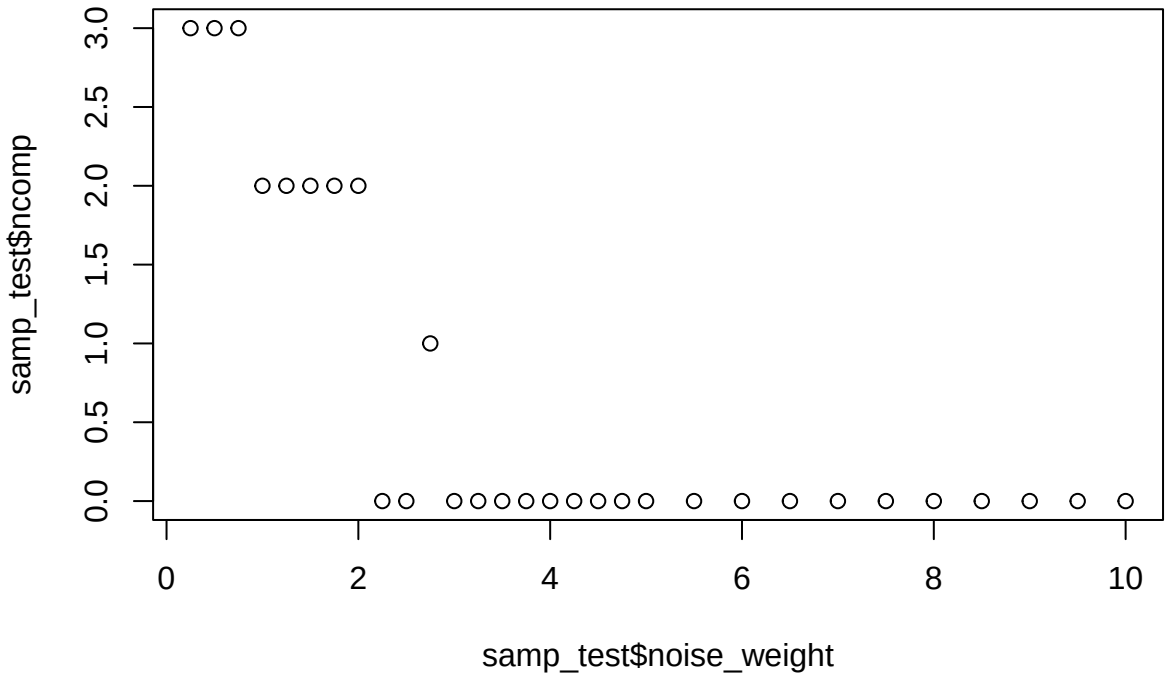
```
reg <- lm(RMSE ~ noise_weight, data = samp_test)

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

New Simulation Noise Test




```
plot(samp_test$noise_weight, samp_test$ncomp)
```



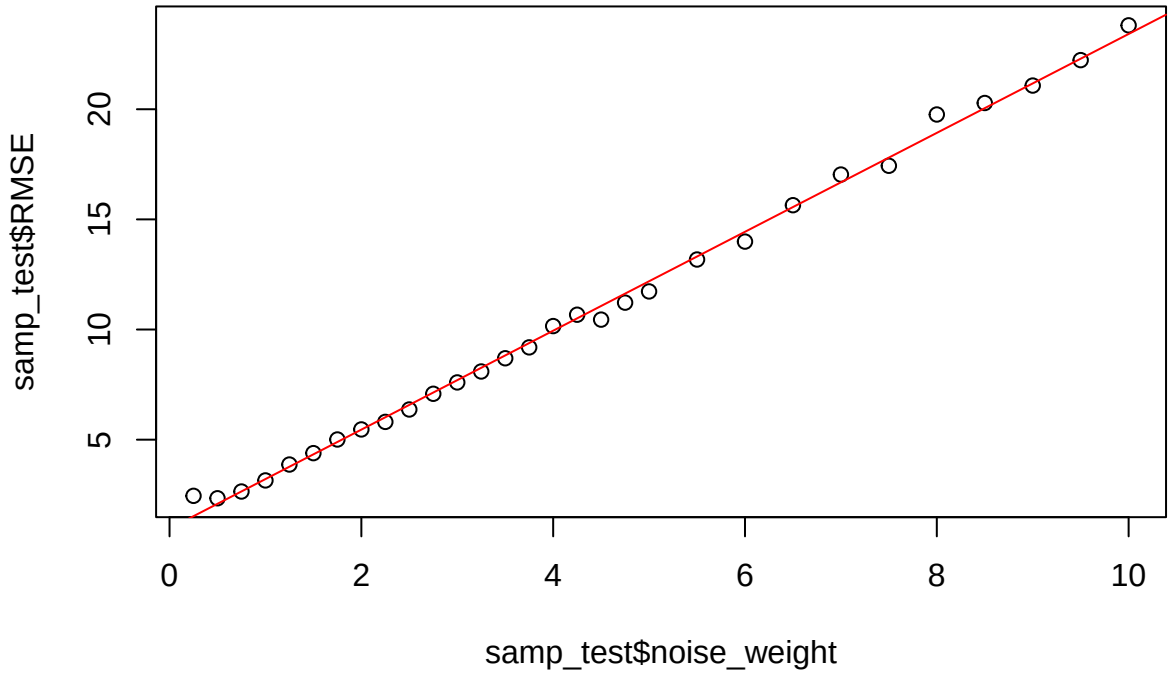

```
samp_test <- apply(matrix(c(seq(from = 0.25, to = 5, by = 0.25),
                            seq(from = 5.5, to = 10, by = 0.5)),
```

```
                          nrow = 1),
                 MARGIN = 2,
                 noise_eval,
                 func = "spls")
```

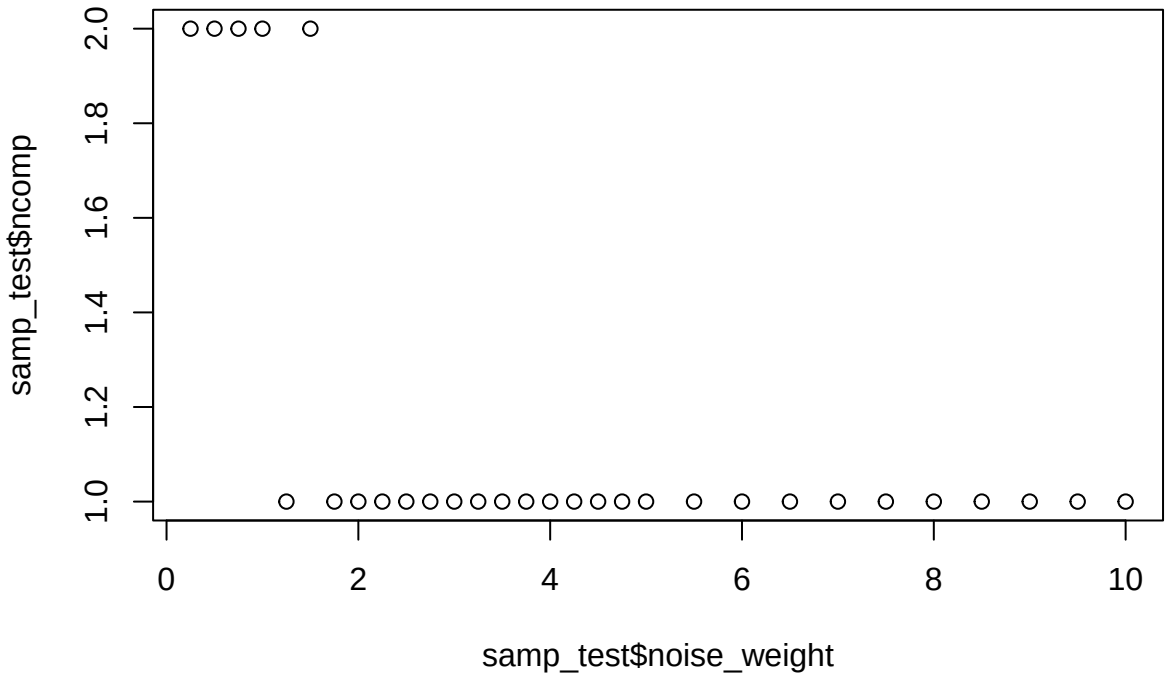
```
samp_test <-read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls_noise_weight_1.csv")[,-1]
colnames(samp_test) <- c("noise_weight", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")

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

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




```
plot(samp_test$noise_weight, samp_test$ncomp)
```

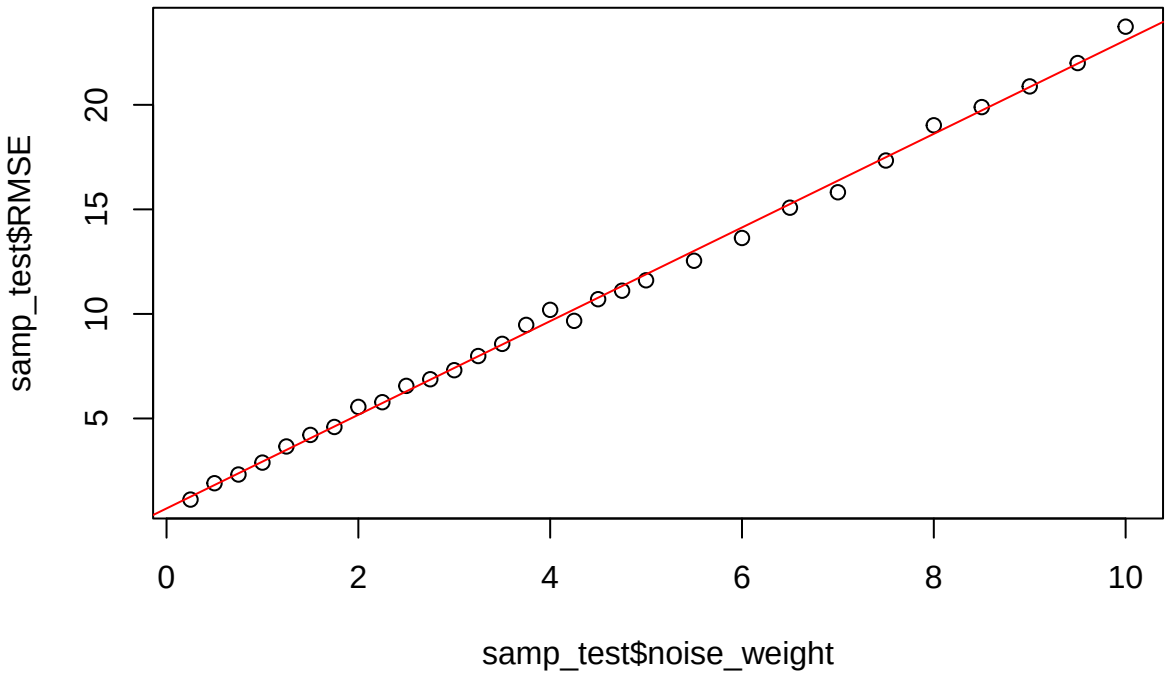



```
samp_test <- apply(matrix(c(seq(from = 0.25, to = 5, by = 0.25),
                            seq(from = 5.5, to = 10, by = 0.5)),
                            nrow = 1),
                   MARGIN = 2,
                   noise_eval,
                   func = "pls")
```

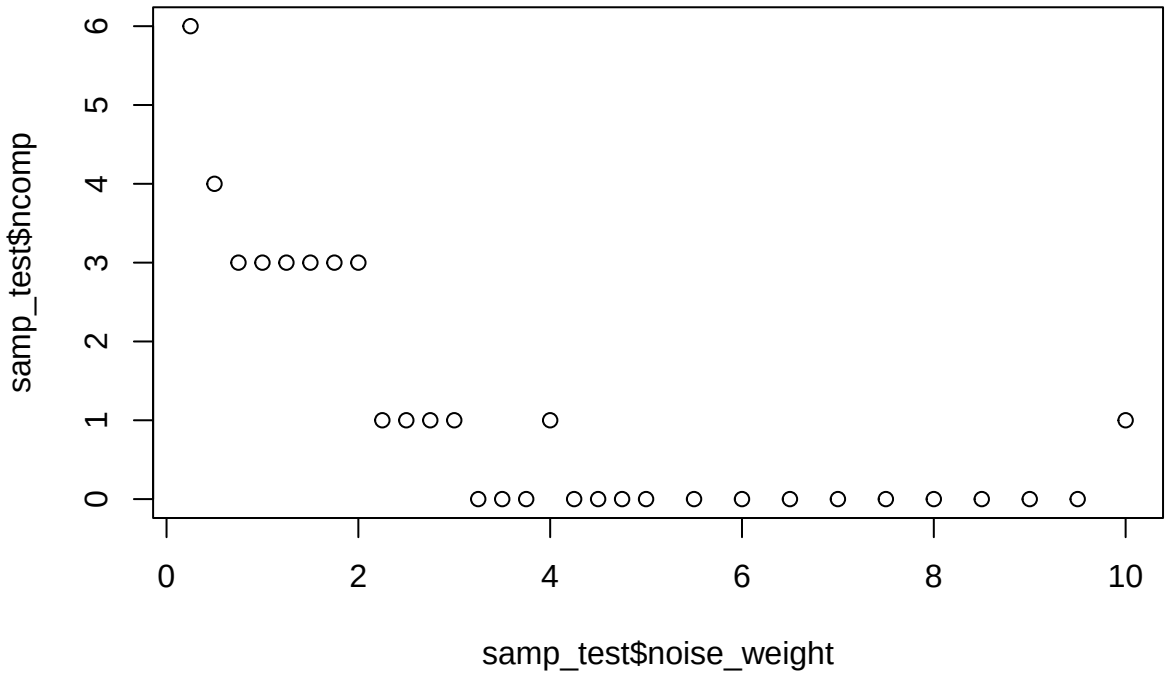
```
samp_test <-read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls_noise_weight_1.csv")

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

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




```
plot(samp_test$noise_weight, samp_test$ncomp)
```




```
noise_samp_spls <- replicate(100, noise_eval(noise_weight = 1, func = "spls"))
```

```
noise_samp_pls <- replicate(100, noise_eval(noise_weight = 1, func = "pls"))
```

```
noise_samp_ddspls <- replicate(100, noise_eval(noise_weight = 1, func = "ddspls"))
```

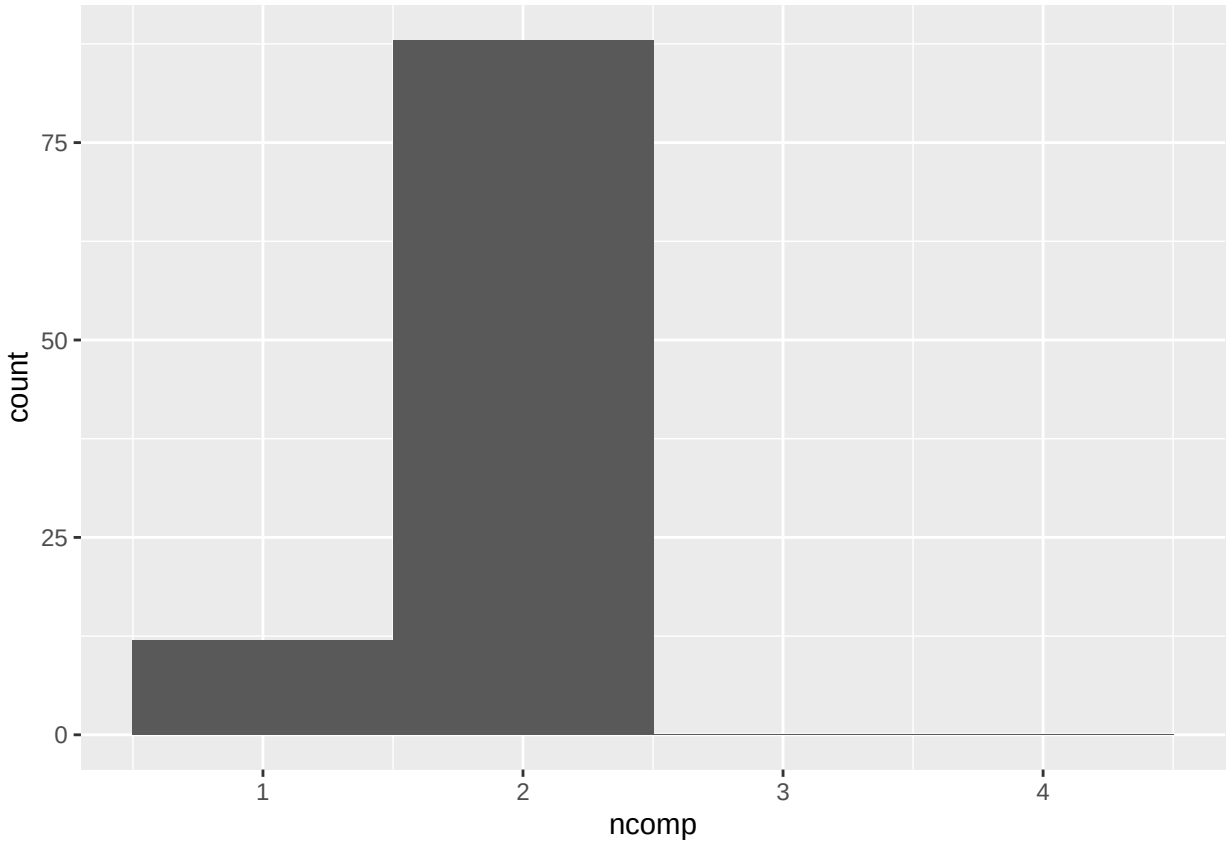
```
noise_samp_spls <- replicate(100, noise_eval(noise_weight = 0.5, func = "spls"))
```

```
noise_samp_pls <- replicate(100, noise_eval(noise_weight = 0.5, func = "pls"))
```

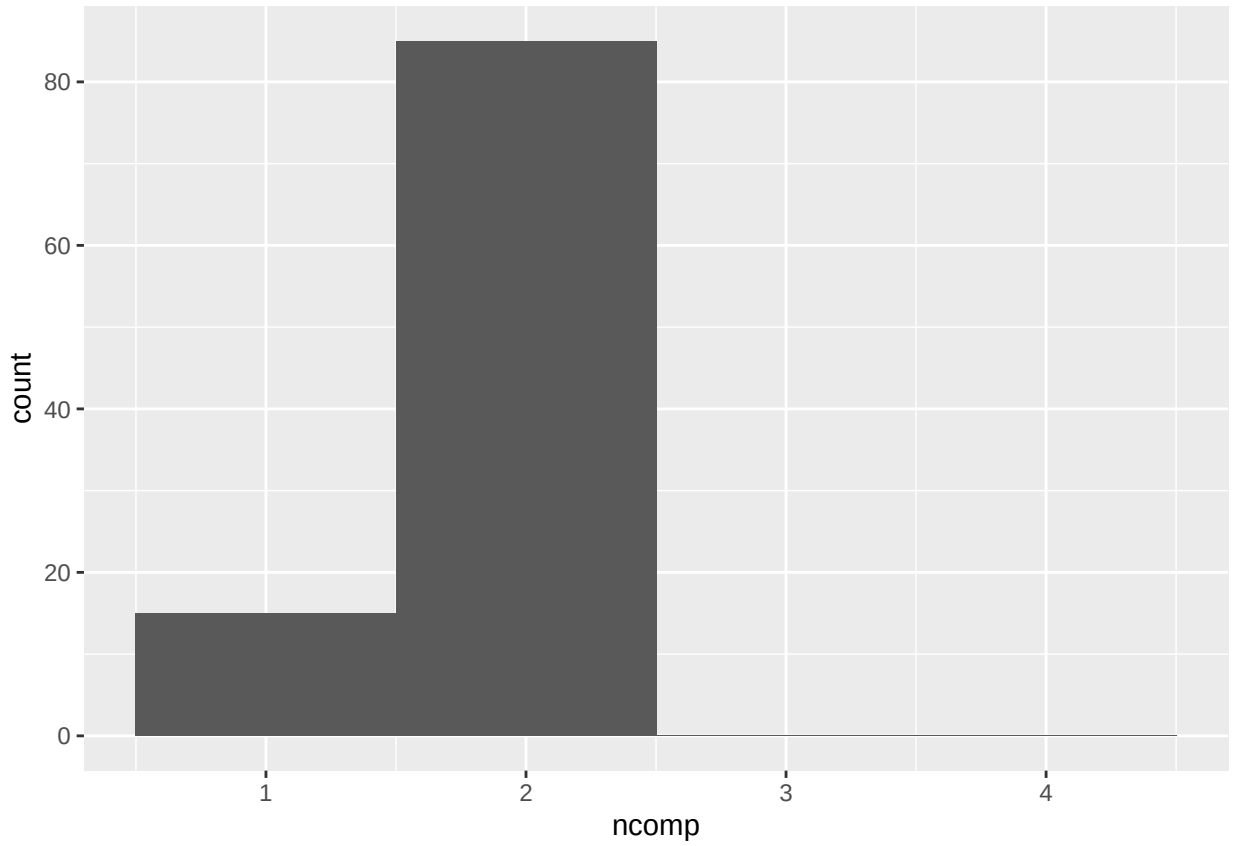
```
noise_samp_ddspls <- replicate(100, noise_eval(noise_weight = 0.5, func = "ddspls"))
```

```
noise_1_ddspls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/noise_1_ddspls.csv")
noise_1_spls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/noise_1_spls.csv")
noise_1_pls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/noise_1_pls.csv")
noise_0.5_ddspls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/noise_0.5_ddspls.csv")
noise_0.5_spls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/noise_0.5_spls.csv")
noise_0.5_pls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/noise_0.5_pls.csv")
```

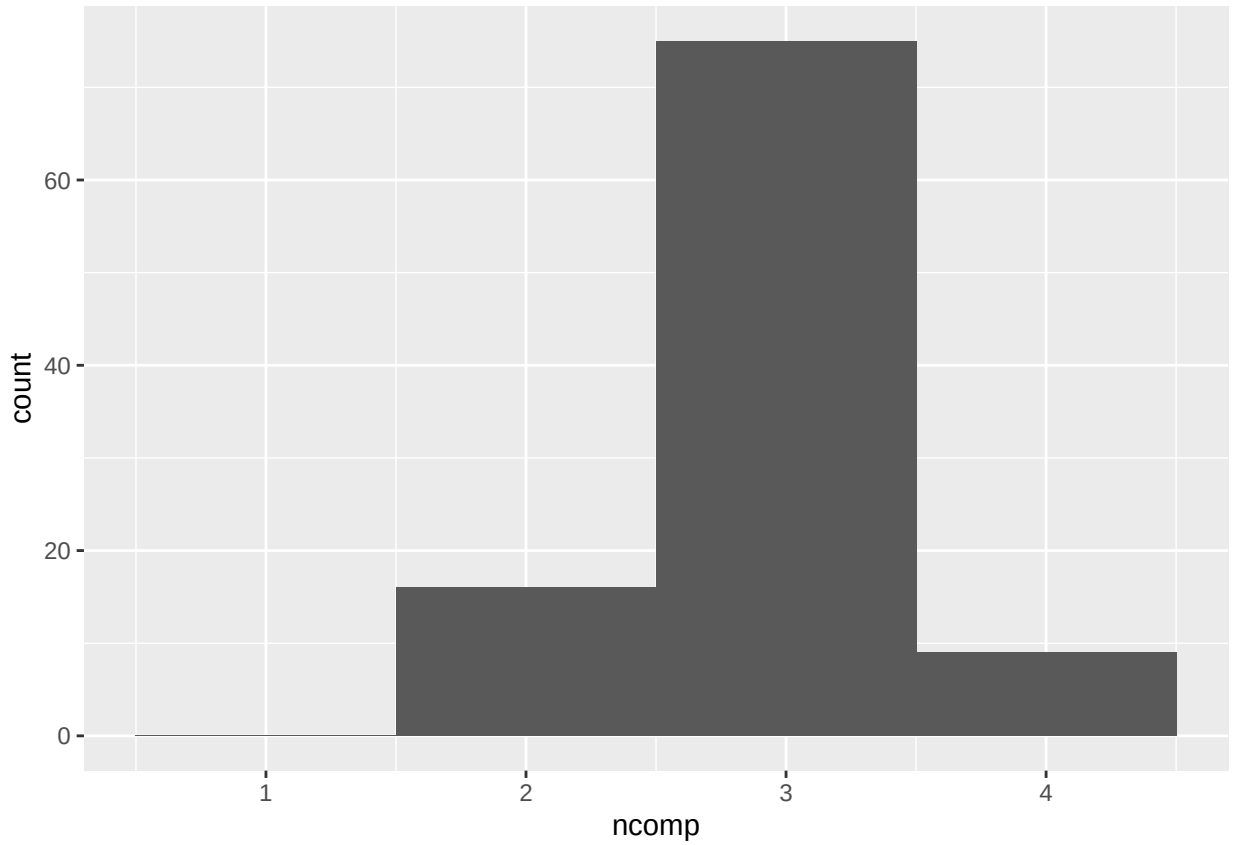
```
ggplot(noise_1_ddspls, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  xlim(0.5, 4.5)
```



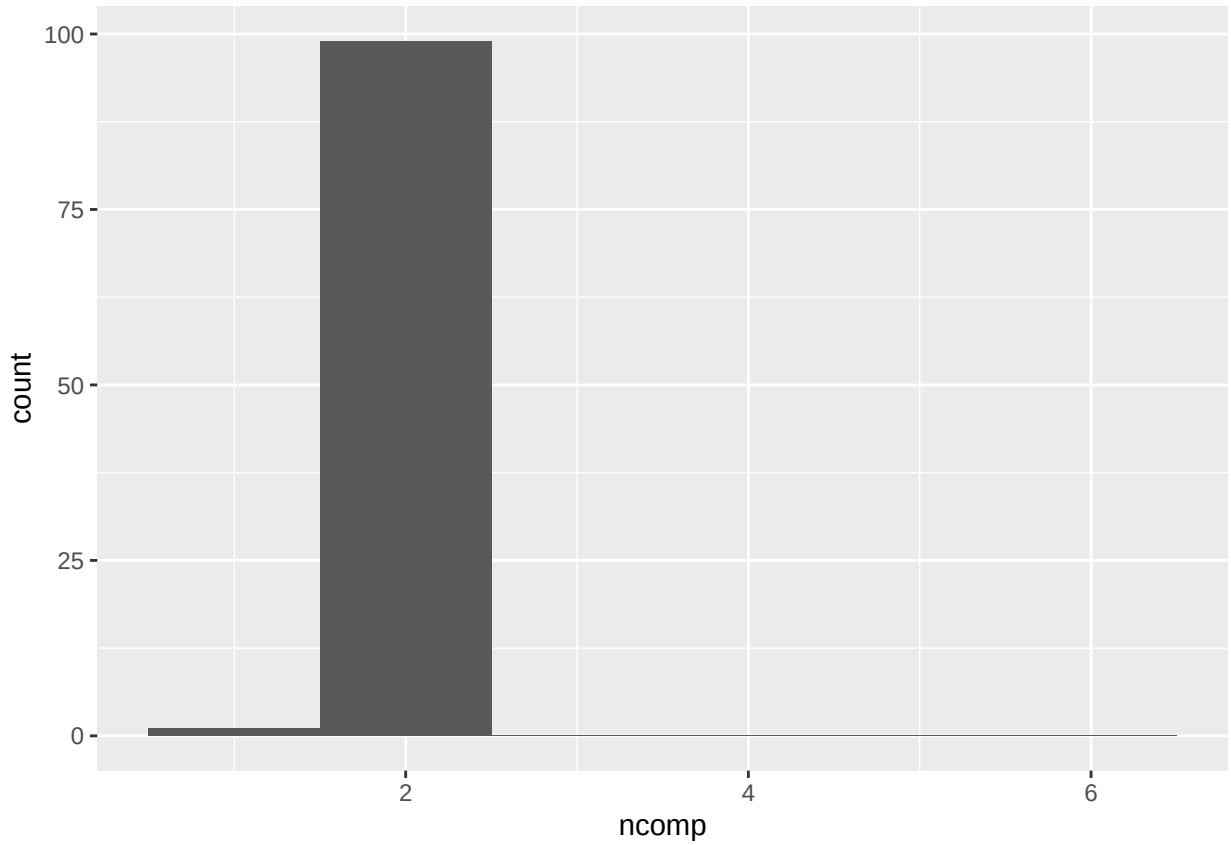

