Regression Summative

Amber Chang

Preliminaries

We investigate a dataset introduced in a publication in Nature by Alizadeh (2000). This dataset reports gene expression profiles (7399 genes) of 240 patients with B-cell Lymphoma (a tumor that developes from B lymphocytes). The response variable corresponds to patient survival times (in years). So, this is a truly high-dimensional regression problem, with p = 7399 predictor variables, but only n = 240 observations. Please use the following steps to read in the data (you may need to install R package HCmodelSets first).

```
#install.packages("HCmodelSets")
require(mvtnorm)
```

```
## Loading required package: mvtnorm
```

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(survival)
```

```
## Loading required package: survival
```

```
require(HCmodelSets)
```

```
## Loading required package: HCmodelSets
```

```
data(LymphomaData)
```

```
names(patient.data)
```

```
## [1] "x"      "time"   "status"
```

```
dim(patient.data$x)
```

```
## [1] 7399  240
```

```
?dim
```

```
X <- t(patient.data$x)
colnames(X) <-paste("G", 1:dim(X)[2], sep="")
```

```
Time <- patient.data$time
X<-as.data.frame(X)
```

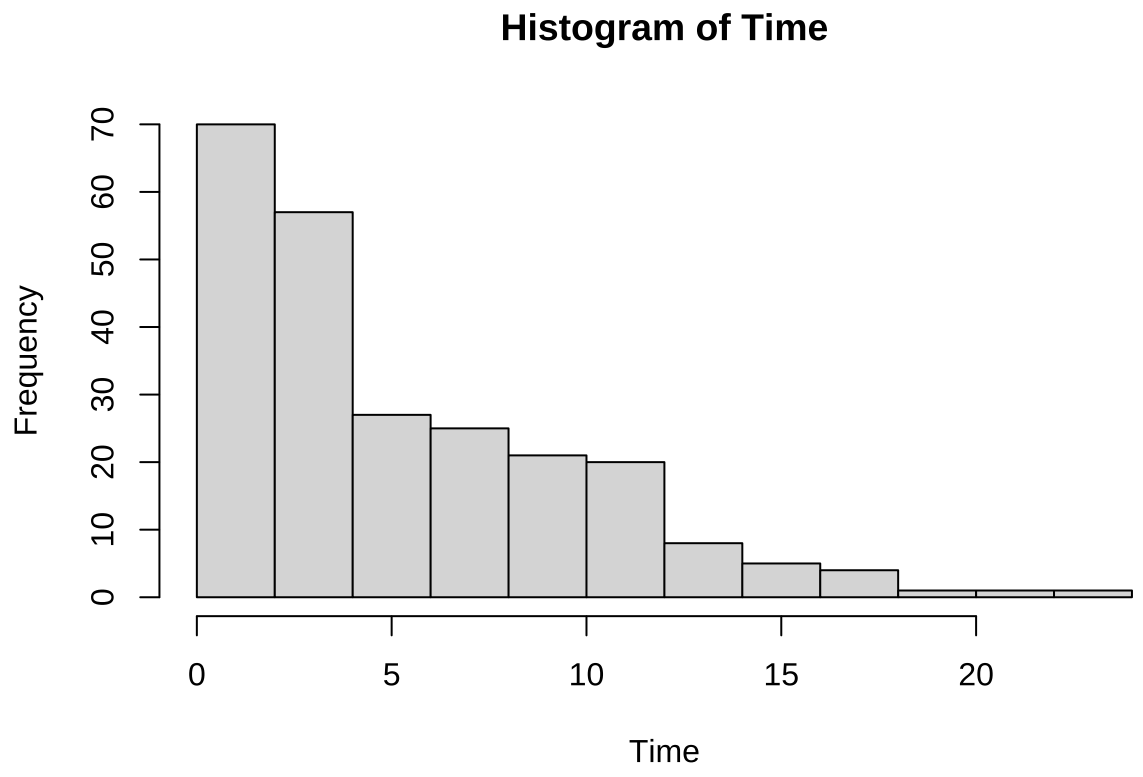
Task 1: Exploratory data analysis (15 marks)

Using appropriate graphical tools, carry out some exploratory analysis to gain a better understanding of the data. For instance, it could be useful to provide a histogram of the response (with explanation of your observation) and carry out principal component analysis (PCA) of the predictor space. Next, create a screeplot using the fviz_eig function from the factoextra package in R (you can find information about the factoextra package at http://www.sthda.com/english/wiki/factoextra-r-package-easy-multivariate- data-analyses-and-elegant-visualization). Explain this plot and determine how many principal components are required to capture 80% of the total variation.

```
dim(X)
```

```
## [1]  240 7399
```

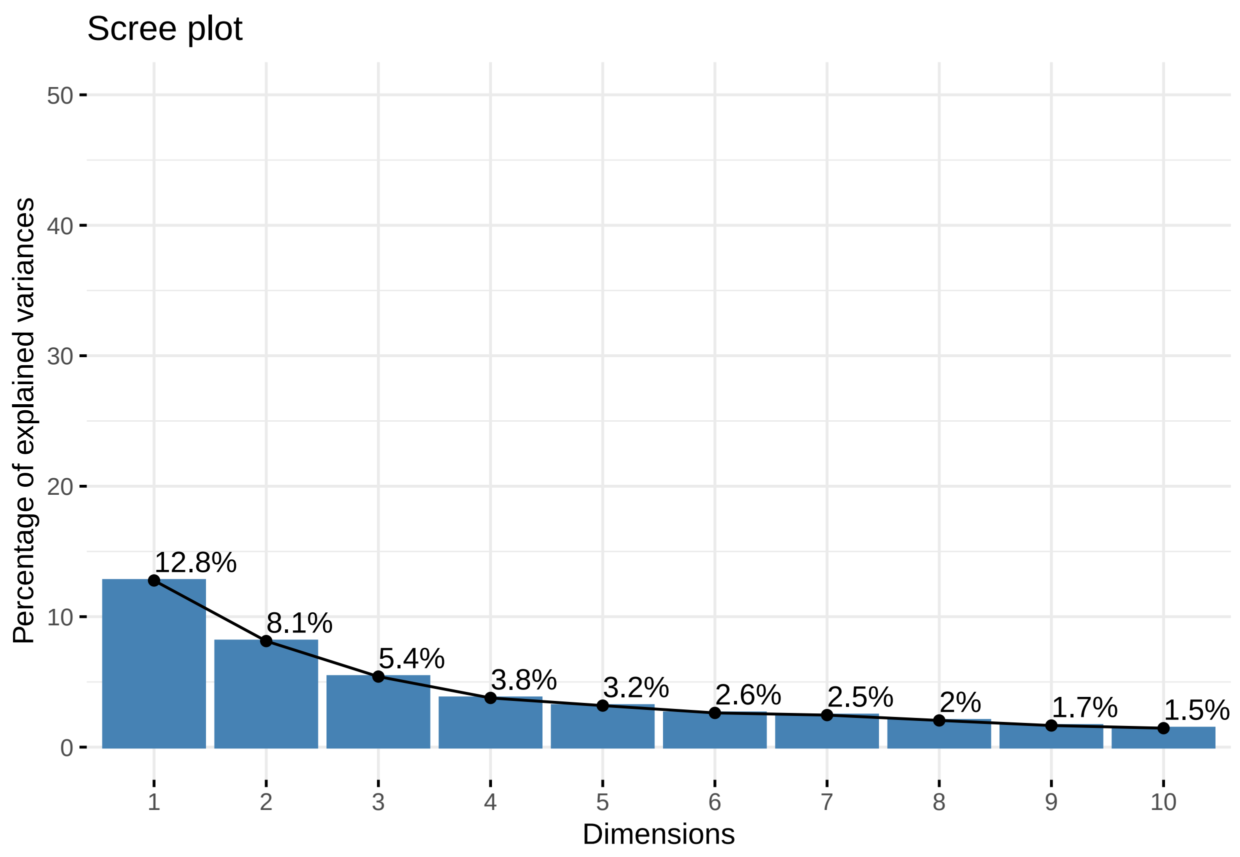
```
hist(Time)
```



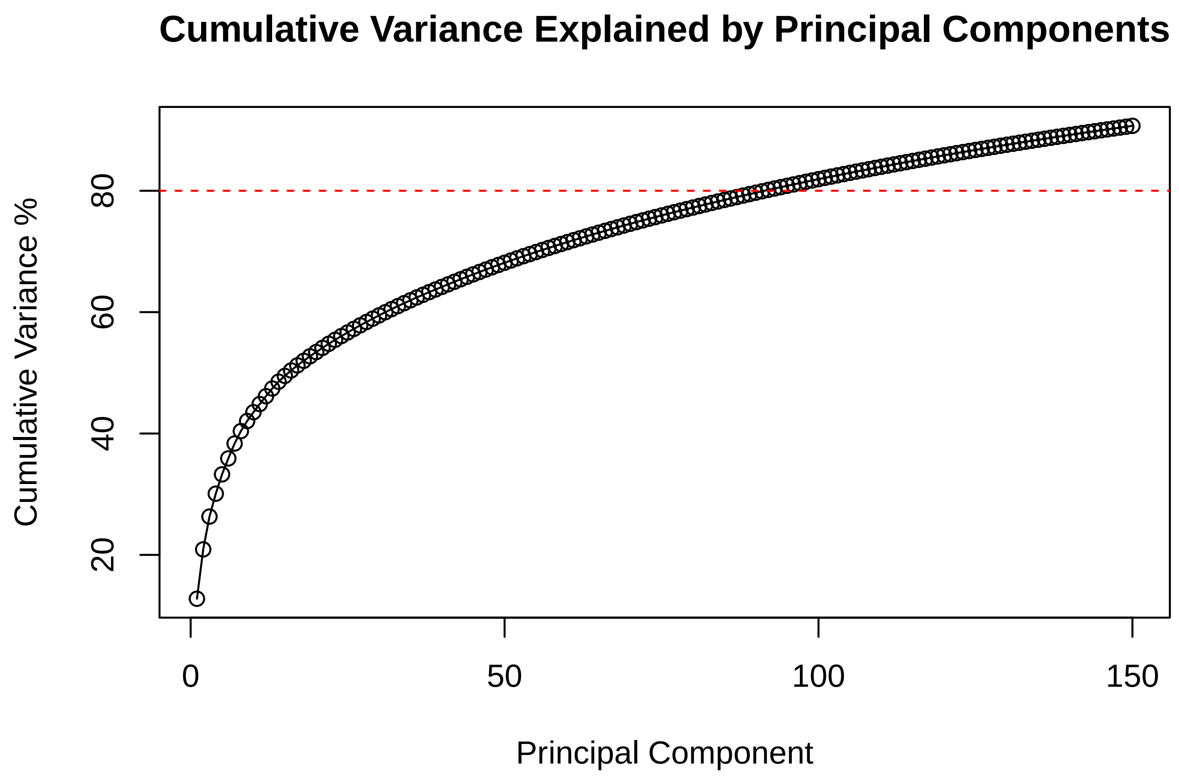
```r
# PCA
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
patient.pca <- prcomp(X,scale=TRUE)

