SImulations 4

Harpeth Lee

4/12/2022

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

New Data Simulation Method

```r
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)]
```

```r
  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

```r
pls_test <- function(n, sim, func, passed_arg, criterion = "diffR2Q2") {
   # 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, criterion = criterion)

     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 Test Function

```
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 = "comp

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

}
```

Predictors Tests

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

   # Randomly simulates data
```

```
  sim <- sim_data(n = n, p = p, q = q, noise_weight = noise_weight, noise_type = "rnorm", struc = stru

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

```
p_test_ddspls <- apply(matrix(seq(from = 50, to = 1000, by = 25),
                              nrow = 1),
                       MARGIN = 2,
                       p_eval,
                       D_method = "complex")
```

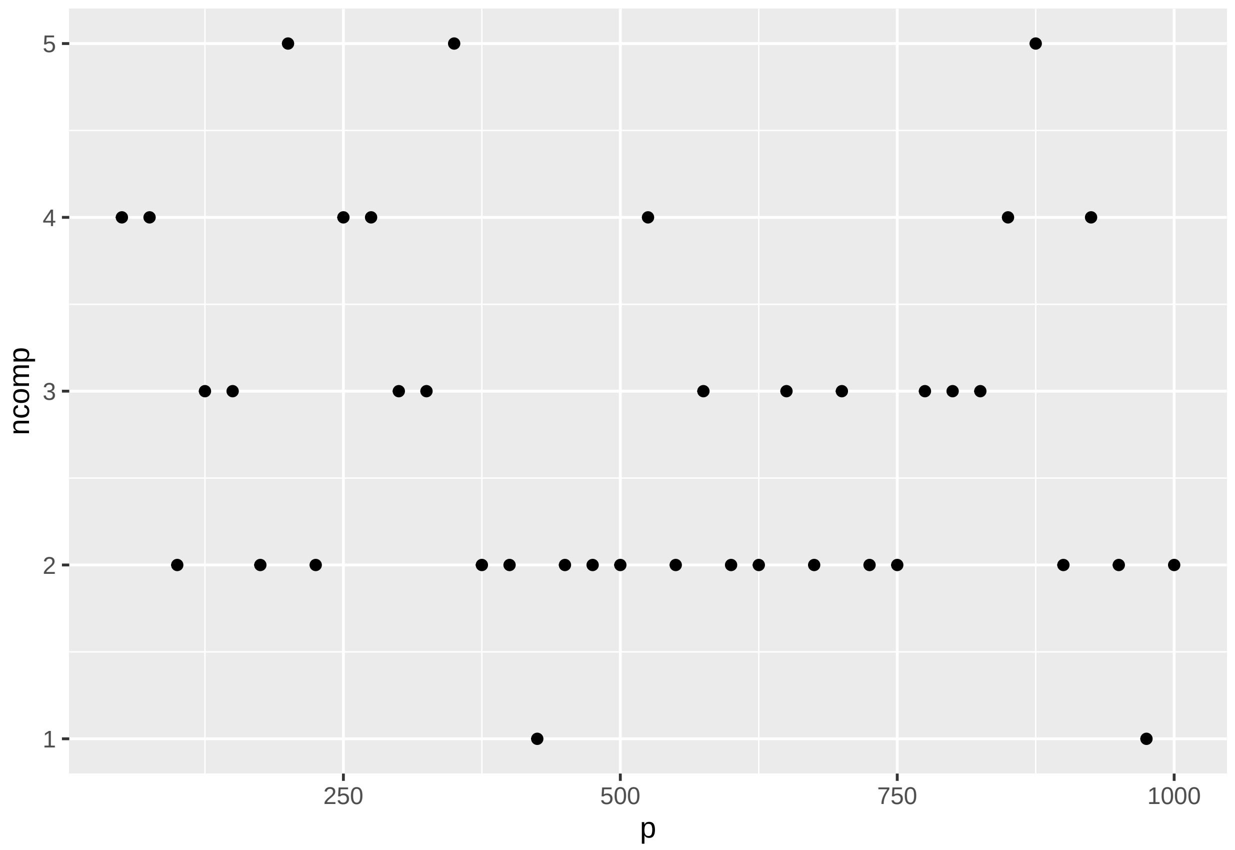
```
p_test_spls <- apply(matrix(seq(from = 50, to = 1000, by = 25),
                            nrow = 1),
                     MARGIN = 2,
                     p_eval,
                     func = "spls",
                     D_method = "complex")
```

```
p_test_pls <- apply(matrix(seq(from = 50, to = 1000, by = 25),
                           nrow = 1),
                    MARGIN = 2,
                    p_eval,
                    func = "pls",
                    D_method = "complex")
```

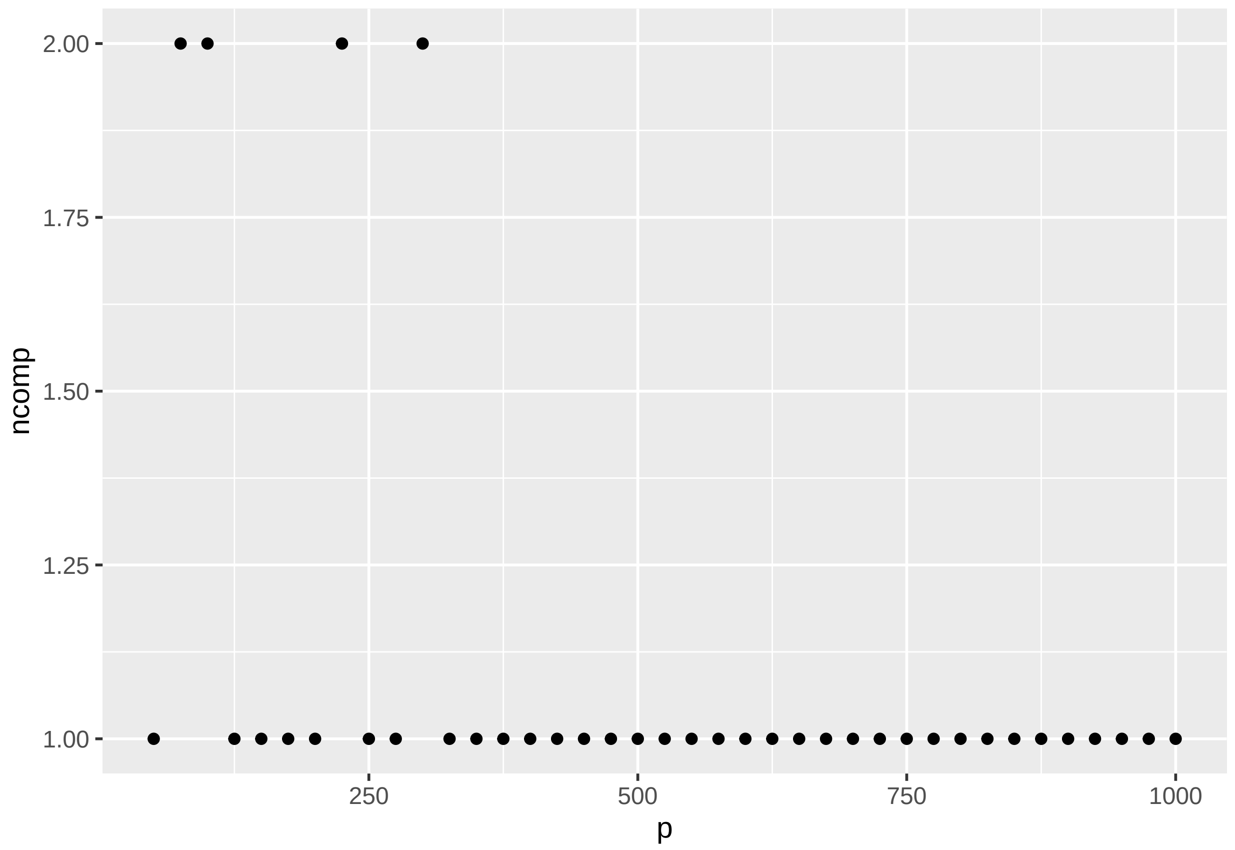
```
p_test_ddspls_Q2 <- apply(matrix(seq(from = 50, to = 1000, by = 25),
                                 nrow = 1),
                          MARGIN = 2,
                          p_eval,
                          D_method = "complex",
                          criterion = "Q2")
```

```
p_test_ddspls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/p_test_ddspls.csv")
p_test_spls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/p_test_spls.csv")
p_test_pls <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/p_test_pls.csv")
p_test_ddspls_Q2 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/p_test_ddspls_Q2.csv"
```

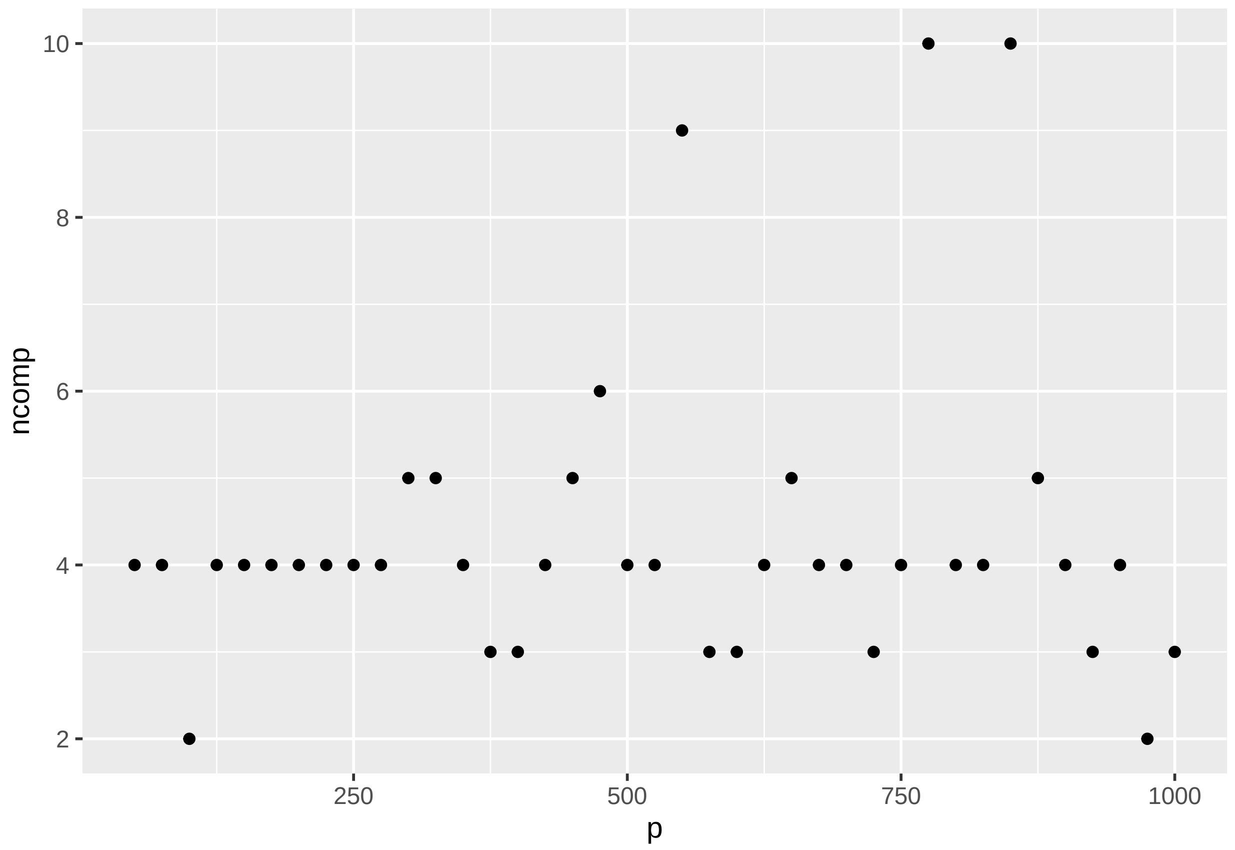
```
ggplot(p_test_ddspls, mapping = aes(x = p, y = ncomp)) +
  geom_point()
```



```
ggplot(p_test_spls, mapping = aes(x = p, y = ncomp)) +
  geom_point()
```

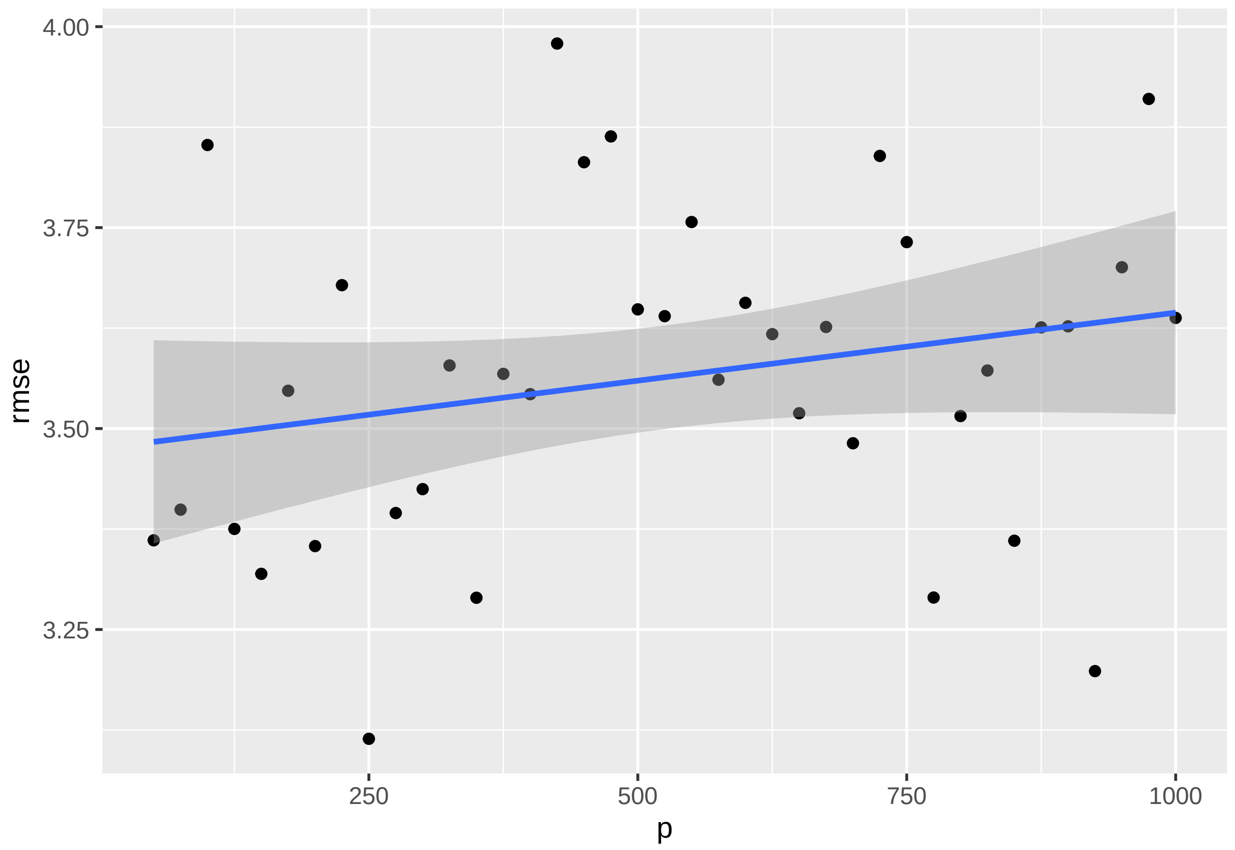


```
ggplot(p_test_pls, mapping = aes(x = p, y = ncomp)) +
  geom_point()
```