```
ggplot(noise_1_spls, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  xlim(0.5, 4.5)
```



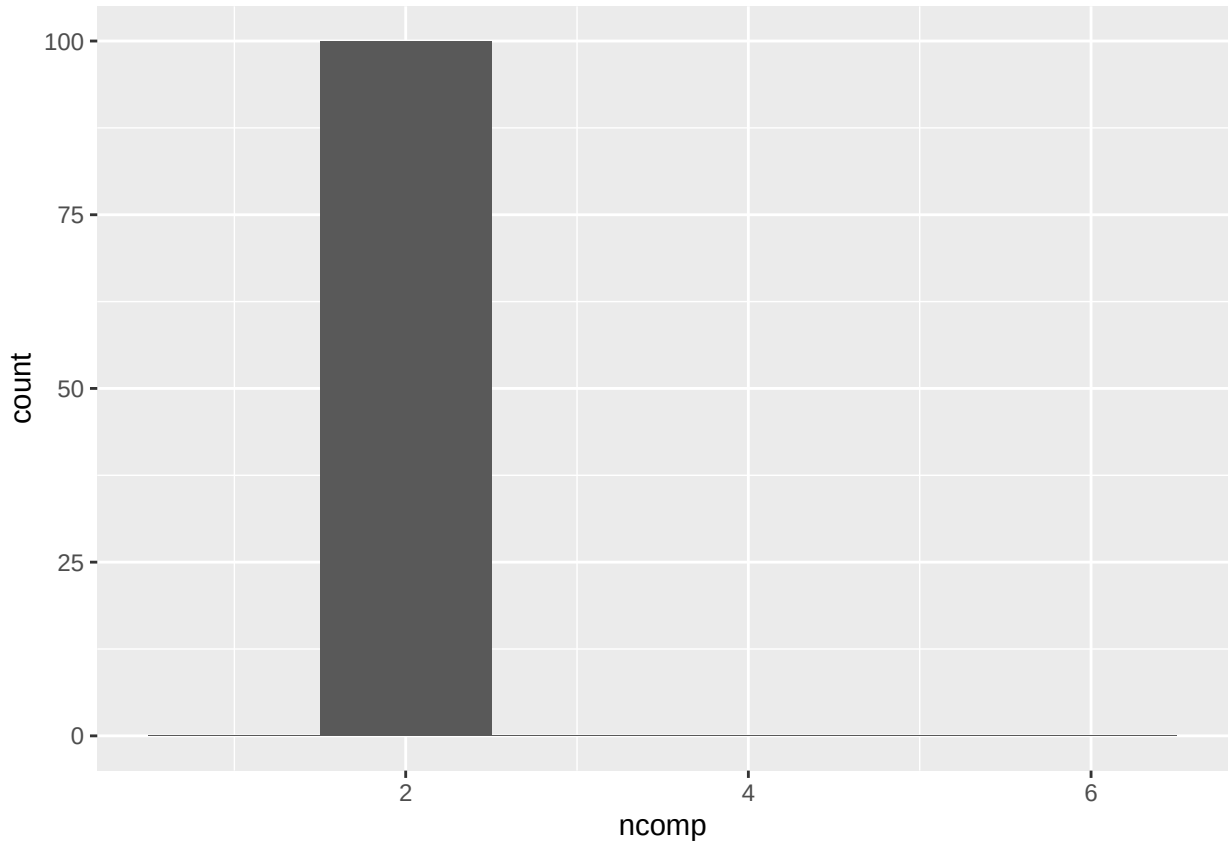

```
ggplot(noise_1_pls, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  xlim(0.5, 4.5)
```



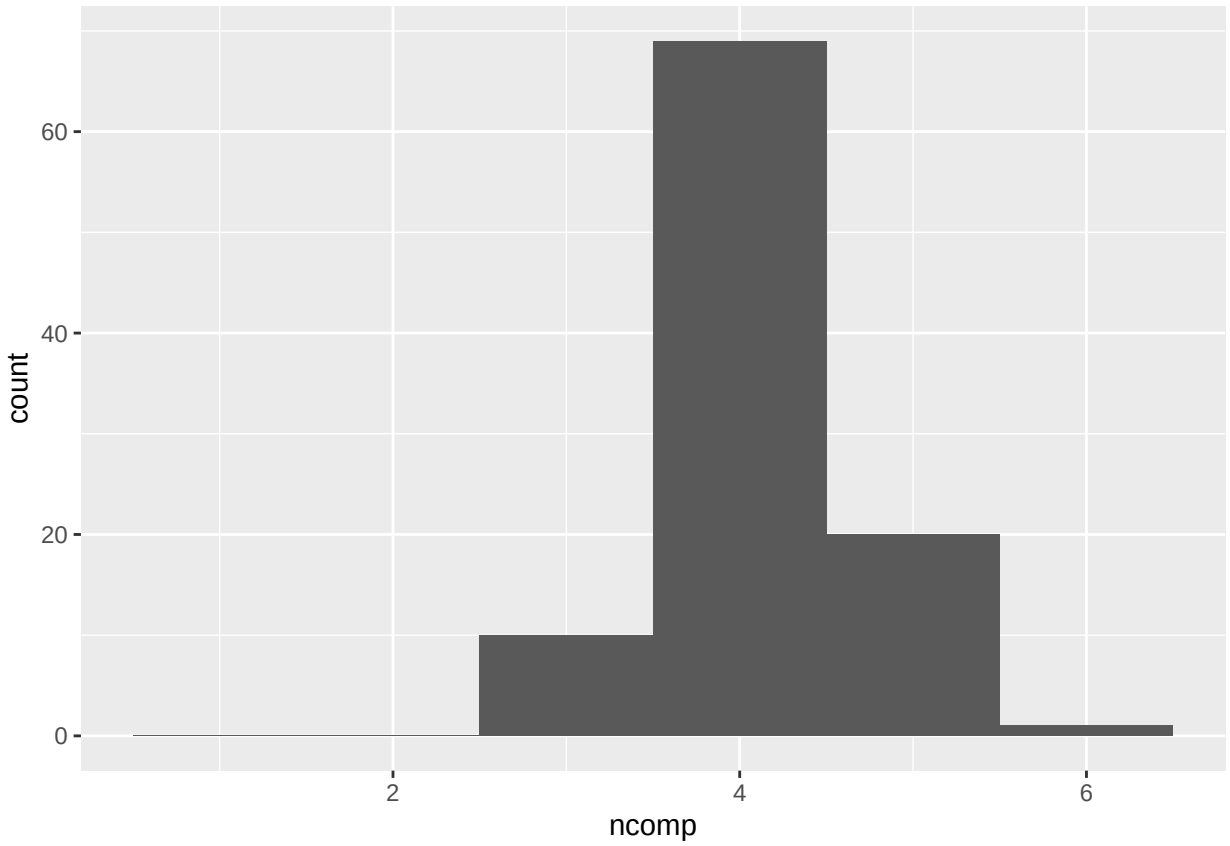

```
ggplot(noise_0.5_ddspls, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  xlim(0.5, 6.5)
```




```
ggplot(noise_0.5_spls, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  xlim(0.5, 6.5)
```




```
ggplot(noise_0.5_pls, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  xlim(0.5, 6.5)
```




```
p_eval <- function(p, noise_weight = 1, n = 150, q = 5,func = "ddsPLS", struc = "complex"){

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

   # Passes Data to test function
   pls_test(n = n, sim = sim, func = func, passed_arg = p)
}
```

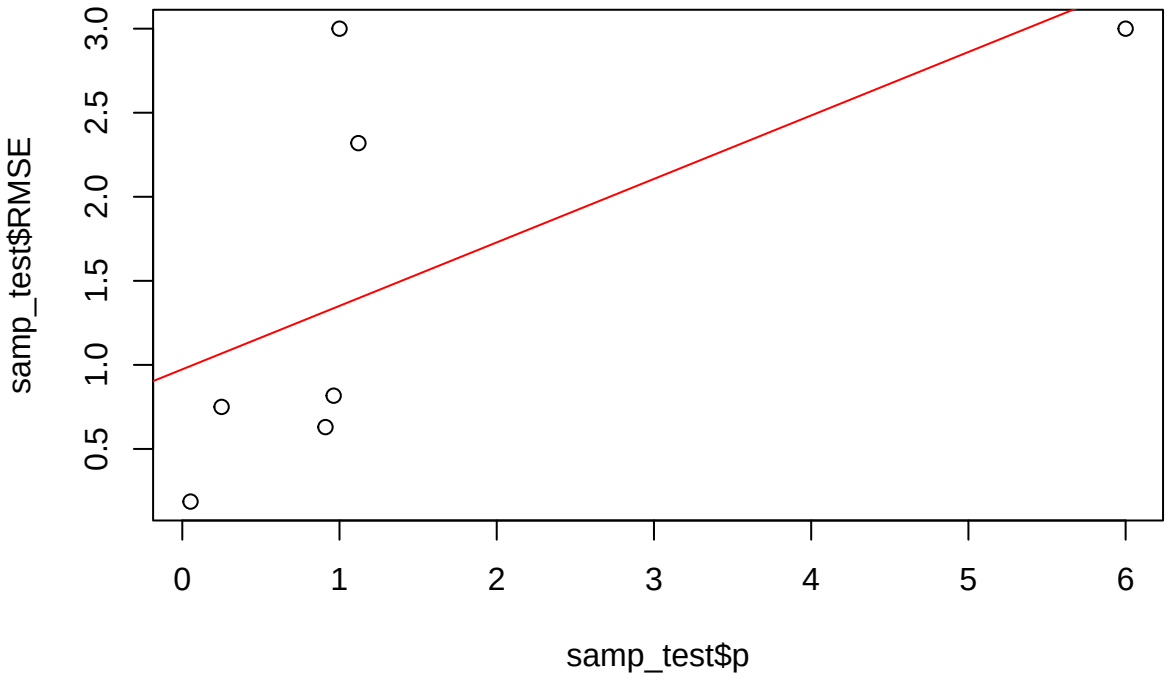
```
samp_test <- apply(matrix(seq(from = 50, to = 1000, by = 50),
                          nrow = 1),
                   MARGIN = 2,
                   p_eval)
```

```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("p", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")

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

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

New Simulation Predictors Test

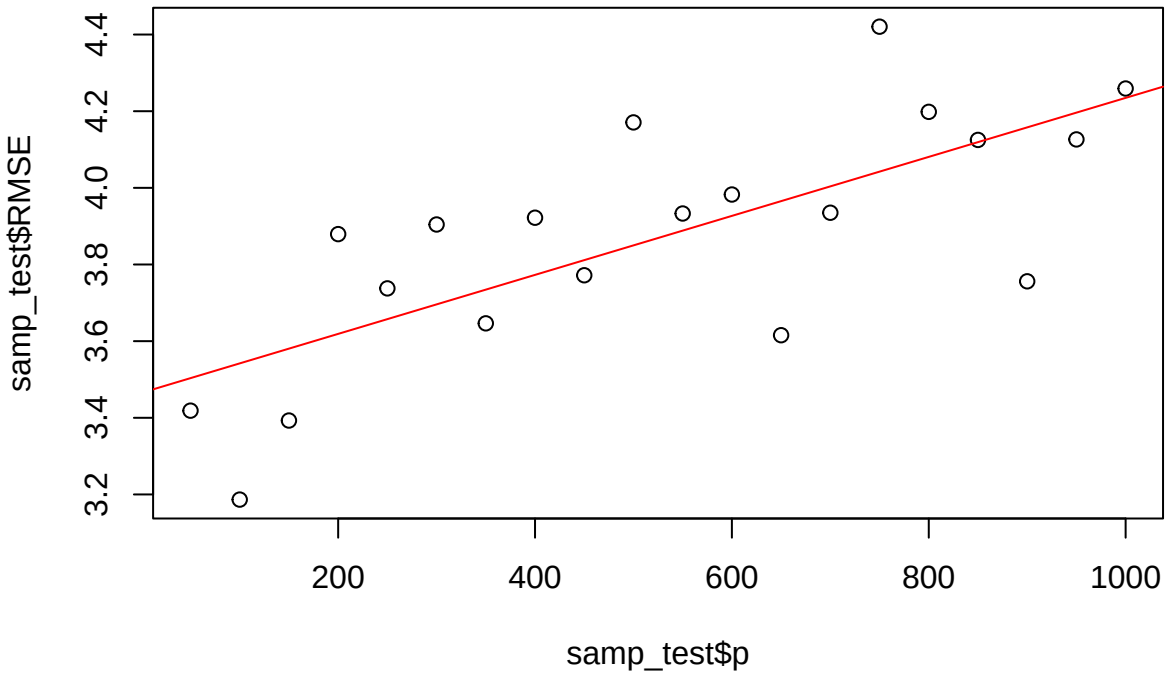



```
samp_test <- apply(matrix(seq(from = 50, to = 1000, by = 50),
                          nrow = 1),
                   MARGIN = 2,
                   p_eval,
                   func = "spls")
```

```
samp_test <-as.data.frame(t(samp_test))
colnames(samp_test) <- c("p", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")

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

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



It looks like ddsPLS performs much better than sPLS when a large number of meaningless predictors are added. I need to figure out how to select the ideal number of components for sPLS models to use and how to calculate the predicted values. Run more simulations to see how model performs. Compare to LASSO. Comparing the number of components. Add LASSO to the theory section.

```
ddspls.p.reps <- replicate(100, p_eval(p = 1000))
```

```
spls.p.reps <- replicate(100, p_eval(p = 1000, func = "spls"))
```

```
pls.p.reps <- replicate(100, p_eval(p = 1000, func = "pls"))
```

```
ddspls.p.reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls.1000.reps.csv")
spls.p.reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls.1000.reps.csv")
pls.p.reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls.1000.reps.csv")

ddspls.p.reps <- as.data.frame(t(ddspls.p.reps)[-1,])
spls.p.reps <- as.data.frame(t(spls.p.reps)[-1,])
pls.p.reps <- as.data.frame(t(pls.p.reps)[-1,])

colnames(ddspls.p.reps) <- c("p", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")
colnames(spls.p.reps) <- c("p", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")
colnames(pls.p.reps) <- c("p", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")
```

```
t.test(ddspls.p.reps$RMSE, spls.p.reps$RMSE)
```

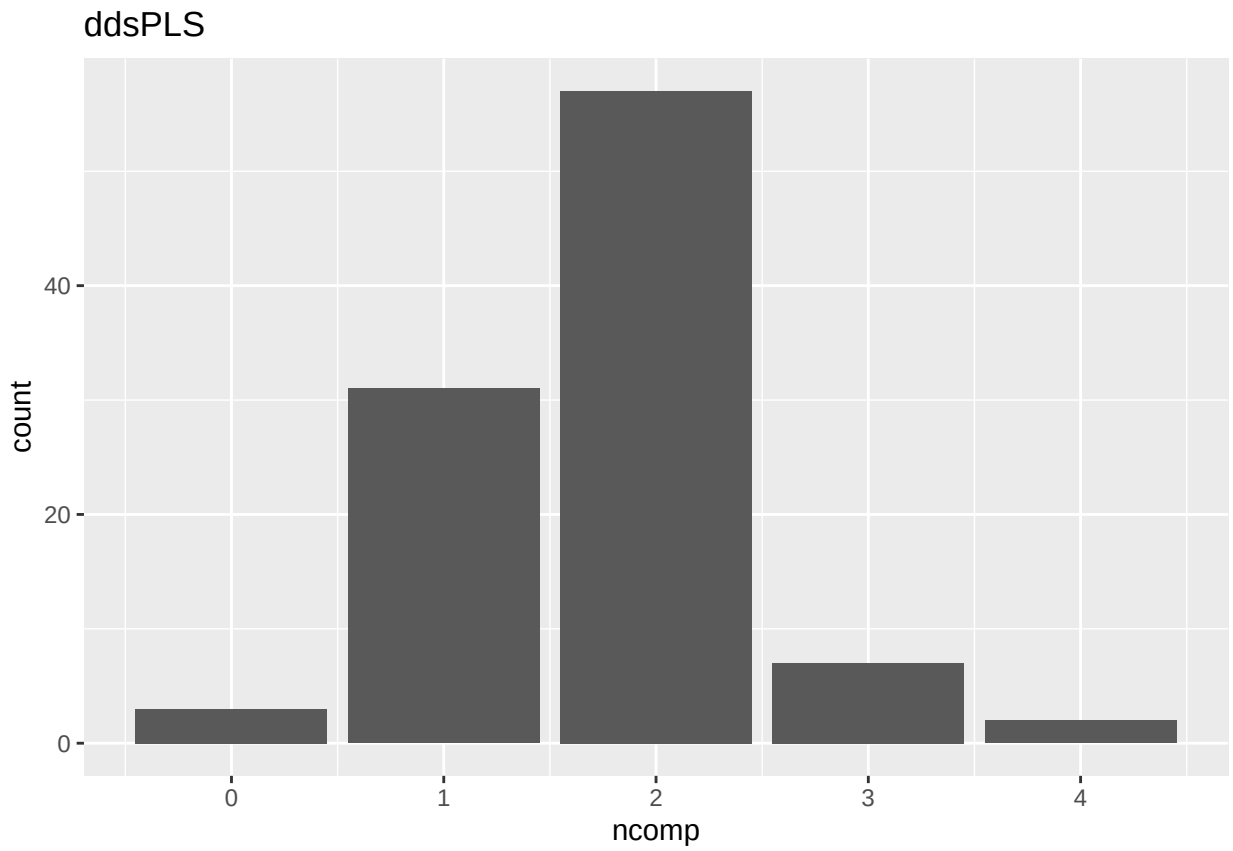
```
##
##  Welch Two Sample t-test
##
## data:  ddspls.p.reps$RMSE and spls.p.reps$RMSE
## t = -11.129, df = 152.53, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.5306905 -0.3706795
## sample estimates:
```

```
## mean of x mean of y
##  3.672778  4.123463
```

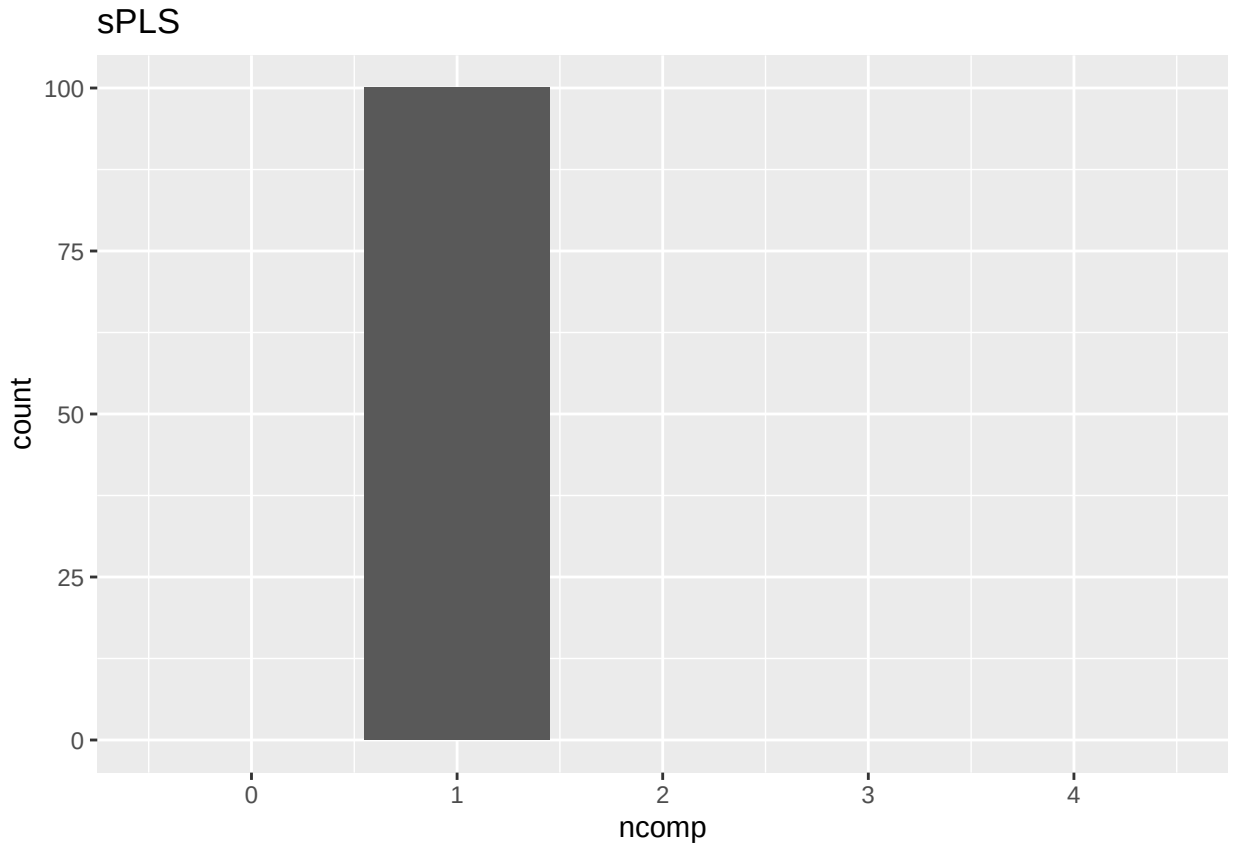
```
t.test(ddspls.p.reps$RMSE, pls.p.reps$RMSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  ddspls.p.reps$RMSE and pls.p.reps$RMSE
## t = -8.1171, df = 133.97, p-value = 2.73e-13
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3908404 -0.2376911
## sample estimates:
## mean of x mean of y
##  3.672778  3.987043
```

```
ggplot(data = ddspls.p.reps, aes(x = ncomp)) +
  geom_bar() +
  labs(title = "ddsPLS")
```

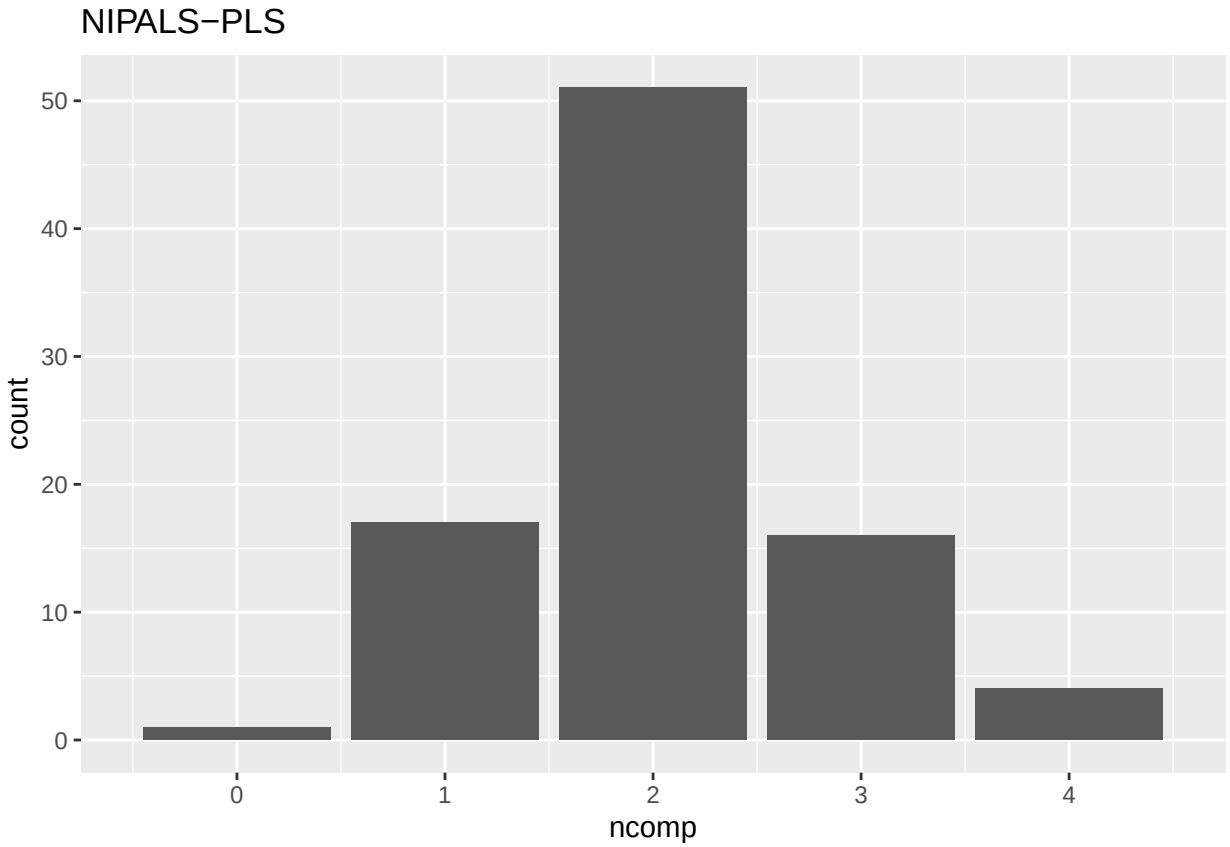



```
ggplot(data = spls.p.reps, aes(x = ncomp)) +
  geom_bar() +
  scale_x_continuous(breaks = 0:4, limits = c(-0.5,4.5))  +
  labs(title = "sPLS")
```




```
ggplot(data = pls.p.reps, aes(x = ncomp)) +
  geom_bar() +
  scale_x_continuous(breaks = 0:4, limits = c(-0.5,4.5))  +
  labs(title = "NIPALS-PLS")
```

```
## Warning: Removed 11 rows containing non-finite values (stat_count).
```



ddsPLS performs better with a large number of predictors uncorrelated with the response. Here the mean RMSE of is significantly lower for ddsPLS compared to sPLS. We can also see that sPLS models fail to find the structure of the data while ddsPLS is able to. Note that the sPLS tuning algorithm must select at least 1 component, this does not mean that the model performs better than mean estimation.