fviz_eig(patient.pca, addlabels = TRUE, ylim = c(0, 50), k=20)
```



```r
table<-get_eigenvalue(patient.pca)
plot(table$cumulative.variance.percent[1:150], type = "o",
     xlab = "Principal Component", ylab = "Cumulative Variance %",
     main = "Cumulative Variance Explained by Principal Components")

# Draw a horizontal line at 80%
abline(h = 80, col = "red", lty = 2)
```



```
#fviz_contrib(patient.pca, choice = "ind", axes = 1:2, top=20)
```

answer From the screeplot above, the first ten dimensions only explains approximately 50 percent of the total variance. To achieve 80% of the information (variances) contained in the data are retained by the first 92 principal components according to the table specified above.

Task 2: The Lasso (15 marks)

We would like to reduce the dimension of the currently 7399-dimensional space of predictors. To this end, apply initially the (frequentist) LASSO onto this data set, using the R function glmnet. The outputs required are

- the trace plot of the fitted regression coefficients;
- a graphical illustration of the cross-validation to find lambda.

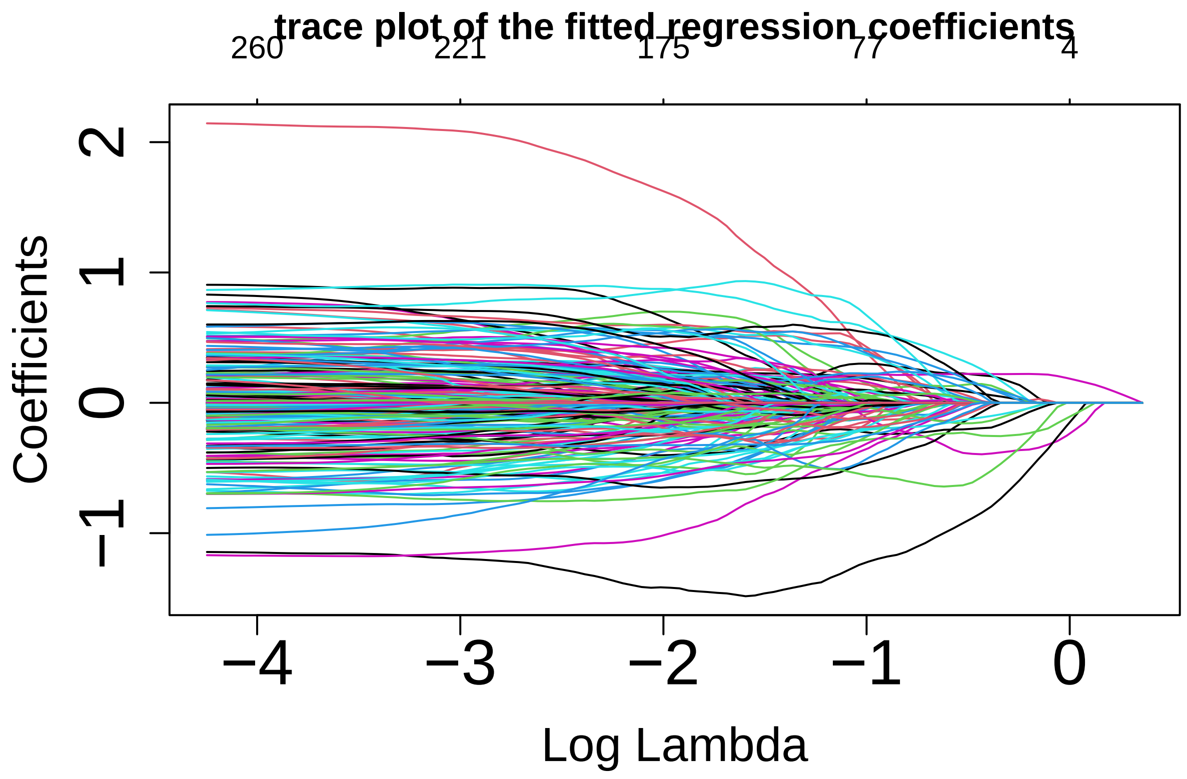
Provide a short statement interpreting the plots. Then, extract and report the value of lamda which minimizes the cross-validation criterion. How many genes are included into the model according to this choice?

```
require(glmnet)
```

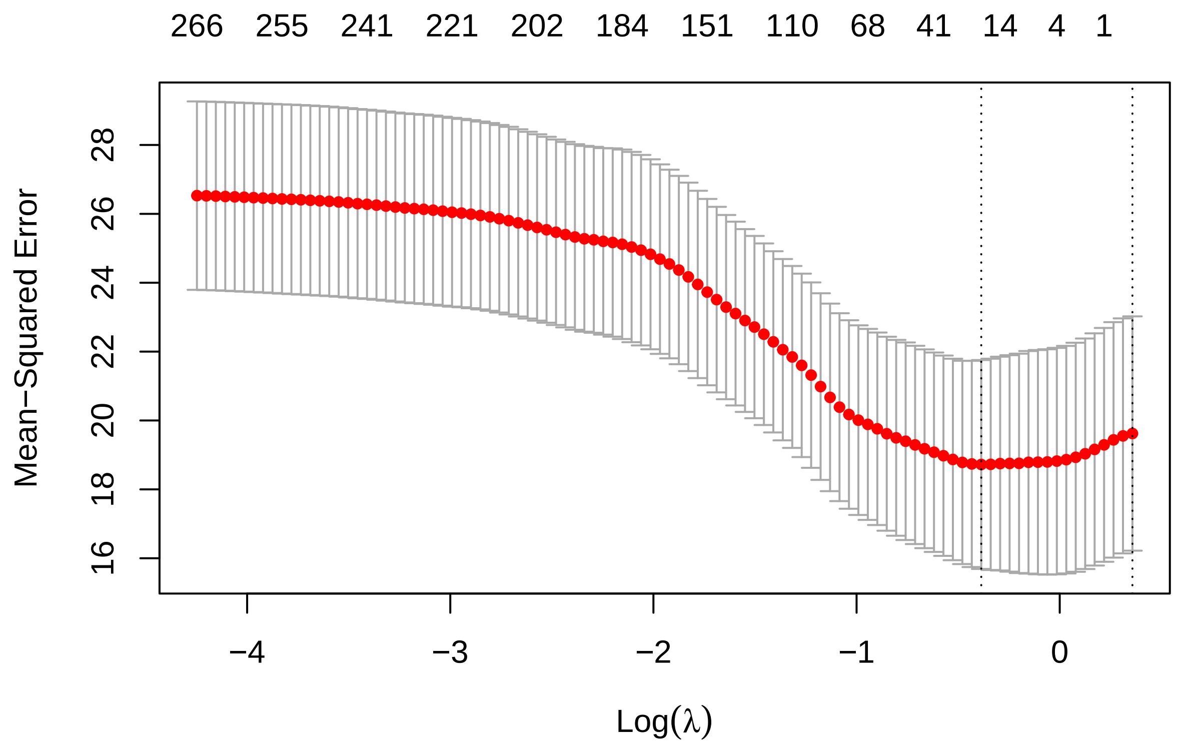
```
## Loading required package: glmnet
```

```
## Loading required package: Matrix
## Loaded glmnet 4.1-8
```

```
X = as.matrix(X)
patient.fit.time= glmnet(X,Time, alpha=1 )
plot(patient.fit.time, xvar="lambda",cex.axis=2, cex.lab=1.5,cex=1,main="trace plot of the fitted regres
```



```
#cross validation for finding best regularization term (lambda)
set.seed(2)
patient.cv.time = cv.glmnet(X,Time, alpha=1 )
plot(patient.cv.time)
```



```
lambda<-patient.cv.time$lambda.min
#cat("min lambda")
#log(lambda)
#lambda1<-patient.cv.time$lambda.1se
#cat("1se lambda")
#log(lambda1)
#cbind(coef(patient.cv.time, s="lambda.min"),coef(patient.cv.time, s="lambda.1se"))