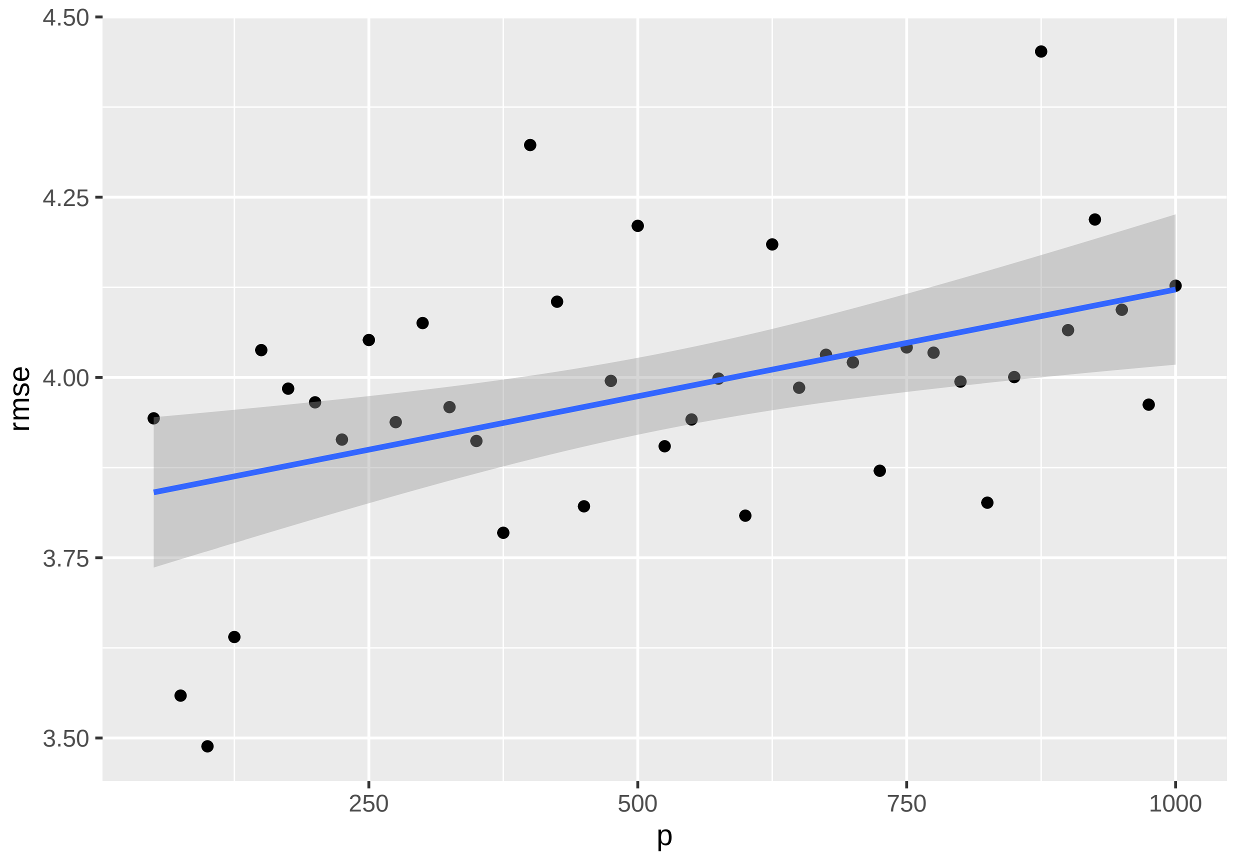
```
ggplot(p_test_ddspls, mapping = aes(x = p, y = rmse)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



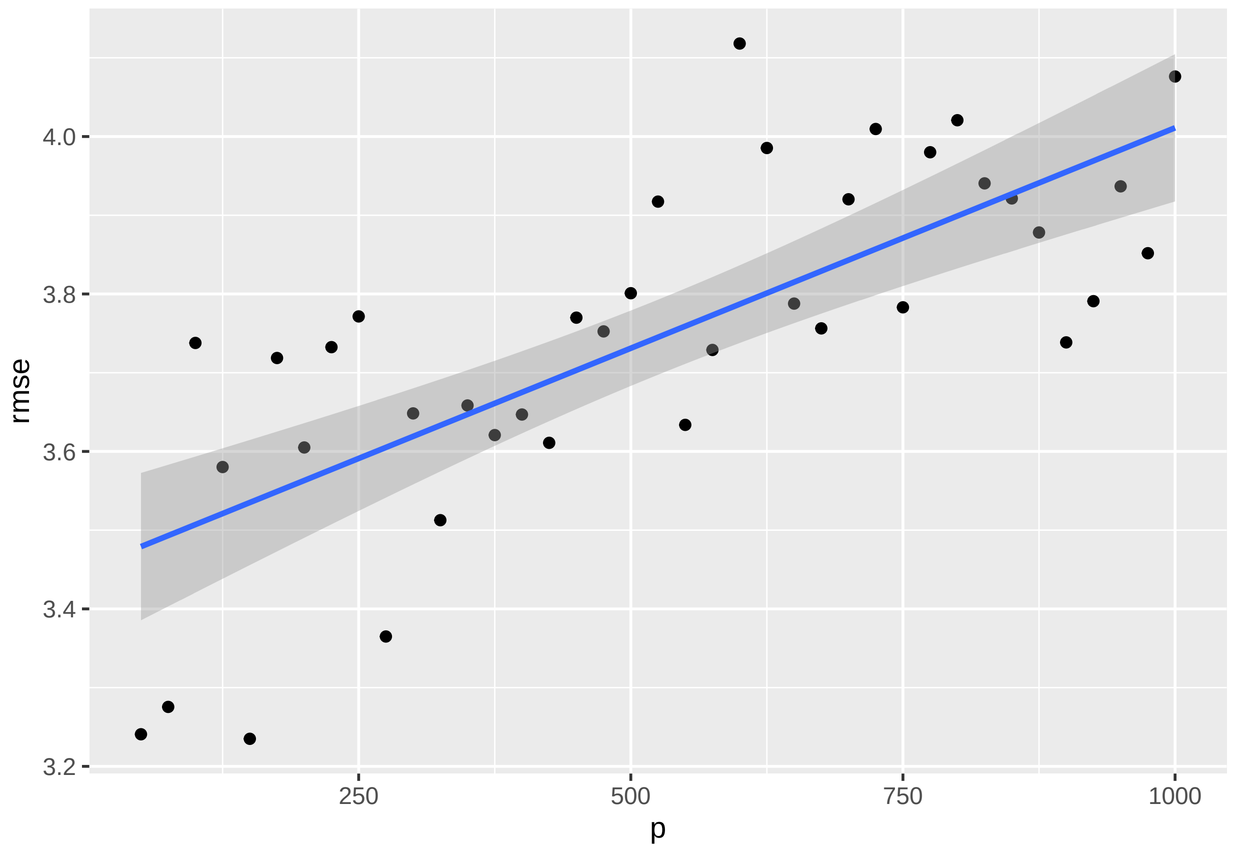
```
ggplot(p_test_spls, mapping = aes(x = p, y = rmse)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



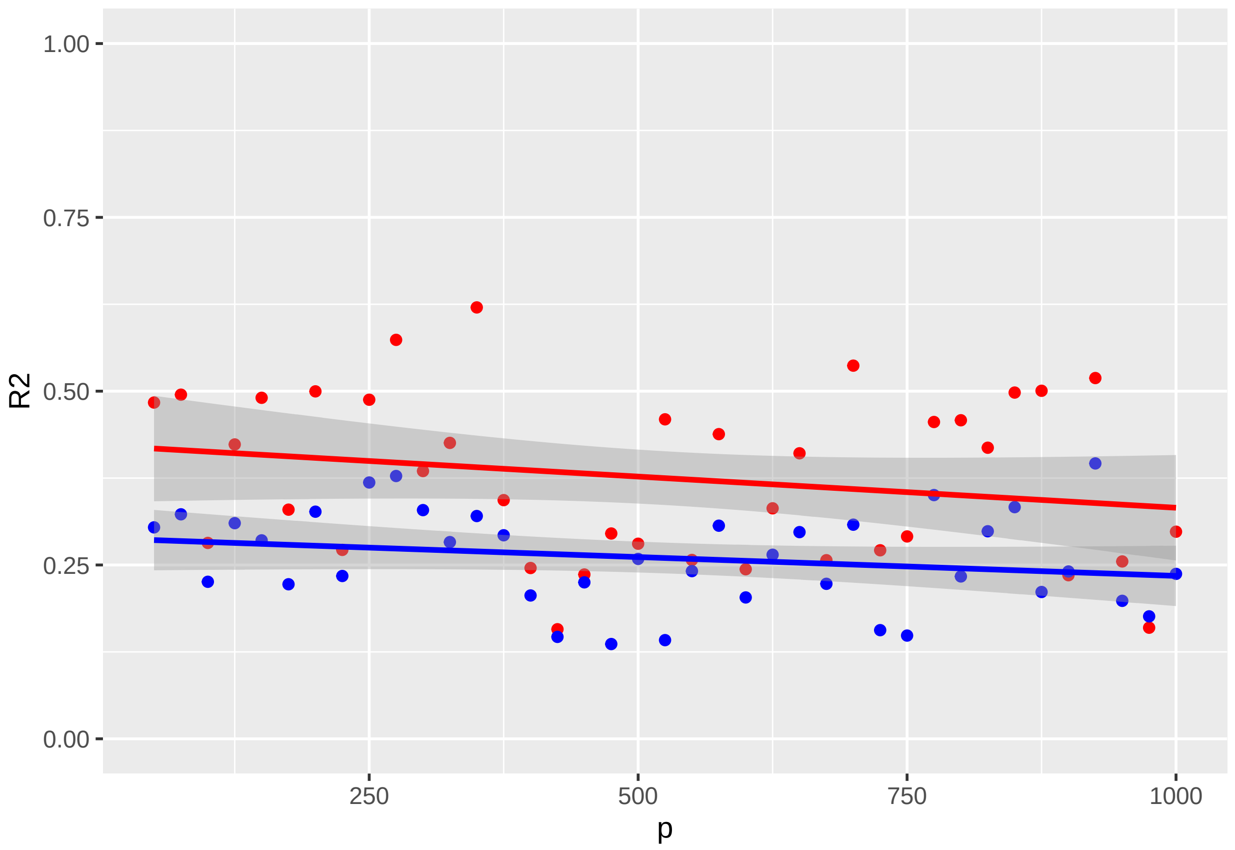
```
ggplot(p_test_pls, mapping = aes(x = p, y = rmse)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



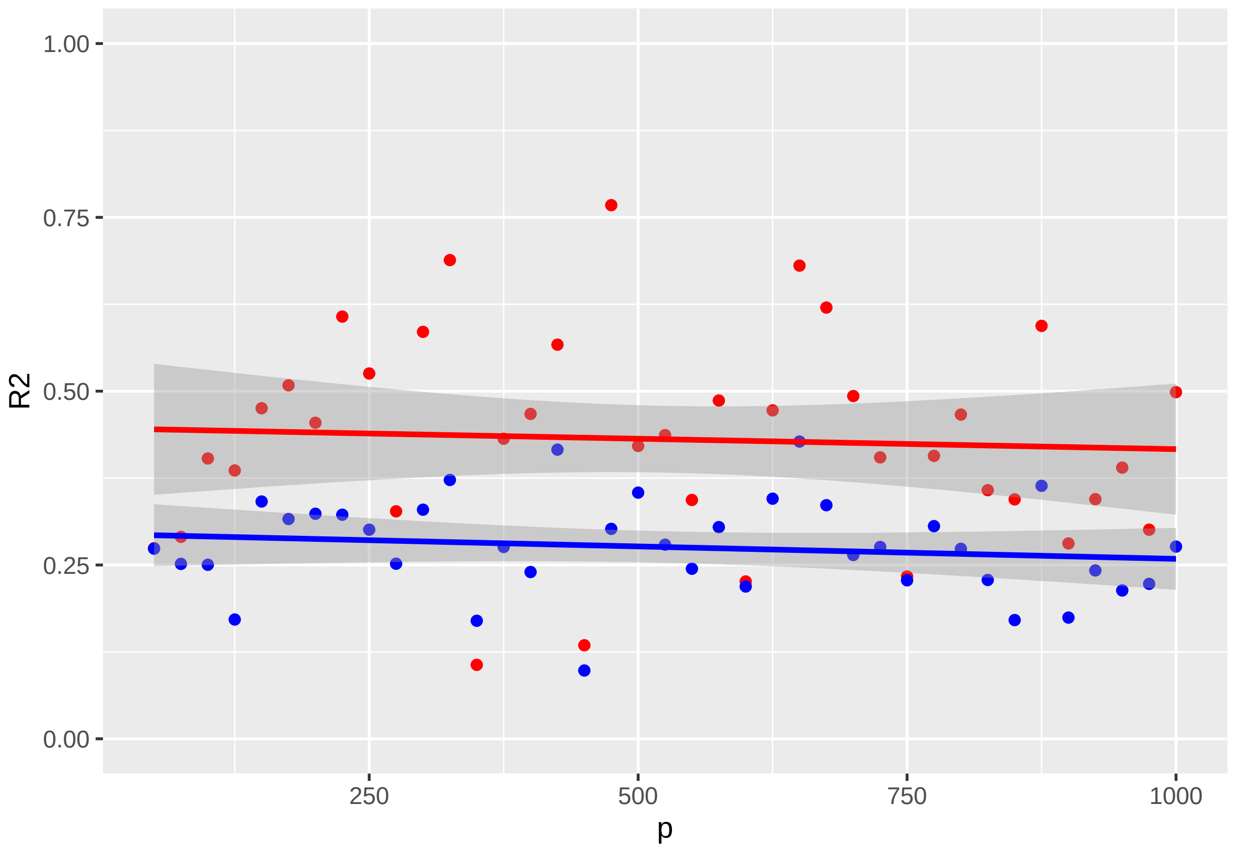
```r
ggplot(p_test_ddspls, mapping = aes(x = p, y = R2)) +
  geom_point(color = "red") +
  geom_point(mapping = aes(x = p, y = Q2), color = "blue") +
  geom_smooth(method = "lm", color = "red") +
  geom_smooth(mapping = aes(x = p, y = Q2), method = "lm", color = "blue") +
  ylim(c(0, 1))
```

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



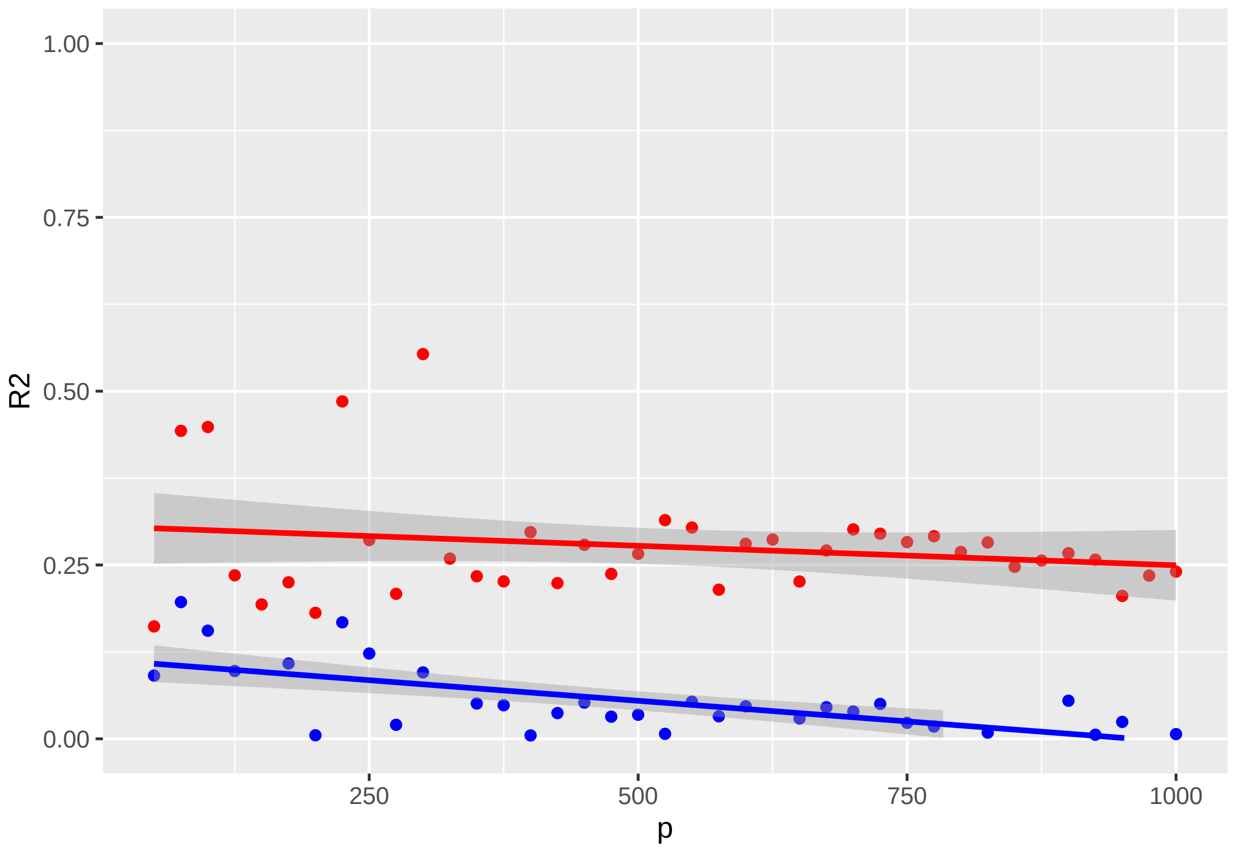
```r
ggplot(p_test_ddspls_Q2, mapping = aes(x = p, y = R2)) +
  geom_point(color = "red") +
  geom_point(mapping = aes(x = p, y = Q2), color = "blue") +
  geom_smooth(method = "lm", color = "red") +
  geom_smooth(mapping = aes(x = p, y = Q2), method = "lm", color = "blue") +
  ylim(c(0, 1))
```

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



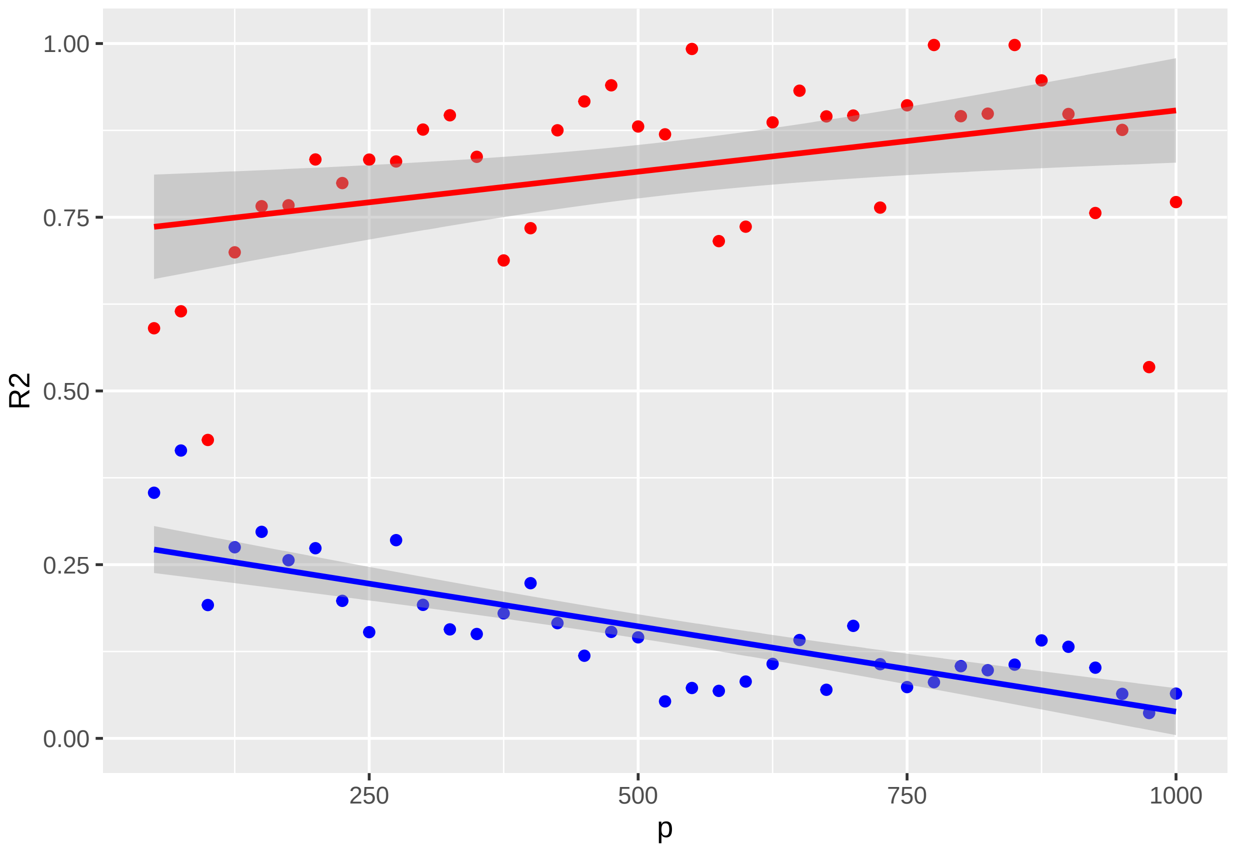
```r
ggplot(p_test_spls, mapping = aes(x = p, y = R2)) +
  geom_point(color = "red") +
  geom_point(mapping = aes(x = p, y = Q2), color = "blue") +
  geom_smooth(method = "lm", color = "red") +
  geom_smooth(mapping = aes(x = p, y = Q2), method = "lm", color = "blue") +
  ylim(c(0, 1))
```

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

## Warning: Removed 7 rows containing non-finite values (stat_smooth).

## Warning: Removed 7 rows containing missing values (geom_point).

## Warning: Removed 4 rows containing missing values (geom_smooth).
```



```
ggplot(p_test_pls, mapping = aes(x = p, y = R2)) +
  geom_point(color = "red") +
  geom_point(mapping = aes(x = p, y = Q2), color = "blue") +
  geom_smooth(method = "lm", color = "red") +
  geom_smooth(mapping = aes(x = p, y = Q2), method = "lm", color = "blue") +
  ylim(c(0, 1))
```

