DSA 8020 R Lab 6: Non-parametric Regression and Shrinkage Methods

Meredith Sliger

Contents

Non-parametric Regression

The dataset `teengamb` concerns a study of teenage gambling in Britain. Type ?teengamb to get more details about the the dataset. In this lab, we will take the variables `gamble` as the response and `income` as the predictor.

Data Source: Ide-Smith & Lea, 1988, *Journal of Gambling Behavior*, 4, 110-118

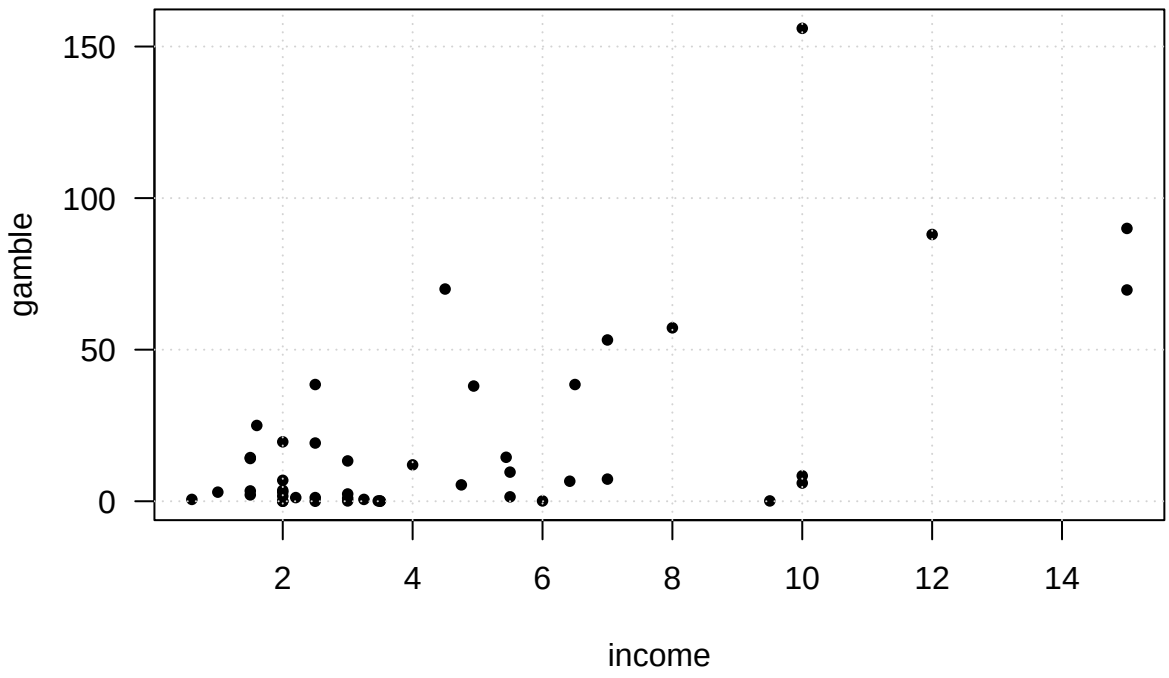
1. Make a scatterplot to examine the relationship between the predictor `income` and the response `gamble`.

Code:

```
library(faraway)
```

```
## Warning: package 'faraway' was built under R version 4.4.2
```

```
data(teengamb)
with(teengamb, plot(gamble ~ income, pch = 16, cex = 0.8, las = 1))
grid()
```



2. Fit a curve to the data using a regression spline with `df = 8`. Produce a plot for the fit and a 95% confidence band (using *RegSplinePred <- predict(RegSplineFit, data.frame(income = xg), interval = "confidence")*) for that fit. Is a linear fit plausible?