#cat(length(which(coef(patient.cv.time, s="lambda.min")!=0)))
```

answer From the Trace plot of coefficients fit by lasso regression for gene diagnosis, we note that when lambda is close to 0, the coefficients shrink towards zero and towards each other.

19 genes are included into the model according to the value of log(lambda) which minimizes the cross-validation criterion. log(lambda) is -0.153718 . We chose not to use lambda.1se because very few gene will therefore be included to fit our model, which is far from ideal.

Task 3: Assessing cross-validation resampling uncertainty (20 marks)

We know that the output of the cross-validation routine is not deterministic. To shed further light on this, please carry out a simple experiment. Run the cross-validation and estimation routine for the (frequentist) LASSO 50 times, each time identifying the value of lambda which minimizes the cross-validation criterion, and each time recording which predictor variables have been selected by the Lasso. When finished, produce a table which lists how often each variable has been included. 2 Build a model which includes all variables which have been selected at least 25 (out of 50) times. Refit this model with the selected variables using ordinary least squares. (Benchmark: The value of R 2 of this model should not be worse than about 0.45, and your model should not make use of more than around 25 genes). Report the names of the selected genes (in terms of the notation defined in the Preliminaries) explicitly

```r
require(glmnet)
set.seed(200)
var_list <- list()
lambda_list<-list()

for(i in 1:50) {
  # Perform LASSO regression with cross-validation
  patient.cv.time <- cv.glmnet(X, Time, alpha = 1)

  lambda<-patient.cv.time$lambda.min

  selected_vars <- which(coef(patient.cv.time, s="lambda.min")!=0)
  selected_vars<-lapply(selected_vars, function(x) x - 1)
  selected_vars<-lapply(selected_vars, function(x) colnames(X)[x])

  lambda_list <- append(lambda_list,lambda )
  var_list <- append(var_list,selected_vars )

}

var_counts_df <- as.data.frame(table(unlist(var_list)))

# Print the table
print(var_counts_df)
```

```
##     Var1 Freq
## 1  G1020   48
## 2  G1021   30
## 3  G1055    3
## 4  G1259   20
## 5  G1456   50
## 6  G1825   50
## 7  G2694    6
## 8  G3395   47
## 9  G3807   47
## 10 G3820    1
## 11 G4099    6
## 12 G4131   50
## 13 G4248   47
## 14  G428    3
## 15 G4762   47
## 16 G5053   20
## 17 G5352   47
## 18 G5476   20
## 19 G5608   30
## 20 G5621   47
## 21 G5950   50
## 22 G6014   30
## 23 G6134   47
## 24 G6321   48
## 25 G6365   20
## 26 G6508   20
## 27 G6757   30
## 28 G7018    3
## 29 G7069    6
## 30 G7070   48
## 31 G7357    6
## 32 G7380    6
```

```
selected_variables <- var_counts_df[var_counts_df$Freq >= 25, "Var1"]
selected_variables<-c(selected_variables)
selected_variables
```

```
##  [1] G1020 G1021 G1456 G1825 G3395 G3807 G4131 G4248 G4762 G5352 G5608 G5621
## [13] G5950 G6014 G6134 G6321 G6757 G7070
## 32 Levels: G1020 G1021 G1055 G1259 G1456 G1825 G2694 G3395 G3807 ... G7380
```

```
selected_cols <- c(
  "G1020", "G1021", "G1456", "G1825", "G3395", "G3807",
  "G4131", "G4248", "G4762", "G5352", "G5608", "G5621",
  "G5950", "G6014", "G6134", "G6321", "G6757", "G7070"
)