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



```r
ddspls_p_reps <- replicate(100, p_eval(p = 1000, D_method = "complex"))
```

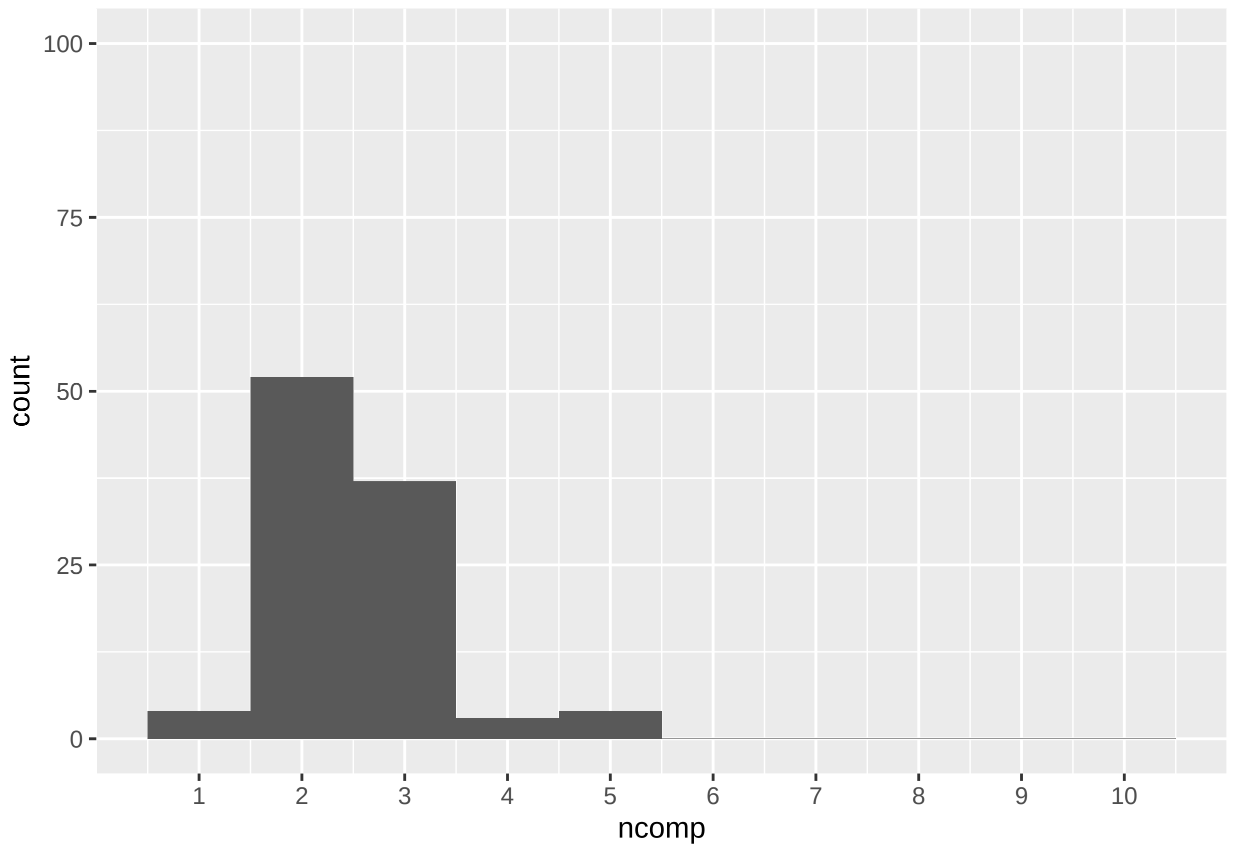
```r
spls_p_reps <- replicate(100, p_eval(p = 1000, D_method = "complex", func = "spls"))
```

```r
ddspls_p_Q2_reps <- replicate(100, p_eval(p = 1000, D_method = "complex", criterion = "Q2"))
```

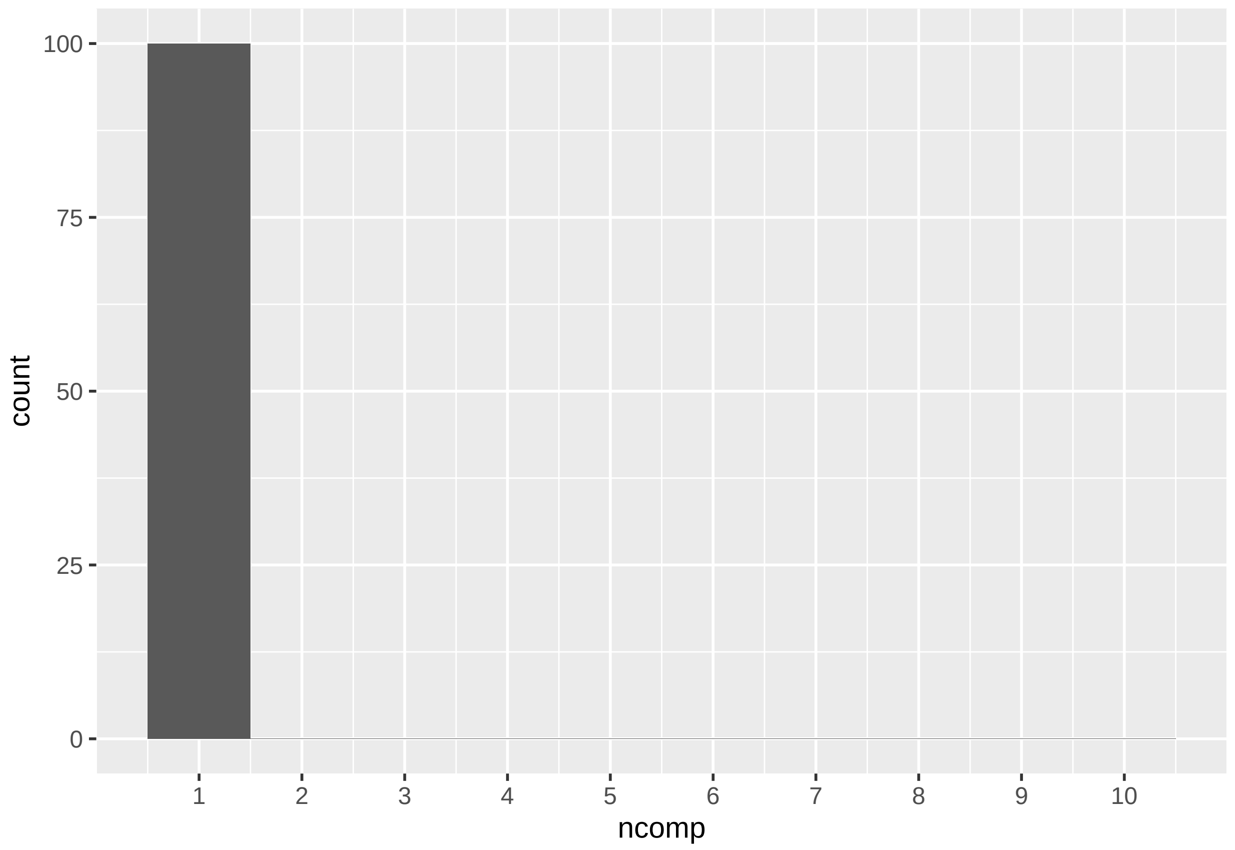
```r
pls_p_reps <- replicate(100, p_eval(p = 1000, D_method = "complex", func = "pls"))
```

```r
ddspls_1000_reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_p_1000.csv")
spls_1000_reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls_p_1000.csv")
pls_1000_reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls_p_1000.csv")
ddspls_1000_reps_Q2 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_p_1000_Q2.c
```

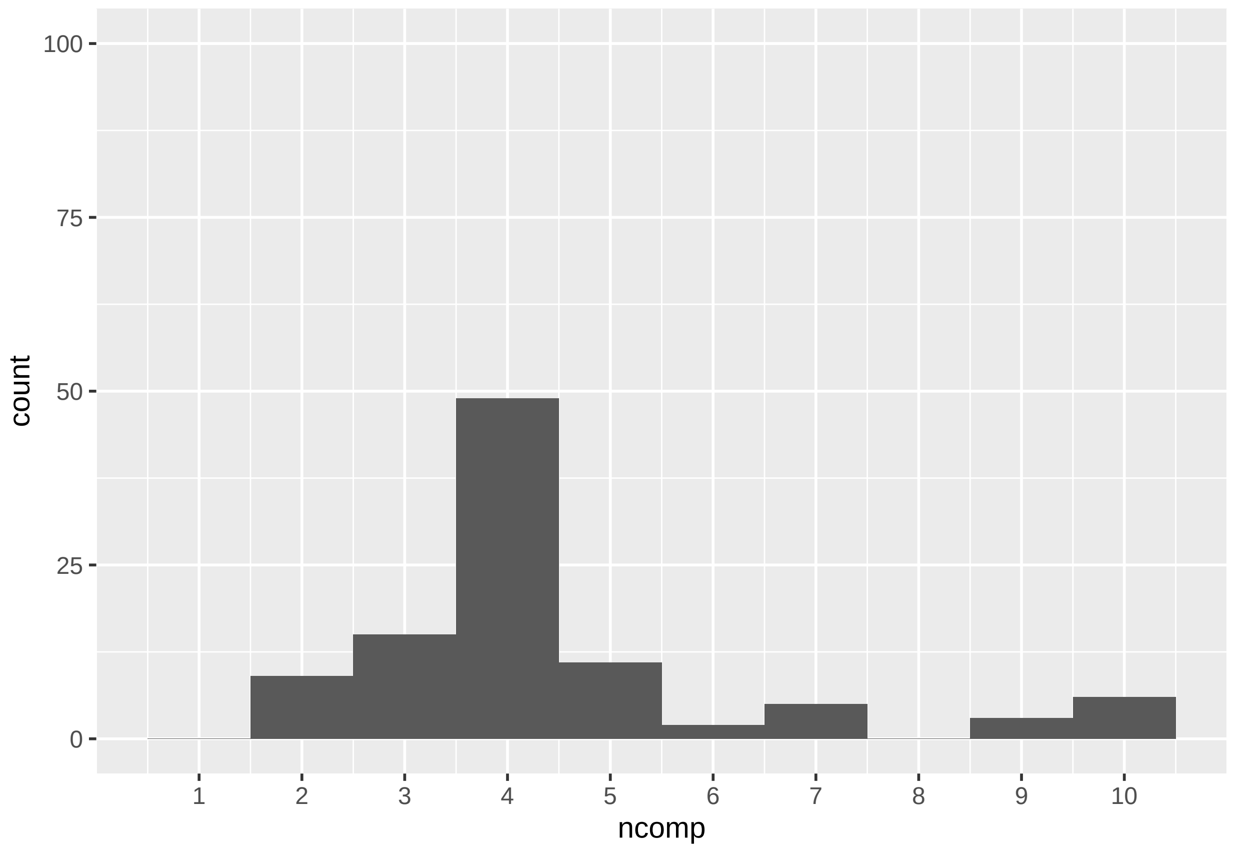
```r
ggplot(ddspls_1000_reps, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5)) +
  ylim(c(0, 100))
```



