hw8

Jiayi

2024-10-31

```
knitr::opts_chunk$set(echo = TRUE)
library(faraway)      # Contains the datasets
library(ggplot2)      # For data visualization
library(pls)          # For principal component regression (PCR)
```

```
##
## Attaching package: 'pls'

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

```
library(glmnet)       # For ridge and lasso regression
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```
library(MASS)
library(ModelMetrics) # For RMSE calculation
```

```
##
## Attaching package: 'ModelMetrics'

## The following object is masked from 'package:base':
##
##     kappa
```

Problem 1:

1.(a)

```
# Load seatpos data
data(seatpos)
?seatpos # Inspect the dataset

# Define response and explanatory variables
y <- seatpos$hipcenter
X <- seatpos[, -1] # Exclude hipcenter for predictors
```

PCA without Scaling

```
# Perform PCA without scaling
pca_no_scale <- prcomp(X, scale = FALSE)
U_no_scale <- pca_no_scale$rotation

# Display the loading matrix U without scaling
print("Loading matrix U without scaling:")
```

```
## [1] "Loading matrix U without scaling:"
```

```
U_no_scale
```

```
##                   PC1        PC2        PC3          PC4          PC5
## Weight    -0.41941105 0.86877641  0.2618673  0.004066278  0.026239405
## HtShoes   -0.14648356 0.11697783 -0.6237812 -0.112185114 -0.065653328
## Ht        -0.14713824 0.11635254 -0.6222619 -0.065799249 -0.012946000
## Seated    -0.05981402 0.05145578 -0.2660020 -0.389895242  0.258653813
## Arm       -0.03365603 0.04071191 -0.1164420  0.468507883 -0.709741656
## Thigh     -0.03724658 0.02253667 -0.1997256  0.780908545  0.561840459
## Leg       -0.04358321 0.02867679 -0.1280023  0.041603736 -0.329433743
## hipcenter  0.87919851 0.46083663 -0.1205071 -0.001209175 -0.002688808
##                    PC6          PC7           PC8
## Weight     0.005509582 -0.003279097 -0.0011280823
## HtShoes    0.181318086 -0.207175677 -0.6950265610
## Ht         0.198254759 -0.145455118  0.7159911154
## Seated    -0.769573376  0.334440438 -0.0019546262
## Arm       -0.490594507 -0.138555428  0.0231871878
## Thigh     -0.063222031  0.161099967 -0.0527388776
## Leg        0.301354075  0.882525559 -0.0311521881
## hipcenter  0.007140811  0.007597360  0.0004640582
```

Perform PCA with scaling