# Subsetting the data frame to include only the selected columns
training_data <- as.data.frame(X[, selected_cols])

# Refit the model using OLS with selected variables
fit <- lm(Time ~ ., data = training_data)

# Calculate R-squared
rsquared <- summary(fit)$r.squared

rsquared
```

```
## [1] 0.4611514
```

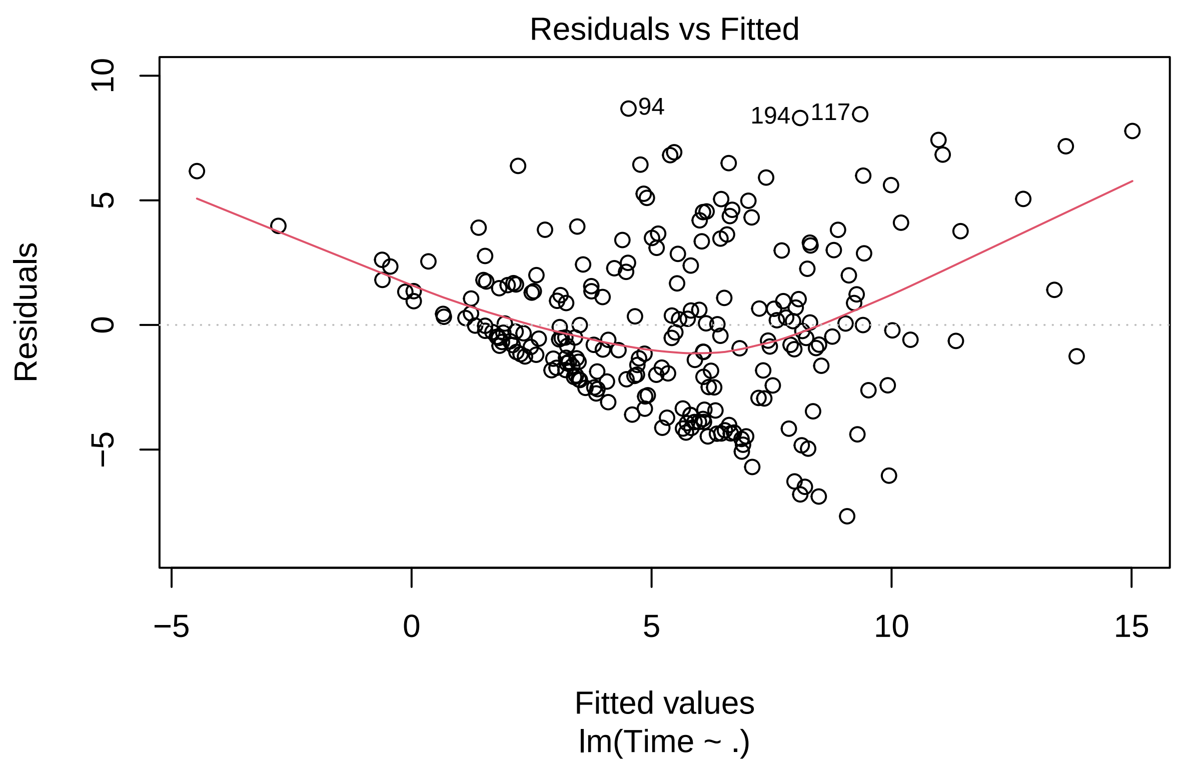
answer "G1020", "G1021", "G1456", "G1825", "G3395", "G3807", "G4131", "G4248", "G4762", "G5352", "G5608", "G5621", "G5950", "G6014", "G6134", "G6321", "G6757", "G7070"

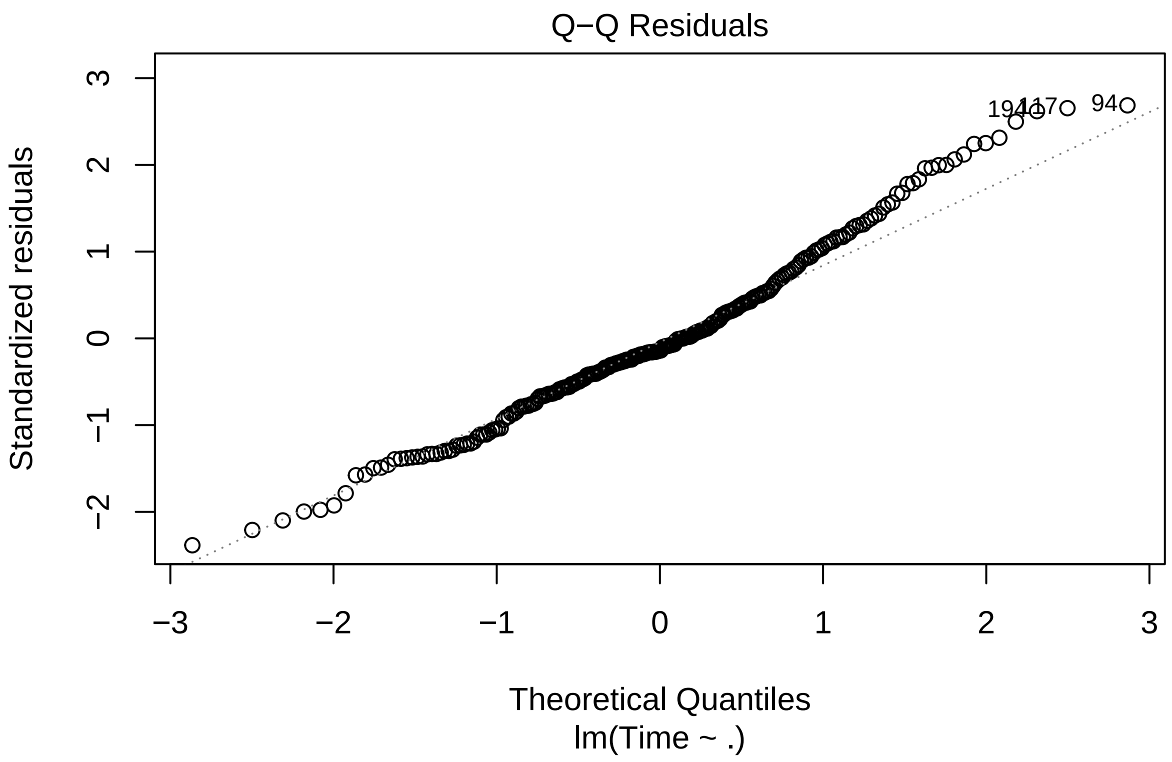
The refitted lasso regression model's Rˆ2 is 0.46

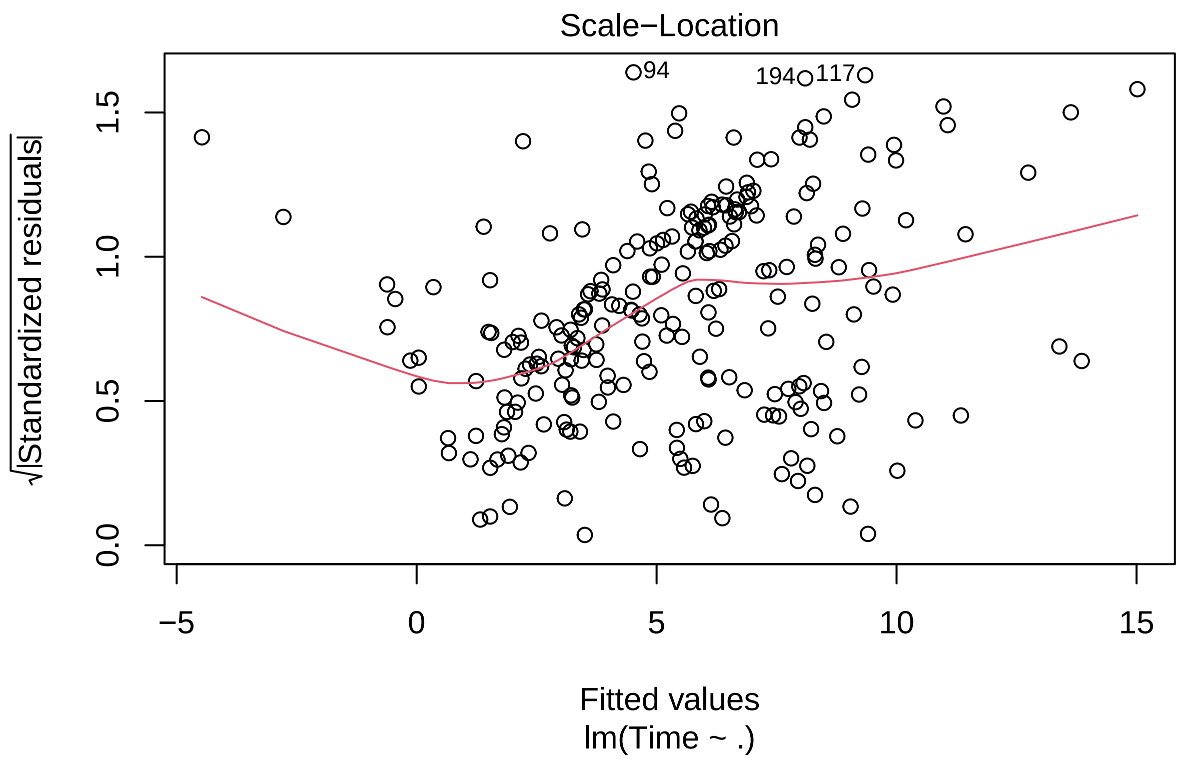
Task 4: Diagnostics (15 marks)

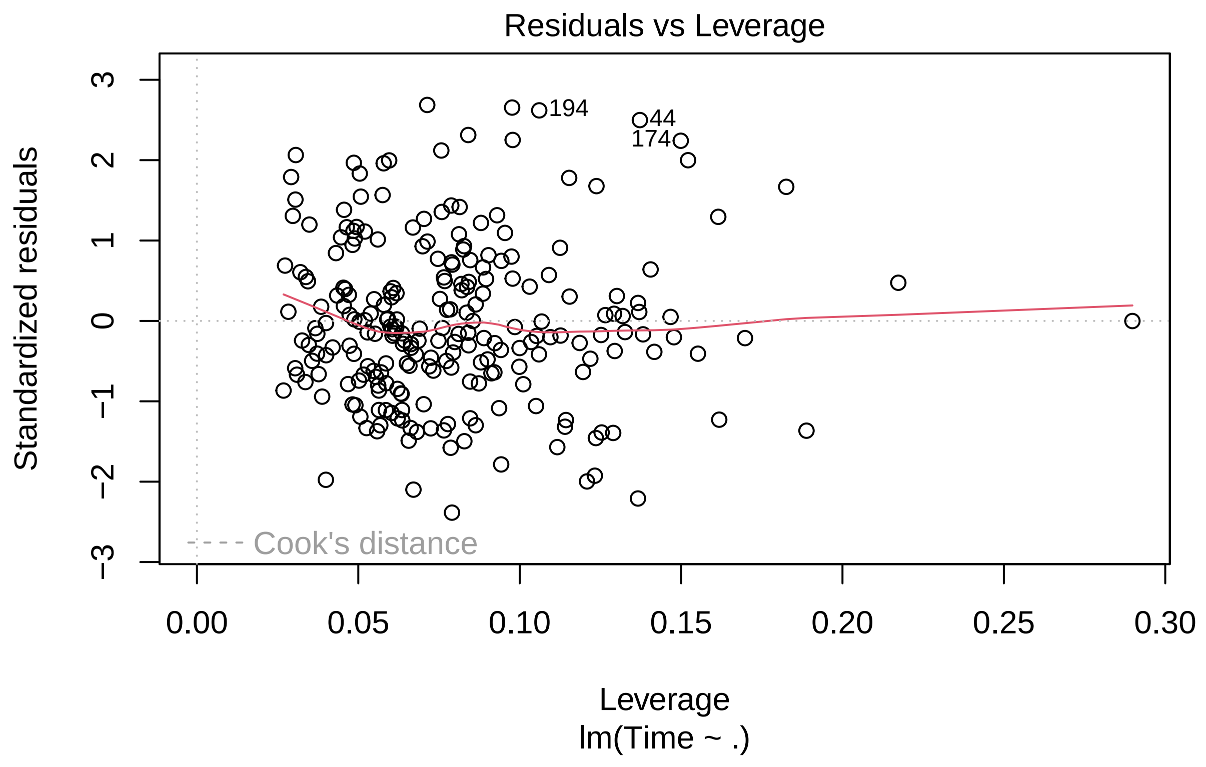
Carry out some residual diagnostics for the model fitted at the end of Task 3, and display the results graphically. Attempt a Box-Cox transformation, and refit the model using the suggested transformation. Repeat the residual diagnostics, and also consider the value of R2 of the transformed model. Give your judgement on whether you would prefer the original or the transformed model