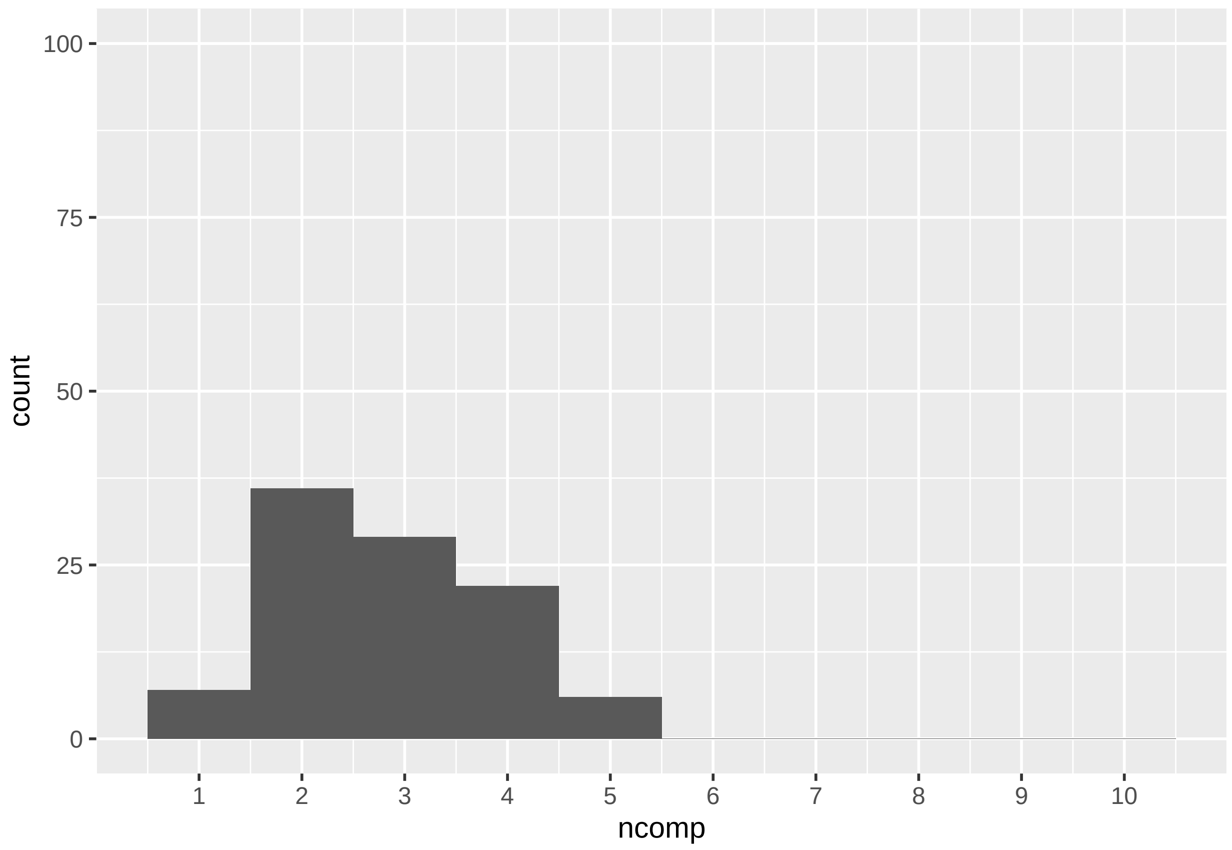
```
ggplot(spls_1000_reps, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5)) +
  ylim(c(0, 100))
```



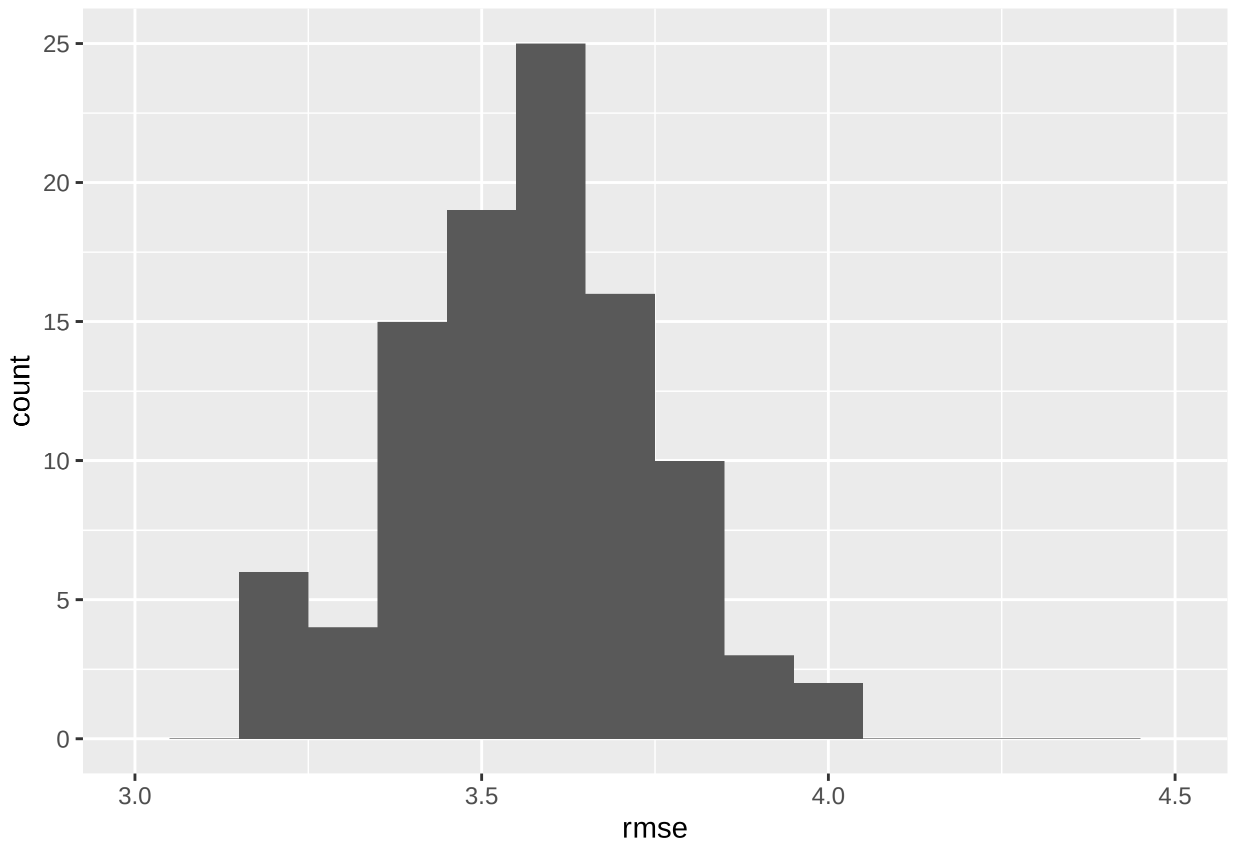
```
ggplot(pls_1000_reps, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5)) +
  ylim(c(0, 100))
```



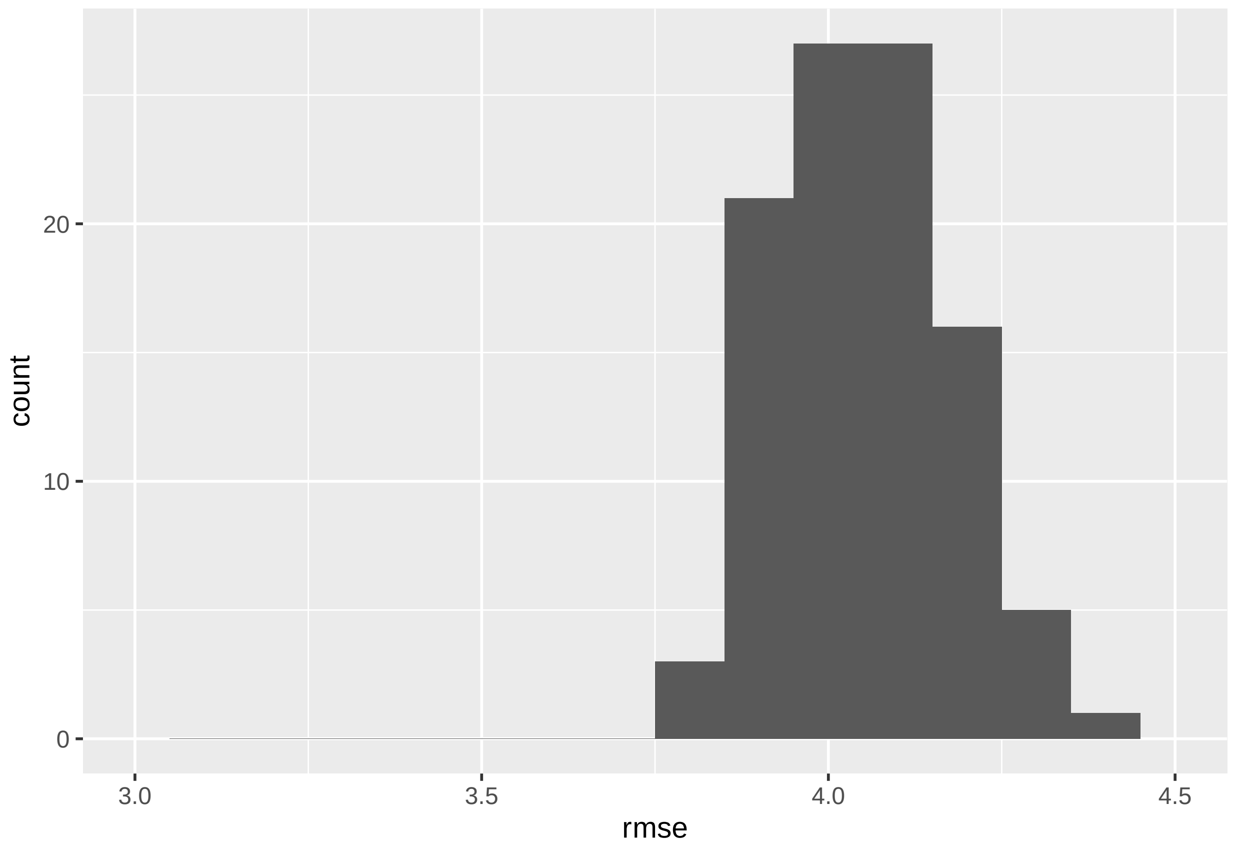
```
ggplot(ddspls_1000_reps_Q2, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5)) +
  ylim(c(0, 100))
```



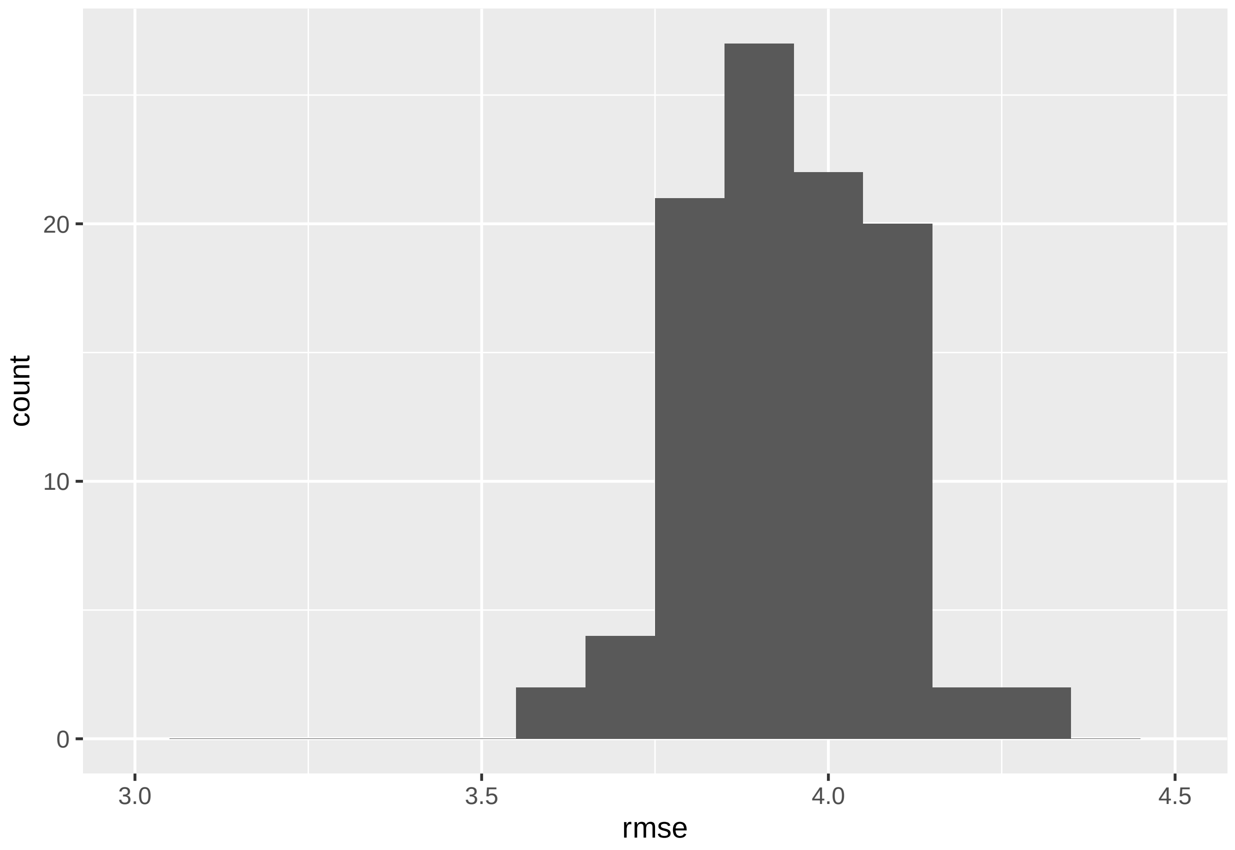
```r
ggplot(ddspls_1000_reps, aes(x = rmse)) +
  geom_histogram(binwidth = 0.1) +
  xlim(c(3, 4.5))
```