```
ddspls.p.reps <- replicate(100, p_eval(p = 500, n=3*250))
```

```
spls.p.reps <- replicate(100, p_eval(p = 500, n=3*250, func = "spls"))
```

```
pls.p.reps <- replicate(100, p_eval(p = 500, n=3*250, func = "pls"))
```

```
ddspls.p.reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls.500.reps.csv")
spls.p.reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls.500.reps.csv")
pls.p.reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls.500.reps.csv")

ddspls.p.reps <- as.data.frame(t(ddspls.p.reps)[-1,])
spls.p.reps <- as.data.frame(t(spls.p.reps)[-1,])
pls.p.reps <- as.data.frame(t(pls.p.reps)[-1,])

colnames(ddspls.p.reps) <- c("p", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")
colnames(spls.p.reps) <- c("p", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")
colnames(pls.p.reps) <- c("p", "ncomp", "RMSE", "R2", "Q2", "R2-Q2")
```

```
t.test(ddspls.p.reps$RMSE, spls.p.reps$RMSE)
```

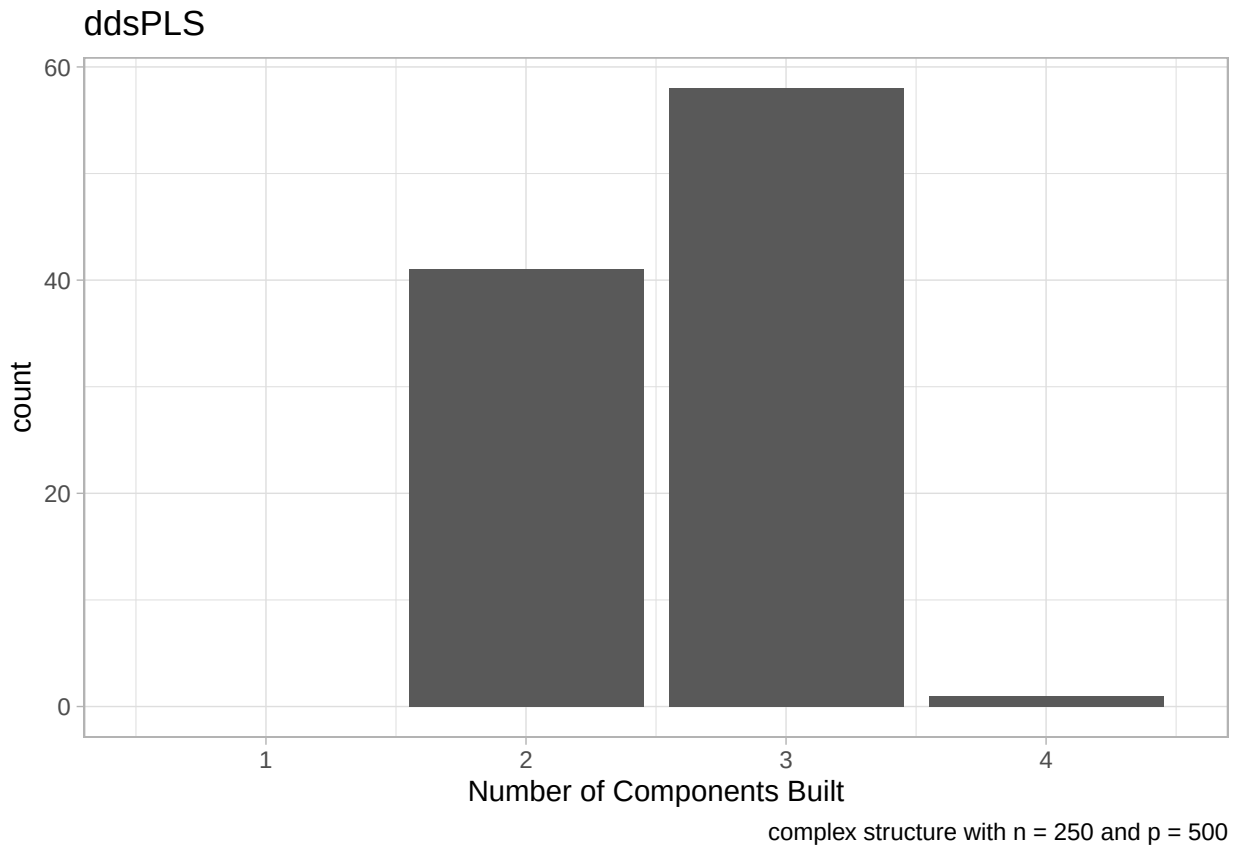
```
##
##  Welch Two Sample t-test
##
## data:  ddspls.p.reps$RMSE and spls.p.reps$RMSE
```

```
## t = -54.715, df = 196.32, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.6130306 -0.5703767
## sample estimates:
## mean of x mean of y
##  2.667779  3.259482
```

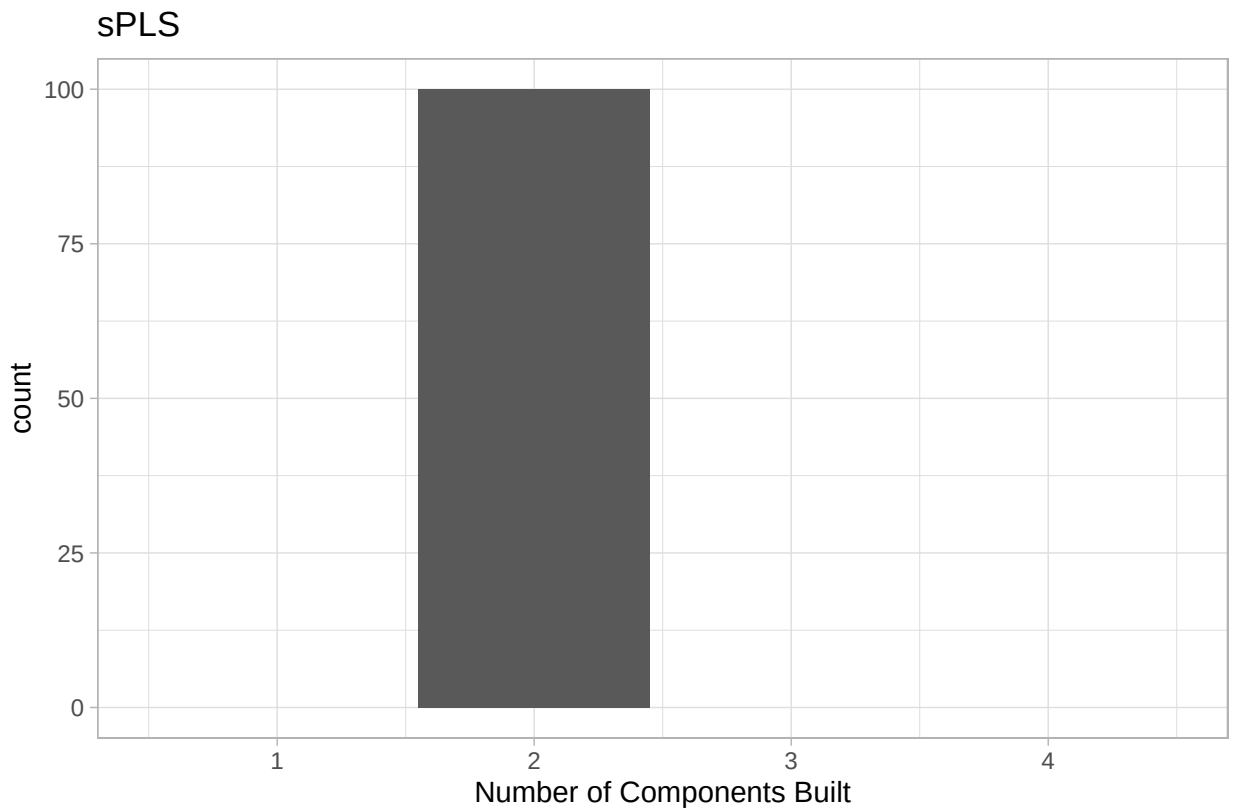
```
t.test(ddspls.p.reps$RMSE, pls.p.reps$RMSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  ddspls.p.reps$RMSE and pls.p.reps$RMSE
## t = -48.886, df = 192.58, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.5319989 -0.4907353
## sample estimates:
## mean of x mean of y
##  2.667779  3.179146
```

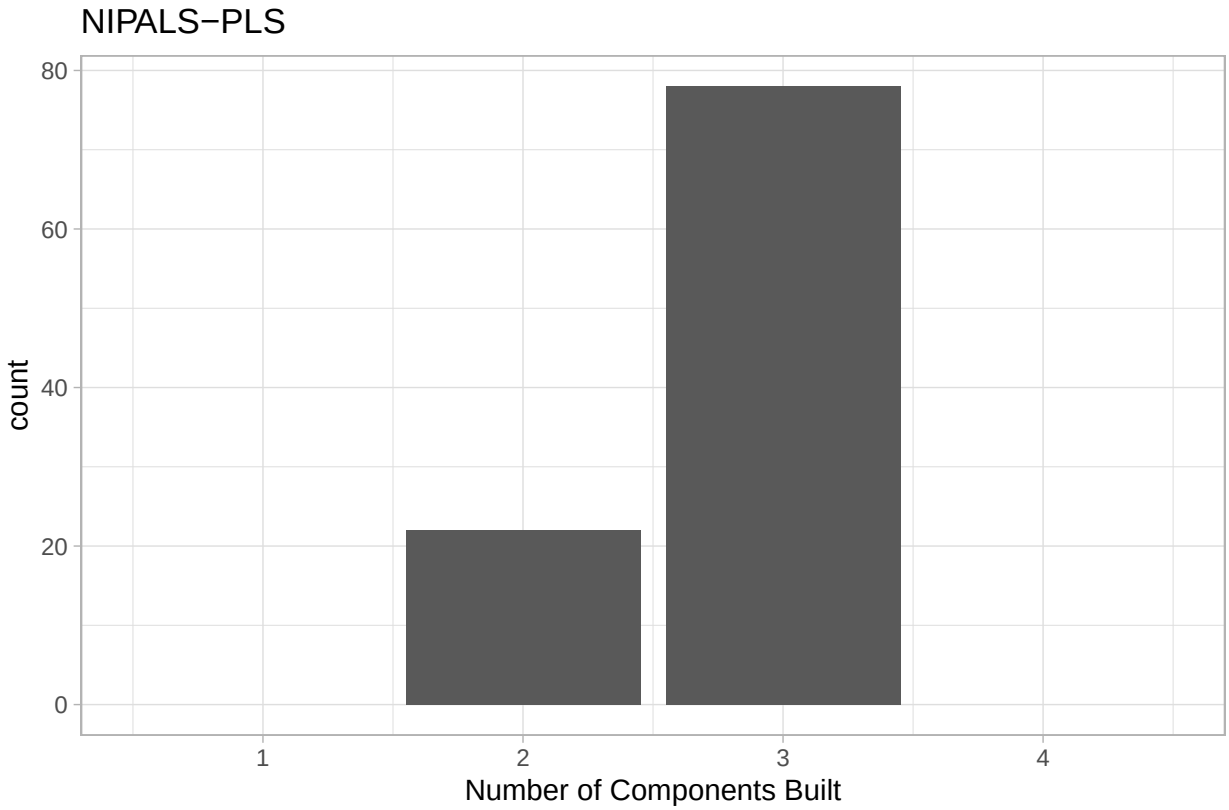
```
ggplot(data = ddspls.p.reps, aes(x = ncomp)) +
  geom_bar() +
  xlab("Number of Components Built") +
  labs(title = "ddsPLS", caption = "complex structure with n = 250 and p = 500") +
  scale_x_continuous(breaks = 1:4, limits = c(0.5,4.5)) +
  theme_light()
```



```
ggplot(data = spls.p.reps, aes(x = ncomp)) +
  geom_bar() +
  xlab("Number of Components Built") +
  scale_x_continuous(breaks = 1:4, limits = c(0.5,4.5))  +
  labs(title = "sPLS", caption = "complex structure with n = 250 and p = 500") +
  theme_light()
```




```
ggplot(data = pls.p.reps, aes(x = ncomp)) +
  geom_bar() +
  xlab("Number of Components Built") +
  scale_x_continuous(breaks = 1:4, limits = c(0.5,4.5))  +
  labs(title = "NIPALS-PLS", caption = "complex structure with n = 250 and p = 500") +
  theme_light()
```



New Simulation Response Test

```
q_eval <- function(q, noise_weight = 0.5, n = 150, p = 100,func = "ddsPLS", struc = "simple", D_method =

  ## Note that if D_method != "simple" there are lower bounds on the value of q

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

   # Passes Data to test function
   pls_test(n = n, sim = sim, func = func, passed_arg = q)
}
```

Range of Q Simulation

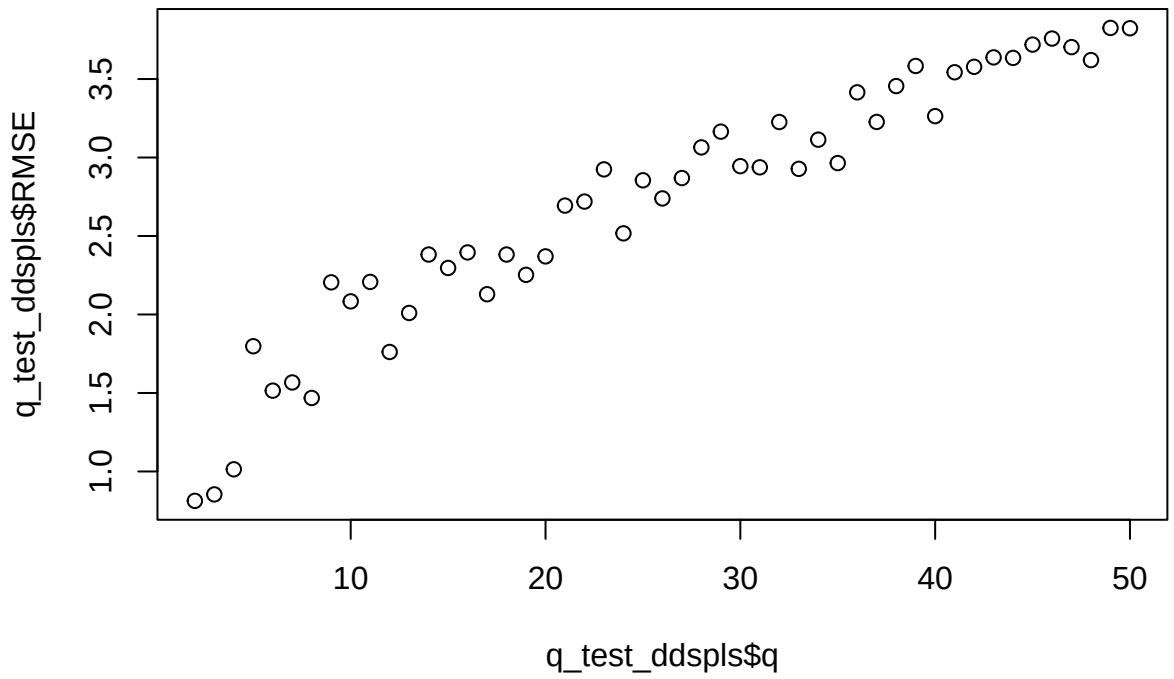
```
q_test_ddspls <- apply(matrix(c(2:50), nrow = 1),
                       MARGIN = 2,
                       q_eval)
```

```
q_test_spls <- apply(matrix(c(2:50),
                            nrow = 1),
                     MARGIN = 2,
                     q_eval,
                     func = "spls")
```

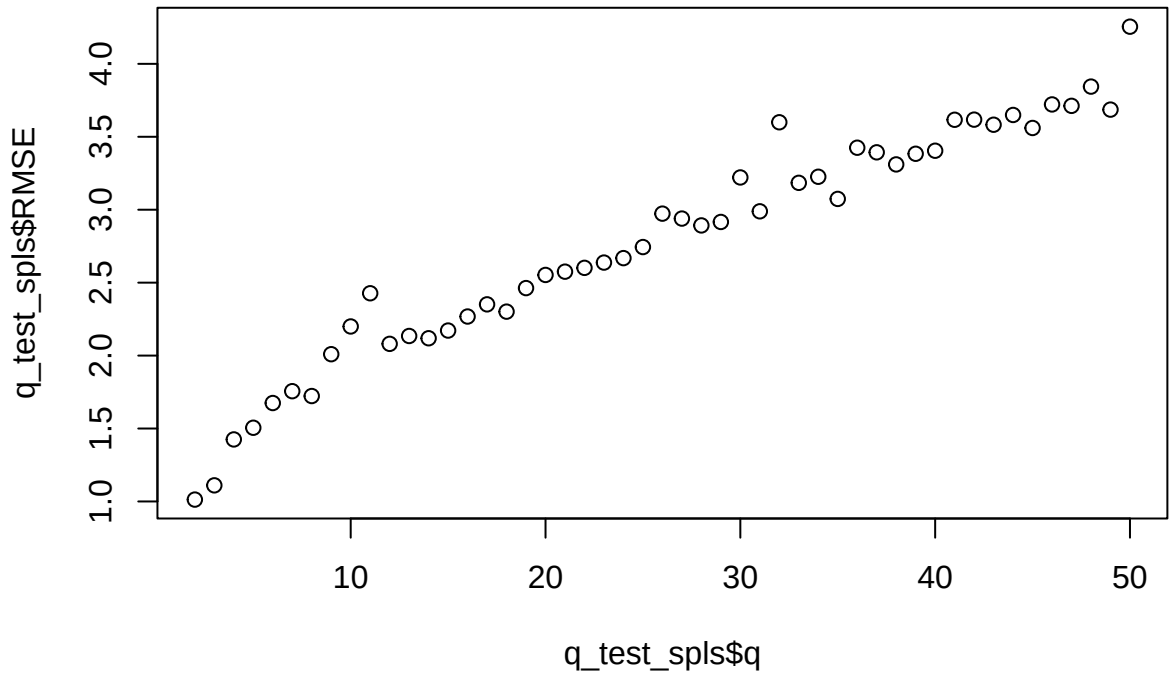
```
q_test_pls <- apply(matrix(c(2:50),
                           nrow = 1),
                    MARGIN = 2,
                    q_eval,
                    func = "pls")
```

```
q_test_ddspls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/q_test_ddspls.csv")
q_test_spls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/q_test_spls.csv")
q_test_pls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/q_test_pls.csv")
```

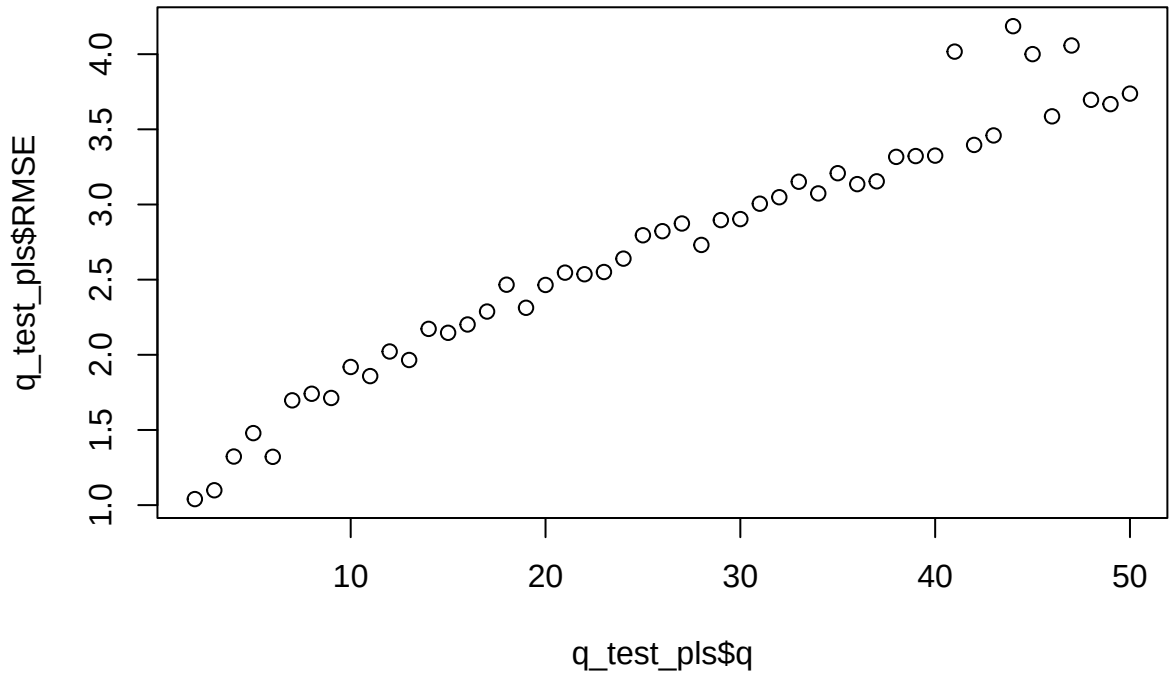
```
plot(q_test_ddspls$q, q_test_ddspls$RMSE)
```



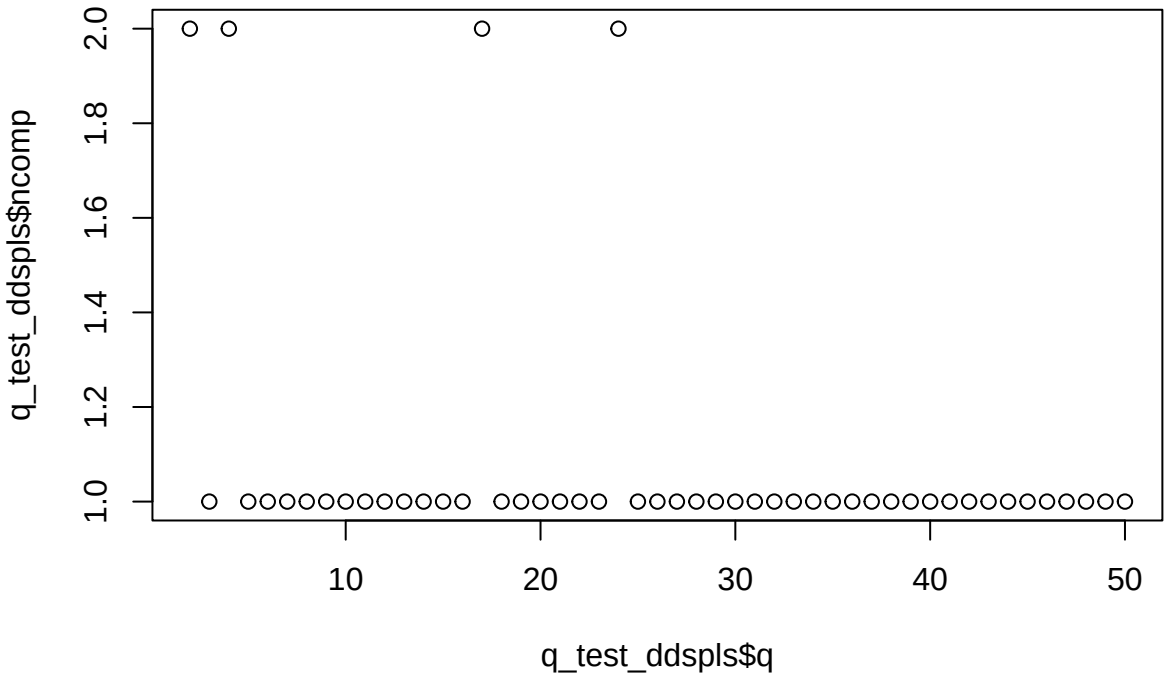