```r
plot(fit)
```

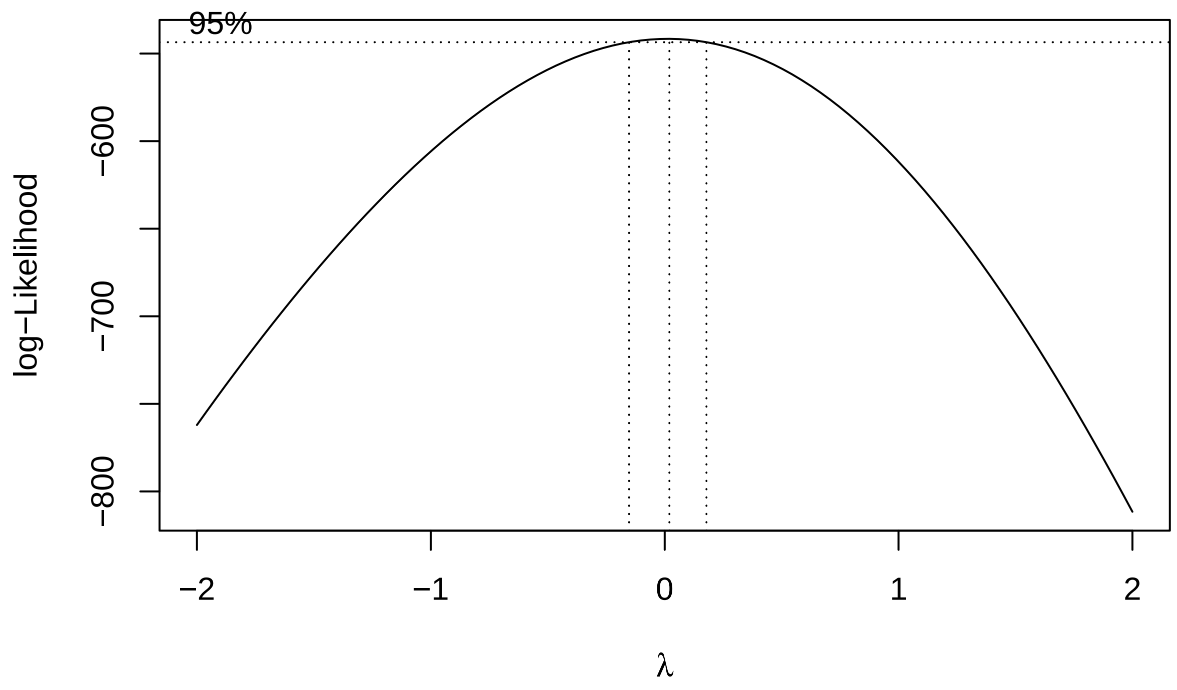

Residuals vs Fitted
Residuals
Fitted values
lm(Time ~ .)
94
194
117
10
5
0
−5
−5
0
5
10
15


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(Time ~ .)
194
117
94
−3
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
−5
0
5
10
15
Fitted values
lm(Time ~ .)
94
194
117



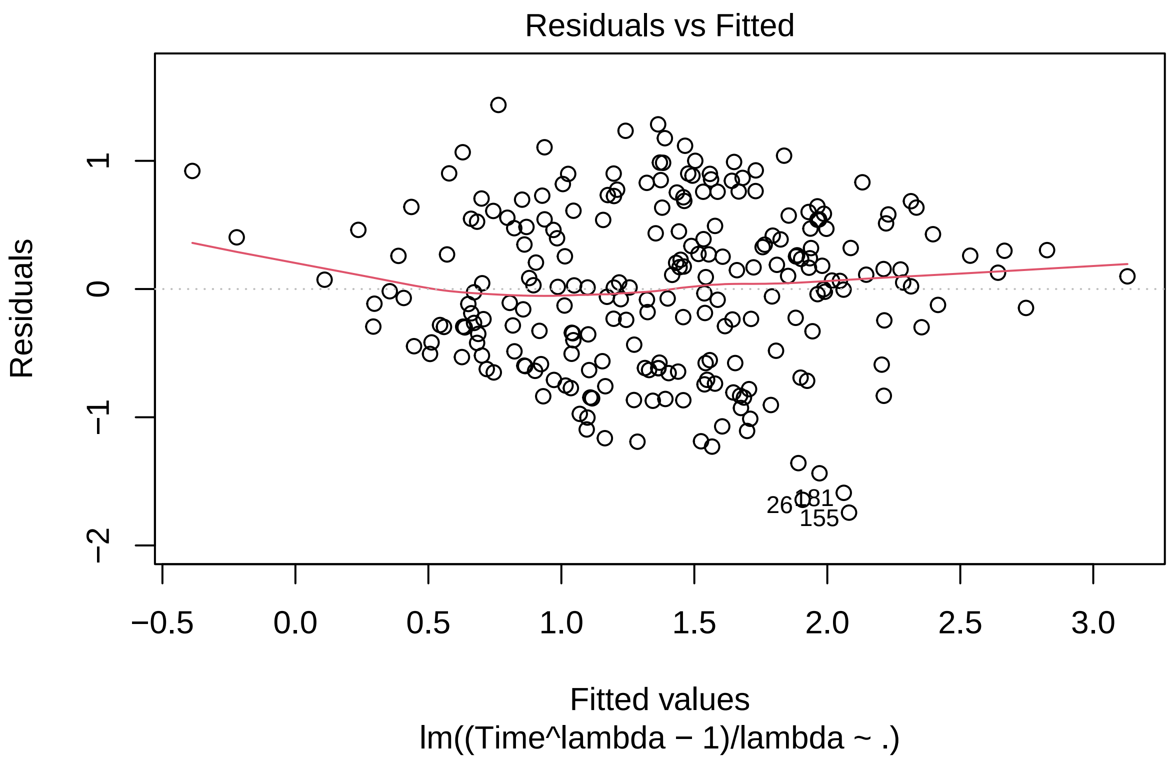
```r
library(MASS)
boxcox_result <- boxcox(fit)
```

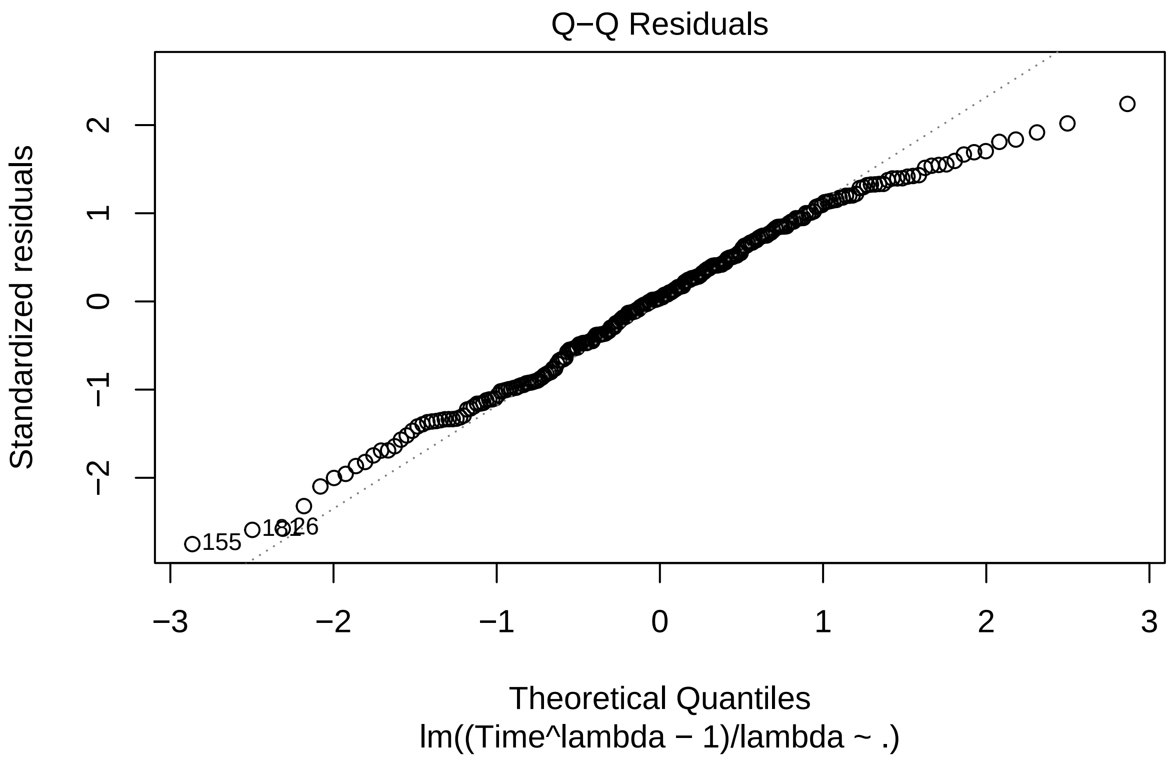


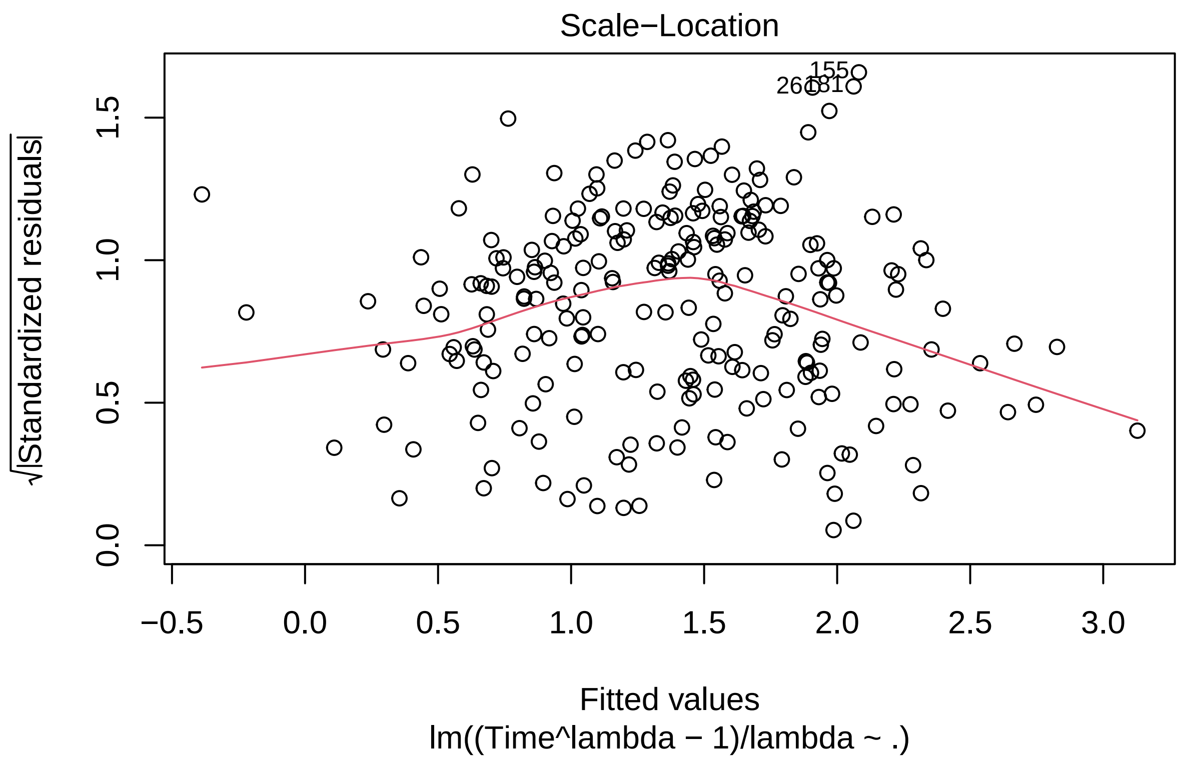
```
# Choose lambda with the maximum log-likelihood
lambda <- boxcox_result$x[which.max(boxcox_result$y)]
cat(lambda,"\n")
```

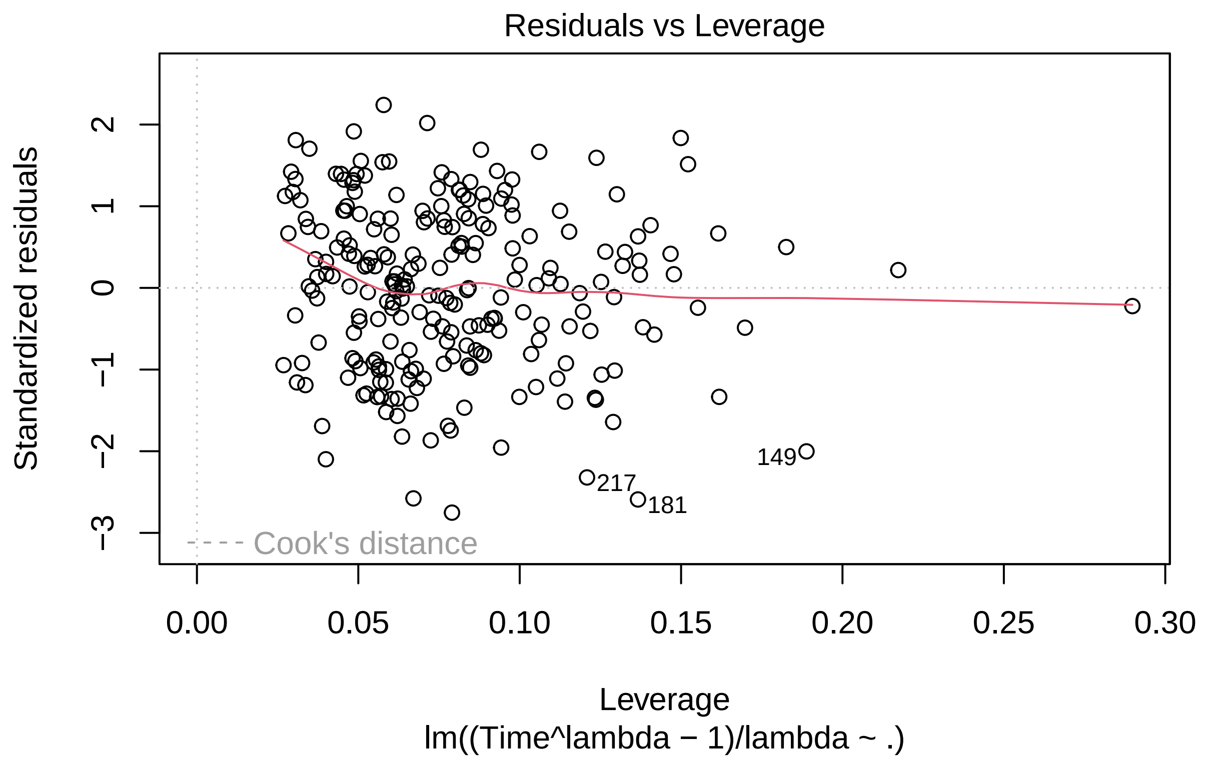
```
## 0.02020202
```