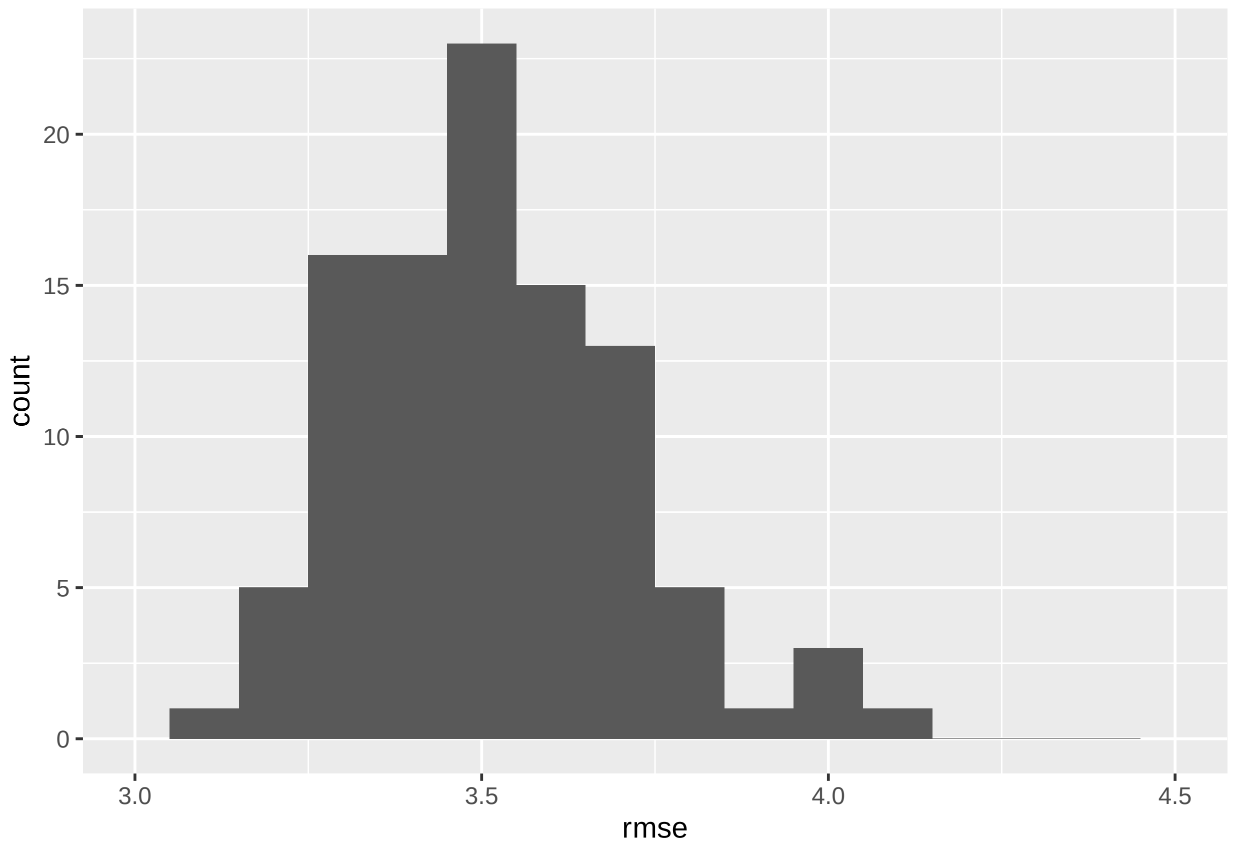
```r
ggplot(spls_1000_reps, aes(x = rmse)) +
  geom_histogram(binwidth = 0.1) +
  xlim(c(3, 4.5))
```



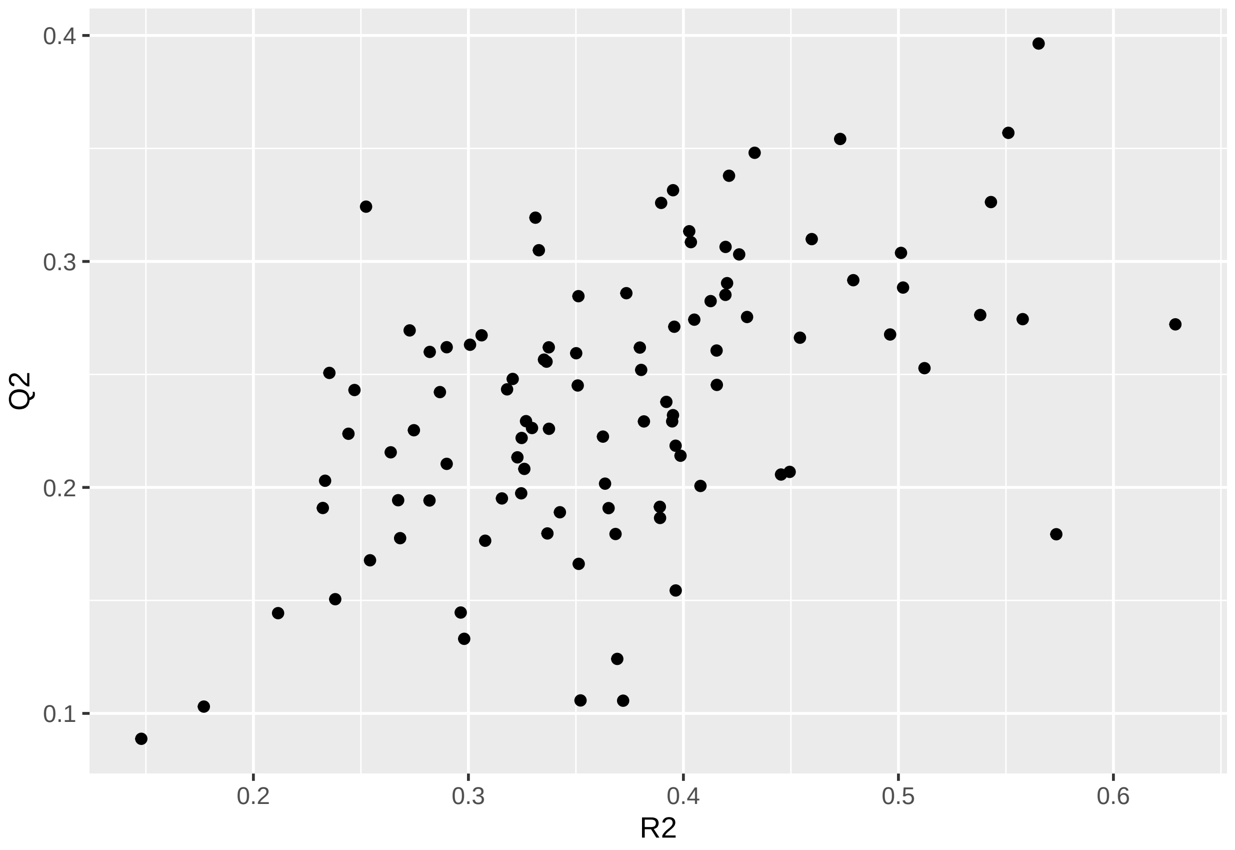
```
ggplot(pls_1000_reps, aes(x = rmse)) +
  geom_histogram(binwidth = 0.1) +
  xlim(c(3, 4.5))
```



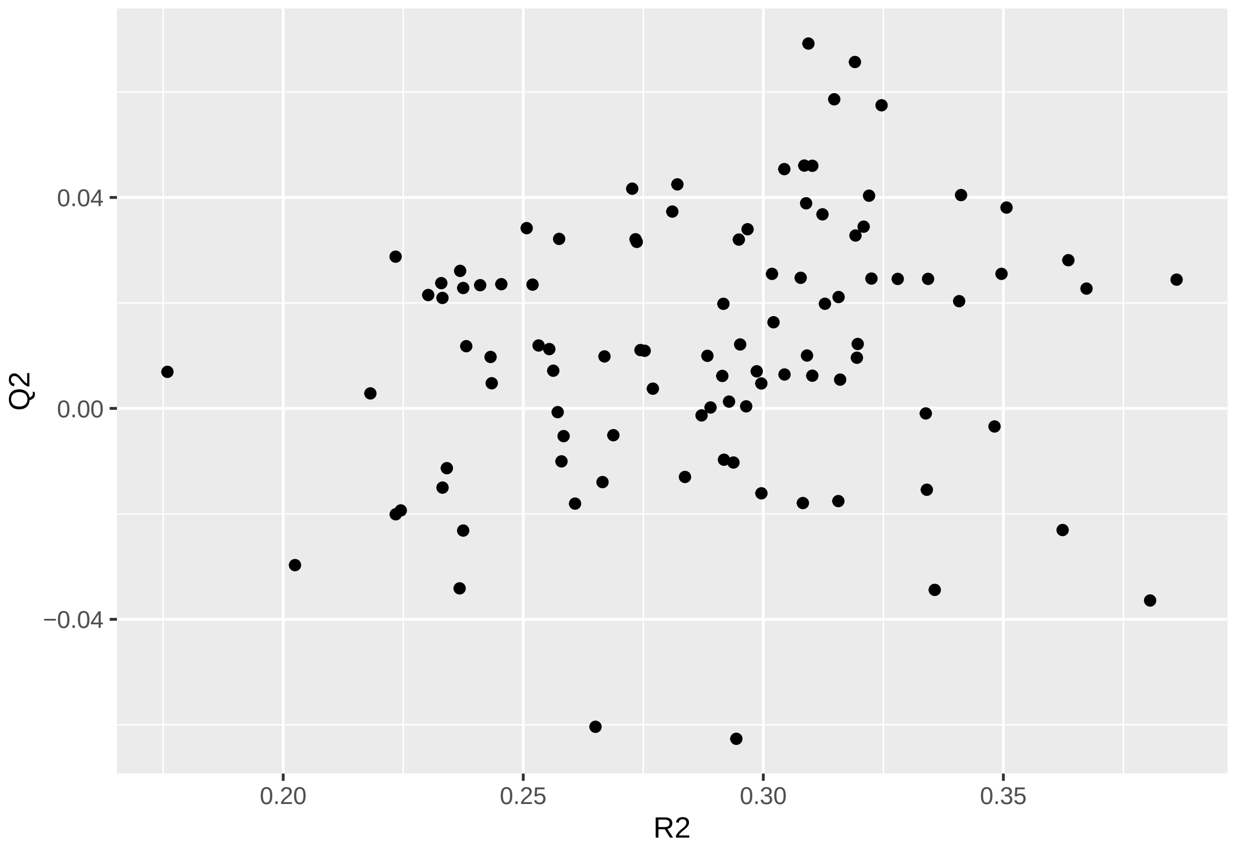
```
ggplot(ddspls_1000_reps_Q2, aes(x = rmse)) +
  geom_histogram(binwidth = 0.1) +
  xlim(c(3, 4.5))
```



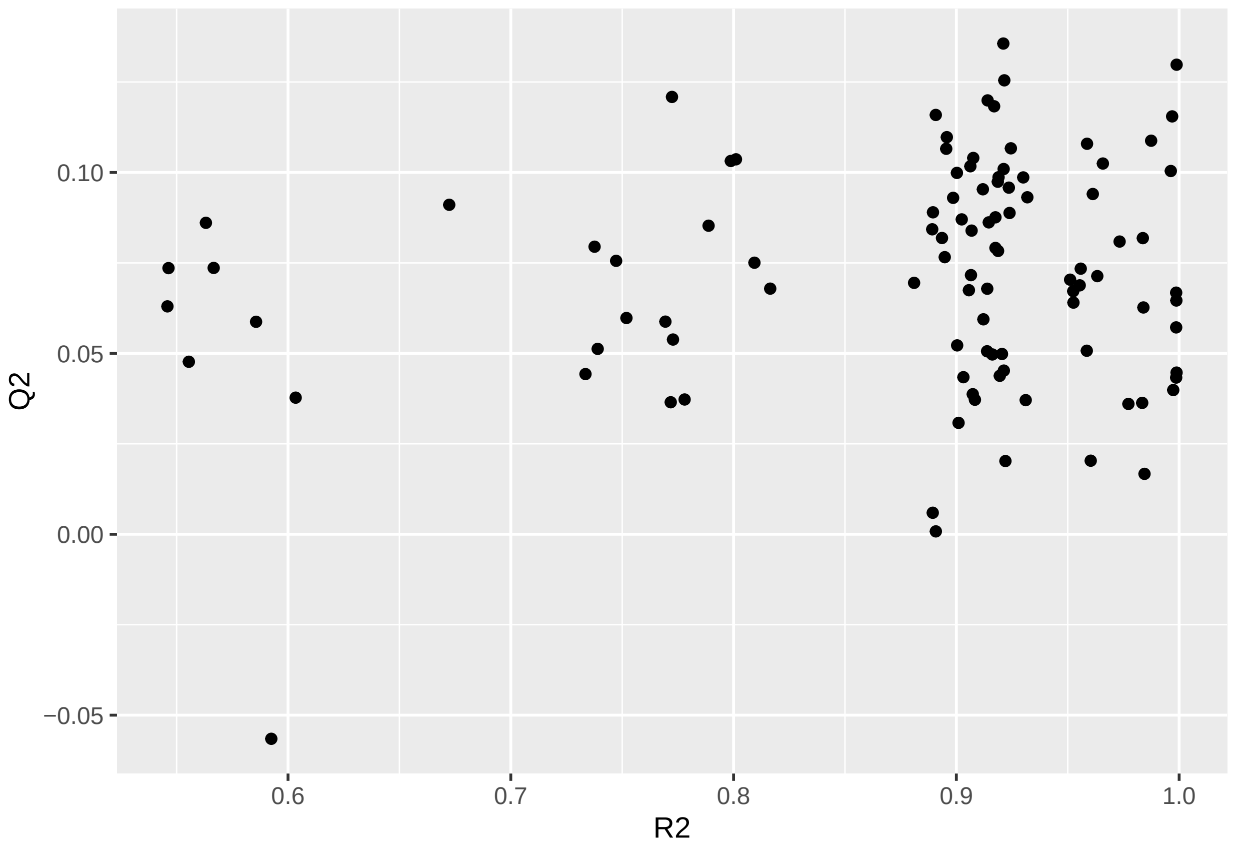
```
ggplot(ddspls_1000_reps, aes(x = R2, y = Q2)) +
  geom_point()
```