```
plot(q_test_spls$q, q_test_spls$RMSE)
```



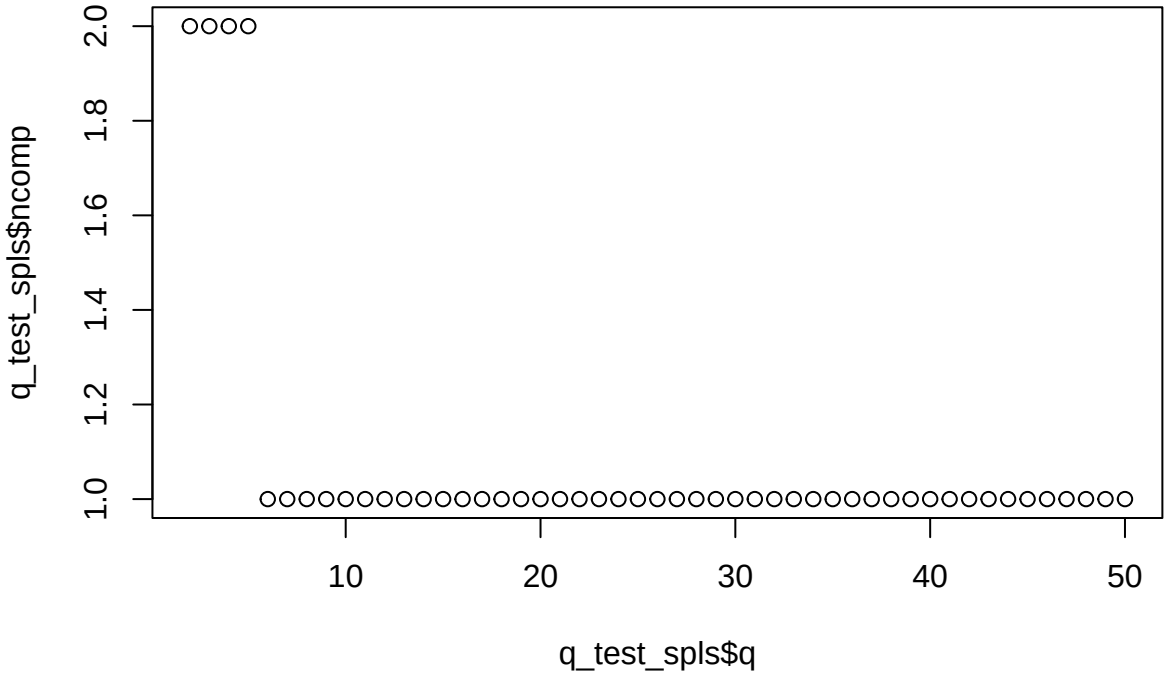

```
plot(q_test_pls$q, q_test_pls$RMSE)
```



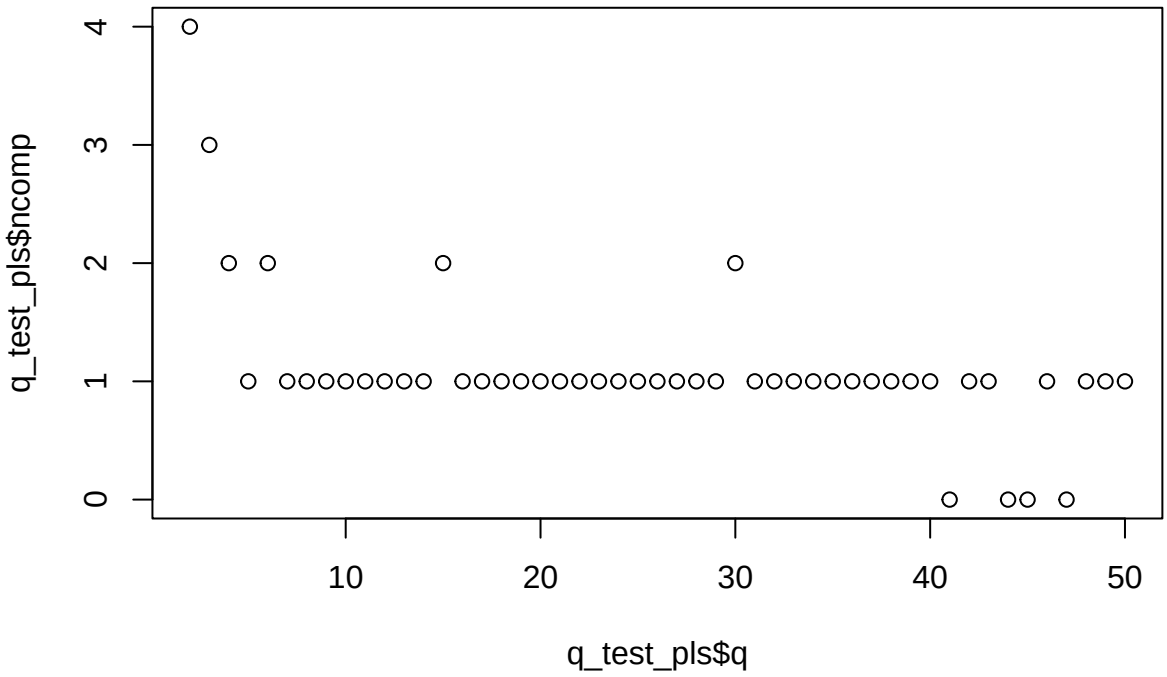

```
plot(q_test_ddspls$q, q_test_ddspls$ncomp)
```




```
plot(q_test_spls$q, q_test_spls$ncomp)
```




```
plot(q_test_pls$q, q_test_pls$ncomp)
```



I think there should just be 1 component built here,

q = 25 Replication

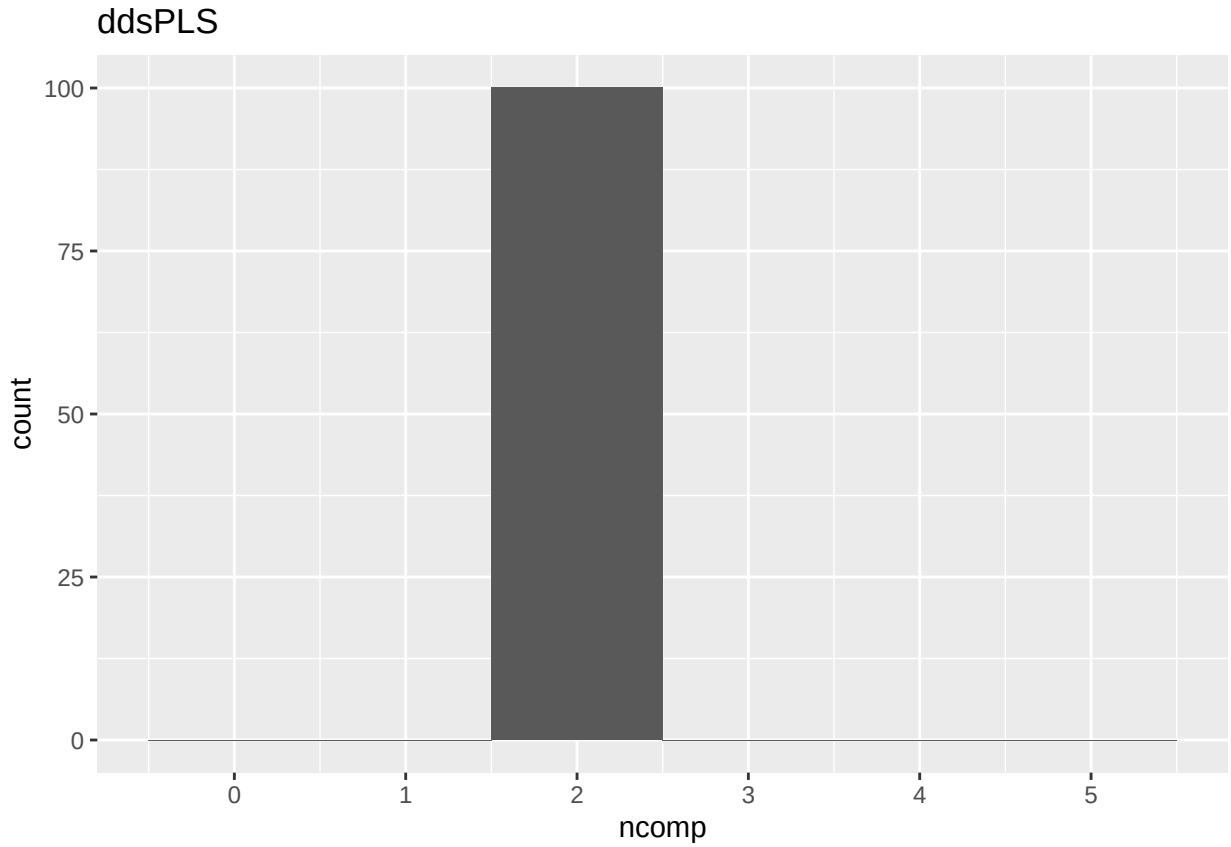
```
ddspls.q.reps <- replicate(100, q_eval(q = 25))
```

```
spls.q.reps <- replicate(100, q_eval(q = 25, func = "spls"))
```

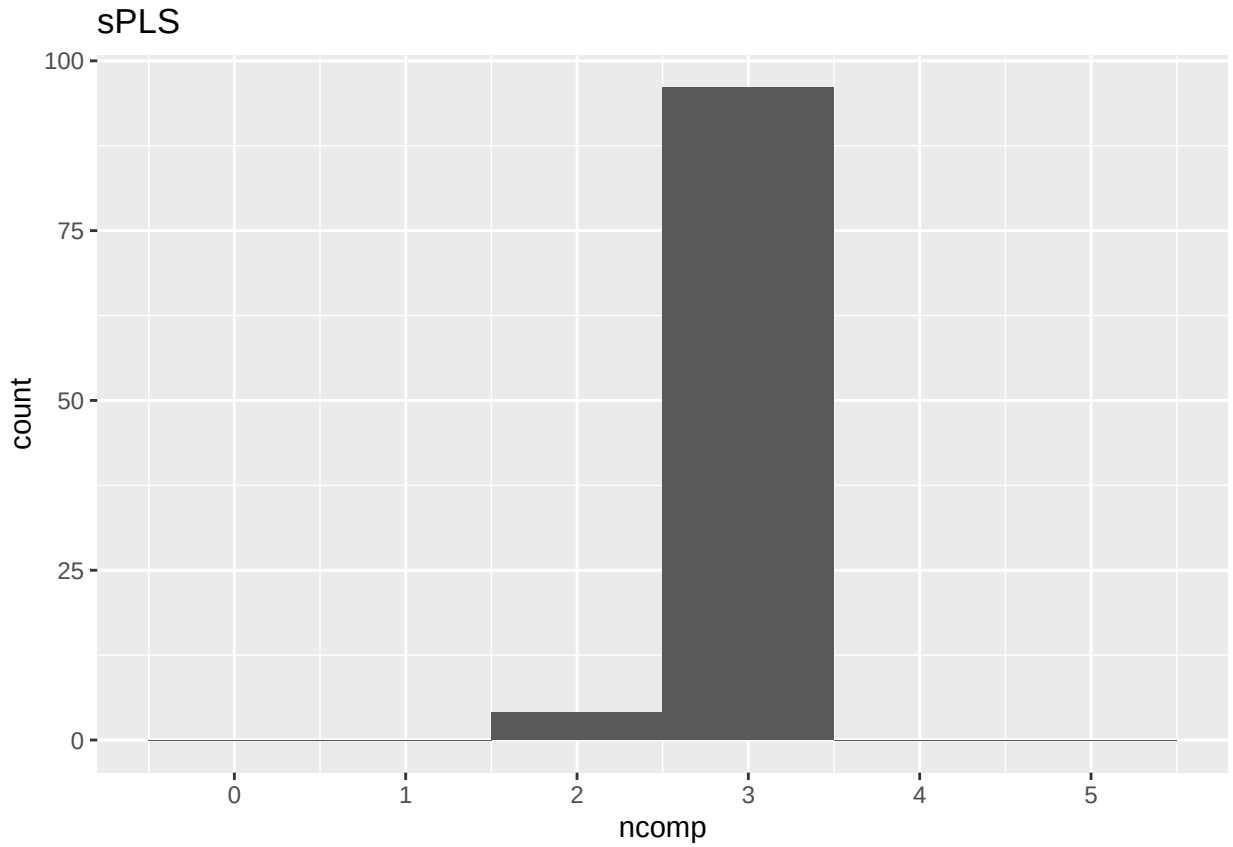
```
pls.q.reps <- replicate(100, q_eval(q = 25, func = "pls"))
```

```
ddspls_q_25 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_q_25.csv")
spls_q_25 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls_q_25.csv")
pls_q_25 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls_q_25.csv")
```

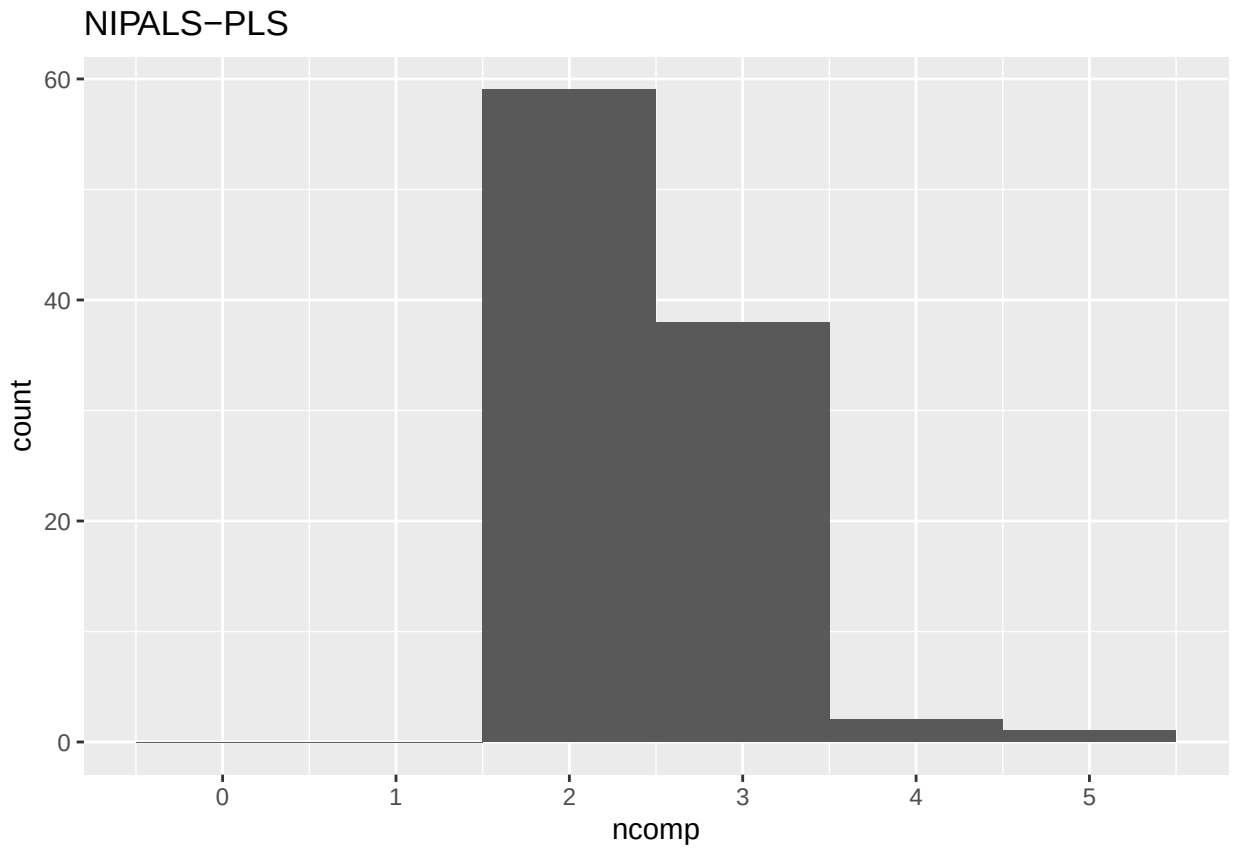
```
ggplot(data = ddspls_q_25, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 0:5, limits = c(-0.5,5.5))  +
  labs(title = "ddsPLS")
```



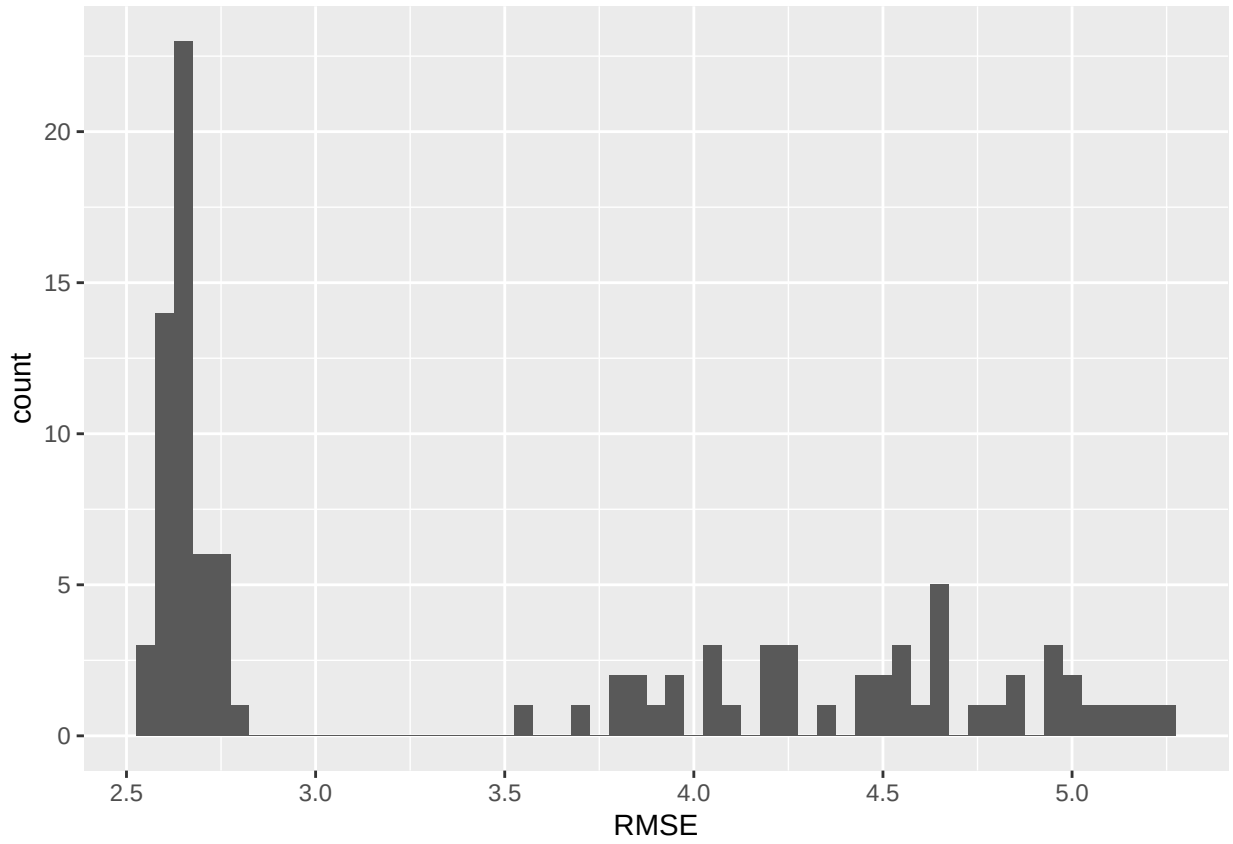

```
ggplot(data = spls_q_25, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 0:5, limits = c(-0.5,5.5))  +
  labs(title = "sPLS")
```



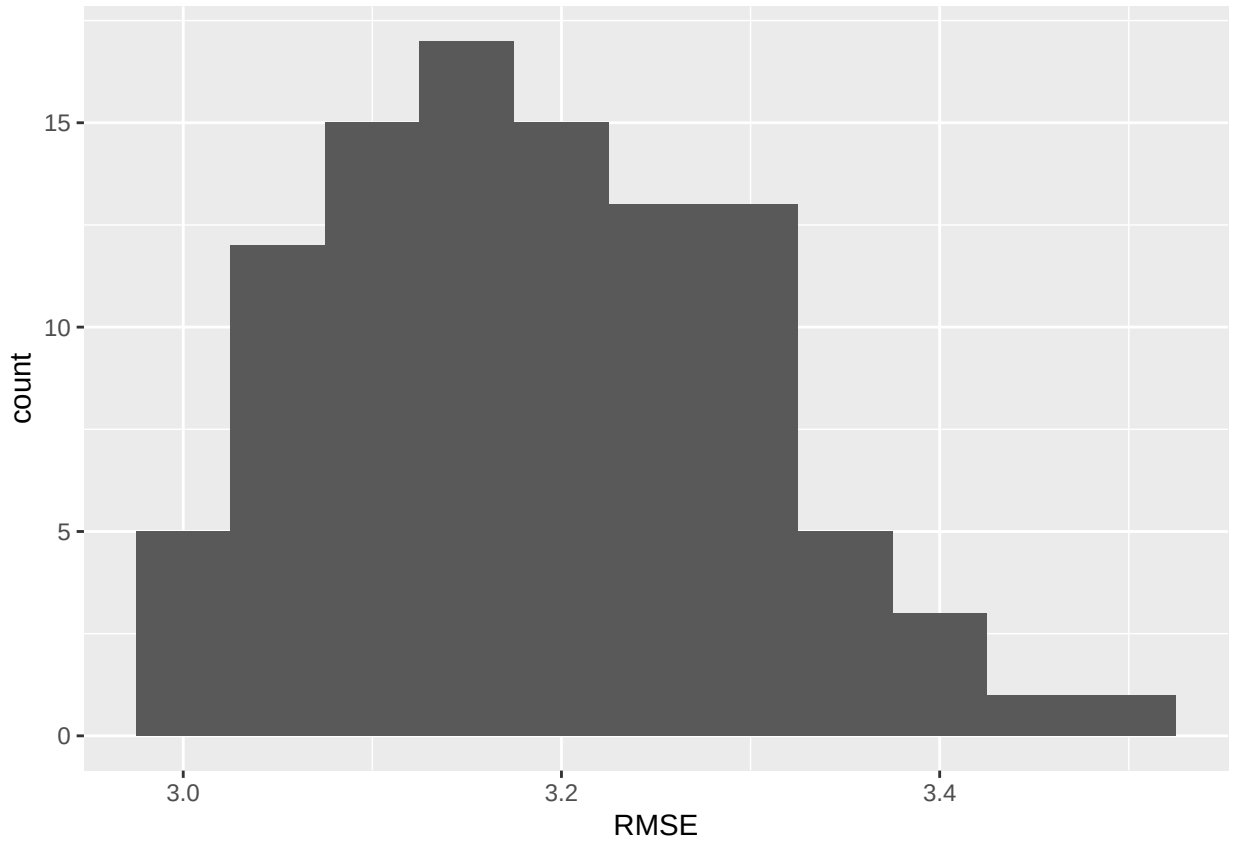

```
ggplot(data = pls_q_25, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 0:5, limits = c(-0.5,5.5))  +
  labs(title = "NIPALS-PLS")
```



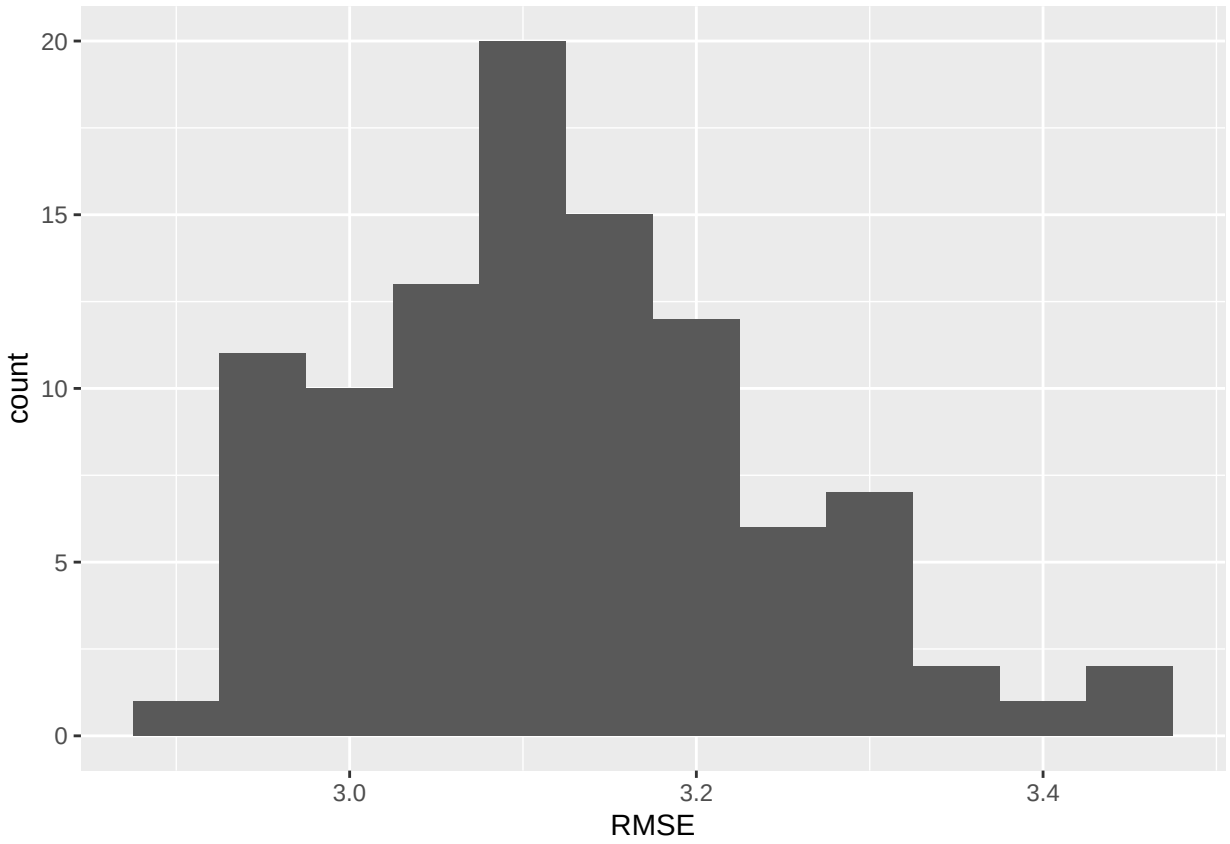