```
# Refit the model using the Box-Cox transformation
lfit <- lm((Time^lambda - 1) / lambda ~., data = training_data)

# Perform residual diagnostics for the transformed model

#par(mfrow=c(1,2), cex=0.6)
#plot(lfit$residuals)
#plot(lfit$fitted, lfit$residuals)

plot(lfit)
```


Residuals vs Fitted
Residuals
1
0
−1
−2
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
261
181
155
Fitted values
lm((Time^lambda − 1)/lambda ~ .)


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm((Time^lambda − 1)/lambda ~ .)
155
181
26
−3
−2
−1
0
1
2
3


Scale−Location
155
261
181
√|Standardized residuals|
0.0
0.5
1.0
1.5
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Fitted values
lm((Time^lambda − 1)/lambda ~ .)



```r
# Evaluate R^2 values for both models
fit_r_squared <- summary(fit)$r.squared
lfit_r_squared <- summary(lfit)$r.squared

cbind(fit_r_squared ,lfit_r_squared)
```

```
##      fit_r_squared lfit_r_squared
## [1,]     0.4611514       0.445667
```

```r
# Provide judgment
if (lfit_r_squared > fit_r_squared) {
  cat("The transformed model with Box-Cox transformation is preferred.")
} else {
  cat("The original model is preferred.")
}
```

```
## The original model is preferred.
```

answer The original model shows a trumpet shape in the diagnostic plot, which indicates a badly fitted model. Given the residual in the transformed model has shown less patterns as opposed to the original model, I would prefer transformed model even though R^2 has slightly been compromised, from 0.461 to 0.445.

Task 5: Nonparametric smoothing (15 marks)

In this task we are interested in modelling Time through a single gene, through a nonparametric, univariate, regression model. Firstly, based on previous analysis, choose a gene which you deem suitable for this task. Provide a scatterplot of the Time (vertical) versus the expression values of that gene (horizontal). Identify a nonparametric smoother of your choice to carry out this task. Based on visual inspection, or trial and error, determine a smoothing parameter which appears suitable, and add the resulting fitted curve to the scatterplot

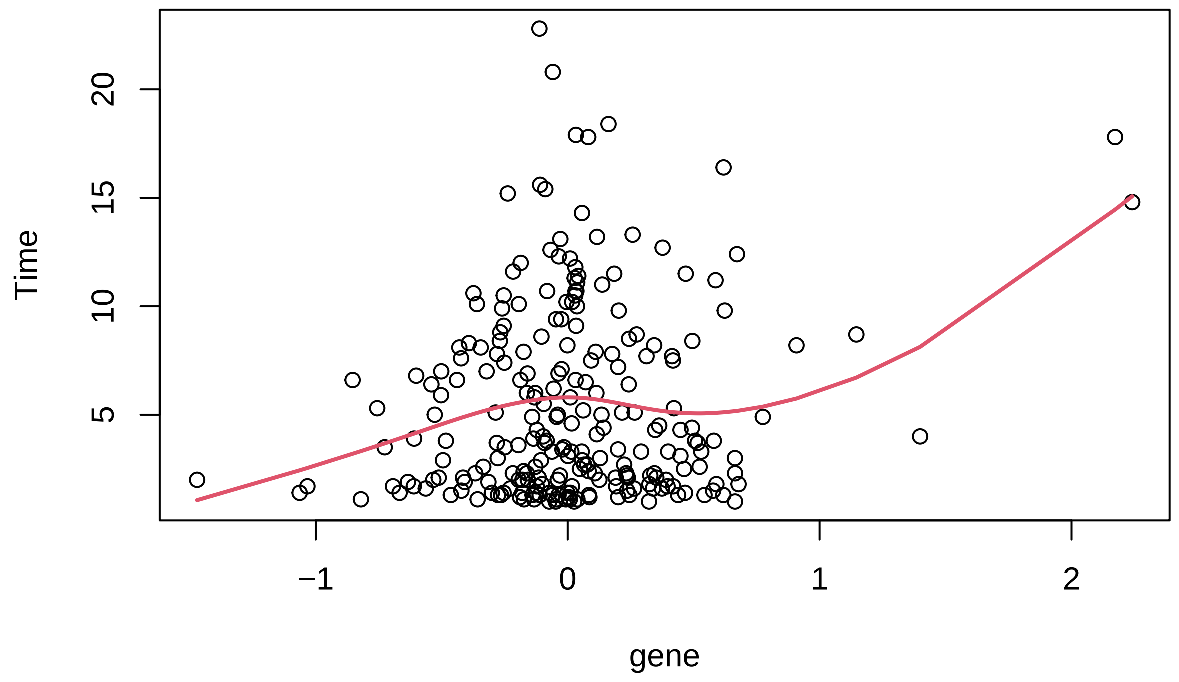
```
colnames(training_data)
```