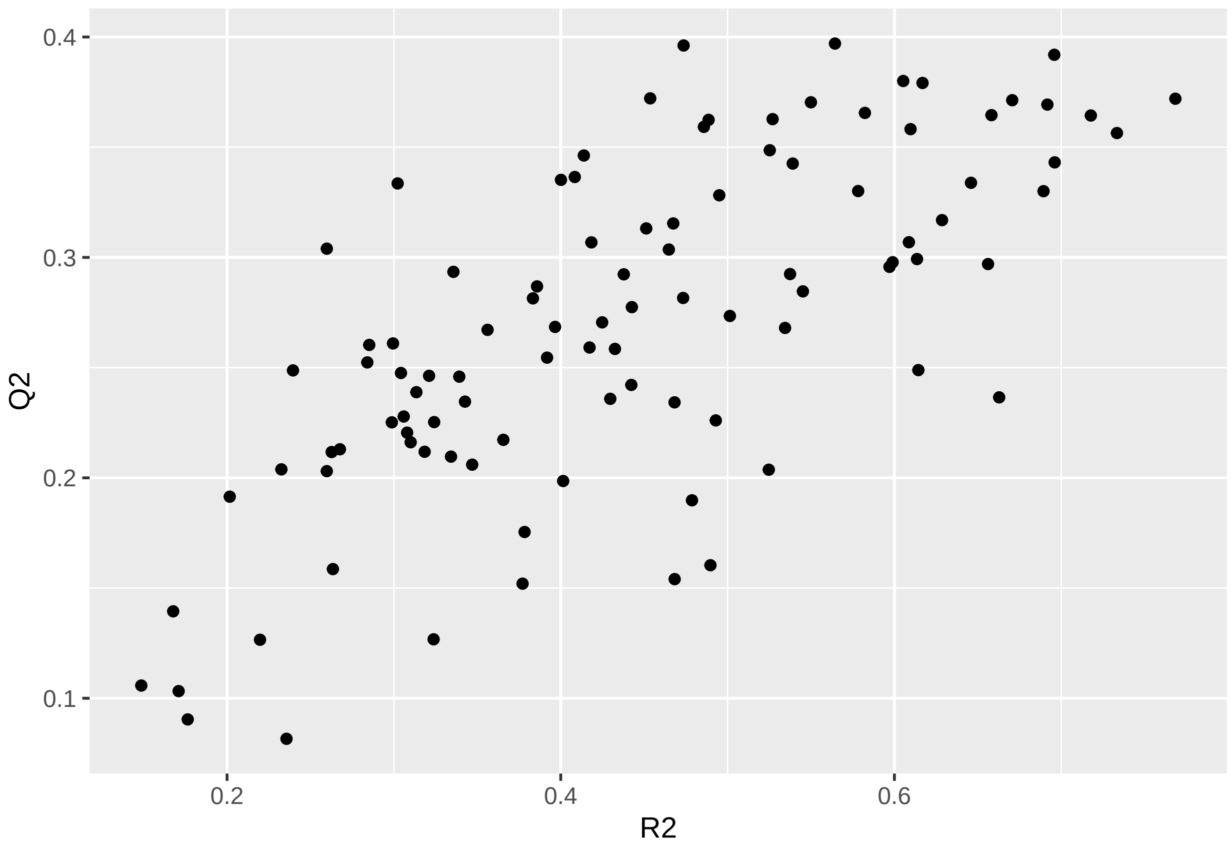
```
ggplot(spls_1000_reps, aes(x = R2, y = Q2)) +
  geom_point()
```



```r
ggplot(pls_1000_reps, aes(x = R2, y = Q2)) +
  geom_point()
```



```
ggplot(ddspls_1000_reps_Q2, aes(x = R2, y = Q2)) +
  geom_point()
```



```
reps_df <- data.frame(c(mean(ddspls_1000_reps$rmse), median(ddspls_1000_reps$rmse), var(ddspls_1000_reps
   c(mean(spls_1000_reps$rmse), median(spls_1000_reps$rmse), var(spls_1000_reps$rmse)),
   c(mean(pls_1000_reps$rmse), median(pls_1000_reps$rmse), var(pls_1000_reps$rmse)),
   c(mean(ddspls_1000_reps_Q2$rmse), median(ddspls_1000_reps_Q2$rmse), var(ddspls_1000_reps_Q2$rmse)))

reps_df <- as.data.frame(t(as.matrix(reps_df)))

colnames(reps_df) <- c("mean", "median", "var")
row.names(reps_df) <- c("ddsPLS", "sPLS", "PLS", "ddsPLS_Q2")

reps_df
```

```
##               mean   median        var
## ddsPLS    3.569282 3.565361 0.02974532
## sPLS      4.052281 4.041317 0.01489767
## PLS       3.944839 3.942986 0.01851942
## ddsPLS_Q2 3.508364 3.492494 0.03847528
```

```
ddspls_p_reps <- replicate(100, p_eval(p = 500, D_method = "complex"))
```

```
spls_p_reps <- replicate(100, p_eval(p = 500, D_method = "complex", func = "spls"))
```

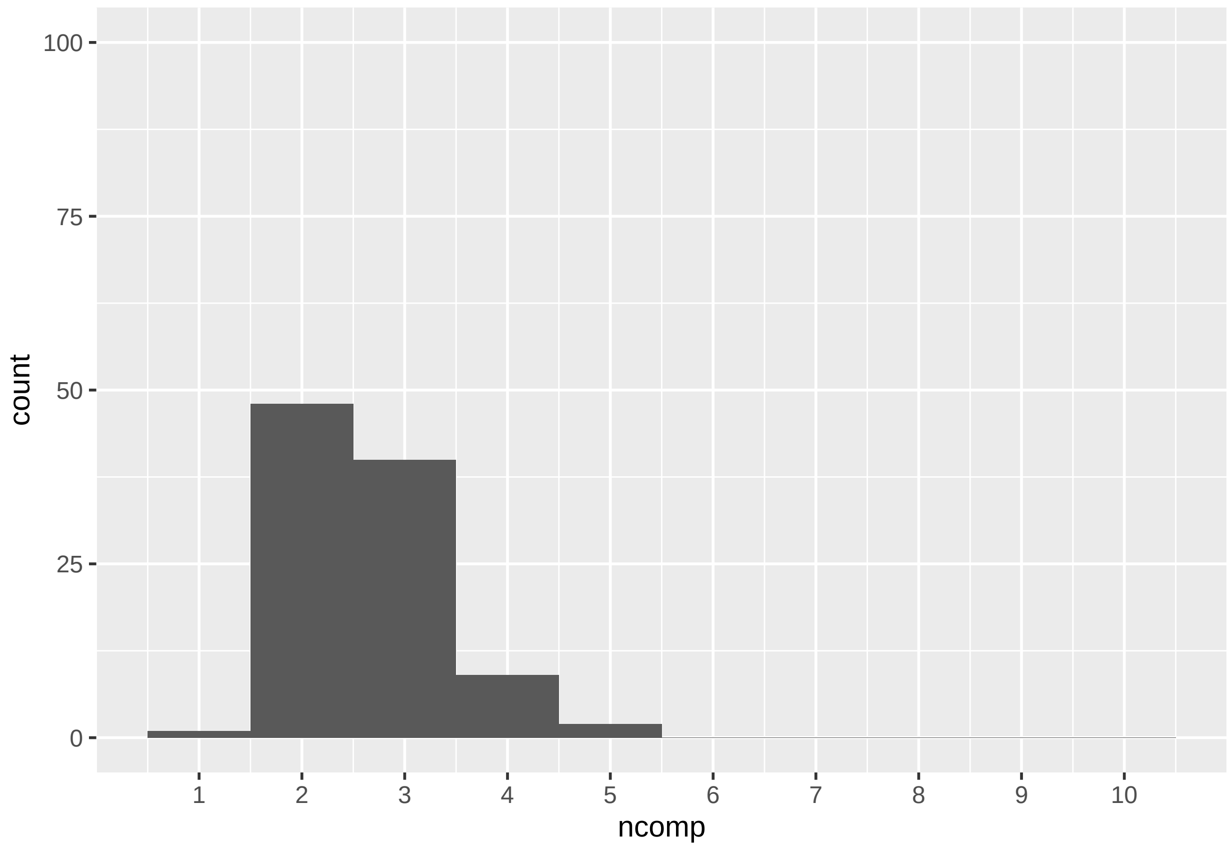
```
ddspls_p_Q2_reps <- replicate(100, p_eval(p = 500, D_method = "complex", criterion = "Q2"))
```

```
pls_p_reps <- replicate(100, p_eval(p = 500, D_method = "complex", func = "pls"))
```

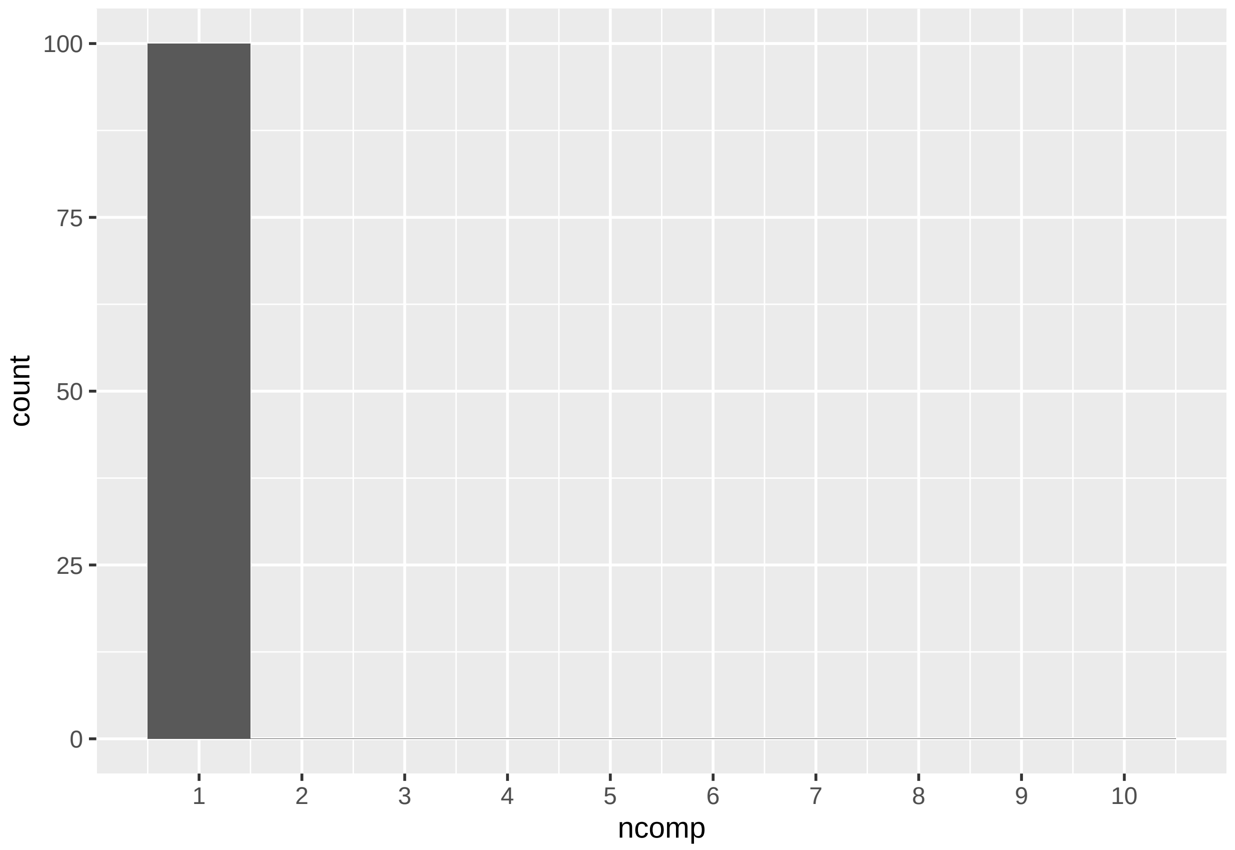
```
ddspls_500_reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_p_500.csv")
spls_500_reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/spls_p_500.csv")
pls_500_reps <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/pls_p_500.csv")
```

```
ddspls_500_reps_Q2 <- read.csv2("/Users/johnlee/R Files/thesis-lee/Simulations/data/ddspls_p_500_Q2.csv
```

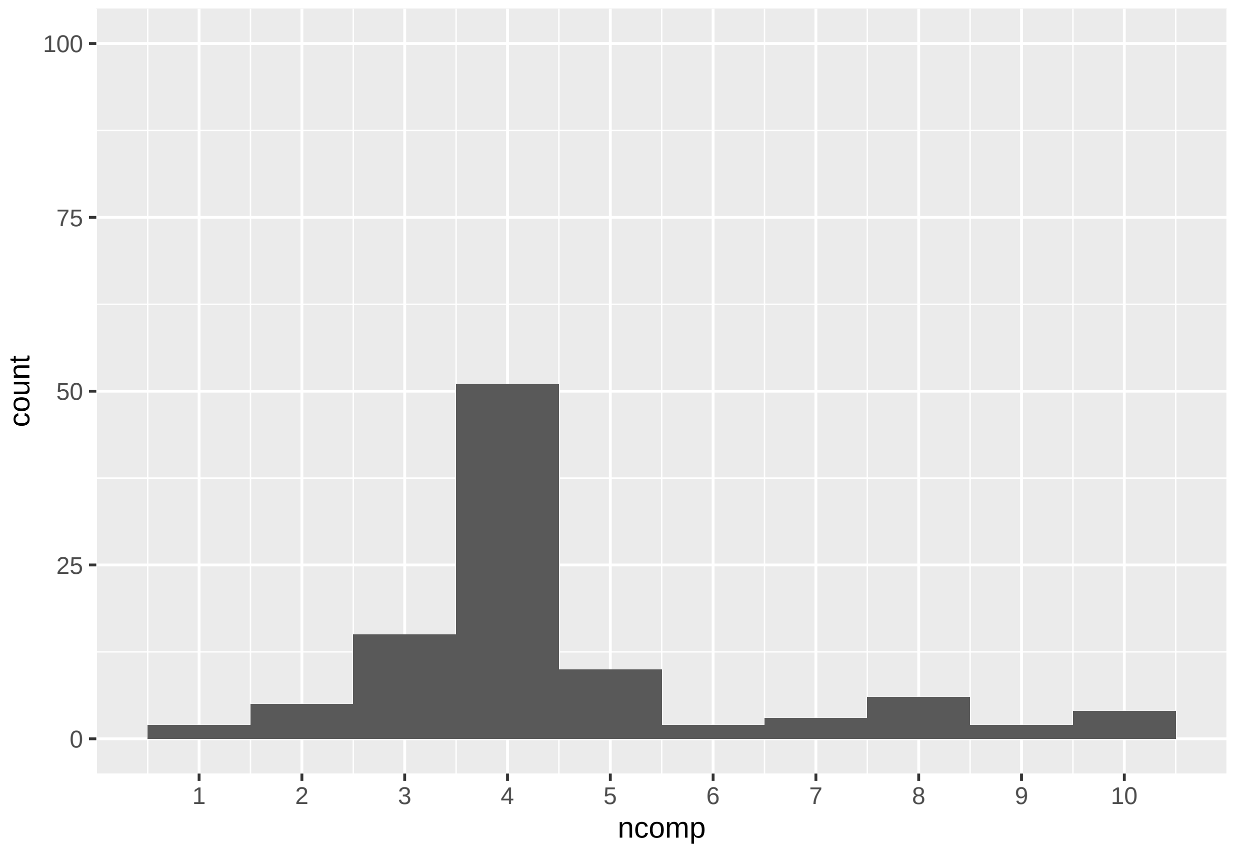
```
ggplot(ddspls_500_reps, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5)) +
  ylim(c(0, 100))
```



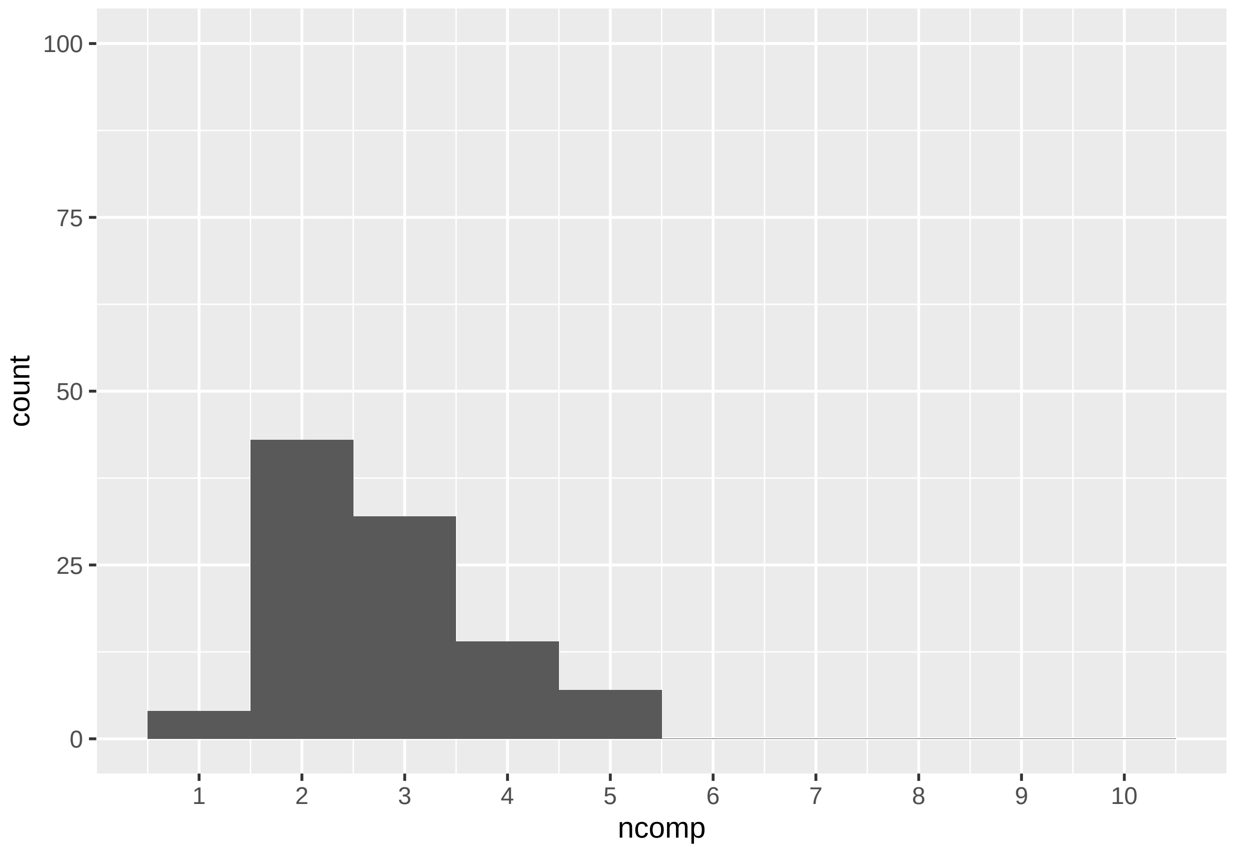
```
ggplot(spls_500_reps, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5)) +
  ylim(c(0, 100))
```