```
ggplot(ddspls_q_25, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```




```
ggplot(spls_q_25, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```




```
ggplot(pls_q_25, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```

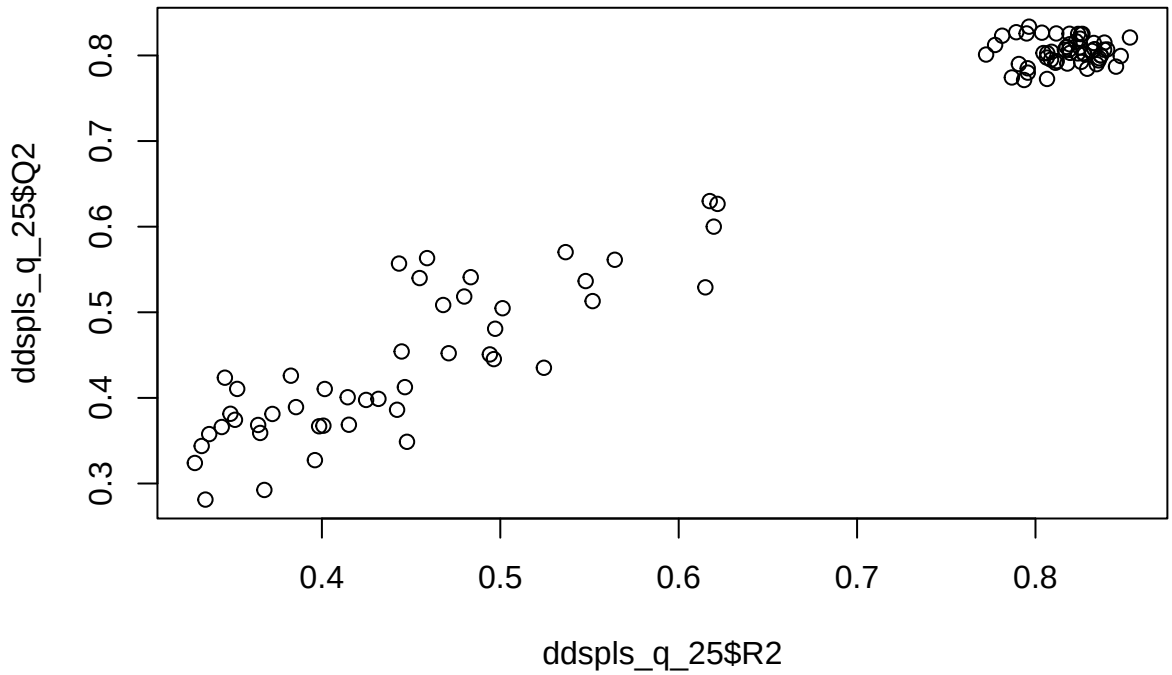



```
q_results <- as.data.frame(rbind(c(mean(ddspls_q_25$RMSE), median(ddspls_q_25$RMSE), var(ddspls_q_25$RMS
      c(mean(spls_q_25$RMSE), median(spls_q_25$RMSE), var(spls_q_25$RMSE)),
      c(mean(pls_q_25$RMSE), median(pls_q_25$RMSE), var(pls_q_25$RMSE))))
q_results
```

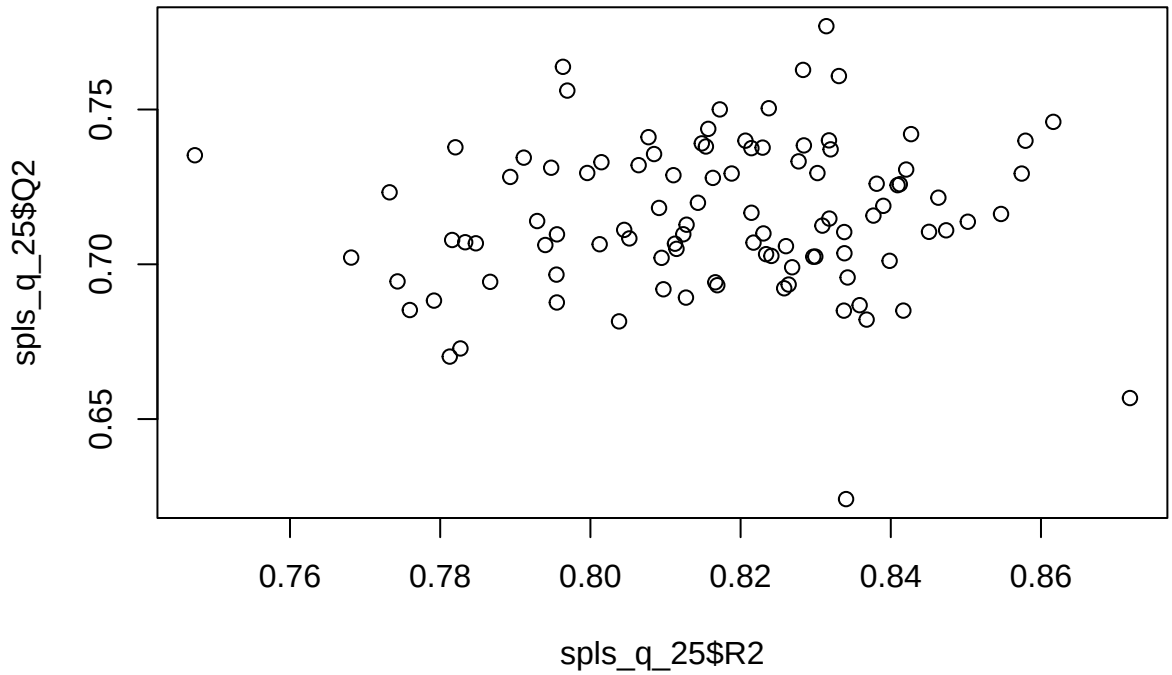
```
##         V1       V2         V3
## 1 3.501722 2.756319 0.92295103
## 2 3.186639 3.178454 0.01184950
## 3 3.123916 3.111958 0.01445585
```

When trying to predict a larger number of responses, many of which are uncorrelated ddsPLS has much more variance in the resulting RMSE. However, it tends to perform better but there is a chance of significantly worse performance.

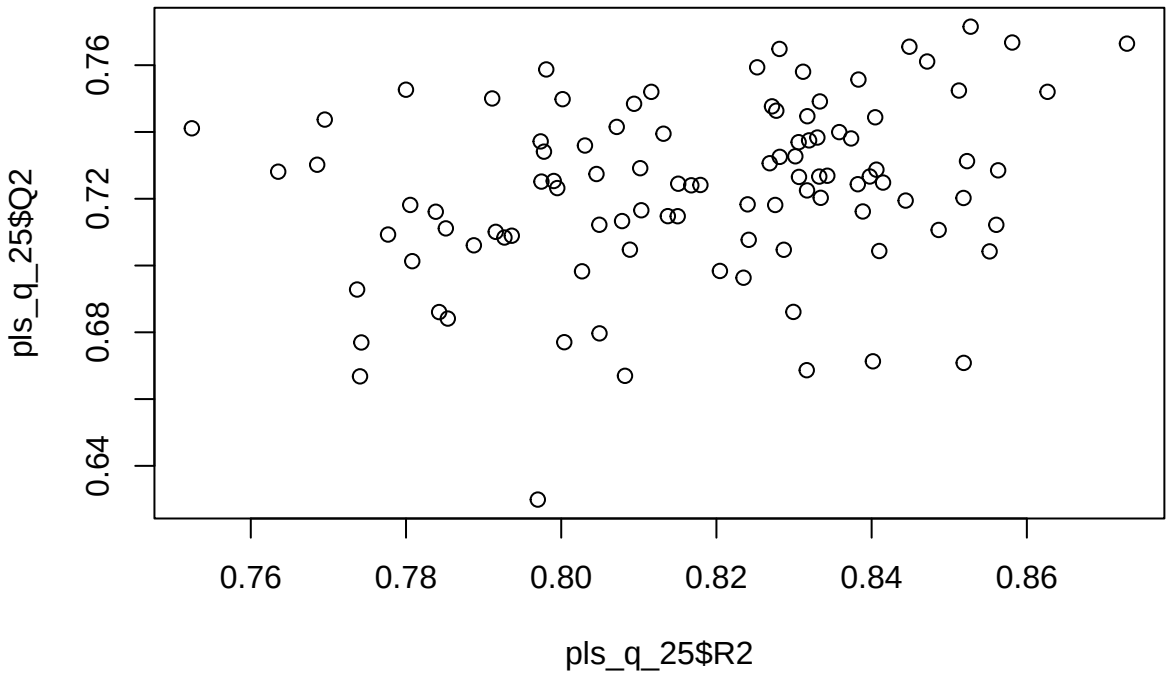
```
plot(ddspls_q_25$R2, ddspls_q_25$Q2)
```




```
plot(spls_q_25$R2, spls_q_25$Q2)
```




```
plot(pls_q_25$R2, pls_q_25$Q2)
```




```
cor(ddspls_q_25$R2, ddspls_q_25$Q2)
```

```
## [1] 0.9815783
```

```
cor(spls_q_25$R2, spls_q_25$Q2)
```

```
## [1] 0.06194807
```

```
cor(pls_q_25$R2, pls_q_25$Q2)
```

```
## [1] 0.2782815
```

Performance issues look like due to problem with ddsPLS fitting the original data. For both PLS and sPLS we observe about what we would expect, R^2 outperforms Q^2 with a positive relation between the two (very weak in the case of PLS). When ddsPLS performs well it far outperforms the other two methods. However, it also performs extremely poorly at time. This is due to the fact that it doesn't fit the data well. If possible it would be good to do more research into what is happening in these cases where the fit is extremely poor. Maybe trying to avoid overfitting the data too much or removing some of the responses that are correlated with the predictors.

Looks like `ddsPLS` function doesn't work when `q=1`, not sure why will try to look thought the source code. Thought I had tested this before, I guess not.

Q Test Newest D method

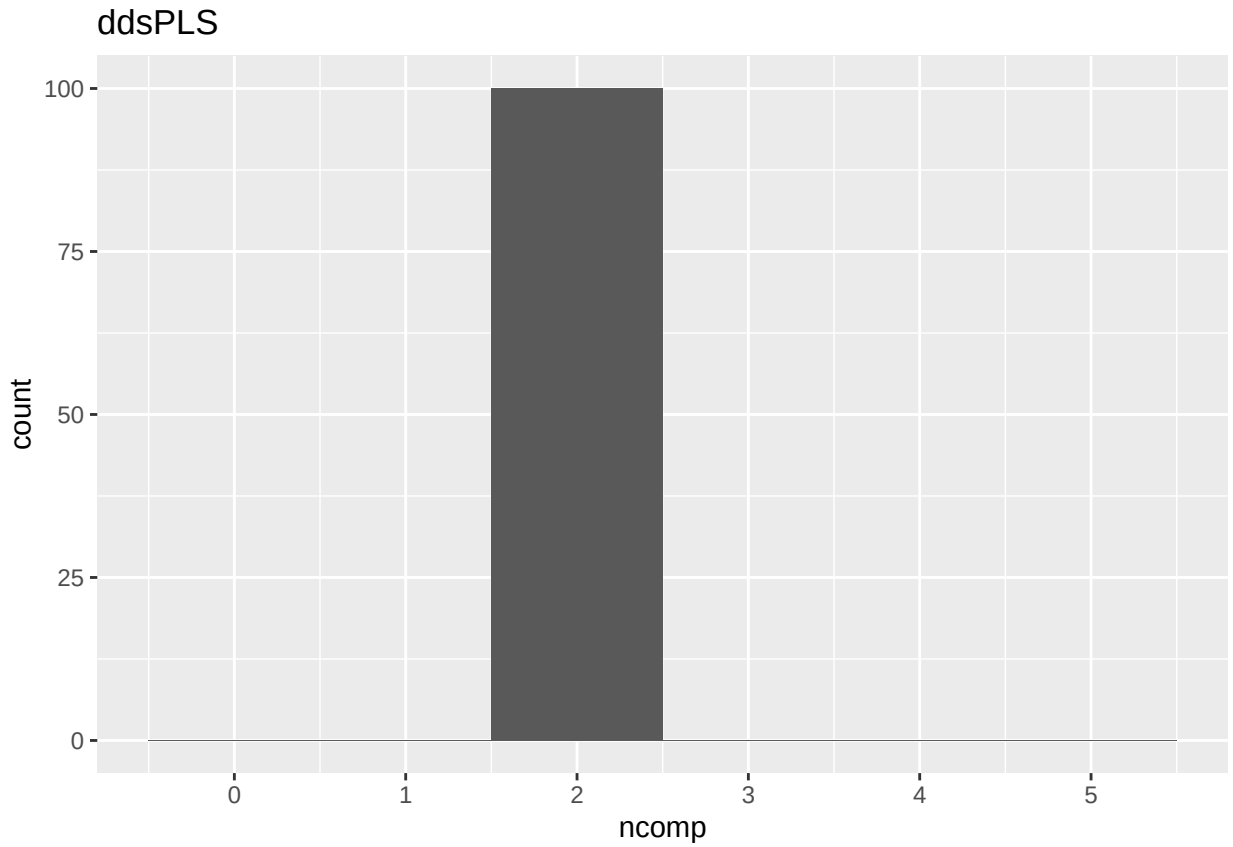
```
ddspls.q.reps <- replicate(100, q_eval(q = 25, D_method = "newest"))
```

```
spls.q.reps <- replicate(100, q_eval(q = 25, func = "spls", D_method = "newest"))
```

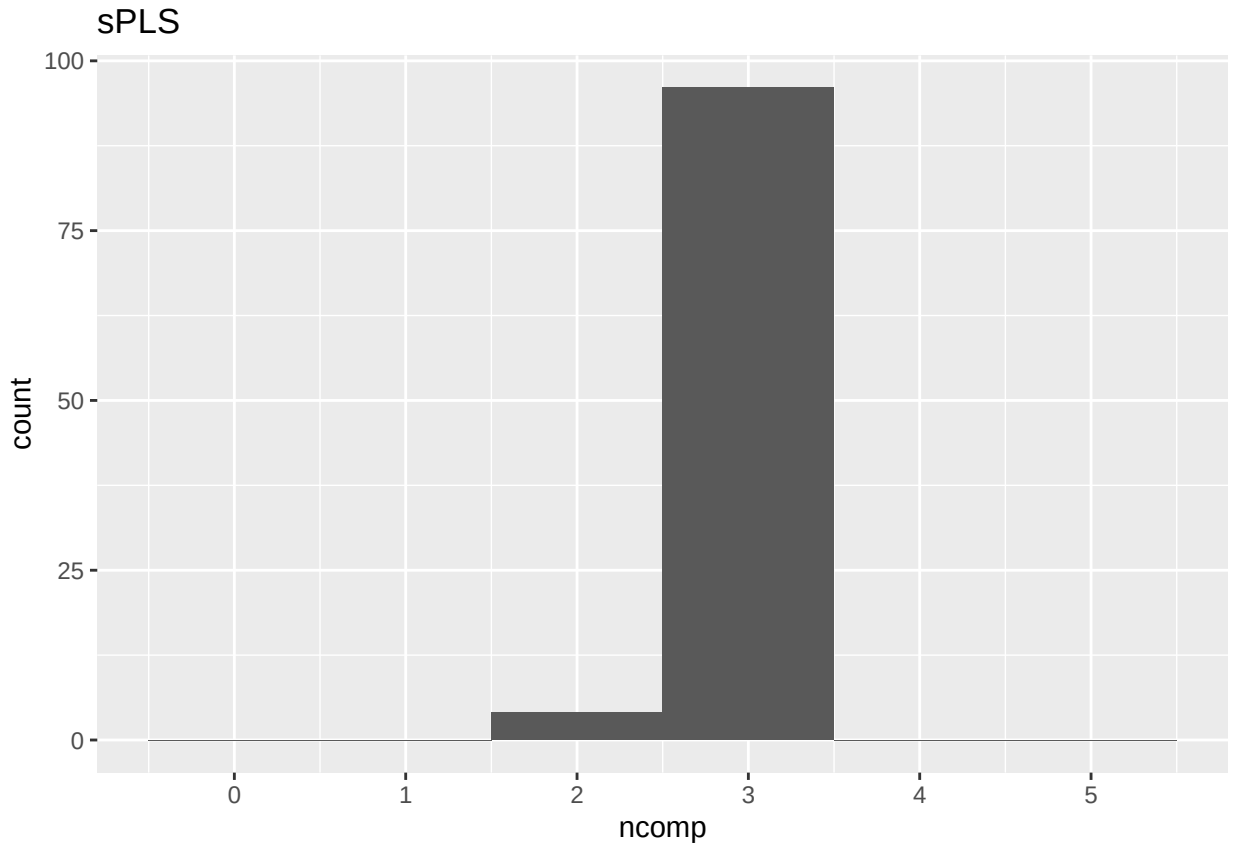
```
pls.q.reps <- replicate(100, q_eval(q = 25, func = "pls", D_method = "newest"))
```

```
ddspls_q_25_Dnewest <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_q_25_Dnewes
spls_q_25_Dnewest <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls_q_25_Dnewest.cs
pls_q_25_Dnewest <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls_q_25_Dnewest.csv"
```

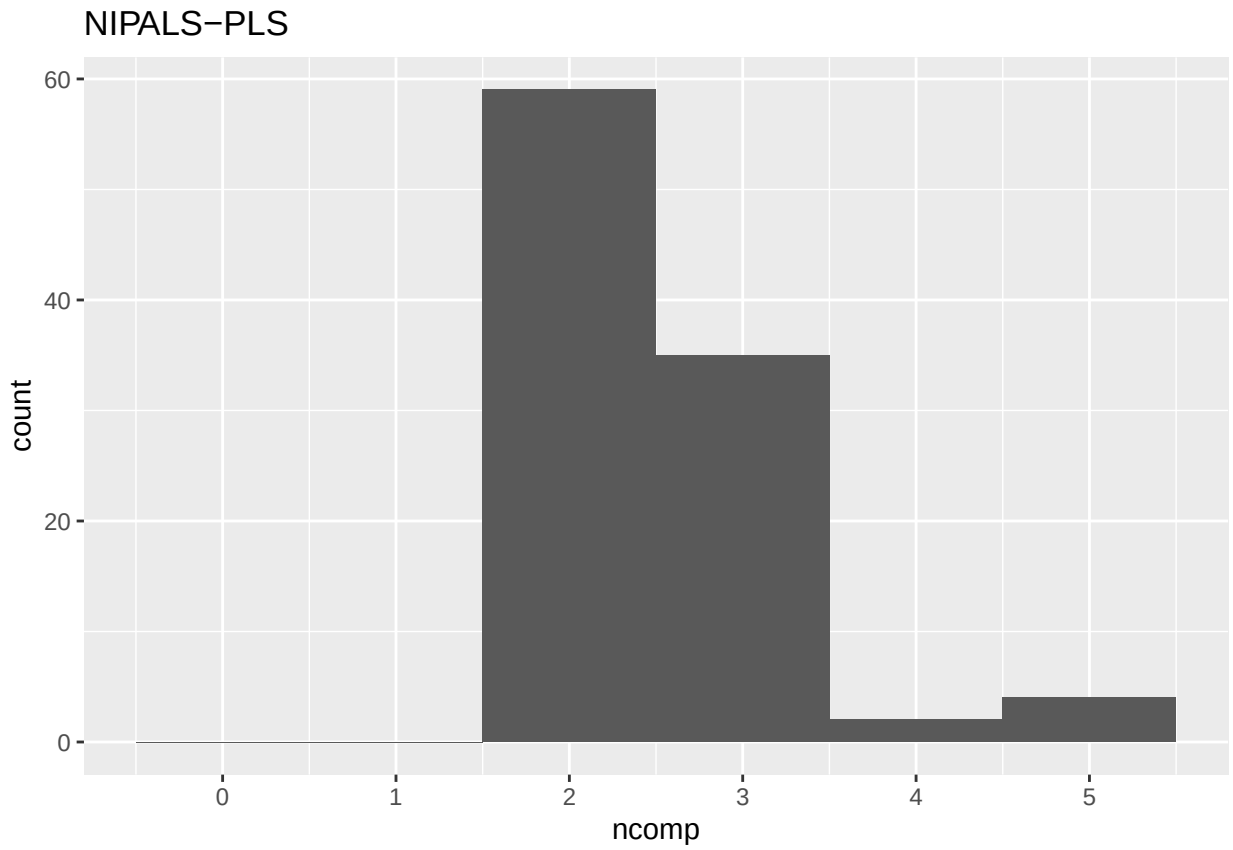
```
ggplot(data = ddspls_q_25_Dnewest, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 0:5, limits = c(-0.5,5.5))  +
  labs(title = "ddsPLS")
```



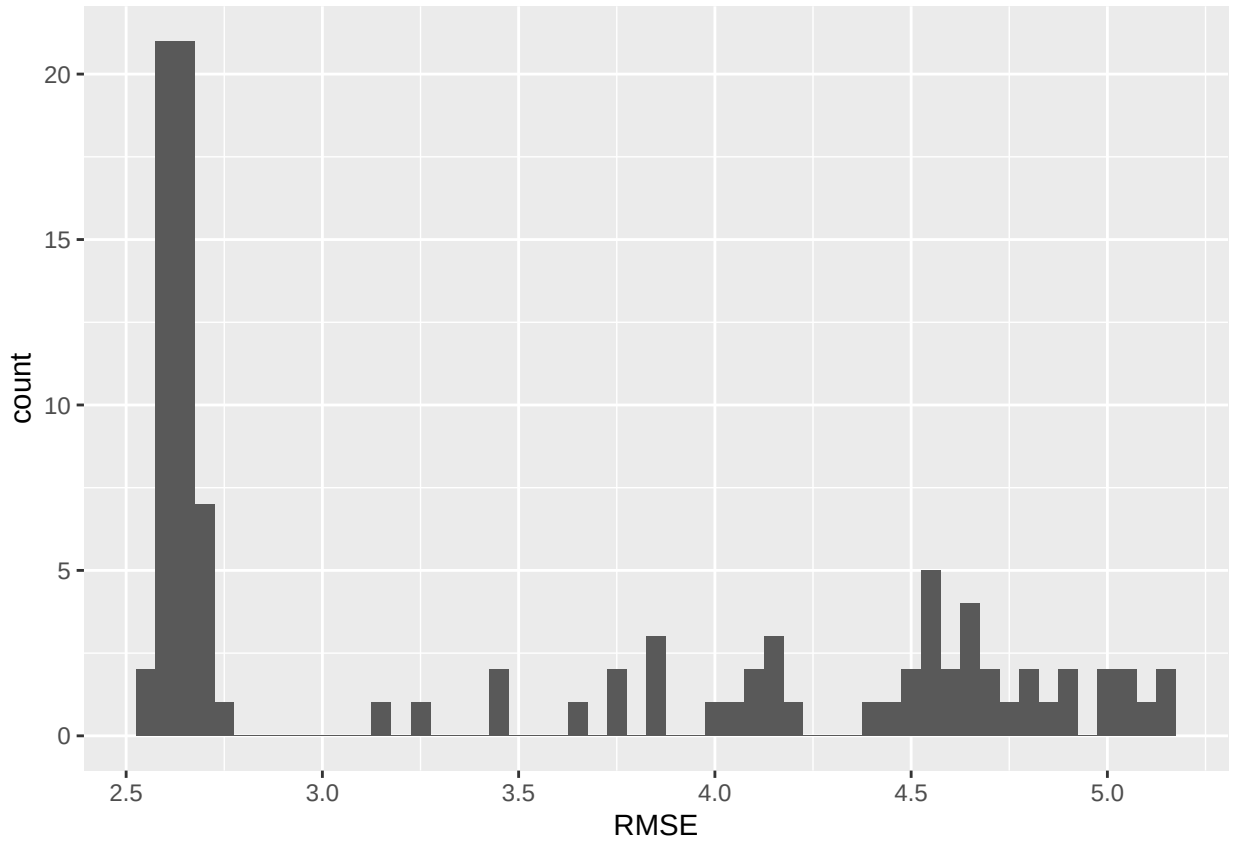

```
ggplot(data = spls_q_25_Dnewest, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 0:5, limits = c(-0.5,5.5))  +
  labs(title = "sPLS")
```