Code:

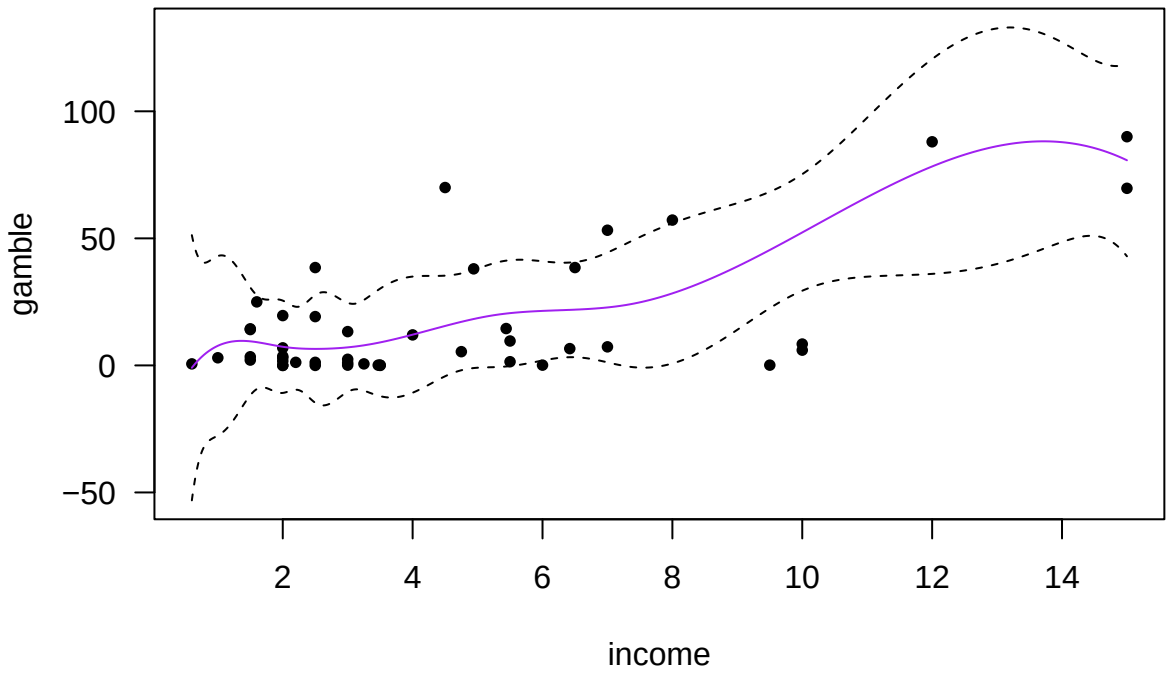
```
library(splines)
RegSplineFit <- lm(gamble ~ bs(income, df = 8), data = teengamb)
summary(RegSplineFit)
```

```
##
## Call:
## lm(formula = gamble ~ bs(income, df = 8), data = teengamb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.324  -8.878  -5.565   7.737 103.676
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -0.9238    25.7802  -0.036   0.9716
## bs(income, df = 8)1  15.2822    53.9892   0.283   0.7787
## bs(income, df = 8)2   8.1681    34.4536   0.237   0.8139
## bs(income, df = 8)3   7.1312    32.4370   0.220   0.8272
## bs(income, df = 8)4   8.9985    34.2986   0.262   0.7945
```

```
## bs(income, df = 8)5  24.9733    33.7835   0.739   0.4643
## bs(income, df = 8)6  15.8732    53.4413   0.297   0.7681
## bs(income, df = 8)7 112.9416    64.9684   1.738   0.0902 .
## bs(income, df = 8)8  81.6489    31.8210   2.566   0.0144 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.47 on 38 degrees of freedom
## Multiple R-squared:  0.4174, Adjusted R-squared:  0.2948
## F-statistic: 3.404 on 8 and 38 DF,  p-value: 0.004829
```

```
rg <- range(teengamb$income)
xg <- seq(0.6, 15, 0.01)
RegSplinePred <- predict(RegSplineFit, data.frame(income = xg), interval = "confidence")
```