```
##  [1] "G1020" "G1021" "G1456" "G1825" "G3395" "G3807" "G4131" "G4248" "G4762"
## [10] "G5352" "G5608" "G5621" "G5950" "G6014" "G6134" "G6321" "G6757" "G7070"
```

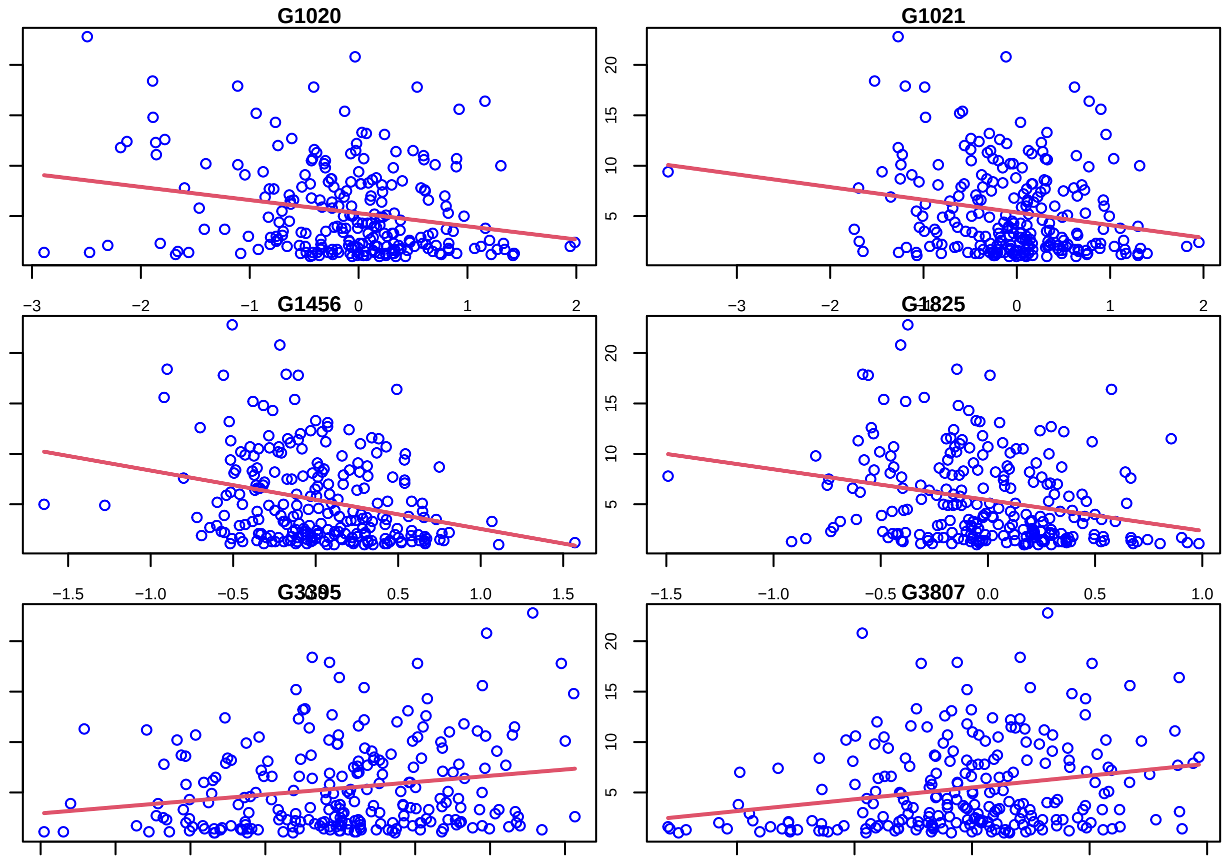
```
#6014, 6757, 5950
require(pspline)
```

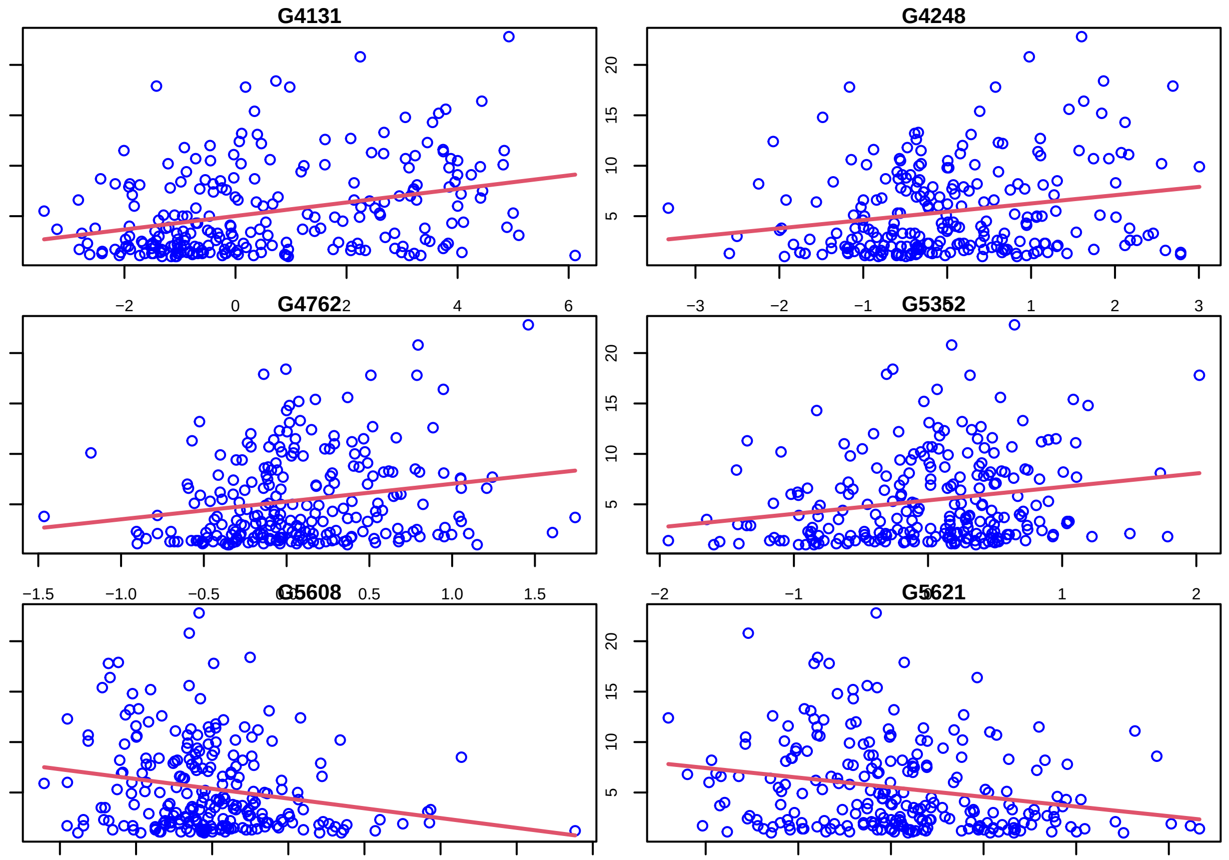
```
## Loading required package: pspline
```

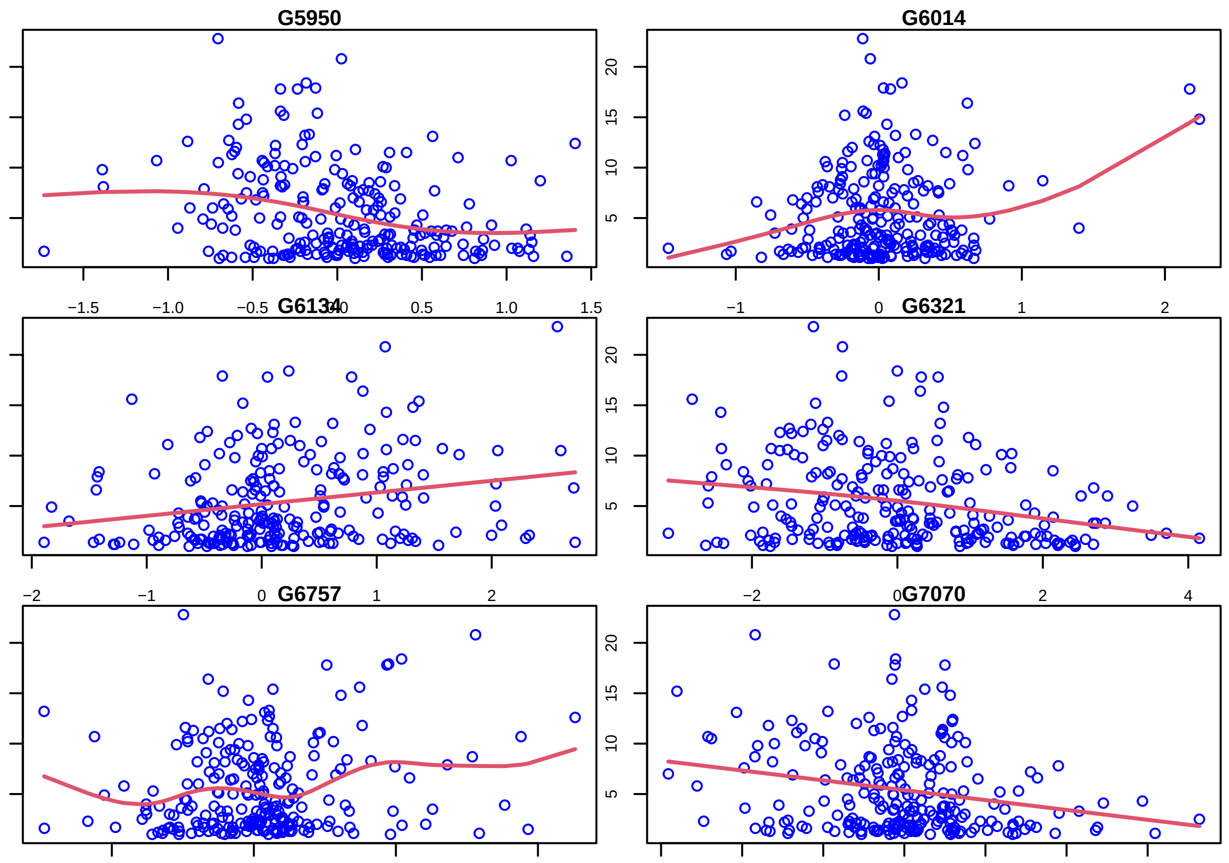
```
gene=training_data$G6014



gene.spline<-sm.spline(gene,Time)

plot(gene, Time)
lines(gene.spline,col=2, lwd=2)
```



```r
# Get column names of the training_data
gene_columns <- colnames(training_data)

# Set up the layout for plots
par(mfrow = c(3, 2), mar = c(1, 1, 1, 1), cex.main = 1, cex.lab = 0.8, cex.axis = 0.8)
?par
# Loop through each gene column
for (j in 1:18) {  # Change the limit based on the actual number of columns
  # Extract gene expression data
  gene <- training_data[[gene_columns[j]]]

  # Plot gene expression against Time
  plot(gene, Time, xlab = "Gene Expression", ylab = "Time",
       main = gene_columns[j], col = "blue")

  gene.spline<-sm.spline(gene,Time)


  lines(gene.spline,col=2, lwd=2)