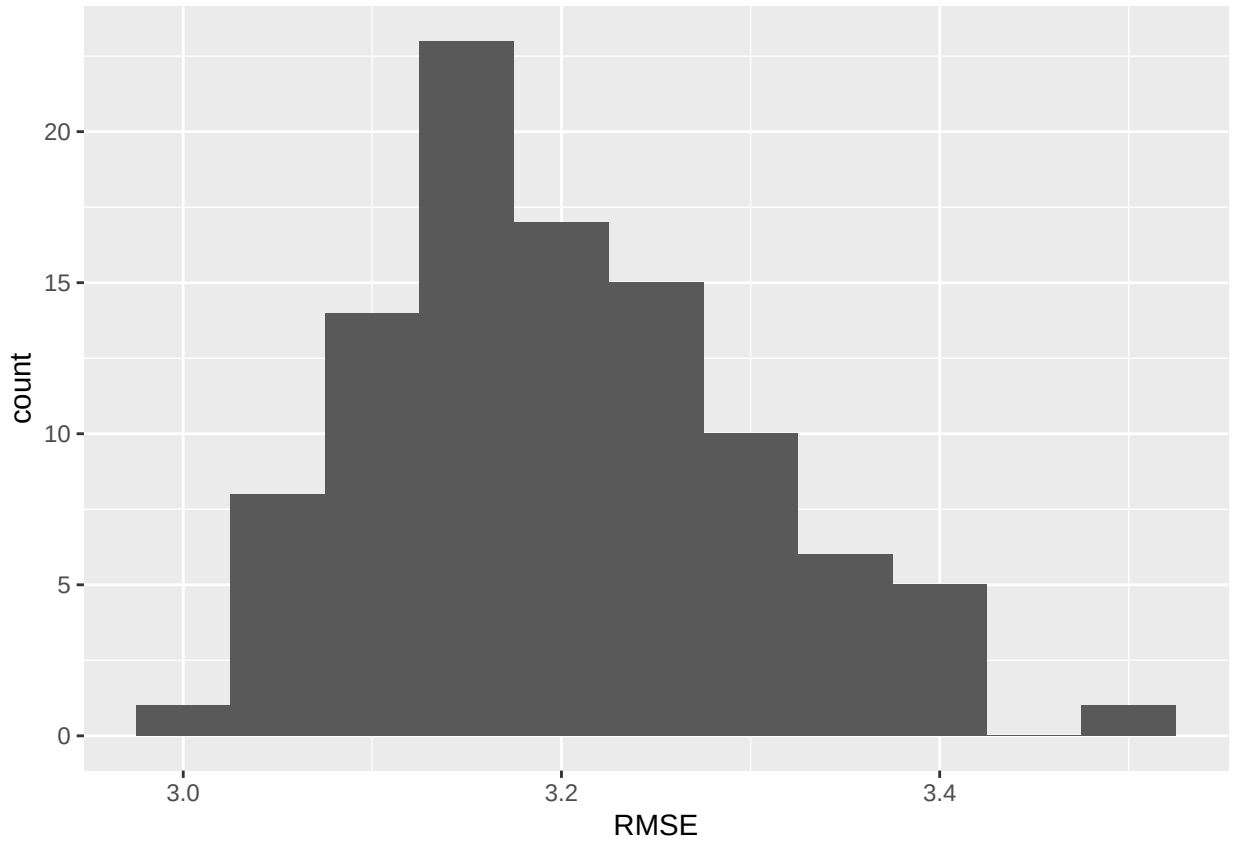

```
ggplot(data = pls_q_25_Dnewest, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 0:5, limits = c(-0.5,5.5))  +
  labs(title = "NIPALS-PLS")
```



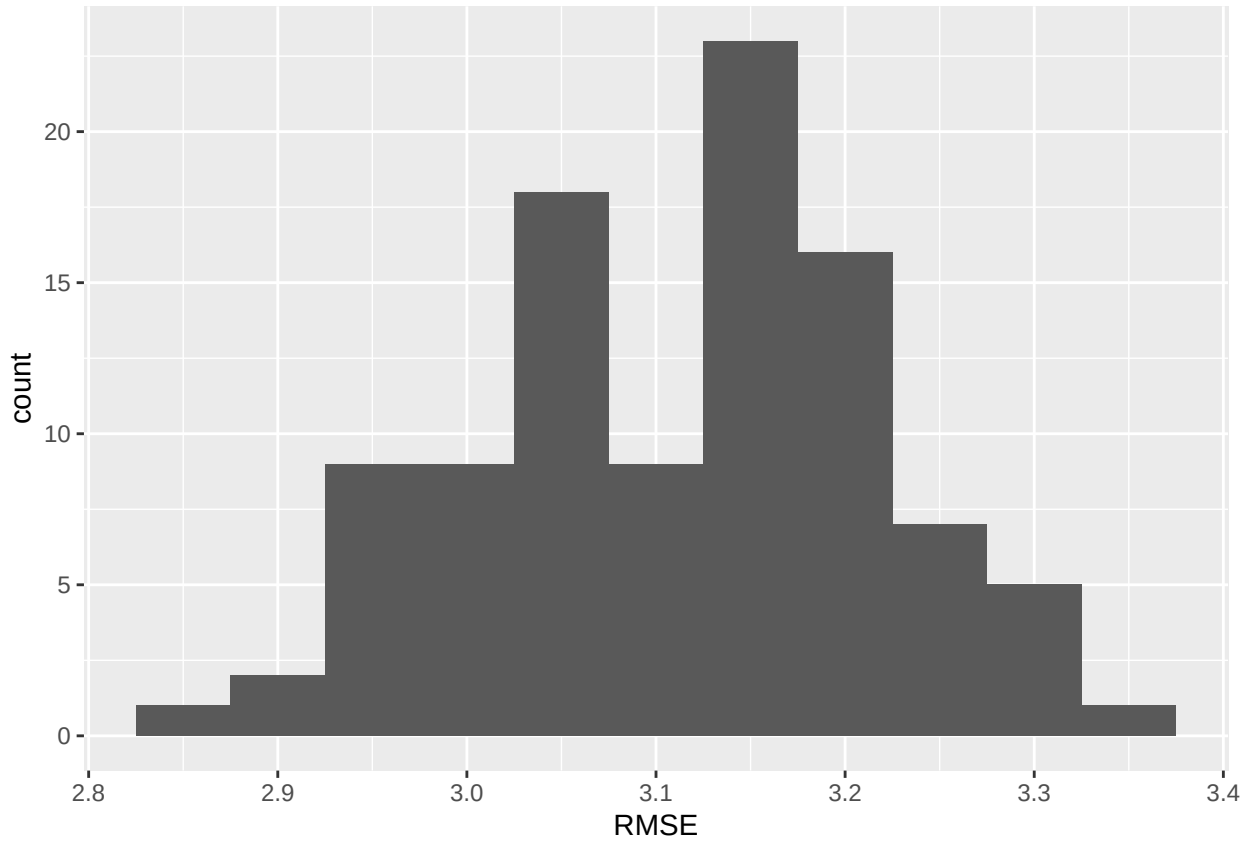

```
ggplot(ddspls_q_25_Dnewest, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```




```
ggplot(spls_q_25_Dnewest, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```




```
ggplot(pls_q_25_Dnewest, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```

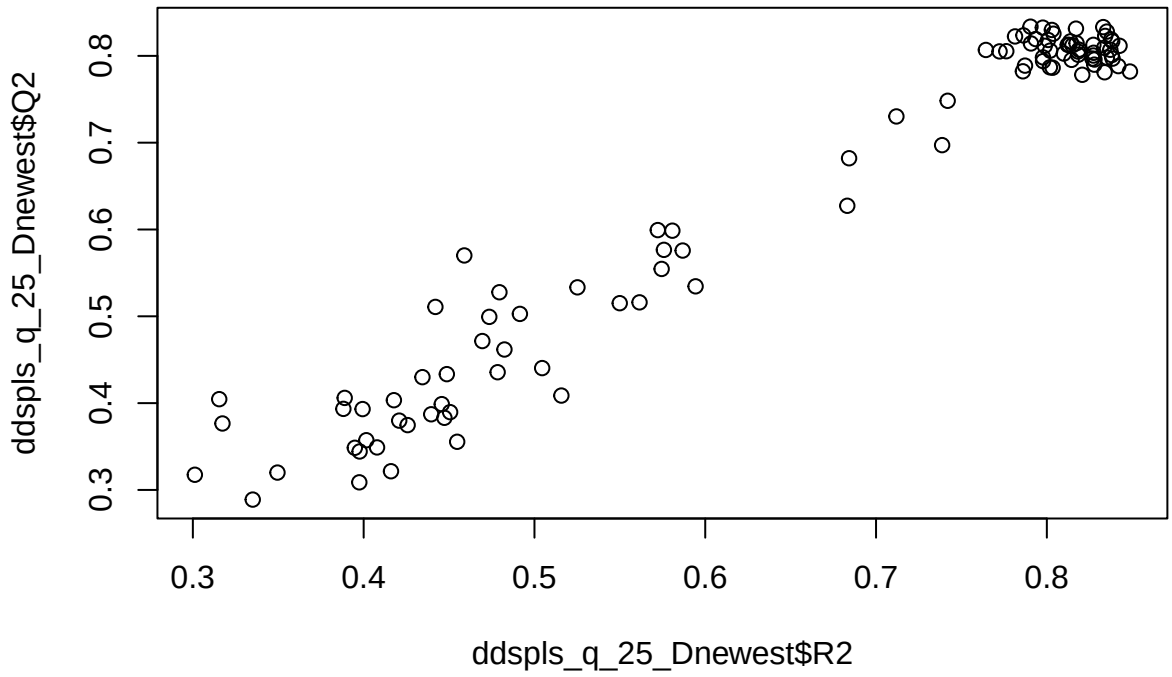



```
q_results_Dnewest <- as.data.frame(rbind(c(mean(ddspls_q_25_Dnewest$RMSE), median(ddspls_q_25_Dnewest$RM
      c(mean(spls_q_25_Dnewest$RMSE), median(spls_q_25_Dnewest$RMSE), var(spls_q_25_Dnewest$RMSE)),
      c(mean(pls_q_25_Dnewest$RMSE), median(pls_q_25_Dnewest$RMSE), var(pls_q_25_Dnewest$RMSE))))
q_results_Dnewest
```

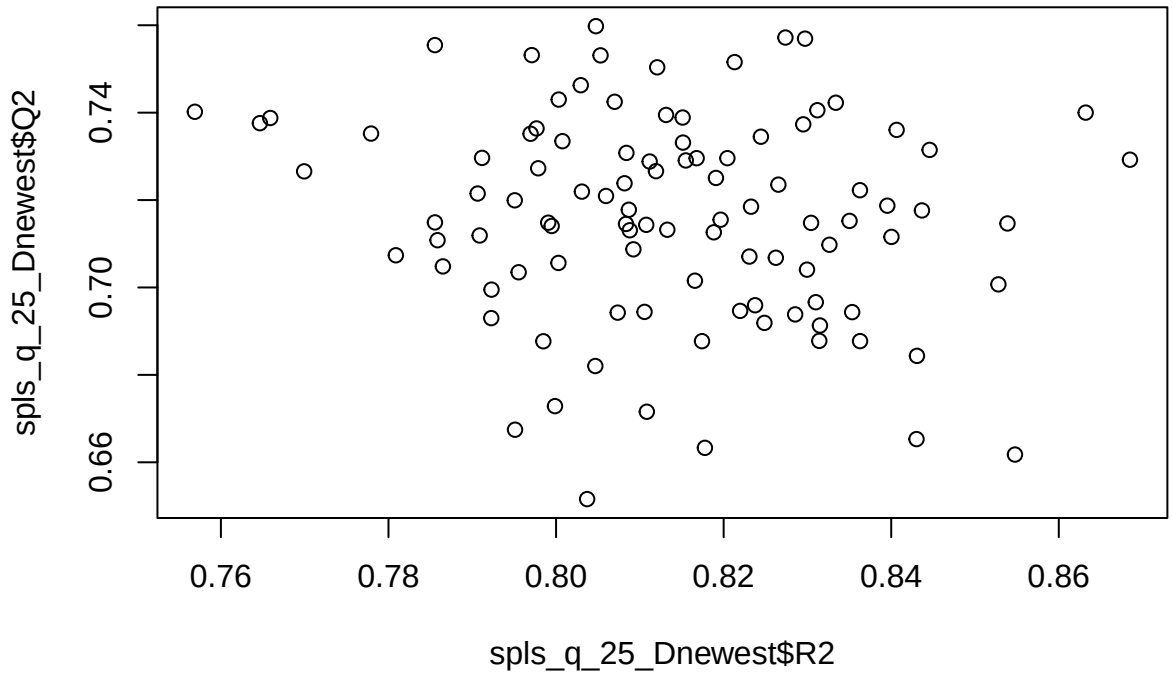
```
##         V1       V2          V3
## 1 3.479514 2.717717 0.914094882
## 2 3.197472 3.180867 0.009682116
## 3 3.112206 3.127018 0.011038909
```

Looks like `ddsPLS` correctly gets structure again with more complex D matrix but has more variable predictions with higher mean RMSE. Might want to try adding predictors. Interestingly, R^2 and Q^2 are negatively correlalted for sPLS.

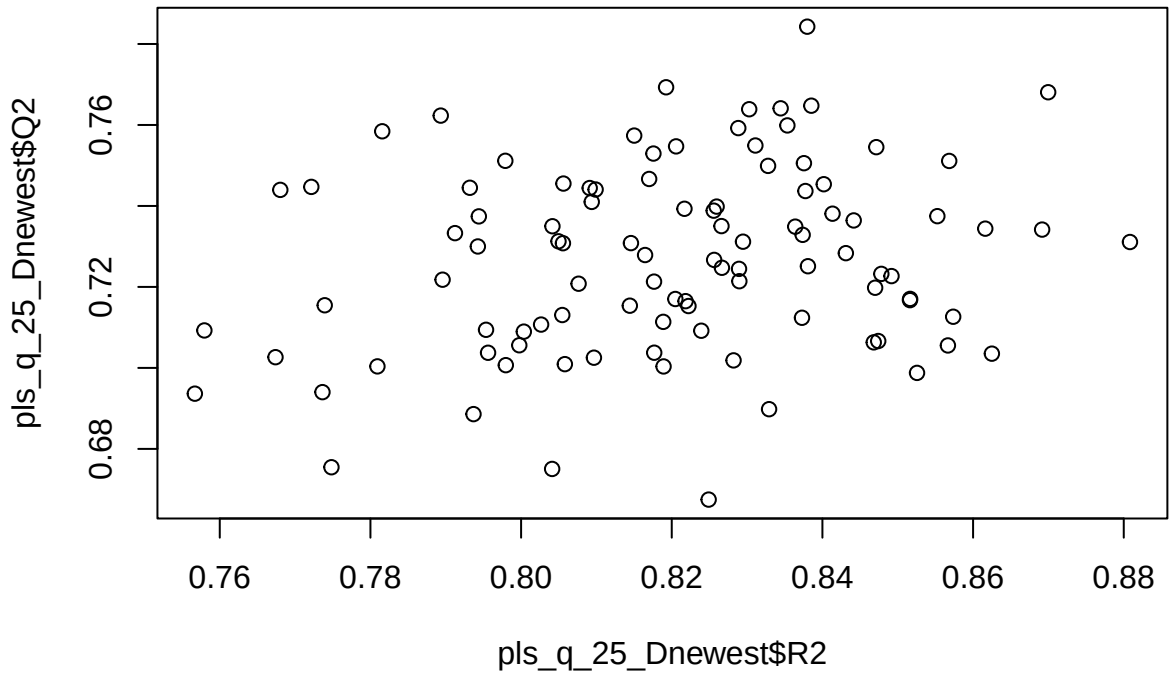
```
plot(ddspls_q_25_Dnewest$R2, ddspls_q_25_Dnewest$Q2)
```




```
plot(spls_q_25_Dnewest$R2, spls_q_25_Dnewest$Q2)
```




```
plot(pls_q_25_Dnewest$R2, pls_q_25_Dnewest$Q2)
```




```
cor(ddspls_q_25_Dnewest$R2, ddspls_q_25_Dnewest$Q2)
```

```
## [1] 0.9798786
```

```
cor(spls_q_25_Dnewest$R2, spls_q_25_Dnewest$Q2)
```

```
## [1] -0.1548873
```

```
cor(pls_q_25_Dnewest$R2, pls_q_25_Dnewest$Q2)
```

```
## [1] 0.2189402
```

New Complex D Structure

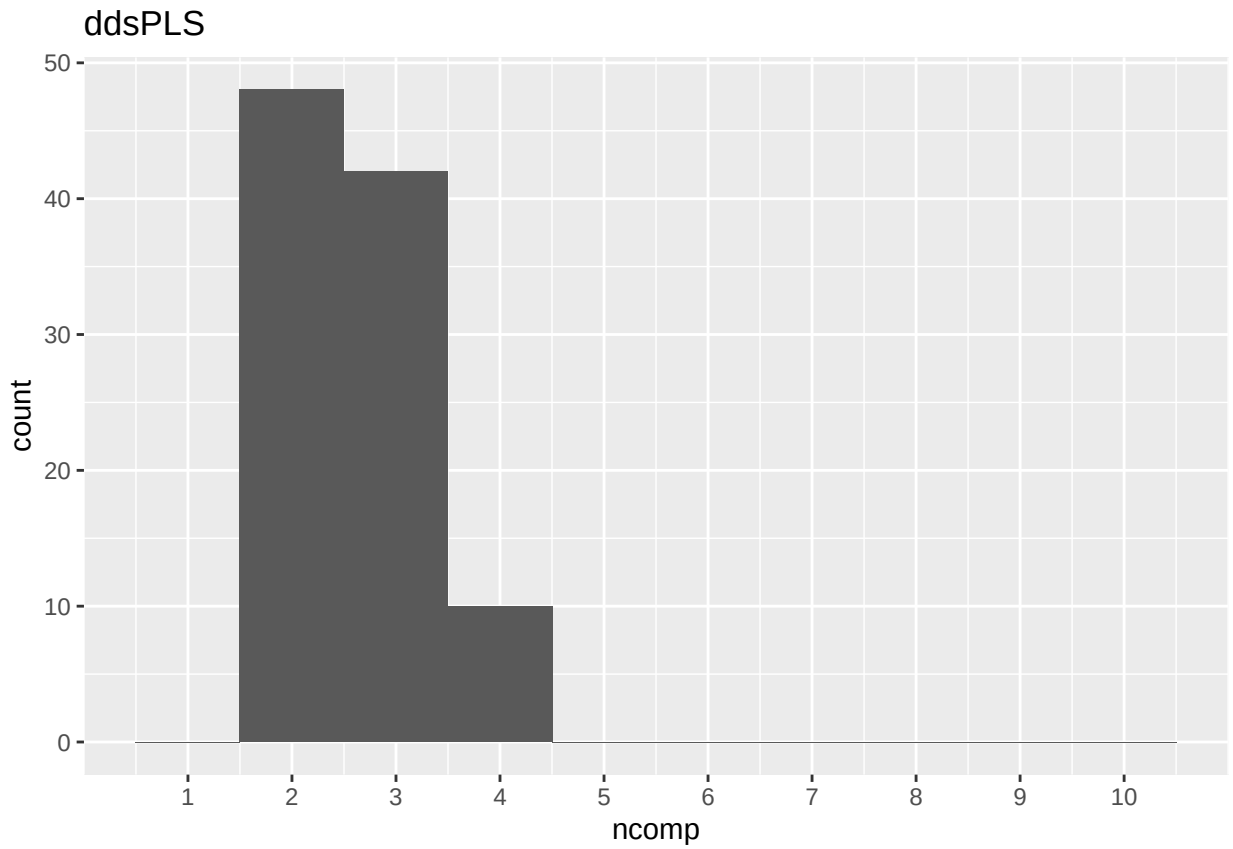
```
ddspls.q.reps <- replicate(100, q_eval(q = 10, struc = "complex", D_method = "complex"))
```

```
spls.q.reps <- replicate(100, q_eval(q = 10, struc = "complex", func = "spls", D_method = "complex"))
```

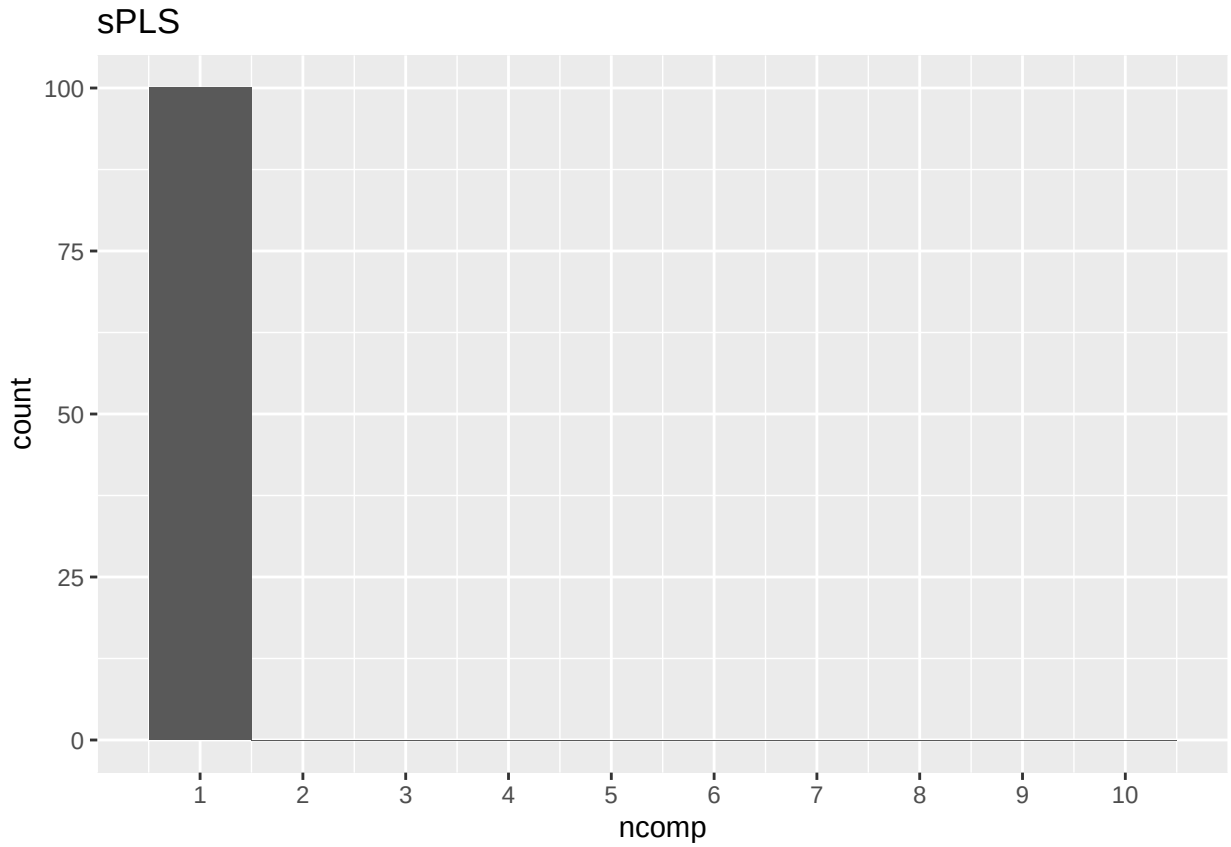
```
pls.q.reps <- replicate(100, q_eval(q = 10, struc = "complex", func = "pls", D_method = "complex"))
```

