Exercise 3

Advanced Methods for Regression and Classification

Rita Selimi

10/28/2024

Load the necessary libraries

Load Data

```
# Load the dataset
load("building.RData")
# str(df)
```

Data Splitting

```
set.seed(12332281)

# Split the data
sample_index <- sample(1:nrow(df), size = 2/3 * nrow(df))
train_data <- df[sample_index, ]
test_data <- df[-sample_index, ]

# Display the number of samples in each set
cat("Training set size:", nrow(train_data), "\n")
```

```
## Training set size: 248
```

```
cat("Test set size:", nrow(test_data), "\n")
```

```
## Test set size: 124
```

Fit regression model on the training data

```
# Fit linear regression model
model <- lm(y ~ ., data = train_data)
# summary(model)
```

Compute RMSE (training data)

```
train_pred <- predict(model, newdata = train_data)

# Compute RMSE for training data
train_rmse <- sqrt(mean((train_data$y - train_pred)^2))
cat("Training RMSE:", train_rmse, "\n")
```

```
## Training RMSE: 0.1928942
```

1. Principal Component Regression (PCR)

a) Applying PCR with Cross-Validation

```
# Fit PCR model with cross-validation and scaling
pcr_model <- pcr(y ~ ., data = train_data, scale = TRUE, validation = "CV", segments = 10)
summary(pcr_model)
```

```
## Data:    X dimension: 248 107
##  Y dimension: 248 1
## Fit method: svdpc
## Number of components considered: 107
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8429   0.6093   0.5938   0.5761   0.5311   0.5321   0.4648
## adjCV       0.8429   0.6091   0.5937   0.5728   0.5300   0.5328   0.4633
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.4715   0.4126   0.3713    0.3739    0.3752    0.3725    0.3700
## adjCV   0.4724   0.4116   0.3683    0.3714    0.3705    0.3698    0.3675
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.3702    0.3730    0.3119    0.2939    0.2923    0.2931    0.2928
## adjCV    0.3677    0.3716    0.3039    0.2895    0.2903    0.2910    0.2909
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV       0.2920    0.2941    0.2967    0.2983    0.3027    0.3019    0.3093
## adjCV    0.2897    0.2919    0.2944    0.2960    0.3003    0.2996    0.3065
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV       0.3076    0.3053    0.2968    0.2938    0.2907    0.2908    0.2802
## adjCV    0.3049    0.3019    0.2936    0.2918    0.2874    0.2880    0.2772
##        35 comps  36 comps  37 comps  38 comps  39 comps  40 comps  41 comps
## CV       0.2797    0.2764    0.2734    0.2743    0.2730    0.2703    0.2736
## adjCV    0.2753    0.2732    0.2699    0.2711    0.2697    0.2676    0.2711
##        42 comps  43 comps  44 comps  45 comps  46 comps  47 comps  48 comps
## CV       0.2750    0.2706    0.2690    0.2713    0.2713    0.2705    0.2707
## adjCV    0.2735    0.2675    0.2657    0.2681    0.2683    0.2671    0.2674
##        49 comps  50 comps  51 comps  52 comps  53 comps  54 comps  55 comps
## CV       0.2732    0.2720    0.2718    0.2728    0.2748    0.2772    0.2775
## adjCV    0.2697    0.2682    0.2682    0.2691    0.2712    0.2736    0.2733
##        56 comps  57 comps  58 comps  59 comps  60 comps  61 comps  62 comps
```