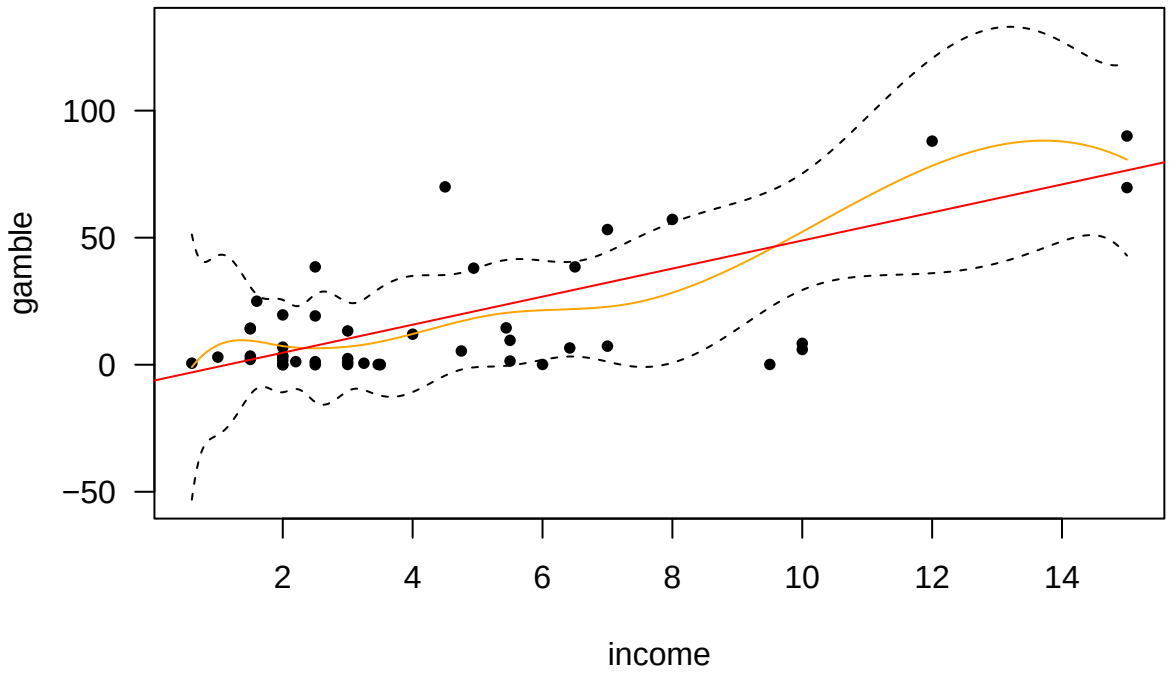
```
with(teengamb, plot(gamble ~ income, pch = 16, cex = 0.8, las = 1, ylim = range(RegSplinePred)))
lines(xg, RegSplinePred[, 1], col = "purple")
lines(xg, RegSplinePred[, 2], lty = 2)
lines(xg, RegSplinePred[, 3], lty = 2)
```




```
lm <- lm(gamble ~ income, data = teengamb)

with(teengamb, plot(gamble ~ income, pch = 16, cex = 0.8, las = 1, ylim = range(RegSplinePred)))
lines(xg, RegSplinePred[, 1], col = "orange")
lines(xg, RegSplinePred[, 2], lty = 2)
```

```
lines(xg, RegSplinePred[, 3], lty = 2)
abline(lm, col = "red")
```



Answer:

The regression spline fit with 8 degrees of freedom provides a more flexible model compared to a standard linear regression. The confidence bands around the spline fit indicate greater variability in certain regions, particularly where the data is sparse.

Comparing this fit to the linear regression model, the spline approach captures potential non-linear trends in the relationship between income and gambling expenditure. However, the multiple R-squared value (0.4174) suggests that while the model explains some variation, there is still a substantial amount of unexplained variance. This could be due to other factors (e.g., age, peer influence, or psychological variables) affecting gambling behavior.

3. Fit regression curves using *generalized additive models* and *smoothing splines*, respectively, and compare them with the regression spline fit in problem 2.