```
ddspls_q_10 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_q_10.csv")
spls_q_10 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls_q_10.csv")
pls_q_10 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls_q_10.csv")
```

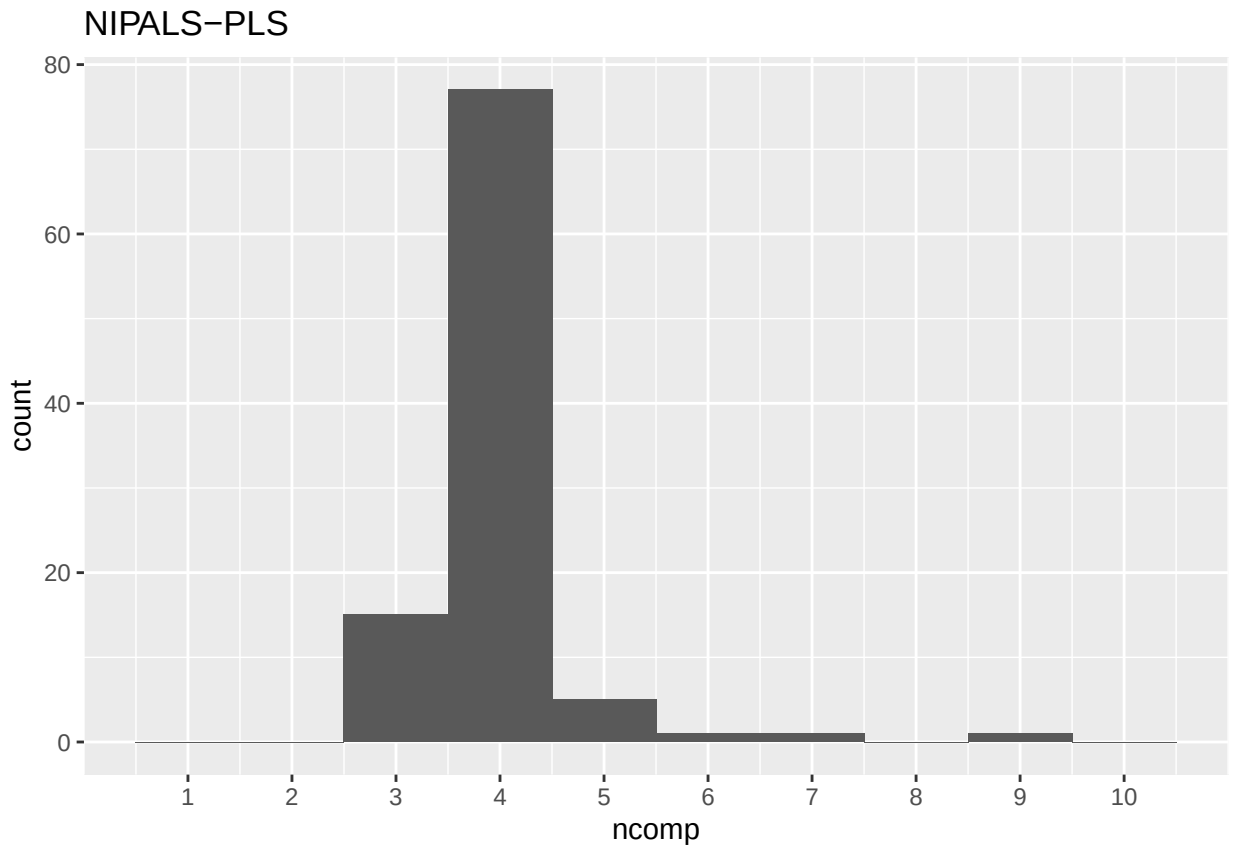
```
ggplot(data = ddspls_q_10, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5))  +
  labs(title = "ddsPLS")
```



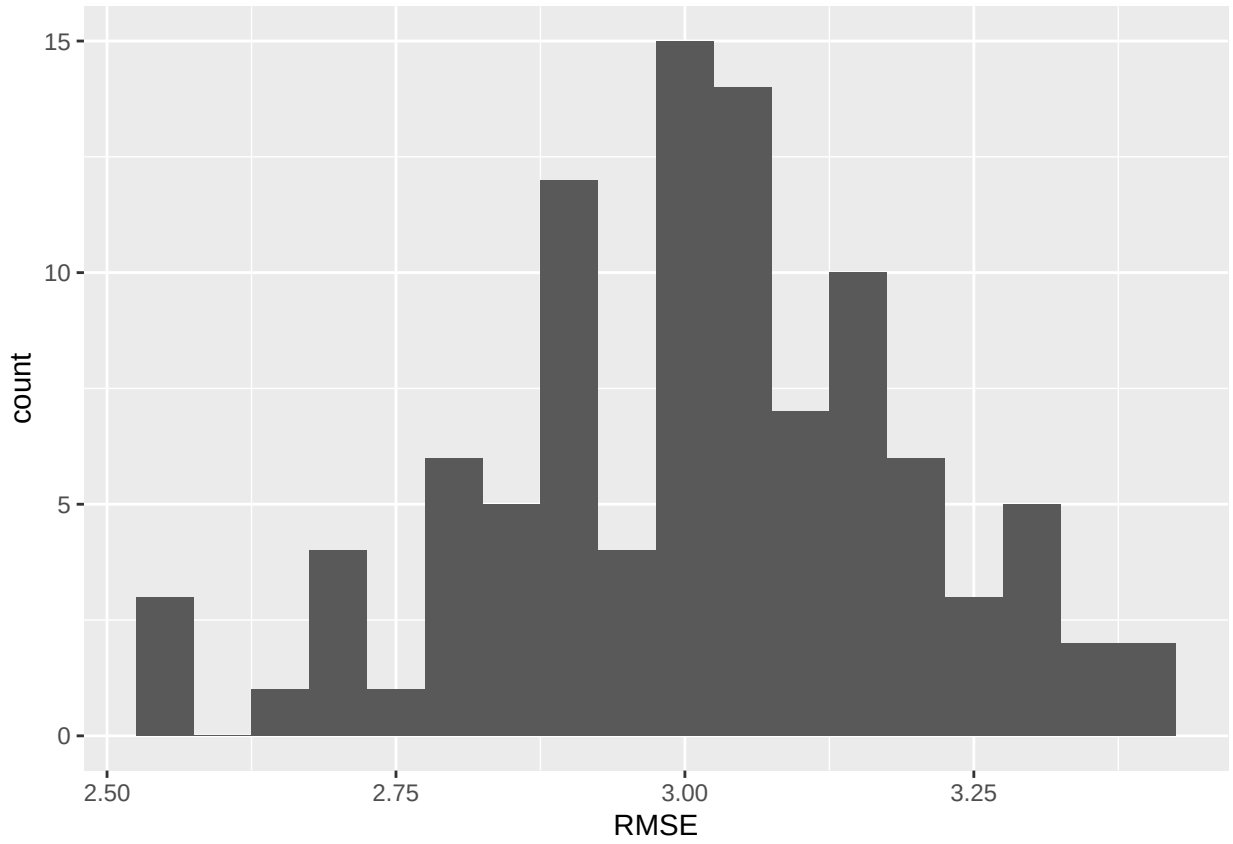

```
ggplot(data = spls_q_10, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5))  +
  labs(title = "sPLS")
```



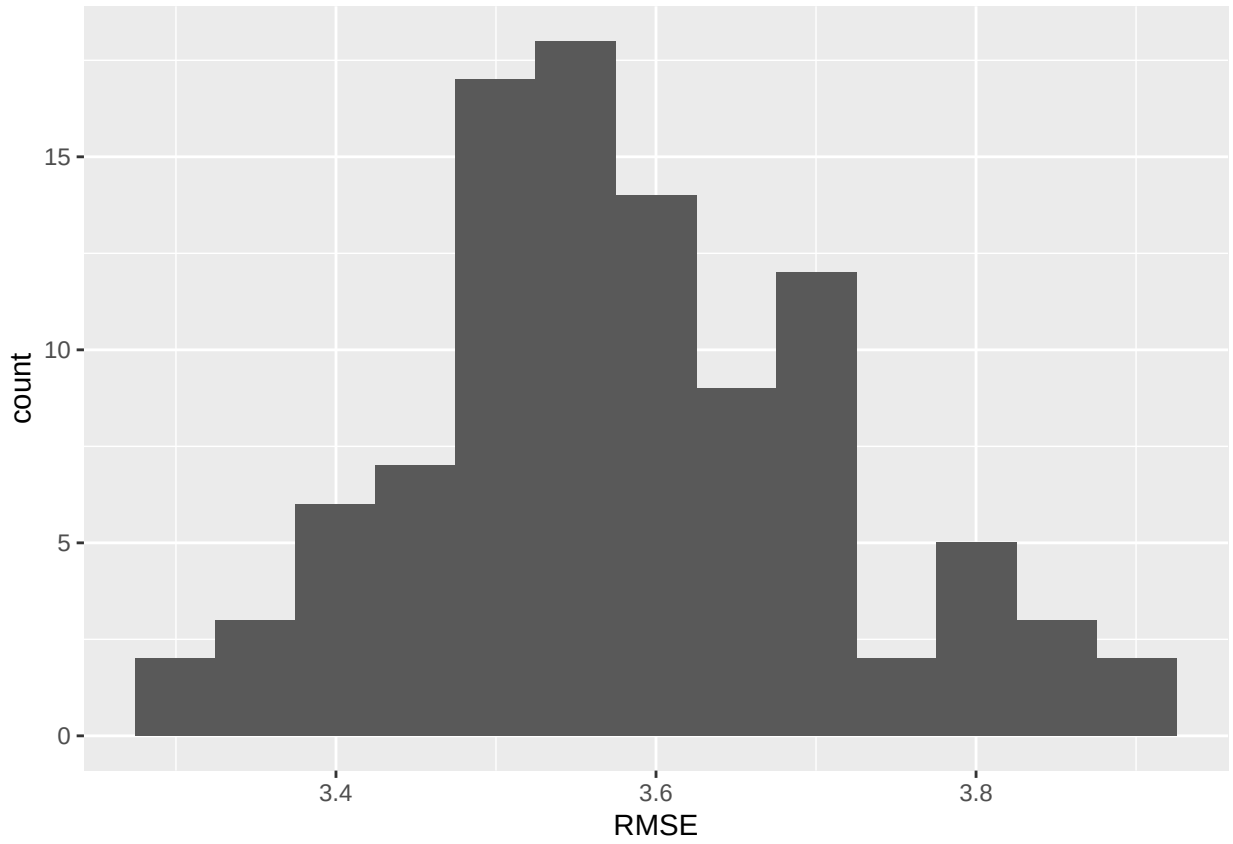

```
ggplot(data = pls_q_10, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5))  +
  labs(title = "NIPALS-PLS")
```



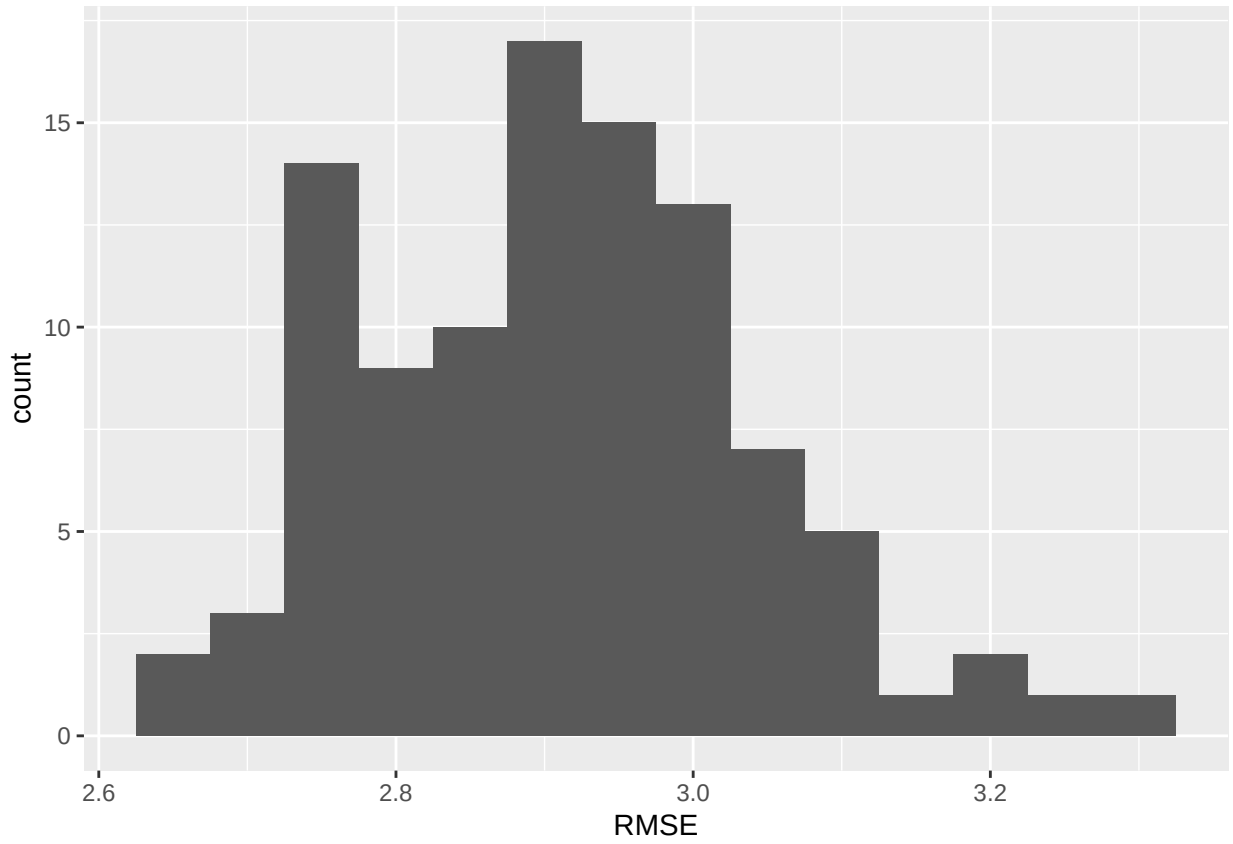

```
ggplot(ddspls_q_10, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```




```
ggplot(spls_q_10, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```



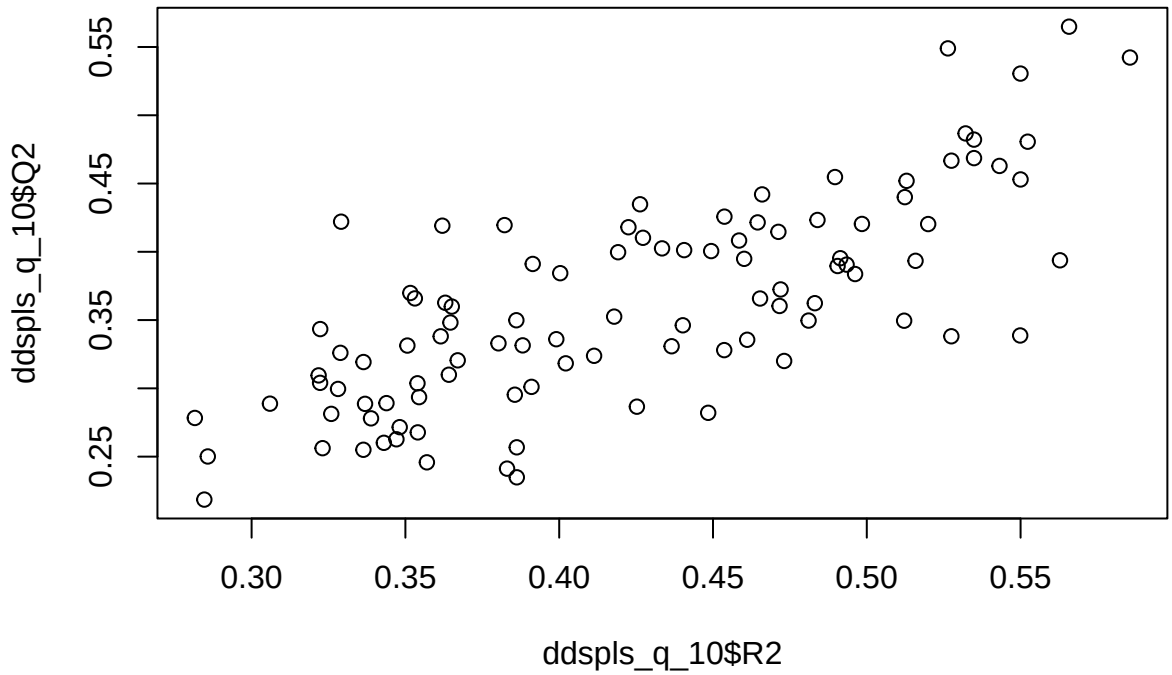

```
ggplot(pls_q_10, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```



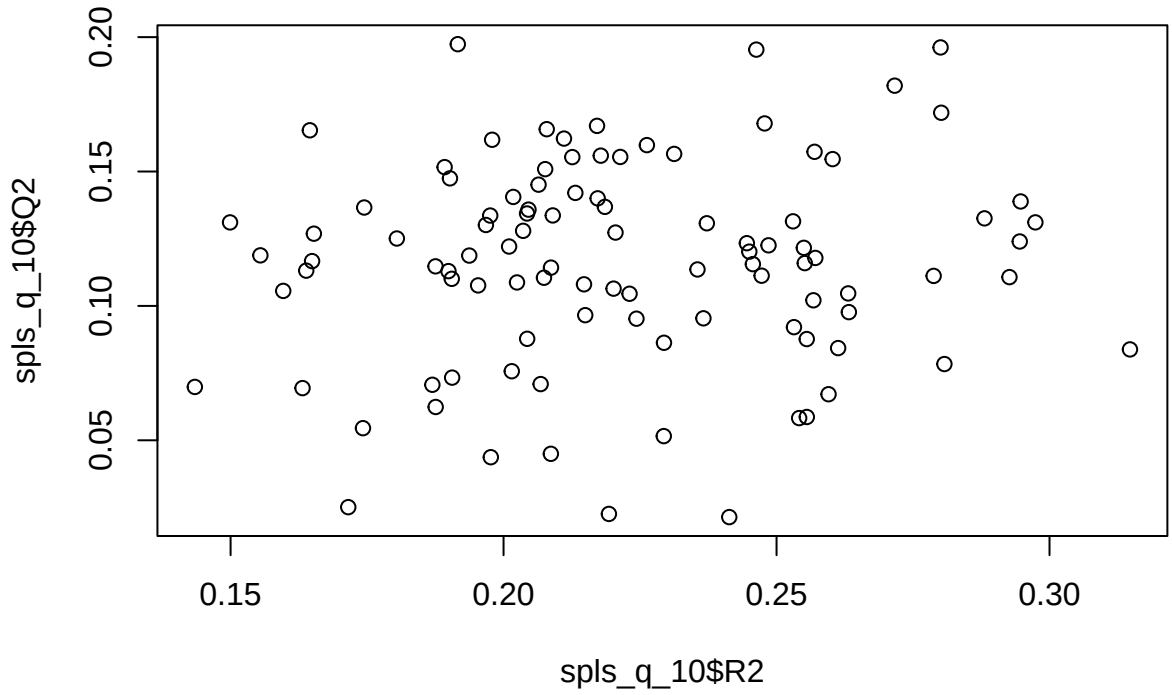

```
q_results_10 <- as.data.frame(rbind(c(mean(ddspls_q_10$RMSE), median(ddspls_q_10$RMSE), var(ddspls_q_10$
      c(mean(spls_q_10$RMSE), median(spls_q_10$RMSE), var(spls_q_10$RMSE)),
      c(mean(pls_q_10$RMSE), median(pls_q_10$RMSE), var(pls_q_10$RMSE))))
q_results_10
```

```
##         V1       V2         V3
## 1 3.016456 3.023800 0.03429483
## 2 3.581873 3.571662 0.01758013
## 3 2.910152 2.893207 0.01692187
```

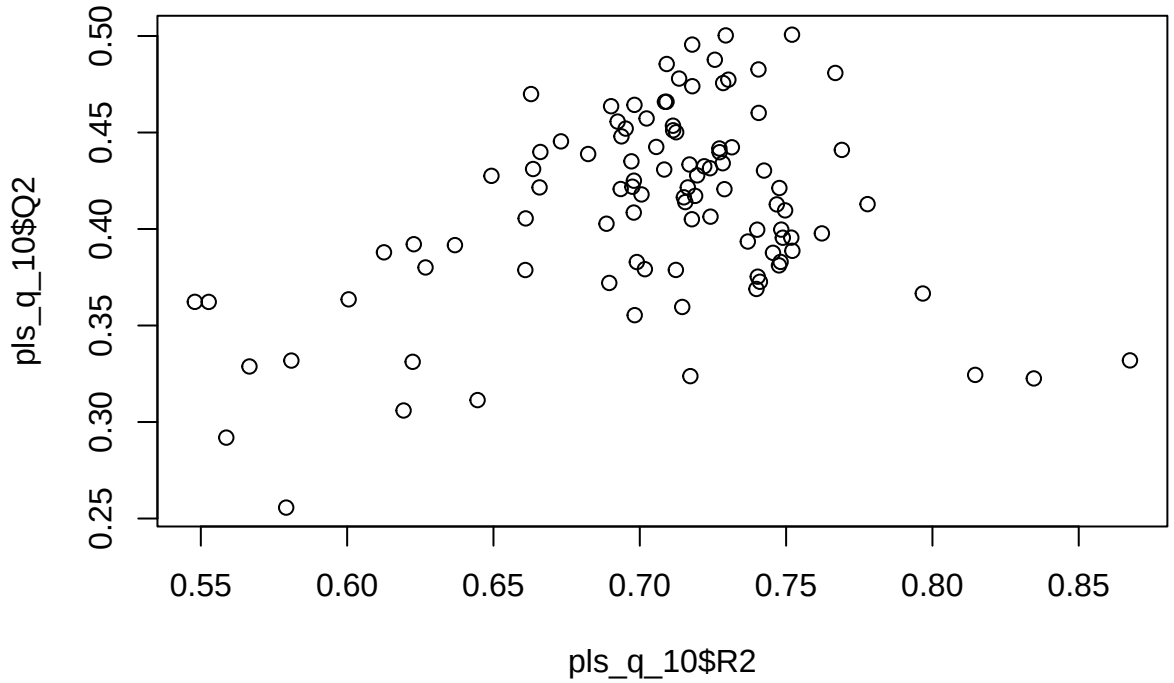
```
plot(ddspls_q_10$R2, ddspls_q_10$Q2)
```




```
plot(spls_q_10$R2, spls_q_10$Q2)
```




```
plot(pls_q_10$R2, pls_q_10$Q2)
```




```
cor(ddspls_q_10$R2, ddspls_q_10$Q2)
```

```
## [1] 0.752564
```

```
cor(spls_q_10$R2, spls_q_10$Q2)
```

```
## [1] 0.08763941
```

```
cor(pls_q_10$R2, pls_q_10$Q2)
```

```
## [1] 0.288633
```

sPLS struggles with many uncorrelated predictors. Interestingly, PLS performs better than ddsPLS. Although PLS looks to overfit the data more, I think it is more likely to make the correct number of components.

Larger Sample Size

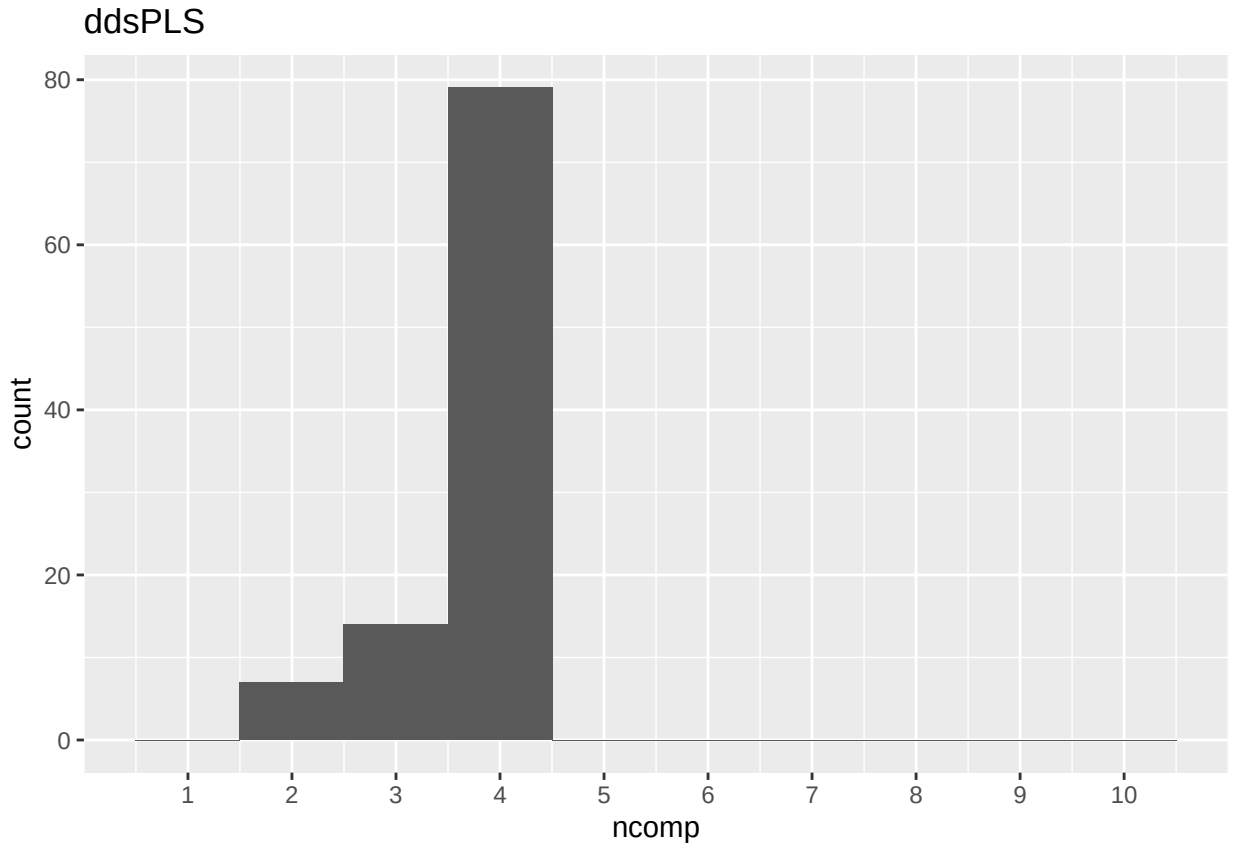
```
ddspls.q.reps <- replicate(100, q_eval(q = 10, struc = "complex", D_method = "complex", n = 300))
```

```
spls.q.reps <- replicate(100, q_eval(q = 10, struc = "complex", func = "spls", D_method = "complex", n =
```