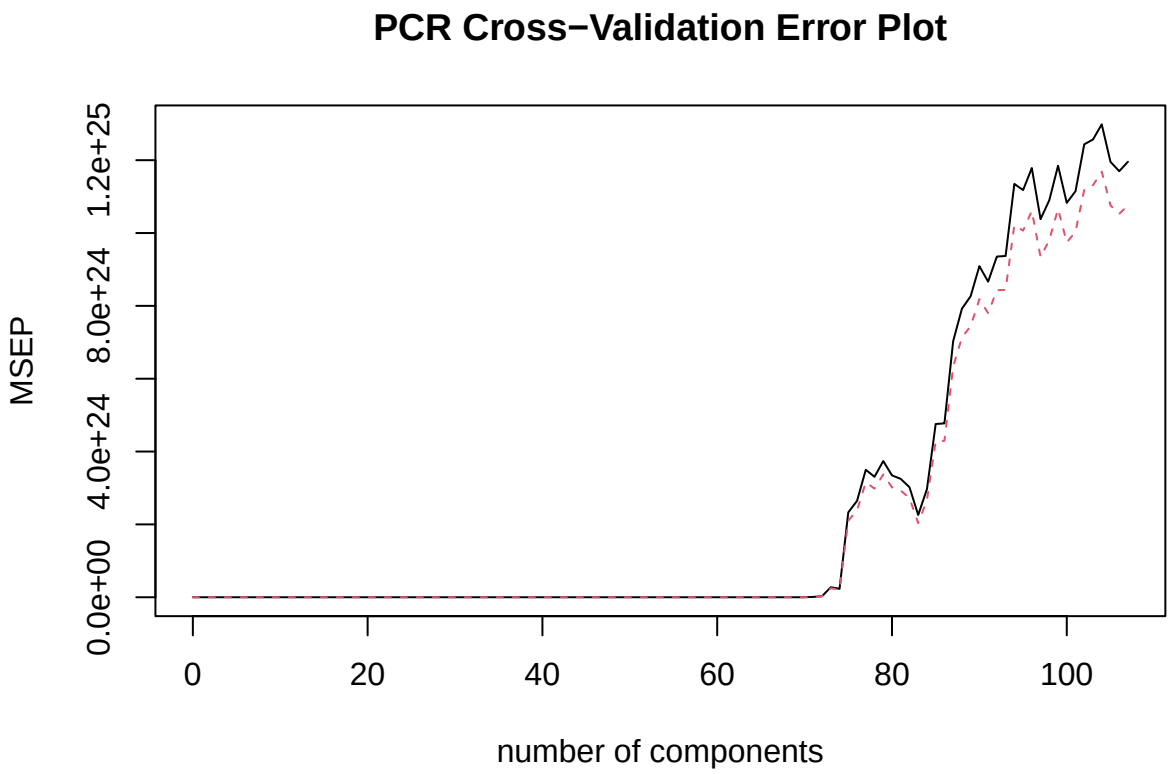
```
## CV       0.2835    0.2852    0.2890    0.2933    0.2957    0.2977    0.2968
## adjCV    0.2790    0.2806    0.2842    0.2884    0.2906    0.2924    0.2918
##        63 comps  64 comps  65 comps  66 comps  67 comps  68 comps  69 comps
## CV       0.2991    0.2983    0.2965    0.2981    0.2979    0.2971    0.3190
## adjCV    0.2946    0.2934    0.2906    0.2918    0.2914    0.2911    0.3118
##         70 comps   71 comps   72 comps   73 comps   74 comps   75 comps
## CV     2.947e+09  9.631e+10  1.727e+11  5.259e+11  4.798e+11  1.527e+12
## adjCV  2.794e+09  9.138e+10  1.638e+11  4.987e+11  4.551e+11  1.450e+12
##         76 comps   77 comps   78 comps   79 comps   80 comps   81 comps
## CV     1.626e+12  1.871e+12  1.819e+12  1.934e+12  1.829e+12  1.803e+12
## adjCV  1.544e+12  1.777e+12  1.727e+12  1.836e+12  1.737e+12  1.711e+12
##         82 comps   83 comps   84 comps   85 comps   86 comps   87 comps
## CV     1.738e+12  1.503e+12  1.724e+12  2.181e+12  2.185e+12  2.652e+12
## adjCV  1.650e+12  1.426e+12  1.635e+12  2.068e+12  2.072e+12  2.516e+12
##         88 comps   89 comps   90 comps   91 comps   92 comps   93 comps
## CV     2.815e+12  2.875e+12  3.015e+12  2.944e+12  3.059e+12  3.061e+12
## adjCV  2.671e+12  2.728e+12  2.861e+12  2.793e+12  2.903e+12  2.905e+12
##         94 comps   95 comps   96 comps   97 comps   98 comps   99 comps
## CV     3.369e+12  3.344e+12  3.433e+12  3.222e+12  3.302e+12  3.442e+12
## adjCV  3.197e+12  3.172e+12  3.257e+12  3.056e+12  3.132e+12  3.265e+12
##        100 comps  101 comps  102 comps  103 comps  104 comps  105 comps
## CV     3.291e+12  3.339e+12  3.526e+12  3.545e+12  3.603e+12  3.458e+12
## adjCV  3.121e+12  3.167e+12  3.346e+12  3.363e+12  3.419e+12  3.280e+12
##        106 comps  107 comps
## CV     3.420e+12  3.458e+12
## adjCV  3.245e+12  3.280e+12
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    64.70    71.75    76.48    80.87    84.20    87.11    89.03    90.74
## y    48.09    51.31    58.89    63.54    63.97    72.89    72.90    79.71
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    92.18     93.26     94.20     95.11     95.85     96.43     96.89
## y    84.00     84.00     84.73     84.73     84.87     85.17     85.18
##    16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X     97.30     97.62     97.90     98.13     98.36     98.55     98.73
## y     90.17     90.33     90.39     90.43     90.47     90.59     90.61
##    23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X     98.88     99.02     99.14     99.24     99.33     99.40     99.46
## y     90.72     90.78     90.92     90.98     91.11     91.18     91.55
##    30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X     99.51     99.56     99.60     99.64     99.68     99.71     99.74
## y     91.77     91.78     92.01     92.04     92.53     92.74     92.77
##    37 comps  38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X     99.77     99.80     99.82     99.84     99.86     99.87     99.89
## y     92.98     93.03     93.16     93.19     93.27     93.27     93.56
##    44 comps  45 comps  46 comps  47 comps  48 comps  49 comps  50 comps
## X     99.90     99.92     99.93     99.94     99.95     99.95     99.96
## y     93.66     93.67     93.69     93.77     93.79     93.91     93.97
##    51 comps  52 comps  53 comps  54 comps  55 comps  56 comps  57 comps
## X     99.97     99.97     99.97     99.98     99.98     99.99     99.99
## y     93.97     93.99     94.00     94.01     94.08     94.08     94.08
##    58 comps  59 comps  60 comps  61 comps  62 comps  63 comps  64 comps
## X     99.99     99.99     99.99    100.00    100.00     100.0    100.00
```

```
## y      94.09     94.09     94.11     94.14     94.18      94.2     94.32
##    65 comps  66 comps  67 comps  68 comps  69 comps  70 comps  71 comps
## X    100.00    100.00    100.00    100.00    100.00    100.00     100.0
## y      94.48     94.54     94.57     94.57     94.59     94.59      94.7
##    72 comps  73 comps  74 comps  75 comps  76 comps  77 comps  78 comps
## X    100.00    100.00    100.00    100.00    100.00    100.00    100.00
## y      94.72     94.72     94.72     94.73     94.73     94.77     94.78
##    79 comps  80 comps  81 comps  82 comps  83 comps  84 comps  85 comps
## X    100.00    100.00    100.00    100.00    100.00    100.00    100.00
## y      94.79     94.81     94.89     94.92     94.95     94.95     95.08
##    86 comps  87 comps  88 comps  89 comps  90 comps  91 comps  92 comps
## X     100.0     100.0    100.00    100.00     100.0    100.00    100.00
## y      95.1      95.1     95.19     95.25      95.3     95.31     95.45
##    93 comps  94 comps  95 comps  96 comps  97 comps  98 comps  99 comps
## X    100.00    100.00    100.00    100.00    100.00    100.00    100.00
## y      95.46     95.46     95.52     95.57     95.57     95.58     95.58
##    100 comps  101 comps  102 comps  103 comps  104 comps  105 comps  106 comps
## X     100.00     100.00     100.00     100.00     100.00     100.00      100.0
## y      95.61      95.63      95.64      95.66      95.66      95.66       95.7
##    107 comps
## X     100.00
## y      95.73
```

This applies Principal Component Regression (PCR) to the training data, scaling the variables and using 10-fold cross-validation to find the best number of components for accurate predictions.

b) Cross-Validation Error Plot and Optimal Components

```r
# Plot cross-validation errors for each component
validationplot(pcr_model, val.type = "MSEP", main = "PCR Cross-Validation Error Plot")
```



How many components seem to be optimal?

Optimal number of components is around 68. The curve shows how the prediction error changes as more components are added. Initially, the error decreases, indicating an improvement in the model's accuracy as components are included. However, after a certain number of components, the error increases, suggesting that adding more components leads to overfitting, where the model becomes too complex and loses its generalizability.