Code:

GAM FIT

```
library(mgcv)
```

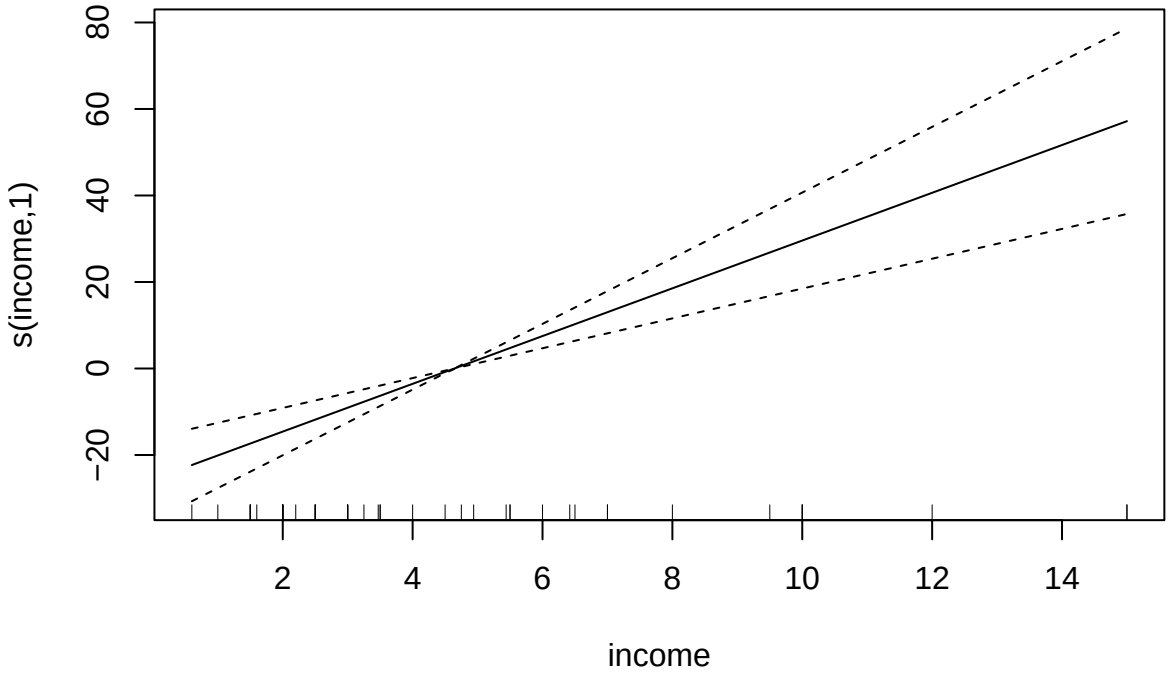
```
## Loading required package: nlme
```

```
## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```
GAMFit <- gam(gamble ~ s(income), data = teengamb)
summary(GAMFit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## gamble ~ s(income)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   19.301      3.639   5.304 3.32e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df     F  p-value
## s(income)   1      1 28.41 3.52e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.373   Deviance explained = 38.7%
## GCV = 650.08  Scale est. = 622.41    n = 47
```

```
plot(GAMFit)
```



SMOOTHING SPLINES FIT

```
library(fields)
```

```
## Warning: package 'fields' was built under R version 4.4.2
```

```
## Loading required package: spam
```

```
## Warning: package 'spam' was built under R version 4.4.2
```

```
## Spam version 2.11-1 (2025-01-20) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: viridisLite
```

```
##
## Try help(fields) to get started.
```