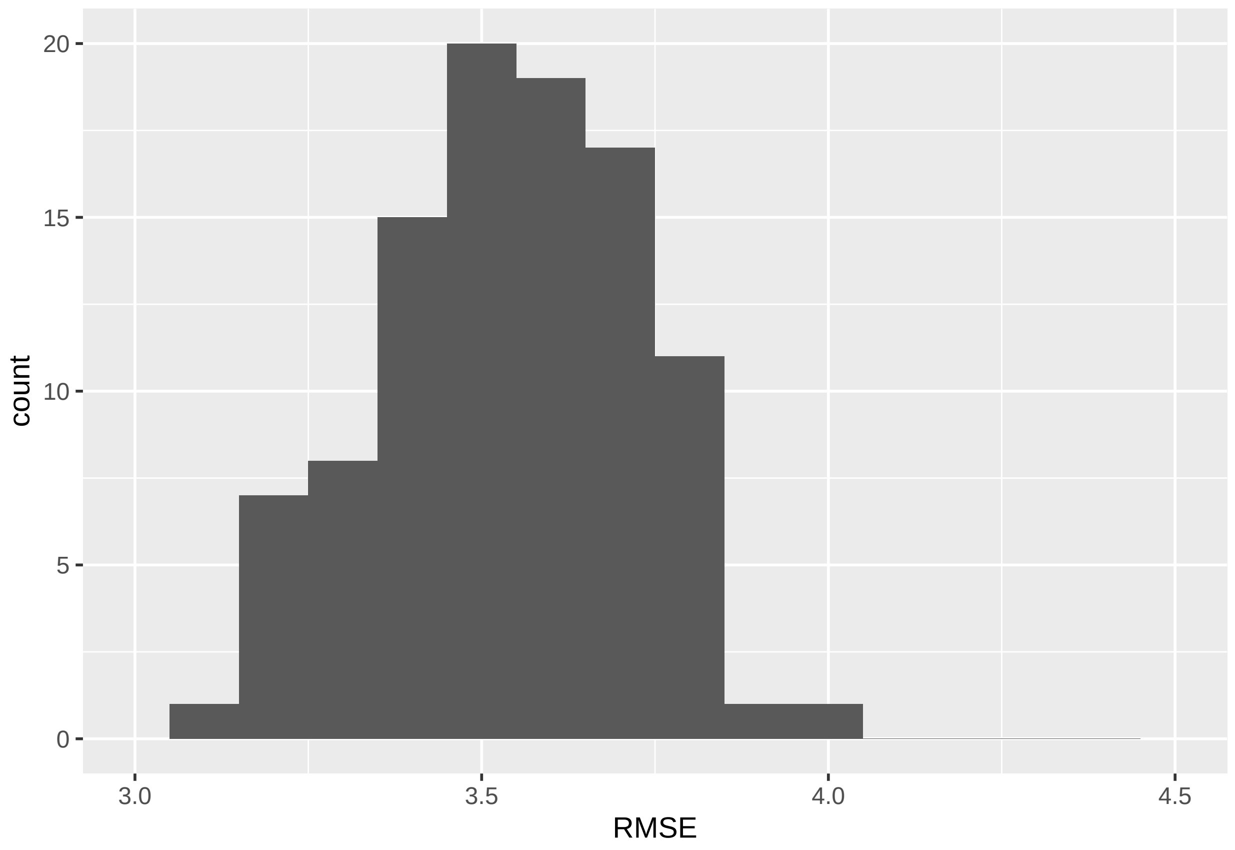
```
ggplot(pls_500_reps, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5)) +
  ylim(c(0, 100))
```



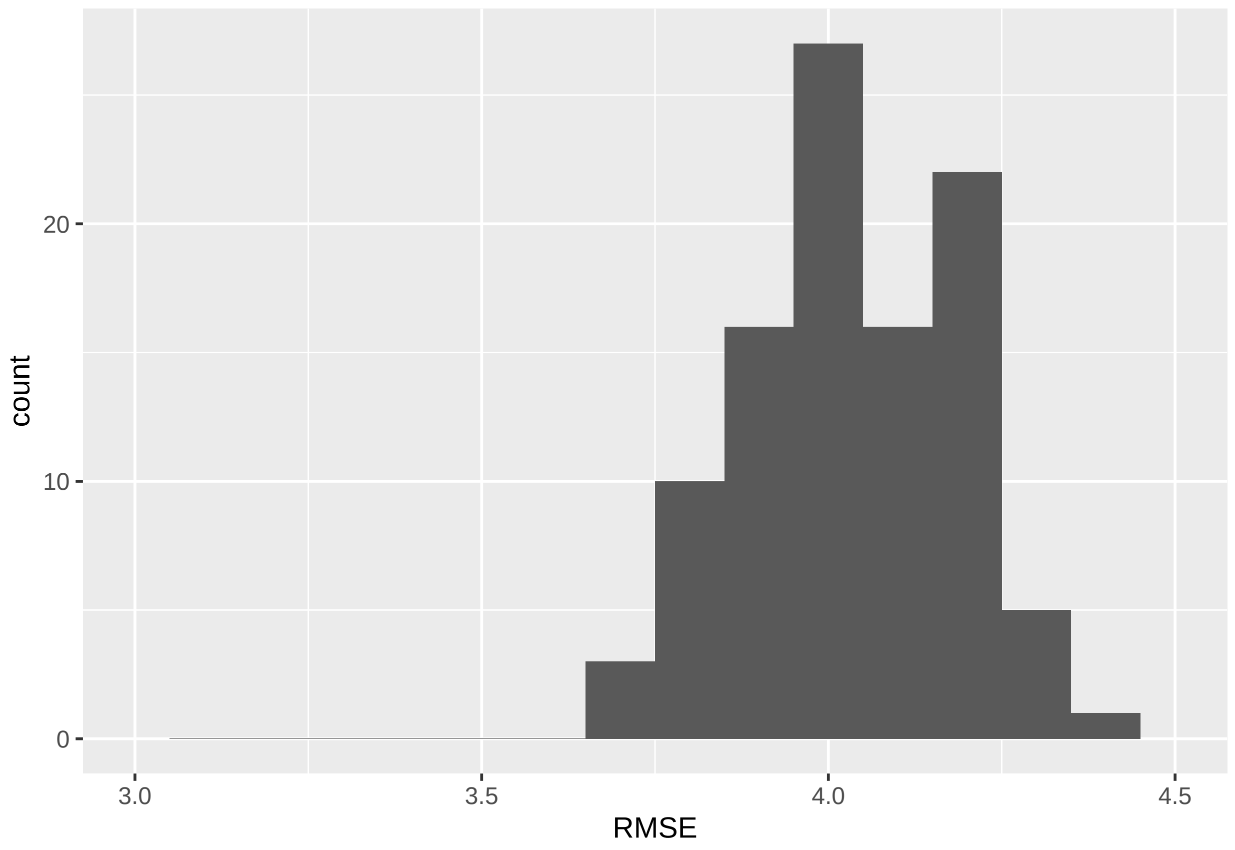
```
ggplot(ddspls_500_reps_Q2, aes(x = ncomp)) +
  geom_histogram(binwidth = 1) +
  scale_x_continuous(breaks = 1:10, limits = c(0.5,10.5)) +
  ylim(c(0, 100))
```



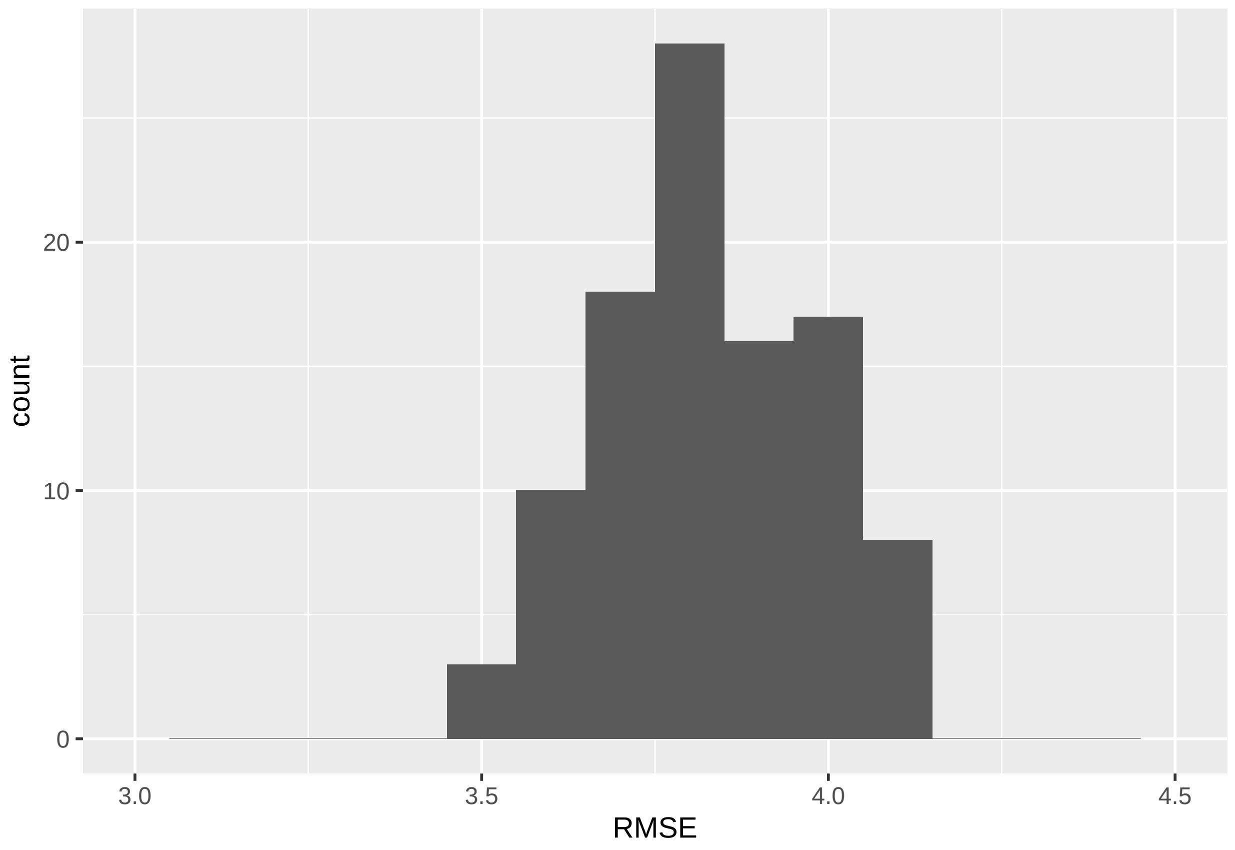
```
ggplot(ddspls_500_reps, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.1) +
  xlim(c(3, 4.5))
```



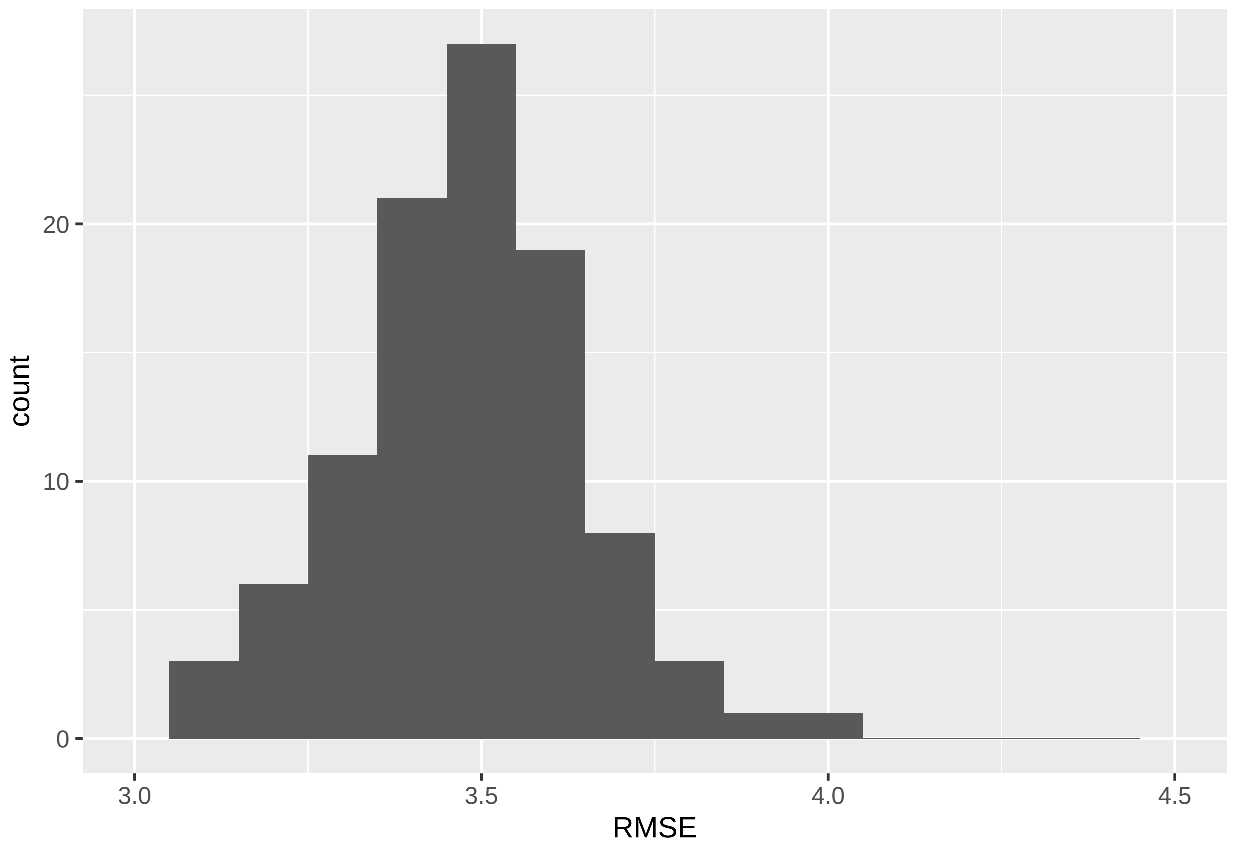
```r
ggplot(spls_500_reps, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.1) +
  xlim(c(3, 4.5))
```