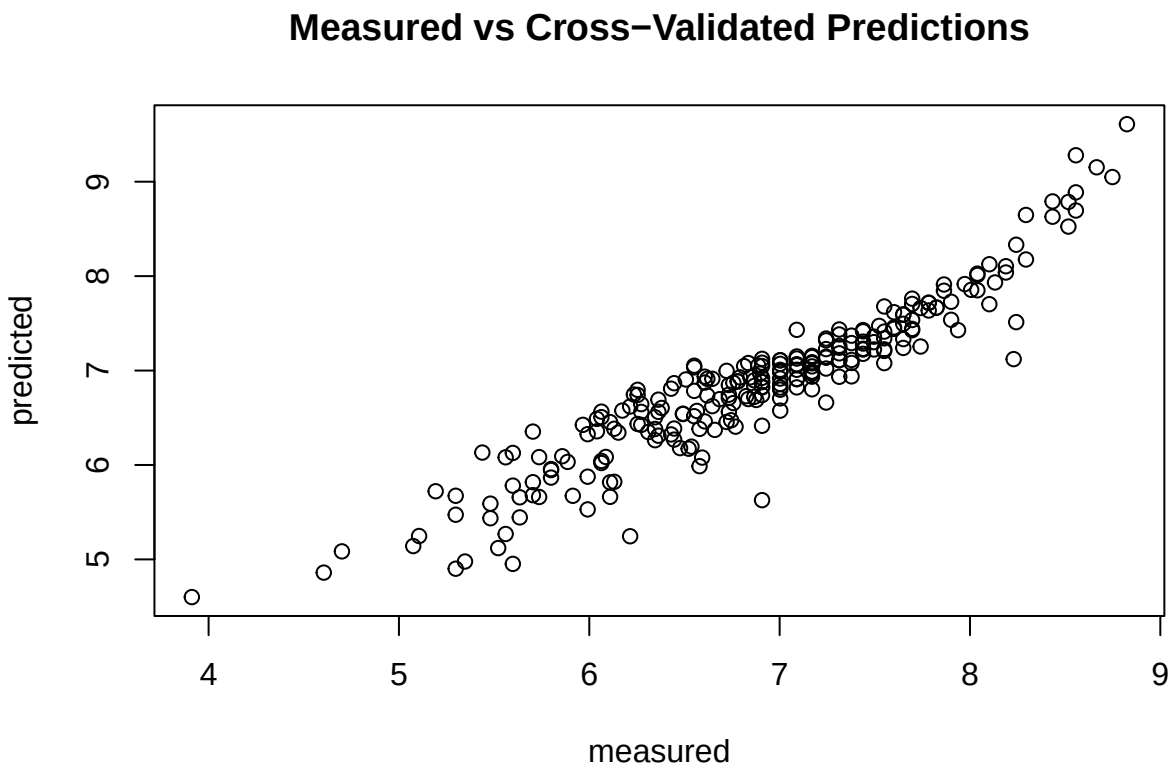
```
rmsep_results <- RMSEP(pcr_model)
rmsep_values <- rmsep_results$val

# Now you can find the optimal RMSE
optimal_components <- 68
optimal_rmse <- rmsep_values[optimal_components]
cat("Resulting RMSE at optimal components:", optimal_rmse, "\n")
```

```
## Resulting RMSE at optimal components: 0.2880382
```

The RMSE increased when moving from a simple training RMSE to a cross-validated RMSE in PCR, bacause: - Overfitting on Training Data: The initial RMSE of 0.1929 was calculated directly on the training data, which means the model was fitted to the same data used to evaluate it. - Cross-Validation for Generalization: When you switch to using cross-validation in PCR, the model is repeatedly trained on different subsets of the data and tested on other subsets. The cross-validated RMSE, better reflects how well the model generalizes to unseen data.

c) Plotting Measured vs. Cross-Validated Predictions

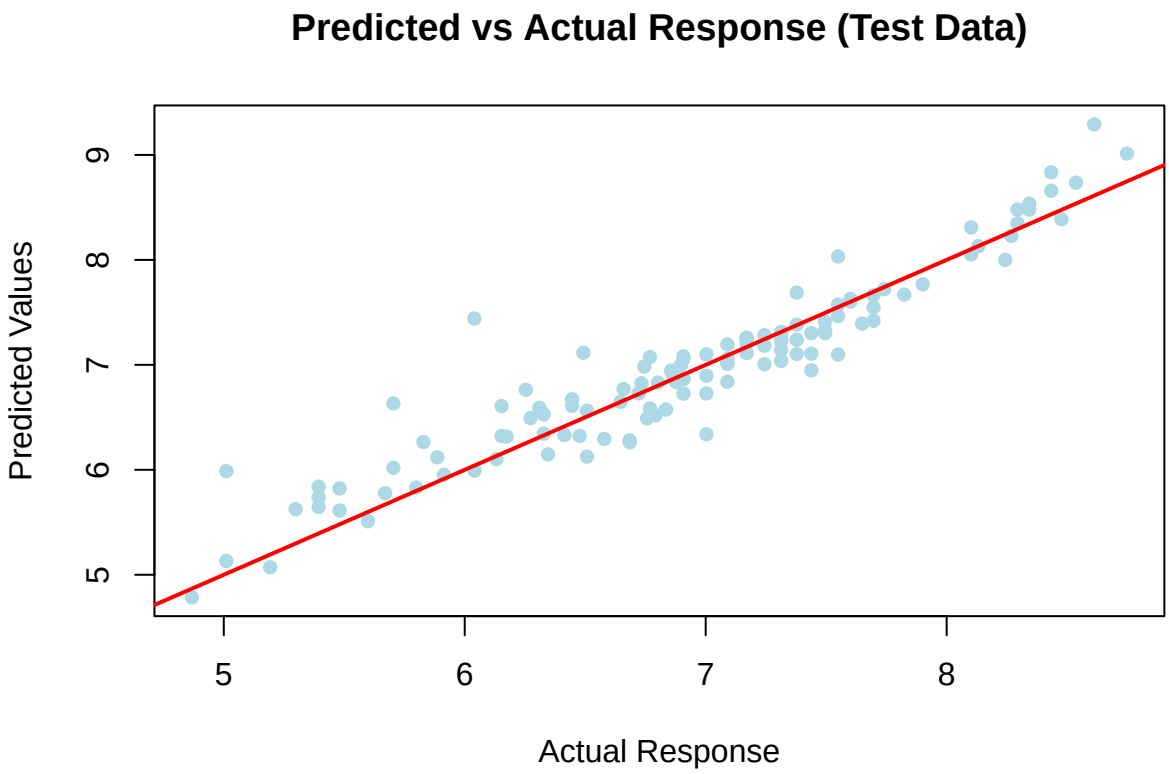
```
# Plot measured vs cross-validated predictions
predplot(pcr_model, ncomp = optimal_components, main = "Measured vs Cross-Validated Predictions")
```