```
pca_scaled <- prcomp(X, scale = TRUE)
U_scaled <- pca_scaled$rotation

# Display the loading matrix U with scaling
print("Loading matrix U with scaling:")
```

```
## [1] "Loading matrix U with scaling:"
```

```
U_scaled
```

```
##                  PC1         PC2         PC3        PC4        PC5         PC6
## Weight    -0.3435122  0.13968422  0.53465981  0.1666215 -0.7393109  0.02704100
## HtShoes   -0.3896381  0.11387569 -0.01748478  0.1573840  0.1996393 -0.02128091
## Ht        -0.3902742  0.10114801 -0.03089331  0.1621804  0.1828684 -0.03316556
## Seated    -0.3608640  0.34074377 -0.01043241  0.3979390  0.3582144  0.37138004
```

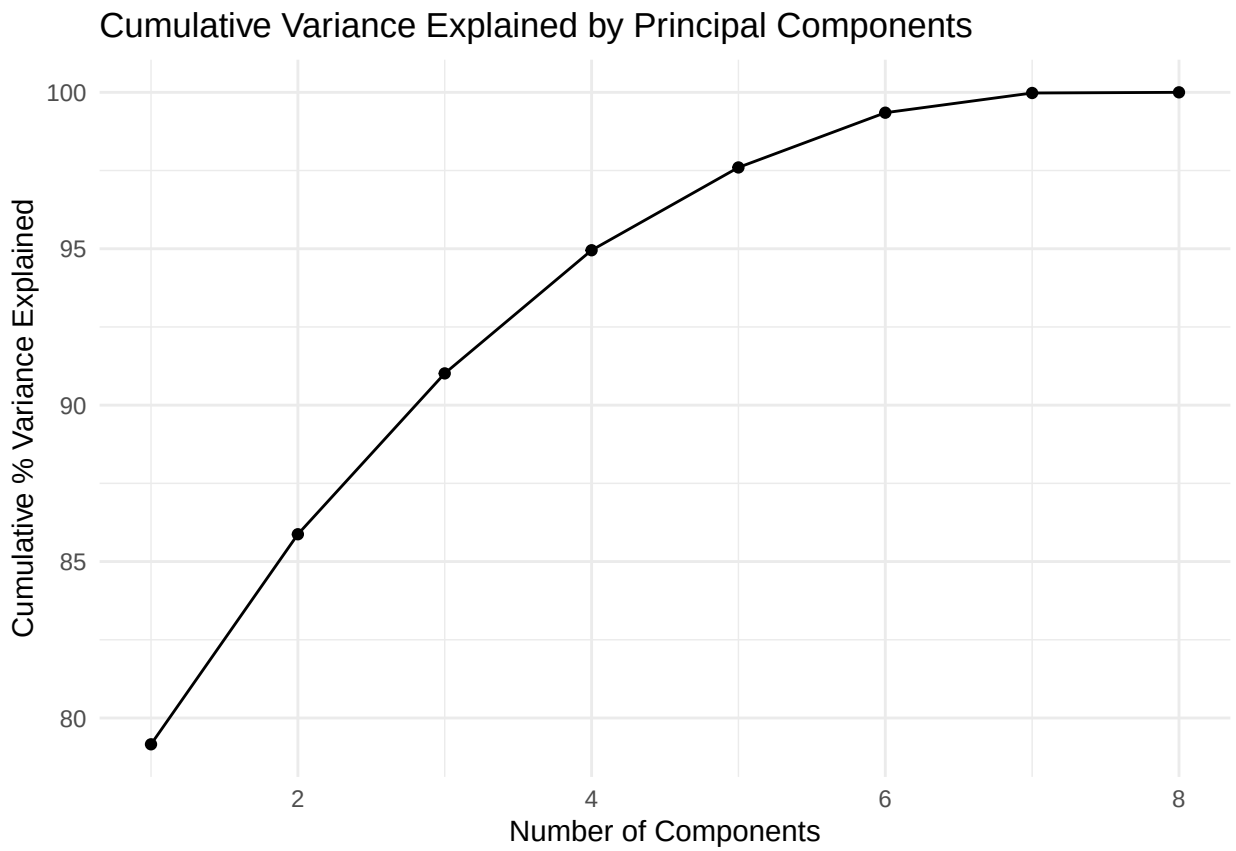
```
## Arm       -0.3247523 -0.51890498  0.44575448 -0.4817514  0.2817000  0.33502343
## Thigh     -0.3075388 -0.69746116 -0.44960224  0.3740599 -0.2248431 -0.07564427
## Leg       -0.3707682  0.09990584  0.04636183 -0.2553677  0.1541699 -0.81340916
## hipcenter  0.3318607 -0.27475039  0.55658000  0.5706064  0.3132529 -0.28318827
##                   PC7           PC8
## Weight     0.03923354 -0.0035692756
## HtShoes   -0.53377218 -0.6963602389
## Ht        -0.51089094  0.7173896502
## Seated     0.57355523 -0.0004552243
## Arm        0.05344395  0.0063375398
## Thigh      0.14254812 -0.0175609799
## Leg        0.31578406 -0.0080812142
## hipcenter -0.02602013  0.0027752603
```