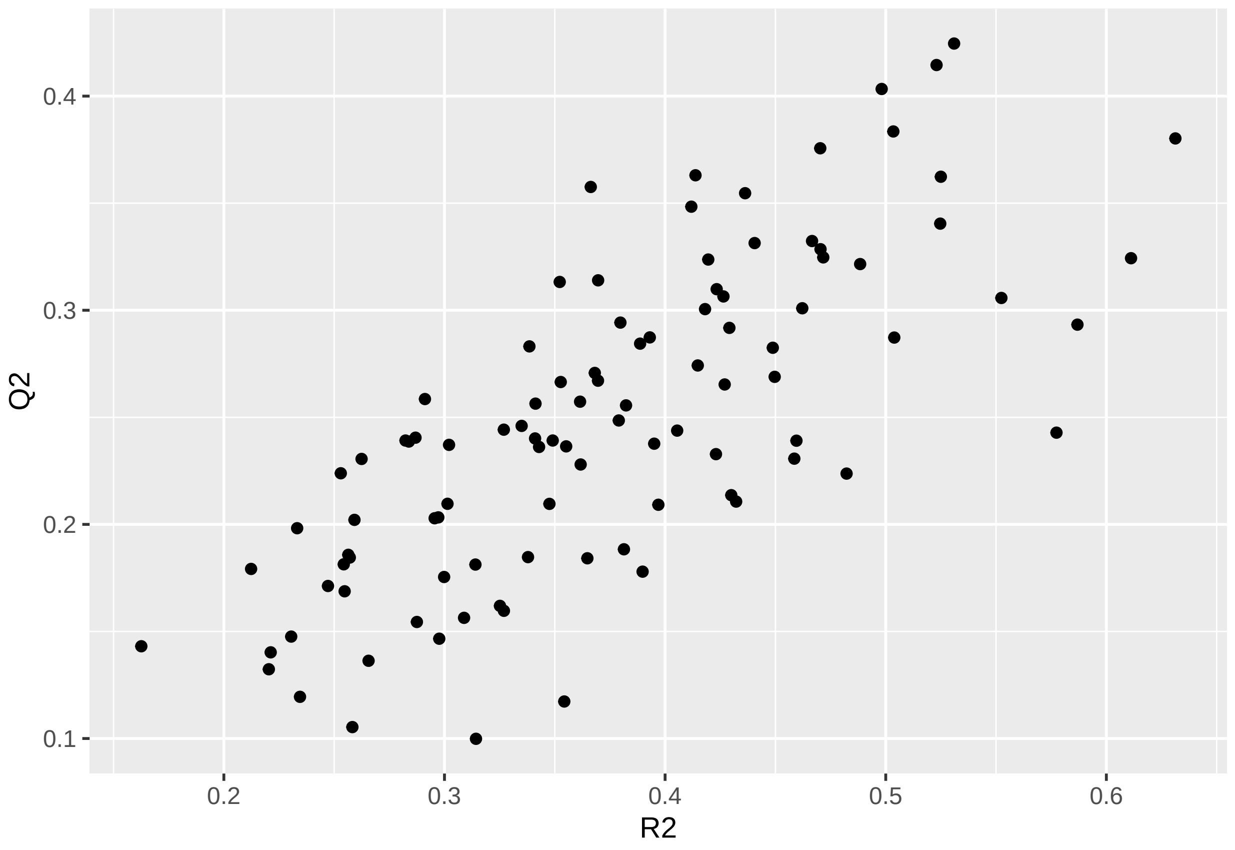
```r
ggplot(pls_500_reps, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.1) +
  xlim(c(3, 4.5))
```



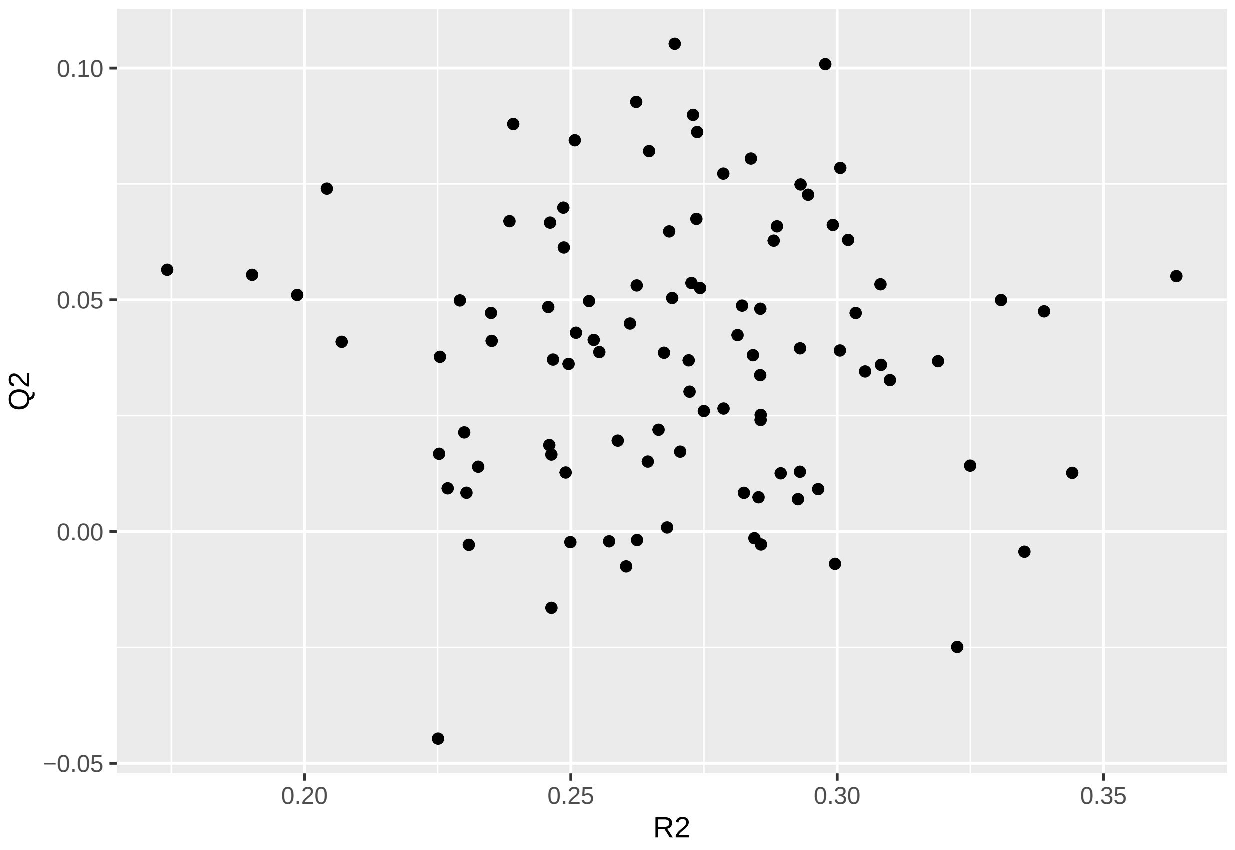
```
ggplot(ddspls_500_reps_Q2, aes(x = RMSE)) +
  geom_histogram(binwidth = 0.1) +
  xlim(c(3, 4.5))
```



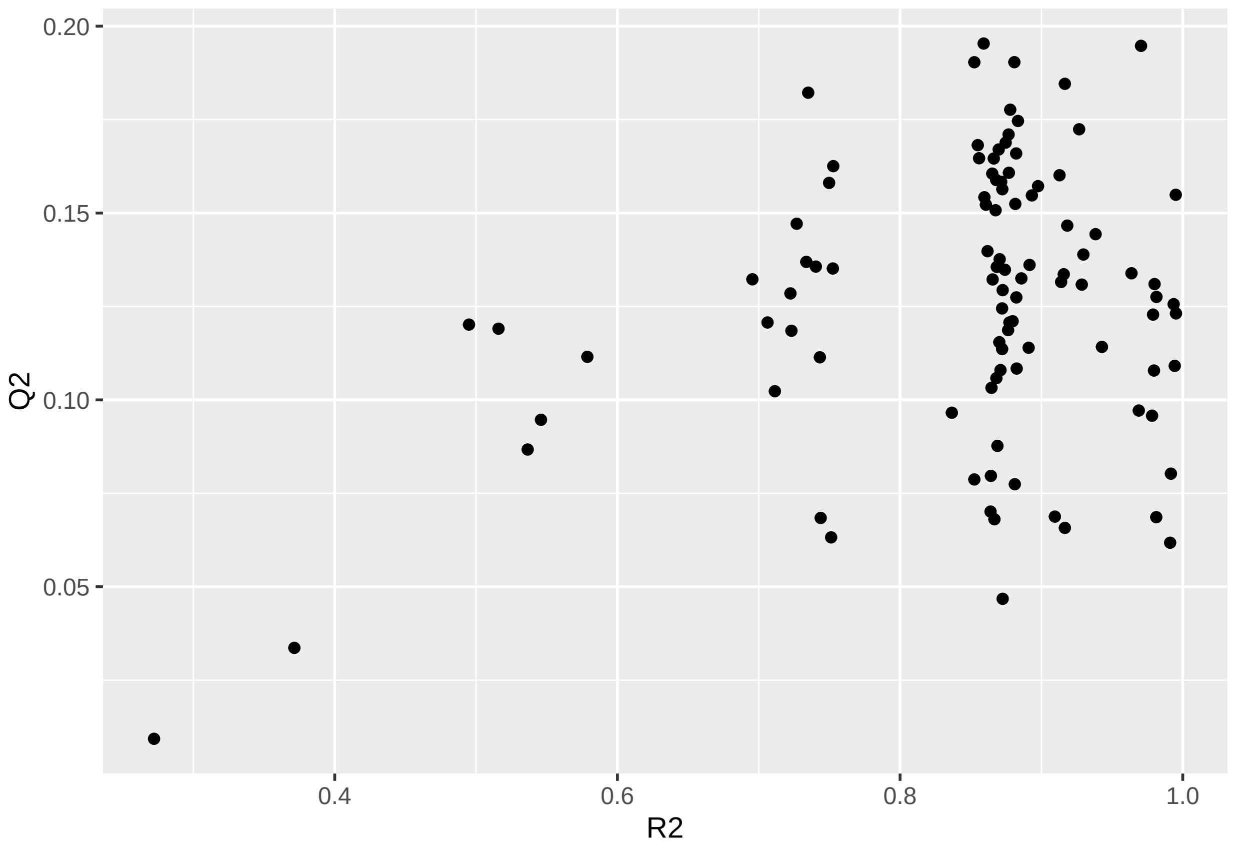
```
ggplot(ddspls_500_reps, aes(x = R2, y = Q2)) +
  geom_point()
```



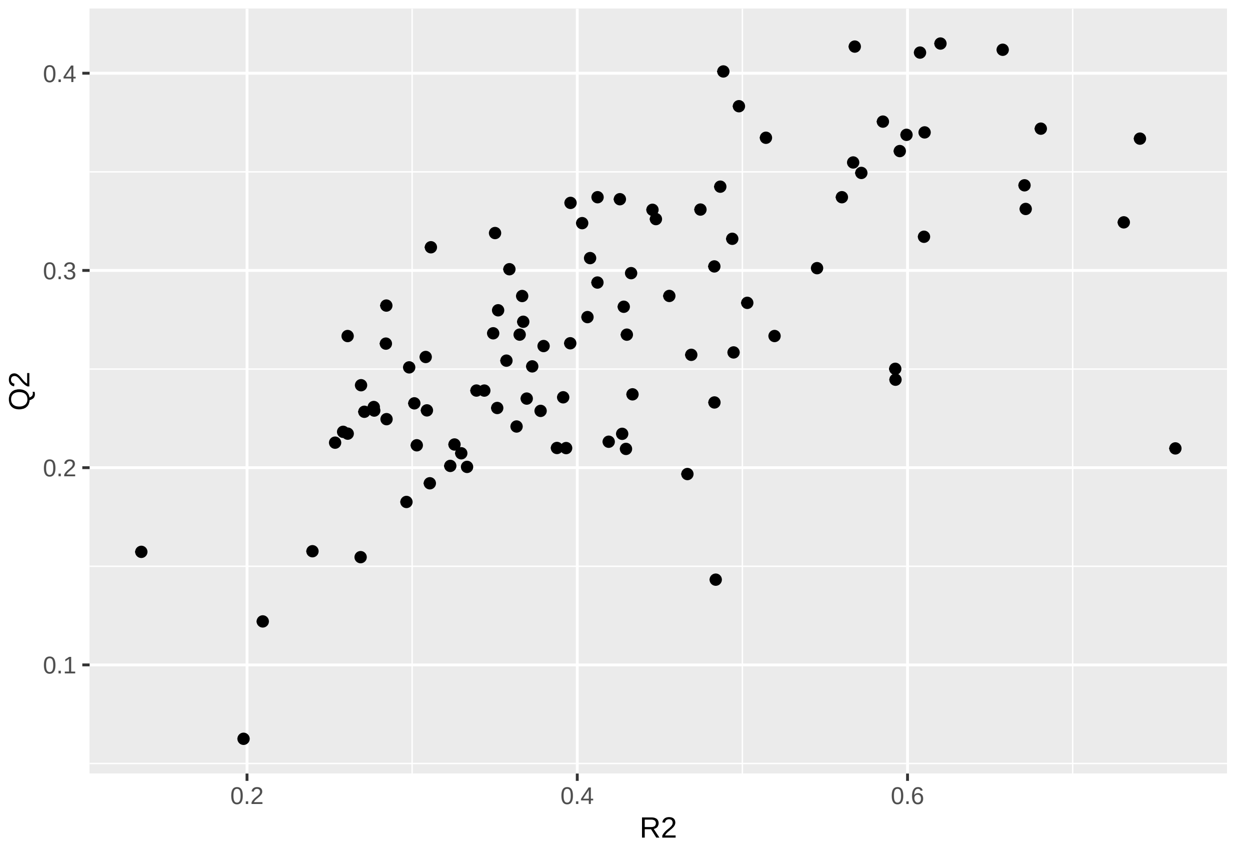
```
ggplot(spls_500_reps, aes(x = R2, y = Q2)) +
  geom_point()
```



```
ggplot(pls_500_reps, aes(x = R2, y = Q2)) +
  geom_point()
```



```
ggplot(ddspls_500_reps_Q2, aes(x = R2, y = Q2)) +
  geom_point()
```



```
reps_df <- data.frame(c(mean(ddspls_500_reps$RMSE), median(ddspls_500_reps$RMSE), var(ddspls_500_reps$R
   c(mean(spls_500_reps$RMSE), median(spls_500_reps$RMSE), var(spls_500_reps$RMSE)),
   c(mean(pls_500_reps$RMSE), median(pls_500_reps$RMSE), var(pls_500_reps$RMSE)),
   c(mean(ddspls_500_reps_Q2$RMSE), median(ddspls_500_reps_Q2$RMSE), var(ddspls_500_reps_Q2$RMSE)))

reps_df <- as.data.frame(t(as.matrix(reps_df)))

colnames(reps_df) <- c("mean", "median", "var")
row.names(reps_df) <- c("ddsPLS", "sPLS", "PLS", "ddsPLS_Q2")

reps_df
```

```
##               mean   median        var
## ddsPLS    3.538211 3.534172 0.03526031
## sPLS      4.035095 4.024334 0.02223995
## PLS       3.829996 3.826565 0.02289107
## ddsPLS_Q2 3.474859 3.483749 0.02864078
```

```
t.test(ddspls_1000_reps$rmse, ddspls_1000_reps_Q2$rmse)
```

```
##
##  Welch Two Sample t-test
##
## data:  ddspls_1000_reps$rmse and ddspls_1000_reps_Q2$rmse
## t = 2.3323, df = 194.81, p-value = 0.02071
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.009405659 0.112430572
## sample estimates:
## mean of x mean of y
##  3.569282  3.508364
```

```
t.test(ddspls_500_reps$RMSE, ddspls_500_reps_Q2$RMSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  ddspls_500_reps$RMSE and ddspls_500_reps_Q2$RMSE
## t = 2.5061, df = 195.9, p-value = 0.01302
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.01349816 0.11320468
## sample estimates:
## mean of x mean of y
##  3.538211  3.474859
```