This plot visualizes the relationship between the actual measured values of the response variable (y) and the cross-validated predictions generated by the PCR model using 68 components. The plot shows that the cross-validated predictions closely align with the actual values along the diagonal, indicating the model's generally accurate fit, with some minor variance at the edges.

d) Plotting Predicted vs Observed Values for Test Data and Computing RMSE

```
test_pred <- predict(pcr_model, newdata = test_data, ncomp = 68)

# Plot Predicted vs Observed values for Test Data
plot(test_data$y, test_pred, main = "Predicted vs Actual Response (Test Data)", xlab = "Actual Response
    ylab = "Predicted Values", pch = 16, col = "lightblue")
abline(0, 1, col = "red", lwd = 2)
```




```
# Calculate RMSE for Test Data
test_rmse <- sqrt(mean((test_data$y - test_pred)^2))
cat("Test RMSE:", test_rmse, "\n")
```

```
## Test RMSE: 0.2872396
```

The plot shows a strong alignment of the predicted values with the actual values for the test data. The relatively low Test RMSE of 0.2872, indicates minimal error in predictions.

2. Partial least squares regression (PLS)

a) Applying PLS with Cross-Validation

```
# Apply PLS on the training data with cross-validation and scaling
pls_model <- plsr(y ~ ., data = train_data, scale = TRUE, validation = "CV", segments = 10)
summary(pls_model)
```

```
## Data:    X dimension: 248 107
##  Y dimension: 248 1
## Fit method: kernelpls
## Number of components considered: 107
##
## VALIDATION: RMSEP
```