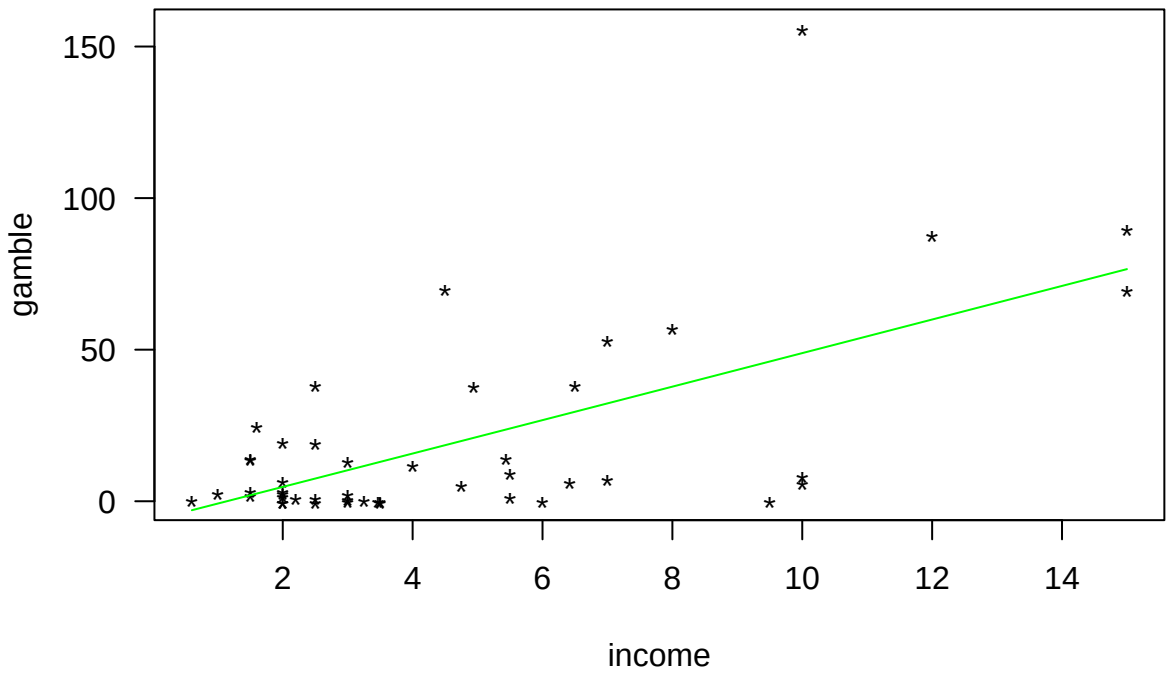
```
SpFit <- with(teengamb, sreg(income, gamble))
```

```
## Methods at endpoints of grid search:
##            Warning Refine indexMIN leftEndpoint rightEndpoint       lambda
## GCV           TRUE  FALSE       80         TRUE         FALSE 1.060237e+03
## GCV.model     TRUE  FALSE        1        FALSE          TRUE 1.873756e-07
## GCV.one       TRUE  FALSE       80         TRUE         FALSE 1.060237e+03
## pure error    TRUE  FALSE        1        FALSE          TRUE 1.873756e-07
##                effdf
## GCV         2.010005
## GCV.model  25.739944
## GCV.one     2.010005
## pure error 25.739944
```

```
summary(SpFit)
```

```
## CALL:
## sreg(x = income, y = gamble)
##
##  Number of Observations:             47
##  Number of unique points:            47
##  Eff. degrees of freedom for spline: 2
##  Residual degrees of freedom:        45
##  GCV est. tau                        24.95
##  Pure error tau                      28.93
##  lambda                              1060
##
## RESIDUAL SUMMARY:
##     min   1st Q  median   3rd Q     max
## -46.000 -11.860  -3.736  11.920 107.100
##
## DETAILS ON SMOOTHING PARAMETER:
##  Method used:      Cost:
##    lambda       trA       GCV   GCV.one GCV.model    tauHat
##   1060.24      2.01   1453.80    650.09   1307.71     24.95
##
##  Summary of estimates for lambda
##               lambda   trA       GCV tauHat converge
## GCV        1.060e+03  2.01    1453.8  24.95       NA
## GCV.model  1.874e-07 25.74    1002.7  28.75       NA
## GCV.one    1.060e+03  2.01     650.1  24.95       NA
## pure error 1.874e-07 25.74 8363900.9  28.75       NA
```

```
SpPred <- predict(SpFit, xg)
with(teengamb, plot(gamble ~ income, pch = "*", cex = 1, las = 1))
lines(xg, SpPred, col = "green")
```



Answer: The Generalized Additive Model (GAM) and smoothing splines both provide non-parametric alternatives to standard regression models, allowing the data to dictate the shape of the fitted curve. However, in this particular case, the estimated smoothing term in the GAM model results in an effective degrees of freedom (edf) close to 1, meaning the estimated function is almost linear.