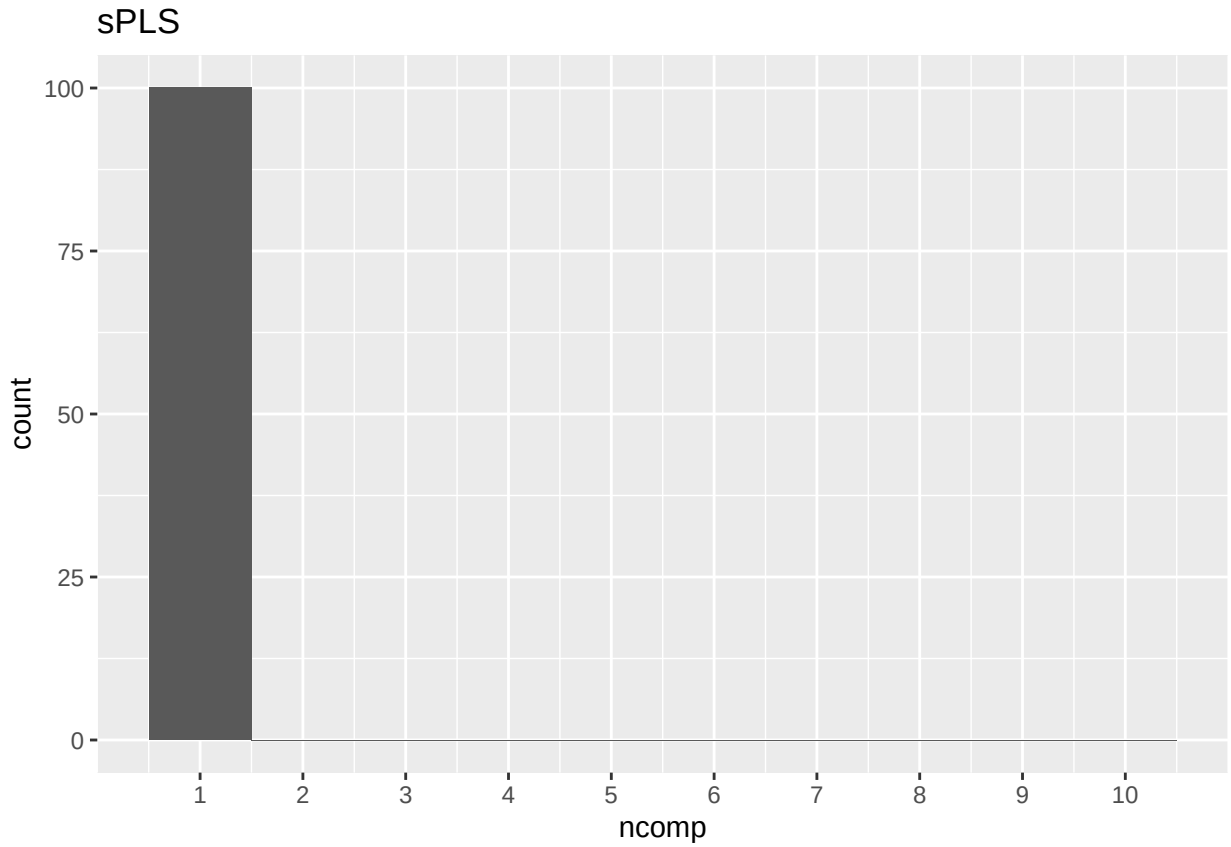
```
pls.q.reps <- replicate(100, q_eval(q = 10, struc = "complex", func = "pls", D_method = "complex", n = 3
```

```
ddspls_q_10_n_300 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_q_10_n_300.csv
spls_q_10_n_300 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls_q_10_n_300.csv")
pls_q_10_n_300 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls_q_10_n_300.csv")
```

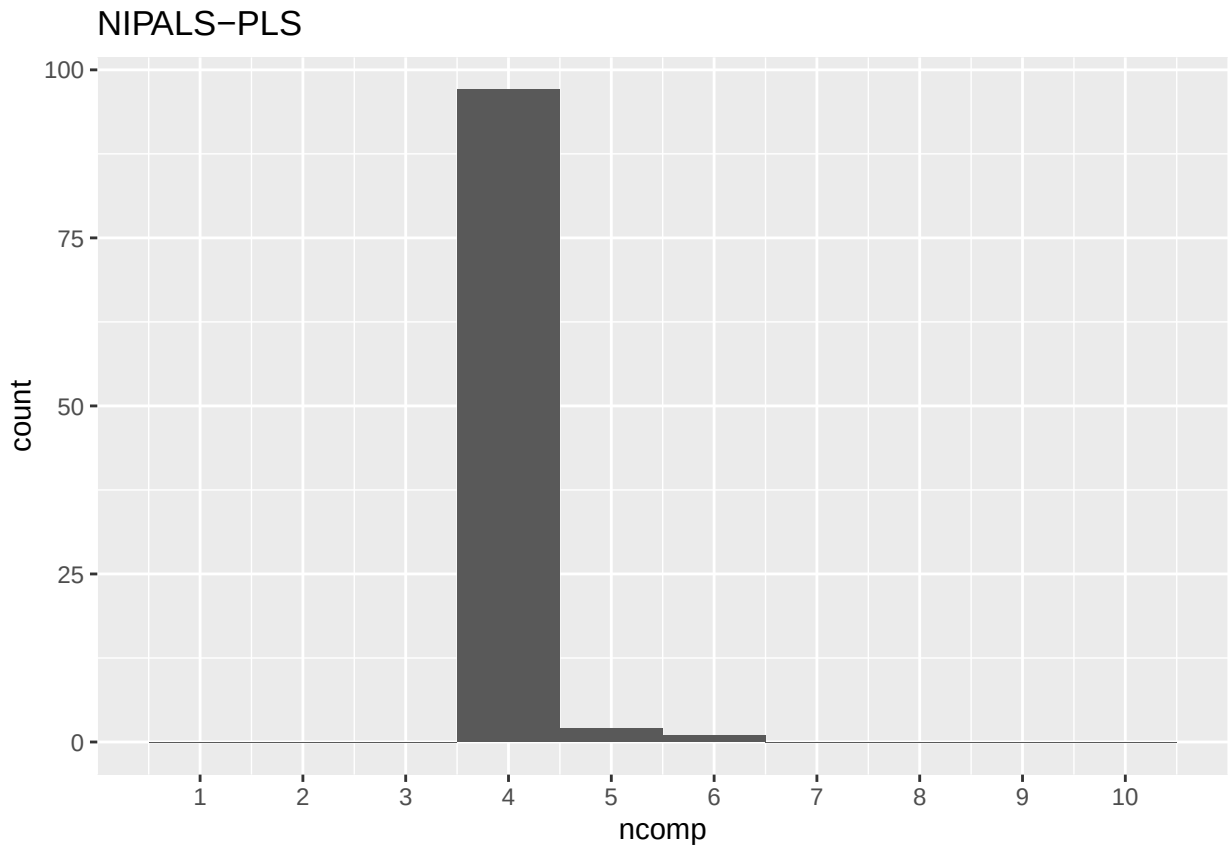
```
ggplot(data = ddspls_q_10_n_300, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5))  +
  labs(title = "ddsPLS")
```



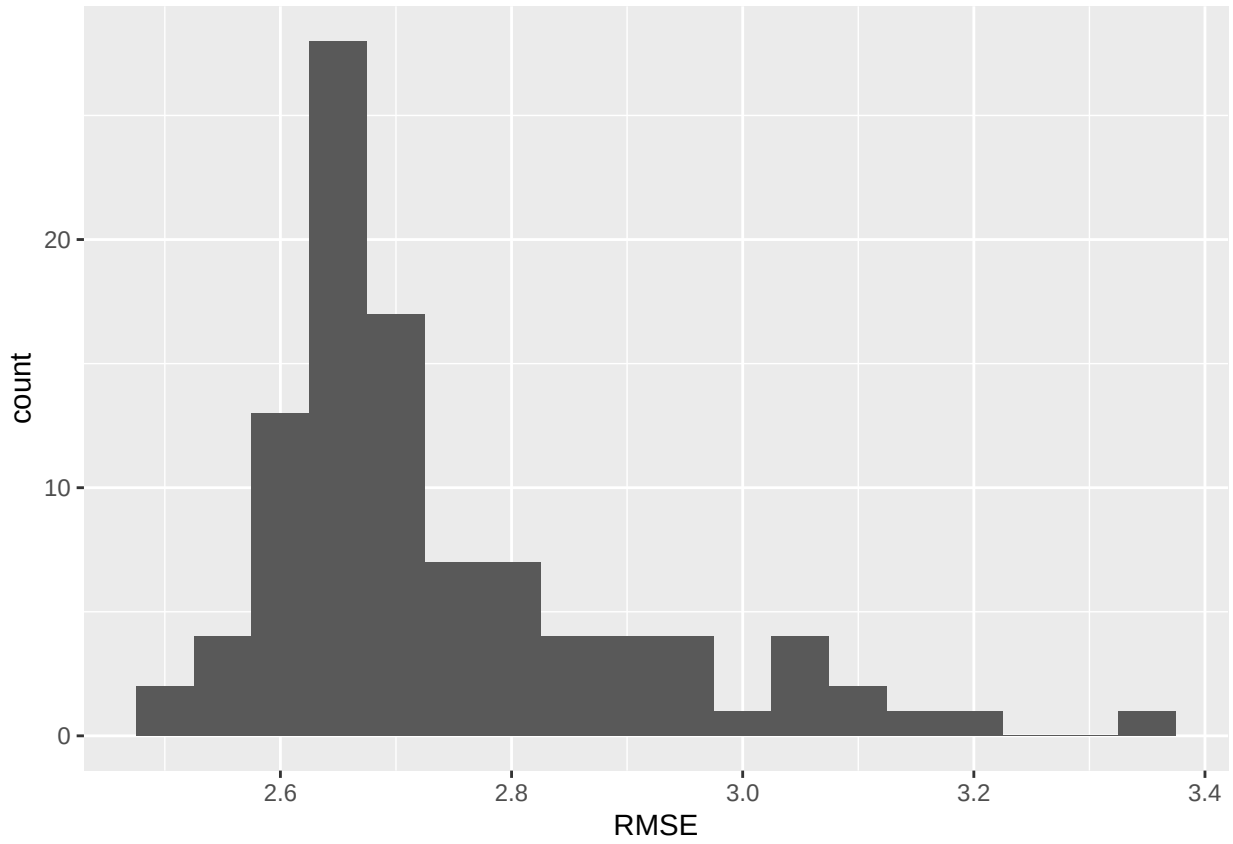

```
ggplot(data = spls_q_10_n_300, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5))  +
  labs(title = "sPLS")
```



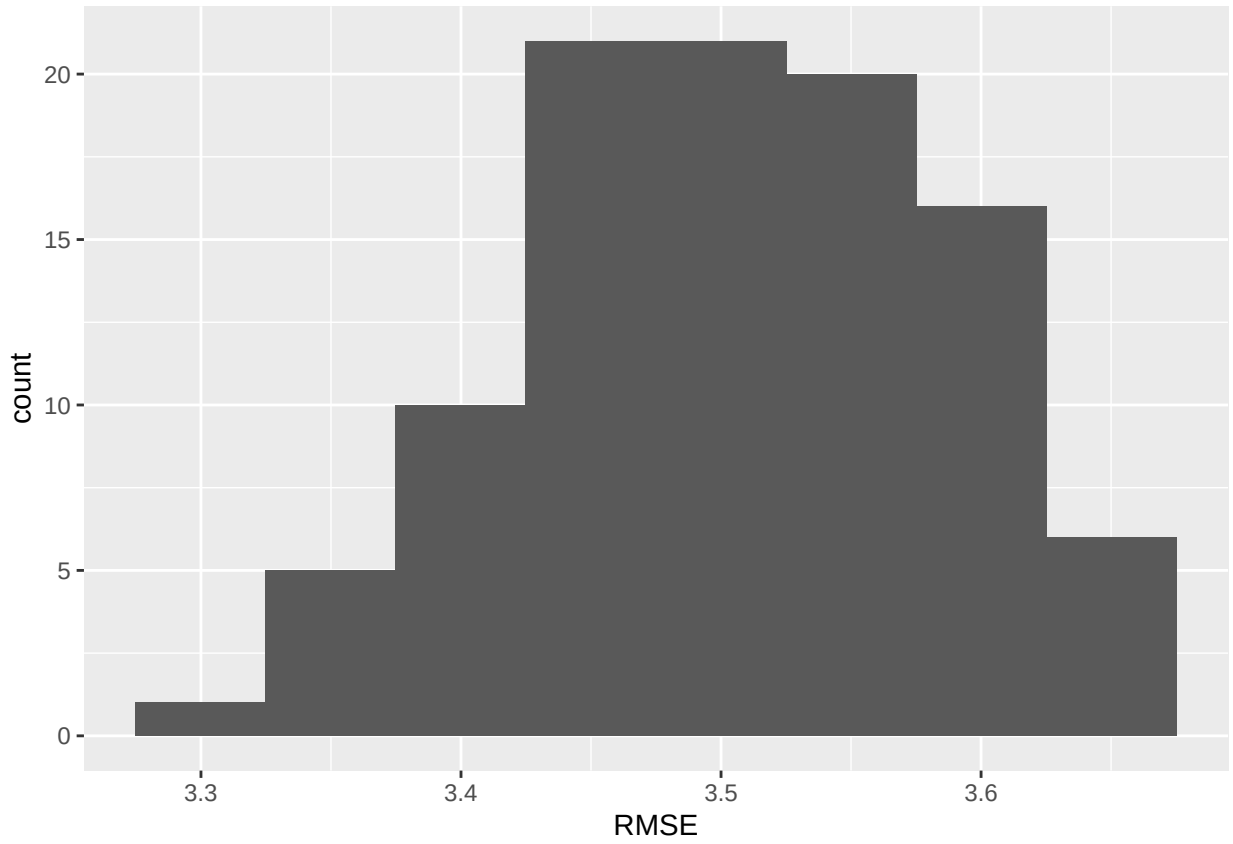

```
ggplot(data = pls_q_10_n_300, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5))  +
  labs(title = "NIPALS-PLS")
```



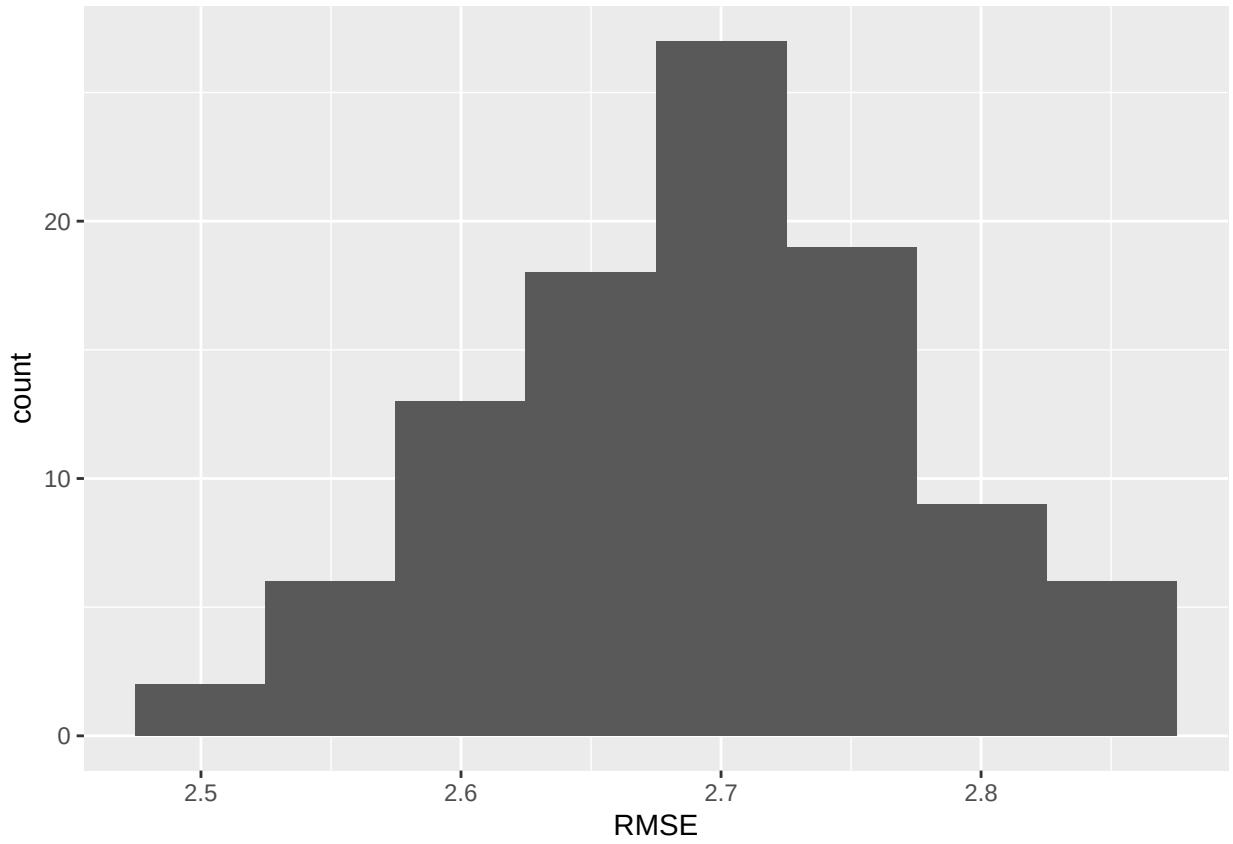

```
ggplot(ddspls_q_10_n_300, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```




```
ggplot(spls_q_10_n_300, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```



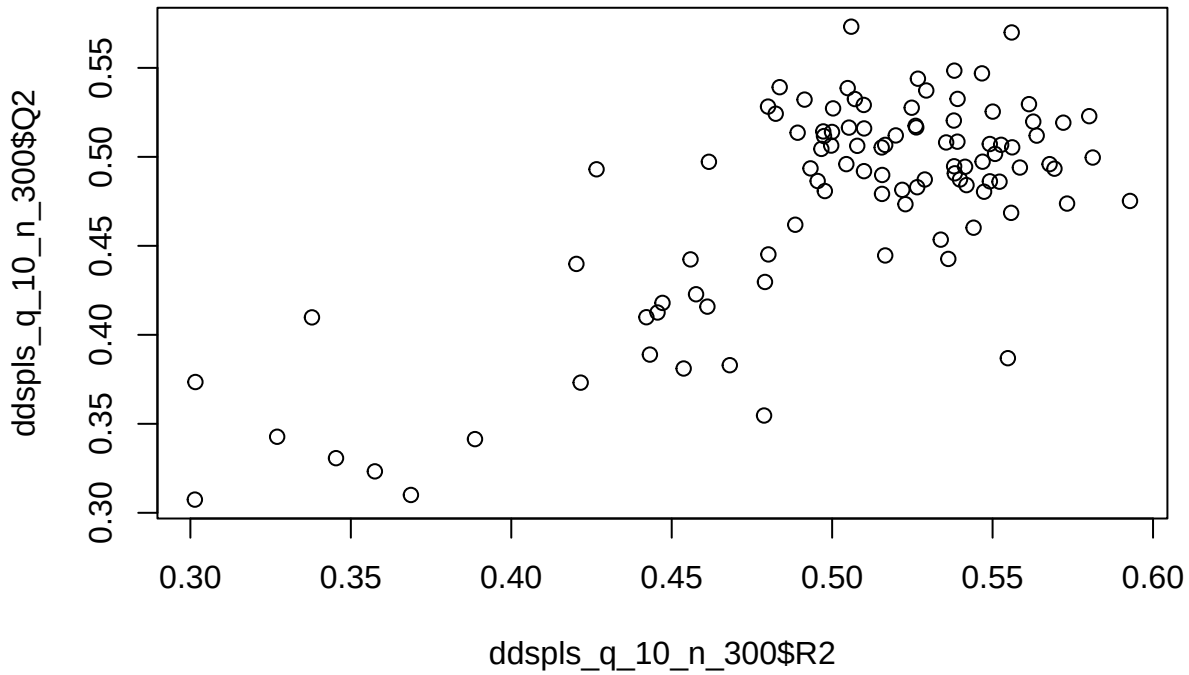

```
ggplot(pls_q_10_n_300, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.05)
```



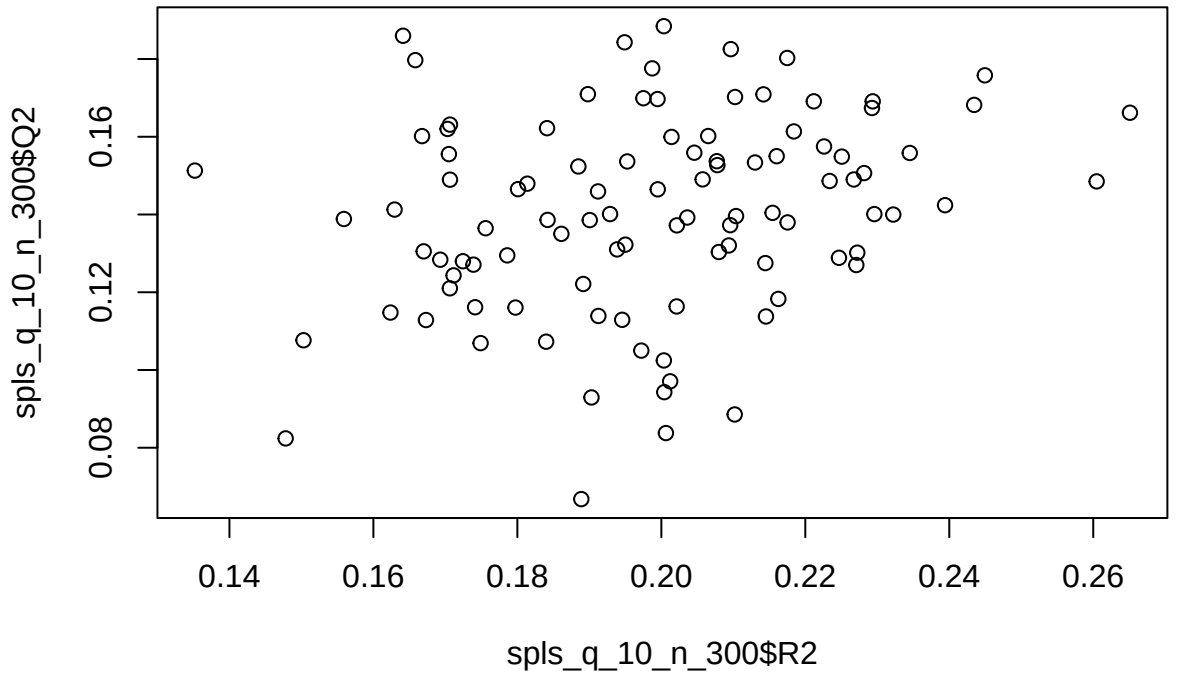

```
q_results_10_n_300 <- as.data.frame(rbind(c(mean(ddspls_q_10_n_300$RMSE), median(ddspls_q_10_n_300$RMSE)
      c(mean(spls_q_10_n_300$RMSE), median(spls_q_10_n_300$RMSE), var(spls_q_10_n_300$RMSE)),
      c(mean(pls_q_10_n_300$RMSE), median(pls_q_10_n_300$RMSE), var(pls_q_10_n_300$RMSE))))
q_results_10_n_300
```

```
##         V1       V2          V3
## 1 2.737081 2.683921 0.025912172
## 2 3.506891 3.511247 0.006376875
## 3 2.691004 2.693719 0.006110431
```

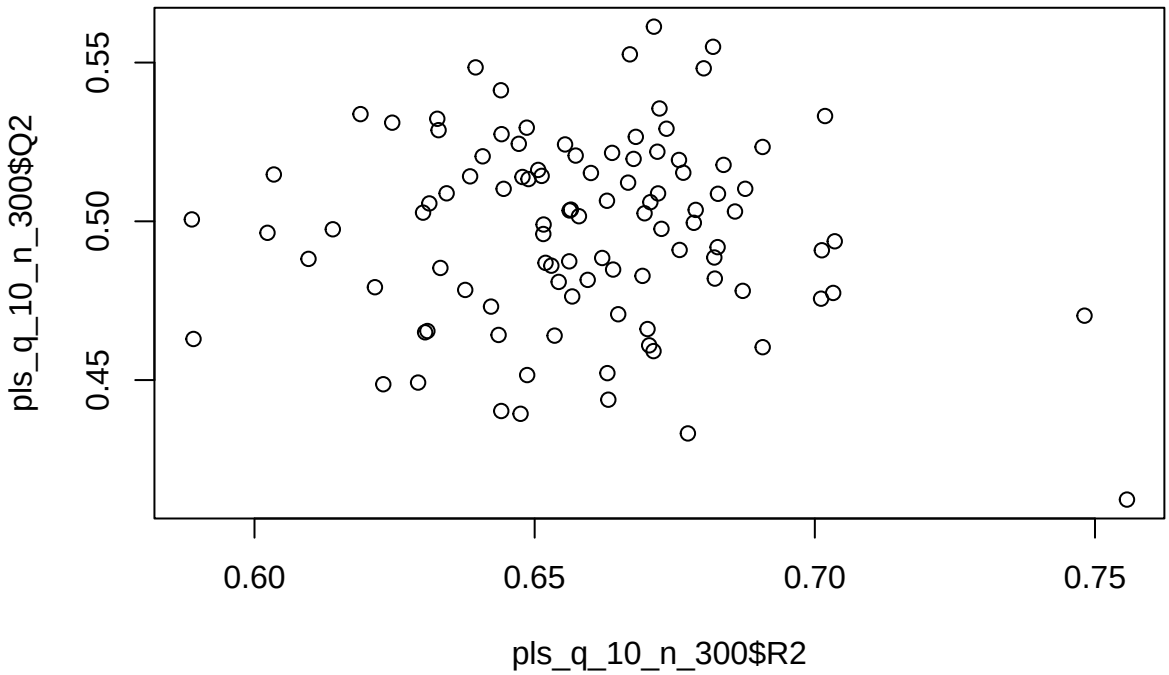
```
plot(ddspls_q_10_n_300$R2, ddspls_q_10_n_300$Q2)
```




```
plot(spls_q_10_n_300$R2, spls_q_10_n_300$Q2)
```




```
plot(pls_q_10_n_300$R2, pls_q_10_n_300$Q2)
```




```
cor(ddspls_q_10_n_300$R2, ddspls_q_10_n_300$Q2)
```

```
## [1] 0.7296526
```

```
cor(spls_q_10_n_300$R2, spls_q_10_n_300$Q2)
```

```
## [1] 0.2539045
```

```
cor(pls_q_10_n_300$R2, pls_q_10_n_300$Q2)
```

```
## [1] -0.07694359
```

Results with a larger sample size aren't significantly different.