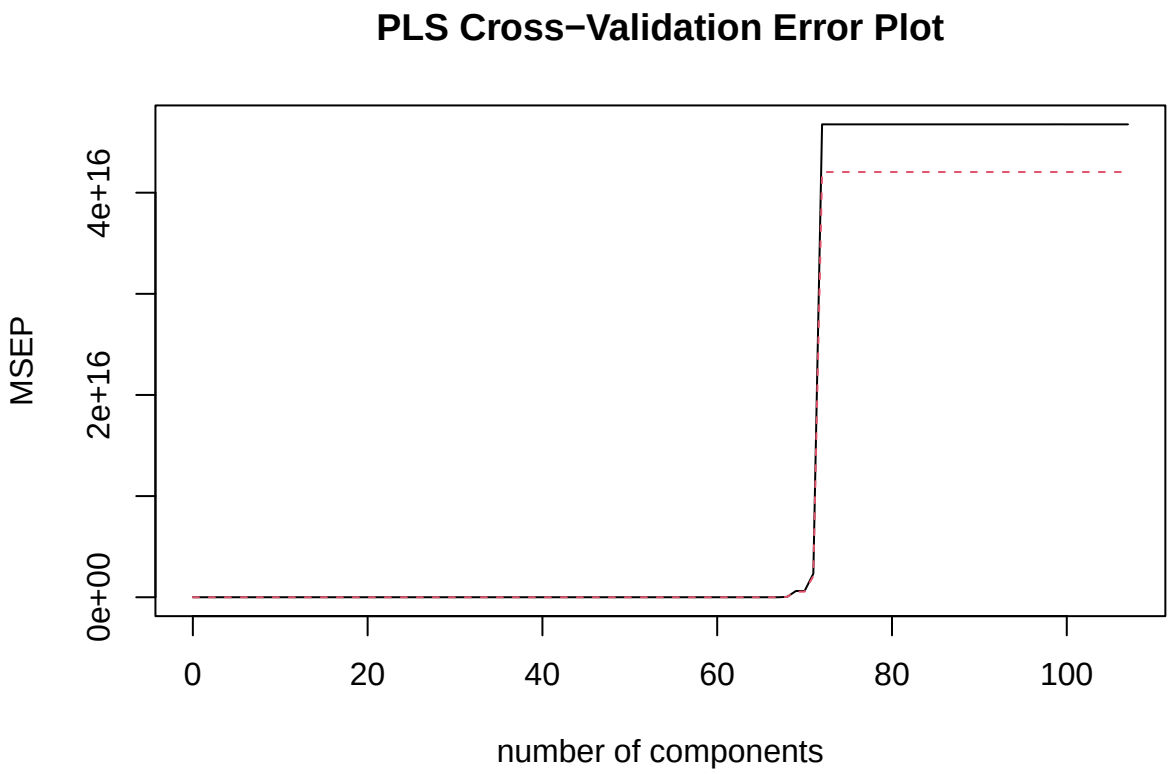
```
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8429   0.5874   0.4073   0.3518   0.3192   0.2944   0.2823
## adjCV       0.8429   0.5873   0.4052   0.3498   0.3171   0.2929   0.2804
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.2829   0.2863   0.2891    0.2932    0.2944    0.2898    0.2815
## adjCV   0.2810   0.2836   0.2859    0.2884    0.2892    0.2841    0.2764
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.2744    0.2725    0.2725    0.2735    0.2762    0.2766    0.2787
## adjCV    0.2707    0.2690    0.2688    0.2696    0.2722    0.2724    0.2745
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV       0.2801    0.2811    0.2827    0.2866    0.2865    0.2899    0.2945
## adjCV    0.2758    0.2767    0.2782    0.2819    0.2818    0.2849    0.2891
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV       0.2973    0.3028    0.3030    0.3059    0.3081    0.3076    0.3105
## adjCV    0.2917    0.2967    0.2968    0.2994    0.3013    0.3008    0.3034
##        35 comps  36 comps  37 comps  38 comps  39 comps  40 comps  41 comps
## CV       0.3122    0.3115    0.3128    0.3123    0.3150    0.3184    0.3197
## adjCV    0.3049    0.3042    0.3053    0.3049    0.3074    0.3106    0.3118
##        42 comps  43 comps  44 comps  45 comps  46 comps  47 comps  48 comps
## CV       0.3230    0.3265    0.3315    0.3372    0.3445    0.3503    0.3539
## adjCV    0.3148    0.3181    0.3226    0.3279    0.3346    0.3400    0.3434
##        49 comps  50 comps  51 comps  52 comps  53 comps  54 comps  55 comps
## CV       0.3574    0.3611    0.3629    0.3648    0.3667    0.3674    0.3682
## adjCV    0.3466    0.3501    0.3516    0.3534    0.3552    0.3558    0.3566
##        56 comps  57 comps  58 comps  59 comps  60 comps  61 comps  62 comps
## CV       0.3686    0.3684    0.3683    0.3687    0.3697    0.3700    0.3700
## adjCV    0.3570    0.3568    0.3567    0.3570    0.3579    0.3582    0.3582
##        63 comps  64 comps  65 comps  66 comps  67 comps  68 comps  69 comps
## CV       0.3703    0.3705    0.3705    0.3705    0.3704   6060898  24816893
## adjCV    0.3585    0.3587    0.3587    0.3587    0.3586   5747296  23532821
##        70 comps  71 comps   72 comps   73 comps   74 comps   75 comps
## CV     24817022  48213846  216233529  216229331  216230956  216230322
## adjCV  23532944  45719190  205045213  205041233  205042773  205042172
##         76 comps   77 comps   78 comps   79 comps   80 comps   81 comps
## CV     216229731  216230773  216230118  216231031  216230406  216231416
## adjCV  205041612  205042600  205041979  205042844  205042252  205043209
##         82 comps   83 comps   84 comps   85 comps   86 comps   87 comps
## CV     216231316  216231089  216230179  216231494  216229654  216231161
## adjCV  205043115  205042899  205042036  205043283  205041539  205042968
##         88 comps   89 comps   90 comps   91 comps   92 comps   93 comps
## CV     216230331  216230826  216229842  216231296  216232423  216231344
## adjCV  205042181  205042650  205041717  205043096  205044164  205043141
##         94 comps   95 comps   96 comps   97 comps   98 comps   99 comps
## CV     216232113  216230959  216230749  216231978  216230177  216232575
## adjCV  205043870  205042777  205042577  205043743  205042034  205044308
##        100 comps  101 comps  102 comps  103 comps  104 comps  105 comps
## CV     216230956  216231194  216231562  216232052  216232291  216232238
## adjCV  205042773  205042999  205043348  205043812  205044039  205043989
##        106 comps  107 comps
## CV     216231325  216231550
## adjCV  205043123  205043336
##
## TRAINING: % variance explained
```

```
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    64.55    69.53    74.23    77.93    80.34    82.10    85.65    87.14
## y    51.98    79.47    85.64    88.38    90.00    91.28    91.57    91.91
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    89.40     90.79     91.64     92.16     93.07     94.37     94.86
## y    92.26     92.82     93.21     93.47     93.63     93.71     93.80
##    16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X     95.24     95.73     96.47     96.67     97.26     97.73     98.05
## y     93.89     93.95     93.99     94.06     94.08     94.11     94.14
##    23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X     98.26     98.44     98.58     98.70     98.81     98.94     99.07
## y     94.17     94.20     94.24     94.27     94.31     94.34     94.37
##    30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X     99.17     99.26     99.35     99.45     99.50     99.57      99.6
## y     94.41     94.46     94.49     94.52     94.55     94.57      94.6
##    37 comps  38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X     99.63     99.66     99.68     99.70     99.73     99.75     99.77
## y     94.61     94.62     94.63     94.64     94.65     94.65     94.66
##    44 comps  45 comps  46 comps  47 comps  48 comps  49 comps  50 comps
## X     99.79     99.81     99.84     99.85     99.87     99.89      99.9
## y     94.67     94.67     94.68     94.68     94.69     94.69      94.7
##    51 comps  52 comps  53 comps  54 comps  55 comps  56 comps  57 comps
## X     99.92     99.93     99.93     99.94     99.95     99.96     99.97
## y     94.70     94.71     94.71     94.71     94.71     94.71     94.71
##    58 comps  59 comps  60 comps  61 comps  62 comps  63 comps  64 comps
## X     99.97     99.98     99.98     99.98     99.99     99.99     99.99
## y     94.71     94.72     94.72     94.72     94.72     94.72     94.72
##    65 comps  66 comps  67 comps  68 comps  69 comps  70 comps  71 comps
## X     99.99    100.00    100.00    100.00    100.00    100.00    100.00
## y     94.72     94.72     94.72     94.72     94.72     94.72     94.72
##    72 comps  73 comps  74 comps  75 comps  76 comps  77 comps  78 comps
## X    100.00    100.00    100.01    100.01    100.01    100.02    100.02
## y     94.72     94.69     94.69     94.69     94.69     94.69     94.69
##    79 comps  80 comps  81 comps  82 comps  83 comps  84 comps  85 comps
## X    100.02    100.03    100.03    100.03    100.04    100.04    100.04
## y     94.69     94.69     94.69     94.69     94.69     94.69     94.69
##    86 comps  87 comps  88 comps  89 comps  90 comps  91 comps  92 comps
## X    100.04    100.05    100.05    100.06    100.06    100.06    100.07
## y     94.69     94.69     94.69     94.69     94.69     94.69     94.69
##    93 comps  94 comps  95 comps  96 comps  97 comps  98 comps  99 comps
## X    100.07    100.07    100.08    100.08    100.08    100.09    100.09
## y     94.69     94.69     94.69     94.69     94.69     94.69     94.69
##    100 comps  101 comps  102 comps  103 comps  104 comps  105 comps  106 comps
## X     100.09     100.10     100.10     100.10     100.11     100.11     100.11
## y      94.69      94.69      94.69      94.69      94.69      94.69      94.69
##    107 comps
## X     100.12
## y      94.69
```

This code fits a PLS model to the train_data dataset, using scaling and 10-fold cross-validation to assess model performance.

b) Cross-Validation Error Plot and Optimal Components

```
# Plot cross-validation errors for each component
validationplot(pls_model, val.type = "MSEP", main = "PLS Cross-Validation Error Plot")
```