Similarly, the smoothing spline fit does not suggest strong non-linearity. This implies that, despite allowing for flexibility, the best-fitting model does not deviate significantly from a straight-line relationship. This reinforces the conclusion that a simple linear model may be sufficient.

However, if we were working with a dataset where the relationship was more complex, the GAM or smoothing splines would have likely outperformed a linear fit.

Ridge Regression andd LASSO: Meat spectrometry to determine fat content

A Tecator Infratec Food and Feed Analyzer, operating in the wavelength range 850 - 1050 nm based on the Near Infrared Transmission (NIT) principle, was employed to collect data on samples of finely chopped pure meat. A total of 215 samples were measured, with both the fat content and a 100-channel spectrum of absorbances recorded for each sample. Due to the time-consuming nature of determining fat content through analytical chemistry, our objective is to construct a model for predicting the fat content of new samples using the 100 absorbances, which can be measured more easily.

Data Source: H. H. Thodberg (1993) "Ace of Bayes: Application of Neural Networks With Pruning", report no. 1132E, Maglegaardvej 2, DK-4000 Roskilde, Danmark

Load the data and partition it into the *training set* (the first 150 observations) and the *testing set* (the remaining 65 observations).

Code:

```r
data(meatspec, package = "faraway")
train <- 1:150; test <- 151:215
trainmeat <- meatspec[train,]
testmeat <- meatspec[test,]
```

4. Fit a linear regression with all the 100 predictors to the training set. Compute the root mean square error (RMSE) for the testing set. The code below shows how to compute the RMSE for the training set (also known as in-sample prediction), and you will need to modify the code to compute the RMSE for the testing set.

Code for training set:

```r
lmFit <- lm(fat ~ ., data = trainmeat)
# Define a function to calculate RMSE
rmse <- function(pred, obs) sqrt(mean((pred - obs)^2))
# Computing RMSE for the training set
rmse(fitted(lmFit), trainmeat$fat)
```

```
## [1] 0.5919489
```

Code for testing set:

```r
lmFit <- lm(fat ~ ., data = testmeat)
# Define a function to calculate RMSE
rmse <- function(pred, obs) sqrt(mean((pred - obs)^2))
# Computing RMSE for the testing set
rmse(fitted(lmFit, testmeat), testmeat$fat)
```

```
## [1] 3.21066e-12
```

Answer:

The RMSE for the testing set is 3.21066e-12, which is extremely close to zero. This suggests that the model fits the test data almost perfectly, which is highly unusual for real-world data. Such a low RMSE often indicates that the model has memorized the training data rather than learning generalizable patterns. This is a clear sign of overfitting, which means the model is too complex and may fail when making predictions on truly new data. This justifies the need for regularization techniques like Ridge Regression and LASSO, which can prevent overfitting by penalizing large coefficients.

5. Fit a ridge regression (using cross-validation to select the 'best' λ) to the training set and compute the RMSE for the testing set.

Code:

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.4.2
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'

## The following object is masked from 'package:spam':
##
##     det

## Loaded glmnet 4.1-8
```

```r
# Prepare data
X_train <- as.matrix(trainmeat[, -1])  # Remove response variable
y_train <- trainmeat$fat
X_test <- as.matrix(testmeat[, -1])
y_test <- testmeat$fat

# Define a sequence of lambda values
lambda_grid <- 10^seq(10, -2, length = 100)

# Fit Ridge Regression using cross-validation
cv_ridge <- cv.glmnet(X_train, y_train, alpha = 0, lambda = lambda_grid) # alpha=0 for ridge