1. (b) Yes, scaling helps in this case. Since the variables likely have different units and ranges (e.g., Weight versus Arm or Leg length), scaling normalizes their contribution to the principal components. This prevents variables with larger scales from dominating the analysis, providing a more interpretable and balanced principal component structure.

I'll explain this by looking at two charts. Chart1: Without Scaling: The loadings for each principal component are heavily influenced by the variables with larger absolute values (like Weight and hipcenter). This is because variables with larger variances tend to dominate the first few components in unscaled PCA, leading to biased interpretations. Chart2: With Scaling: After scaling, each variable has a mean of zero and a standard deviation of one, so each variable contributes equally to the principal components regardless of their original units. The loadings are now more balanced, and the PCA reflects the underlying structure of the data rather than being skewed by the scale of individual variables.

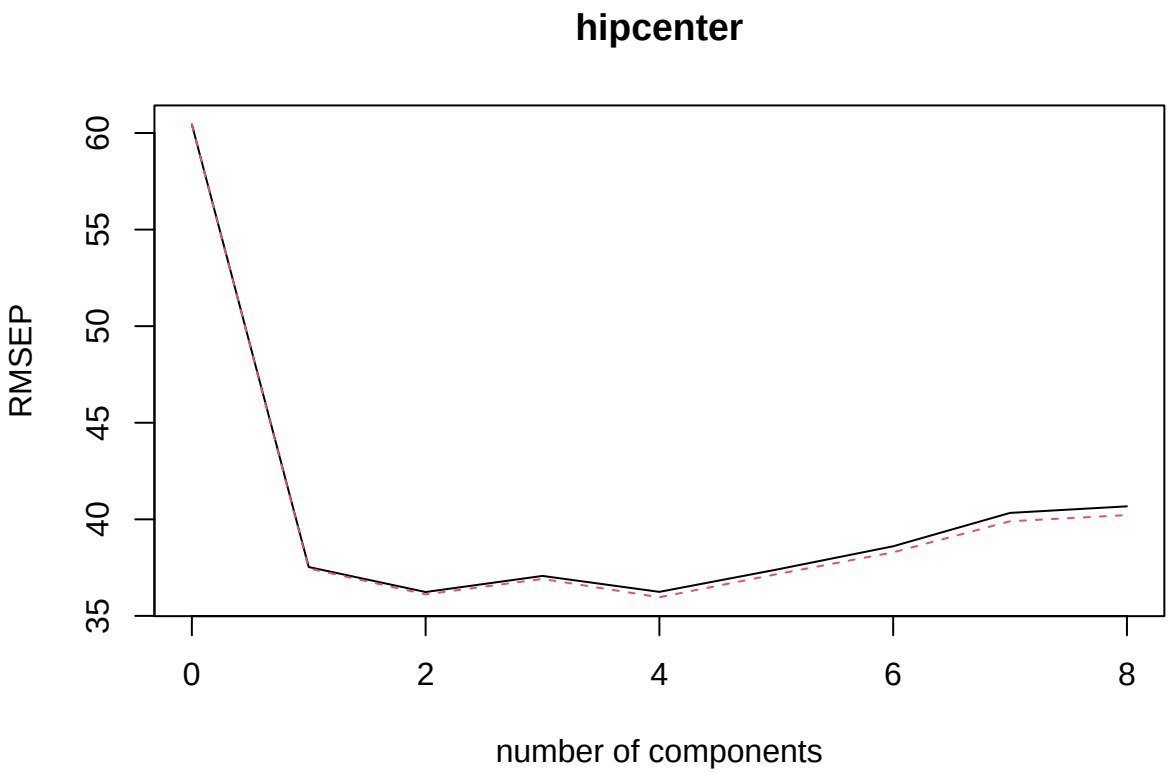
1. (c)