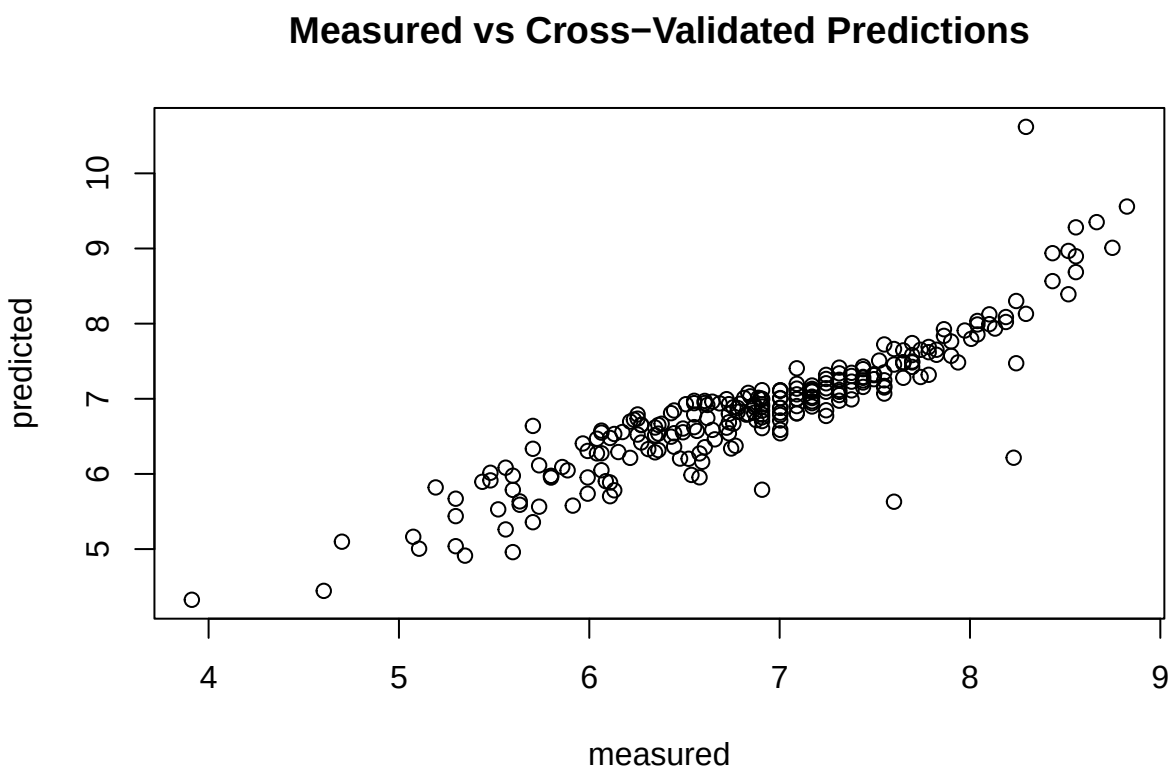
How many components seem to be optimal? In the PLS Cross-Validation Error Plot, the optimal number of components appears to be where the Mean Squared Error of Prediction (MSEP) is at its lowest before any significant increase. From the plot, this seems to occur at around 65 components. After this point, the MSEP value increases sharply, indicating that adding more components does not improve the model's performance and may lead to overfitting.

```
rmsep_results2 <- RMSEP(pls_model)
rmsep_values2 <- rmsep_results2$val

# Now you can find the optimal RMSE
optimal_components <- 65
optimal_rmse <- rmsep_values2[optimal_components]
cat("Resulting RMSE at optimal components:", optimal_rmse, "\n")
```

```
## Resulting RMSE at optimal components: 0.3080849
```

c) Plotting Measured vs. Cross-Validated Predictions

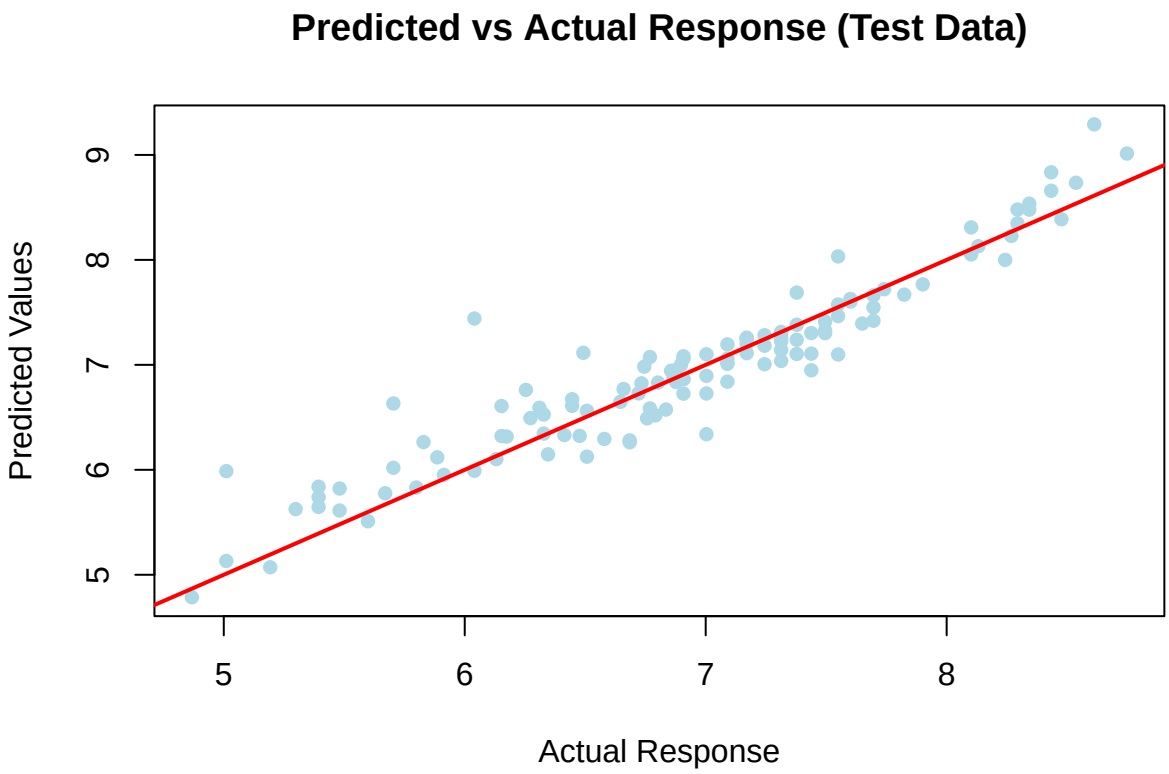
```r
# Plot measured vs cross-validated predictions
predplot(pls_model, ncomp = optimal_components, main = "Measured vs Cross-Validated Predictions")
```