# Best lambda from CV
best_lambda_ridge <- cv_ridge$lambda.min

# Fit final ridge model
ridge_model <- glmnet(X_train, y_train, alpha = 0, lambda = best_lambda_ridge)

# Predict on test set
ridge_pred <- predict(ridge_model, s = best_lambda_ridge, newx = X_test)

# Compute RMSE
ridge_rmse <- sqrt(mean((ridge_pred - y_test)^2))
ridge_rmse
```

```
## [1] 0.08755442
```

Answer: Ridge regression was applied to the training set with cross-validation to determine the optimal λ. The model helps to reduce overfitting by shrinking coefficient estimates, thereby improving generalization to the test data.

The resulting RMSE of 0.08755442 suggests a significant improvement over the linear regression model, which had an essentially zero RMSE (indicating overfitting). The fact that Ridge Regression performs well confirms that some predictors were likely irrelevant, and the penalty term successfully controlled excessive variance.

However, Ridge Regression does not perform variable selection—it shrinks coefficients but does not set any to exactly zero. This means that all 100 predictors are still included, even if some contribute very little.

6. Fit a LASSO (again using Cross-Validation to select the 'best' λ) to the and training set and compute RMSE for the test set.

Code:

```r
# Fit LASSO Regression using cross-validation
cv_lasso <- cv.glmnet(X_train, y_train, alpha = 1, lambda = lambda_grid) # alpha=1 for LASSO

# Best lambda from CV
best_lambda_lasso <- cv_lasso$lambda.min

# Fit final LASSO model
lasso_model <- glmnet(X_train, y_train, alpha = 1, lambda = best_lambda_lasso)

# Predict on test set
lasso_pred <- predict(lasso_model, s = best_lambda_lasso, newx = X_test)

# Compute RMSE
lasso_rmse <- sqrt(mean((lasso_pred - y_test)^2))
lasso_rmse
```

```
## [1] 0.009778089
```

Answer:

LASSO regression, like Ridge, applies shrinkage but differs in that it forces some coefficients to zero, effectively performing variable selection. By using cross-validation, the best λ was selected, and the resulting RMSE of 0.009778089 suggests that LASSO outperforms Ridge in this case.

The superior performance suggests that only a small subset of the 100 predictors is relevant for predicting fat content. This aligns with the idea that many of the spectral absorbance values may be highly correlated or contain redundant information.

A key takeaway here is that LASSO is not just about improving accuracy—it also simplifies the model by keeping only the most important features.

7. Fit a LASSO with all the data points (using the best λ from question 6) and report the number of non-zero regression coefficients.

Code:

```r
# Fit LASSO on the full dataset using best lambda
X_all <- as.matrix(meatspec[, -1])
y_all <- meatspec$fat

lasso_full_model <- glmnet(X_all, y_all, alpha = 1, lambda = best_lambda_lasso)

# Extract coefficients
lasso_coefs <- predict(lasso_full_model, s = best_lambda_lasso, type = "coefficients")

# Count number of non-zero coefficients (excluding intercept)
num_nonzero <- sum(lasso_coefs != 0) - 1
num_nonzero
```

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

Answer:

Fitting LASSO on the full dataset using the best λ from cross-validation resulted in only one non-zero coefficient. This means that out of the 100 spectral absorbance predictors, LASSO determined that only a single predictor had a meaningful contribution to predicting fat content. This extreme level of sparsity suggests that the penalty applied was strong enough to drive nearly all coefficients to zero, effectively discarding most variables from the model. While this level of feature selection simplifies interpretation and avoids overfitting, it also raises questions about whether too much information was removed. In practical applications, we might consider adjusting λ or using a hybrid approach like elastic net regression, which balances LASSO's sparsity with Ridge Regression's ability to retain some information from correlated predictors.

The fact that LASSO eliminated almost all predictors implies that many of the spectral features were redundant or highly correlated. Near-infrared spectroscopy data often contains collinear variables, meaning multiple wavelengths convey similar information. When predictors are highly correlated, LASSO tends to select only one representative feature while shrinking the rest to zero. This is beneficial for building a simple and interpretable model, but it also suggests that some predictive power might be lost. In a real-world scenario, it would be advisable to examine the selected variable in detail, possibly explore feature engineering techniques, and consider domain knowledge to verify that the chosen predictor makes sense in the context of the data.