```r
# Calculate % variance explained by each component
var_explained <- pca_scaled$sdev^2 / sum(pca_scaled$sdev^2) * 100
cum_var_explained <- cumsum(var_explained)

# Plot cumulative variance explained
ggplot(data.frame(Components = 1:length(var_explained), Variance = cum_var_explained), aes(x = Componen
  geom_line() + geom_point() +
  labs(title = "Cumulative Variance Explained by Principal Components", x = "Number of Components", y =
  theme_minimal()
```



1.(d)

```r
# Perform PCR with cross-validation to select optimal number of components
pcr_model <- pcr(hipcenter ~ ., data = seatpos, scale = TRUE, validation = "CV")

# Plot RMSEP to find optimal components
validationplot(pcr_model, val.type = "RMSEP")
```




```
# Choose the optimal number of components based on cross-validation (e.g., the point where RMSEP is min
summary(pcr_model)
```

```
## Data:    X dimension: 38 8
##  Y dimension: 38 1
## Fit method: svdpc
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           60.45    37.53    36.24    37.07    36.24    37.39    38.61
## adjCV        60.45    37.46    36.11    36.91    35.97    37.16    38.29
##        7 comps  8 comps
## CV       40.33    40.67
## adjCV    39.90    40.22
##
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X            70.91    86.37    92.17    95.18    97.61    99.35    99.98
## hipcenter    61.89    66.34    66.48    67.82    68.02    68.58    68.63
##            8 comps
## X           100.00
## hipcenter    68.66
```

Problem 2:

```
# Load fat data
data(fat)
?fat # Inspect the dataset

# Define response and predictor variables
y_fat <- fat$brozek
X_fat <- as.matrix(fat[, -1]) # Convert to matrix for glmnet compatibility
```

```
# Ridge regression with cross-validation to find optimal lambda
ridge_cv_fat <- cv.glmnet(X_fat, y_fat, alpha = 0)
ridge_model_fat <- glmnet(X_fat, y_fat, alpha = 0, lambda = ridge_cv_fat$lambda.min)

# Display Ridge coefficients for fat data
print("Ridge Regression Coefficients:")
```

```
## [1] "Ridge Regression Coefficients:"
```

```
coef(ridge_model_fat)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  1.376150e+02
## siri         3.815850e-01
## density     -1.355276e+02
## age          8.711004e-03
## weight       1.817193e-02
## height       3.174374e-02
## adipos       4.035218e-02
## free        -7.816695e-02
## neck         1.460480e-02
## chest        4.384218e-02
## abdom        6.273469e-02
## hip          3.752830e-02
## thigh        4.434819e-02
## knee         5.536282e-02
## ankle        1.358959e-02
## biceps       1.479529e-02
## forearm      6.869092e-02
## wrist       -3.560631e-02
```

```
# Calculate and display RMSE for Ridge Regression
ridge_rmse <- rmse(y_fat, predict(ridge_model_fat, X_fat, s = ridge_cv_fat$lambda.min))
paste("RMSE for Ridge Regression:", ridge_rmse)
```

```
## [1] "RMSE for Ridge Regression: 0.701227608975924"
```

Results for Ridge Regression: - Coefficients: Ridge regression retained all predictors with shrunk coefficients. - RMSE: 0.7012.

All predictors have non-zero coefficients because Ridge regression shrinks the coefficients towards zero but doesn't set them to zero. Ridge retains all predictors with reduced magnitudes, useful when we believe most predictors contribute to the response.

```
# LASSO regression with cross-validation to find optimal lambda
lasso_cv_fat <- cv.glmnet(X_fat, y_fat, alpha = 1)
lasso_model_fat <- glmnet(X_fat, y_fat, alpha = 1, lambda = lasso_cv_fat$lambda.min)

# Display LASSO coefficients for fat data
print("LASSO Regression Coefficients:")
```

```
## [1] "LASSO Regression Coefficients:"
```

```
coef(lasso_model_fat)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept)  1.9690262
## siri         0.9049931
## density     -0.3428184
## age          .
## weight       .
## height       .
## adipos       .
## free         .
## neck         .
## chest        .
## abdom        .
## hip          .
## thigh        .
## knee         .
## ankle        .
## biceps       .
## forearm      .
## wrist        .
```

```
# Calculate and display RMSE for LASSO Regression
lasso_rmse <- rmse(y_fat, predict(lasso_model_fat, X_fat, s = lasso_cv_fat$lambda.min))
paste("RMSE for LASSO Regression:", lasso_rmse)
```

```
## [1] "RMSE for LASSO Regression: 0.242712168504842"
```

Results for LASSO Regression: - Coefficients: LASSO selected only siri and density as important predictors, setting all other coefficients to zero. - RMSE: 0.2427 (LASSO shows a better fit than Ridge in this case. The model performs well.)

LASSO eliminates less important predictors, performing both variable selection and regularization, which is helpful when you suspect that only a few predictors are truly important.