This plot visualizes the relationship between the actual measured values of the response variable (y) and the cross-validated predictions generated by the PLS model using 65 components. The plot shows that the cross-validated predictions closely align with the actual values along the diagonal, indicating the model's generally accurate fit, with some minor variance at the edges.

d) Plotting Predicted vs Observed Values for Test Data and Computing RMSE

```r
pls_test_pred <- predict(pls_model, newdata = test_data, ncomp = 65)

# Plot Predicted vs Observed values for Test Data
plot(test_data$y, test_pred, main = "Predicted vs Actual Response (Test Data)", xlab = "Actual Response
    ylab = "Predicted Values", pch = 16, col = "lightblue")
abline(0, 1, col = "red", lwd = 2)
```




```
# Calculate RMSE for test data predictions
pls_test_rmse <- sqrt(mean((test_data$y - pls_test_pred)^2))
cat("Test RMSE for PLS:", pls_test_rmse, "\n")
```

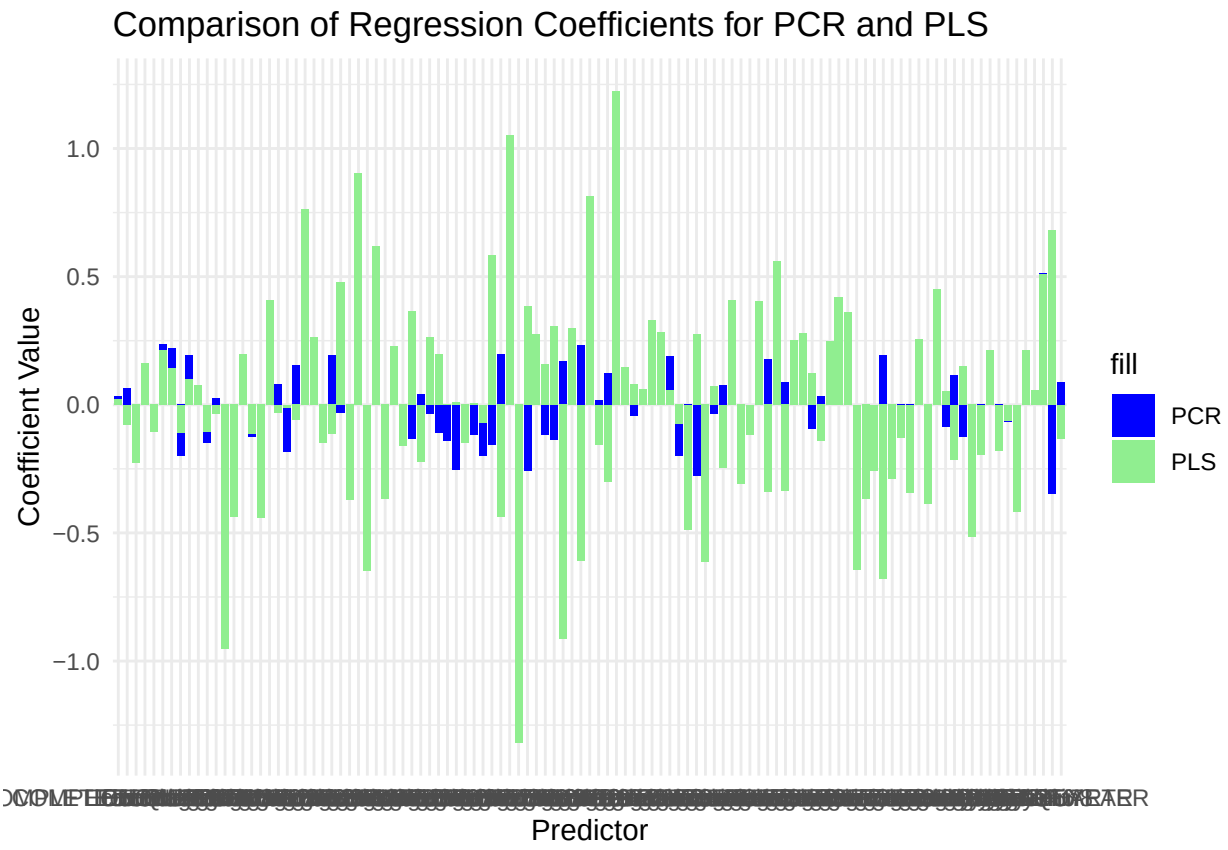
```
## Test RMSE for PLS: 0.2988856
```

The plot shows a strong alignment of the predicted values with the actual values for the test data. The relatively low Test RMSE of 0.2992037, indicates minimal error in predictions.

e)) Compare the regression coefficients from PCR and PLS

```
# Extract coefficients for the optimal number of components in PCR and PLS
pcr_coefs <- coef(pcr_model, ncomp = optimal_components)
pls_coefs <- coef(pls_model, ncomp = optimal_components)

# Convert to data frame for easy plotting
coef_data <- data.frame(Predictor = rownames(pcr_coefs), PCR = as.vector(pcr_coefs),
    PLS = as.vector(pls_coefs))

# Plot the coefficients for PCR and PLS
library(ggplot2)
ggplot(coef_data, aes(x = Predictor)) + geom_bar(aes(y = PCR, fill = "PCR"), stat = "identity",
    position = "dodge") + geom_bar(aes(y = PLS, fill = "PLS"), stat = "identity",
```

```
    position = "dodge") + labs(title = "Comparison of Regression Coefficients for PCR and PLS",
    y = "Coefficient Value") + scale_fill_manual(values = c(PCR = "blue", PLS = "lightgreen")) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) + theme_minimal()
```



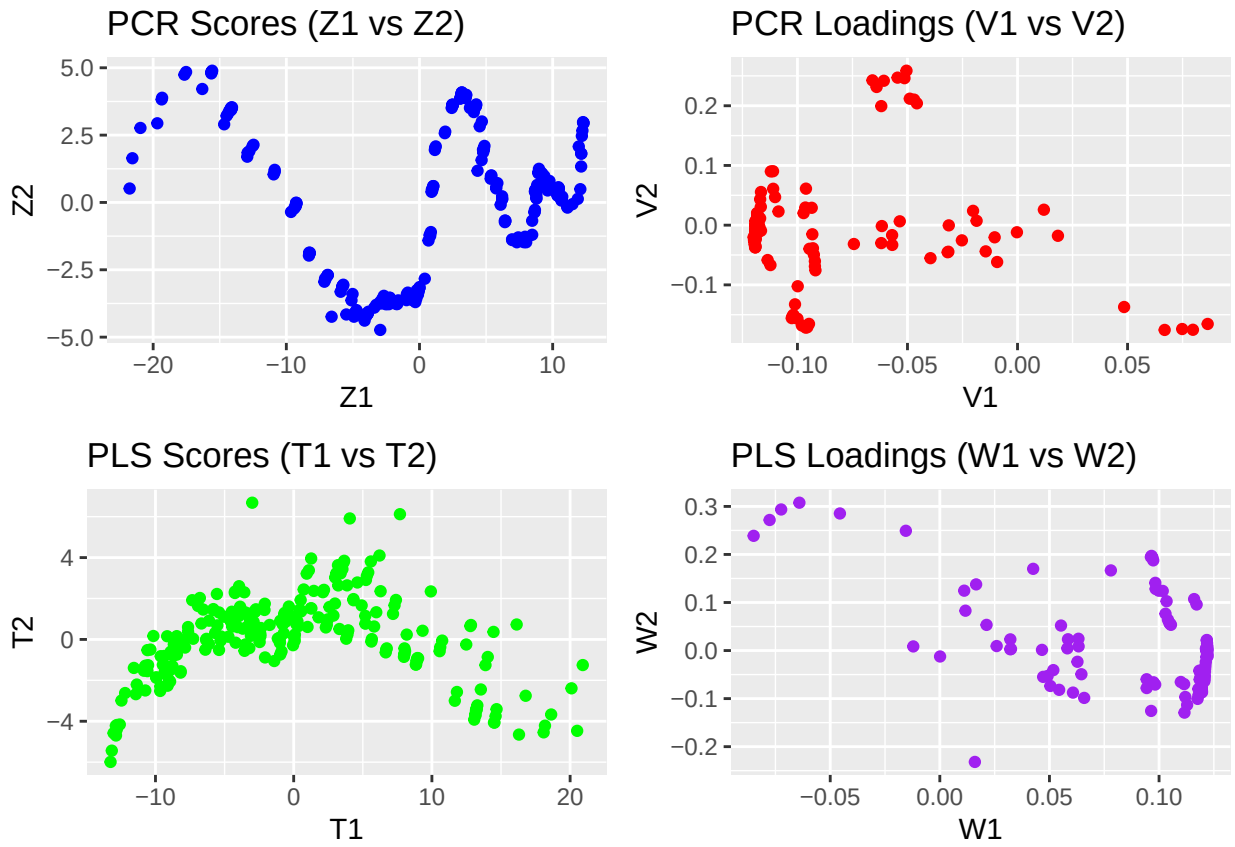
From this comparison plot of regression coefficients for PCR (blue) and PLS (green), we can observe:

1. **Coefficient Magnitudes**: PLS generally shows larger coefficients than PCR across several predictors, suggesting that PLS places more emphasis on certain variables than PCR. This is because PLS aims to maximize the covariance between predictors and the response variable, while PCR focuses on capturing the variance of predictors.
2. **Direction of Coefficients**: In some predictors, the direction of the coefficients differs between PCR and PLS, indicating that each method might be capturing slightly different relationships between predictors and the response.
3. **Stability and Selection**: PCR seems to yield smaller coefficients, particularly around zero for several predictors, which could indicate that only the most influential predictors have noticeable effects.

3. Score vectors and loadings vectors

```
library(ggplot2)
library(gridExtra)

# Extract scores and loadings for PCR
```

```r
pcr_scores <- pcr_model$scores[, 1:2]  # First two score vectors (Z1 and Z2)
pcr_loadings <- pcr_model$loadings[, 1:2]  # First two loading vectors (V1 and V2)

# Extract scores and loadings for PLS
pls_scores <- pls_model$scores[, 1:2]  # First two score vectors (T1 and T2)
pls_loadings <- pls_model$loadings[, 1:2]  # First two loading vectors (W1 and W2)

# Convert to data frames for ggplot
pcr_scores_df <- as.data.frame(pcr_scores)
pcr_loadings_df <- as.data.frame(pcr_loadings)
pls_scores_df <- as.data.frame(pls_scores)
pls_loadings_df <- as.data.frame(pls_loadings)

# Plot PCR scores
p1 <- ggplot(pcr_scores_df, aes(x = `Comp 1`, y = `Comp 2`)) + geom_point(color = "blue") +
    labs(title = "PCR Scores (Z1 vs Z2)", x = "Z1", y = "Z2")

# Plot PCR loadings
p2 <- ggplot(pcr_loadings_df, aes(x = `Comp 1`, y = `Comp 2`)) + geom_point(color = "red") +
    labs(title = "PCR Loadings (V1 vs V2)", x = "V1", y = "V2")

# Plot PLS scores
p3 <- ggplot(pls_scores_df, aes(x = `Comp 1`, y = `Comp 2`)) + geom_point(color = "green") +
    labs(title = "PLS Scores (T1 vs T2)", x = "T1", y = "T2")

# Plot PLS loadings
p4 <- ggplot(pls_loadings_df, aes(x = `Comp 1`, y = `Comp 2`)) + geom_point(color = "purple") +
    labs(title = "PLS Loadings (W1 vs W2)", x = "W1", y = "W2")

# Arrange the plots in a 2x2 grid
grid.arrange(p1, p2, p3, p4, ncol = 2)
```



1. **PCR Scores (Z1 vs Z2)**: The plot of the first two principal component scores (Z1 and Z2) shows a clear pattern, indicating that the first two components capture a meaningful structure in the data. There seems to be a trend within the points, suggesting a relationship between the variables that these components represent.

2. **PCR Loadings (V1 vs V2)**: This plot of the loadings (V1 and V2) shows how the original predictors contribute to the first two principal components. The spread of points here implies that certain variables are more heavily weighted on either V1 or V2, helping shape the data's primary directions.

3. **PLS Scores (T1 vs T2)**: The PLS scores (T1 and T2) also show a pattern, with points spread across a curve. This indicates that PLS, like PCR, captures a trend in the data. However, PLS components focus on maximizing the correlation with the response variable, so these patterns are influenced by that goal.

4. **PLS Loadings (W1 vs W2)**: The loadings for PLS (W1 vs W2) indicate how the original variables contribute to the first two PLS components. The distribution of points suggests some variables contribute more to these components, capturing relationships specifically targeted to predict the response.