}
```


G1020
G1021
G1456
G1825
G3395
G3807
5
10
15
20
−3
−2
−1
0
1
2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


G4131
G4248
G4762
G5352
G5608
G5621
5
10
15
20
−2
0
2
4
6
−3
−2
−1
0
1
2
3
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
−2
−1
0
1
2

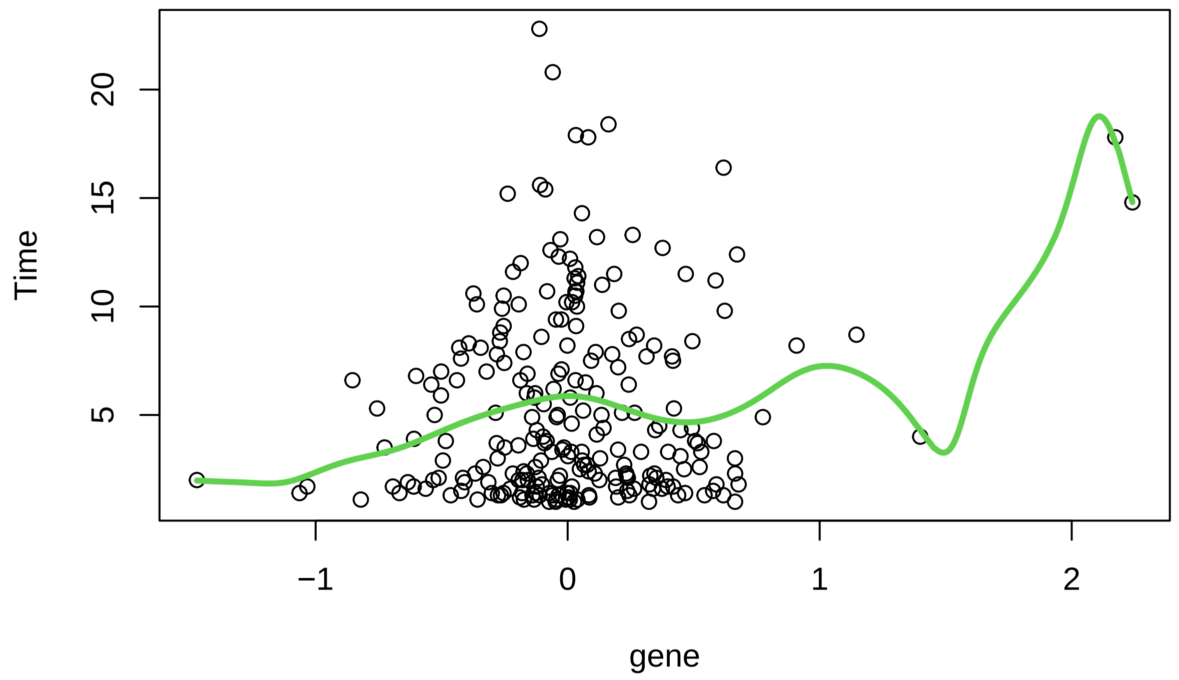


```
set.seed(124)
require(KernSmooth)
```

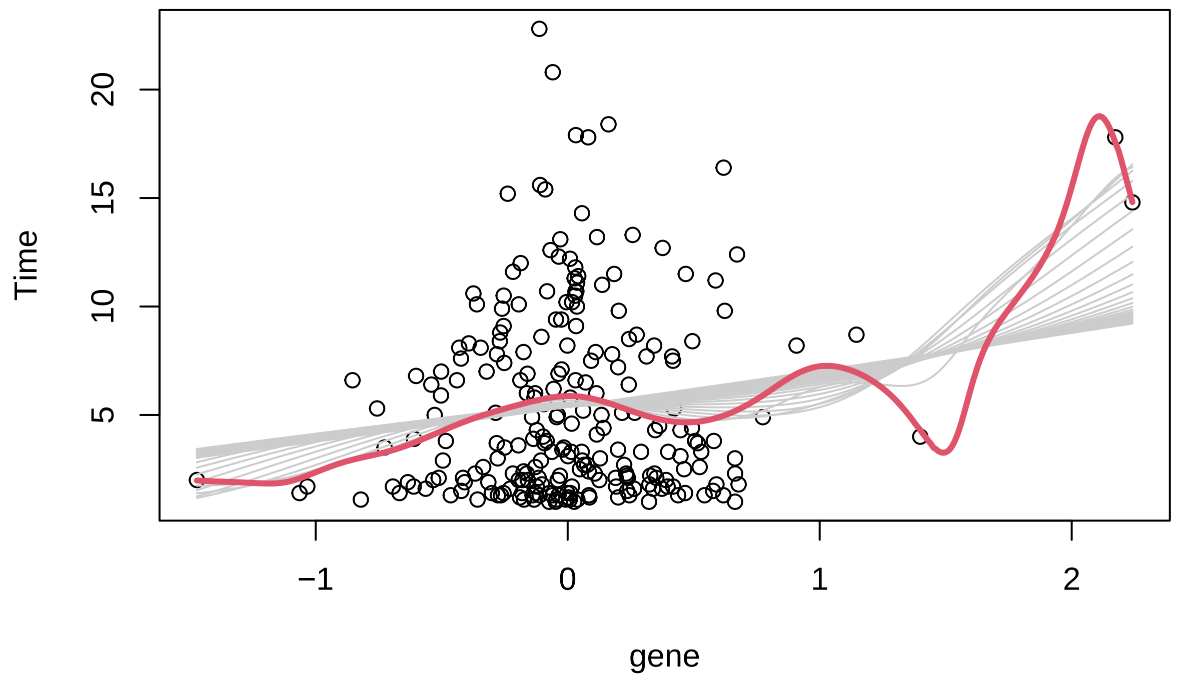
```
## Loading required package: KernSmooth
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
gene=training_data$G6014

h <- dpill(gene, Time,  truncate=FALSE)
gene.loc <- locpoly(gene, Time, bandwidth=h)
plot(gene, Time)
lines(gene.loc$x, gene.loc$y,  col=3, lwd=3)
```



```
fam.fit <- matrix(0,50,401)
h.grid <- seq(0.1,5, by=0.1)
for (j in 1:50){
  fam.fit[j,]<-locpoly(gene, Time,
     bandwidth=h.grid[j])$y
}
plot(gene, Time)
for ( j in 3:50){
  lines(gene.loc$x, fam.fit[j,], col="grey80")
}
lines(gene.loc$x, gene.loc$y, col=2, lwd=3)
legend(100, 0.70725, legend="dpill solution",
lwd=3, col=2)
```



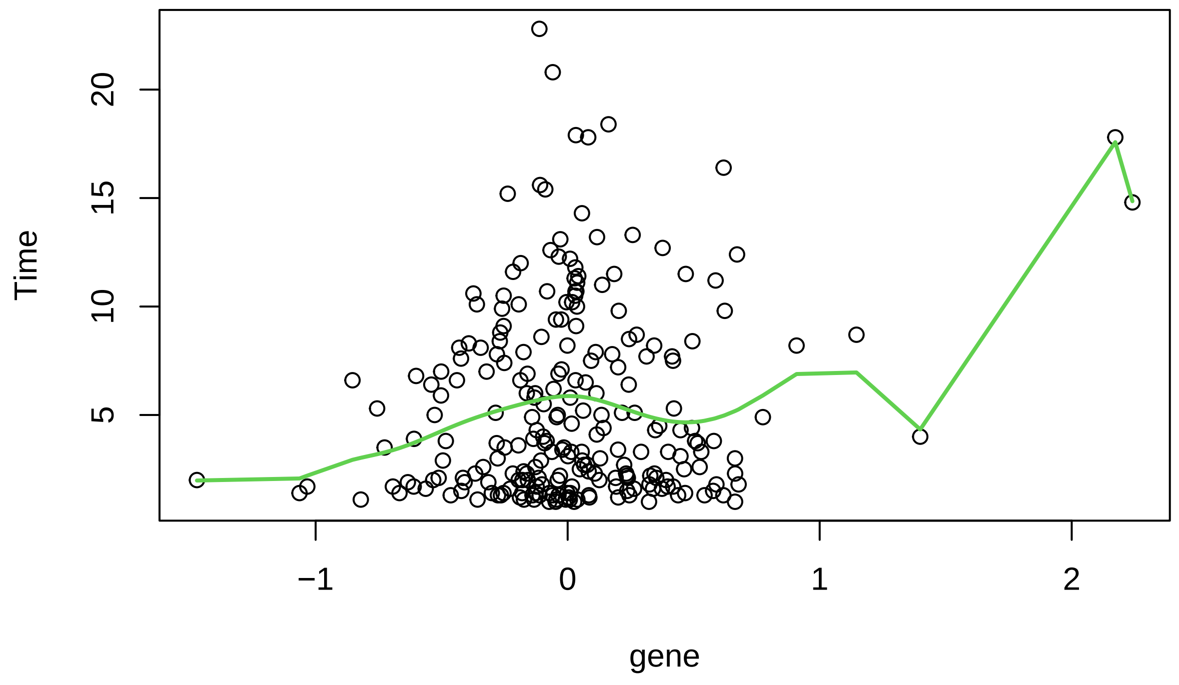
Task 6: Bootstrap confidence intervals (20 marks)

Continuing from Task 5 (with the same, single, predictor variable, and the same response Time), proceed with a more systematic analysis. Specifically, produce a nonparametric smoother featuring

- a principled way to select the smoothing parameter;
- bootstrapped confidence bands.

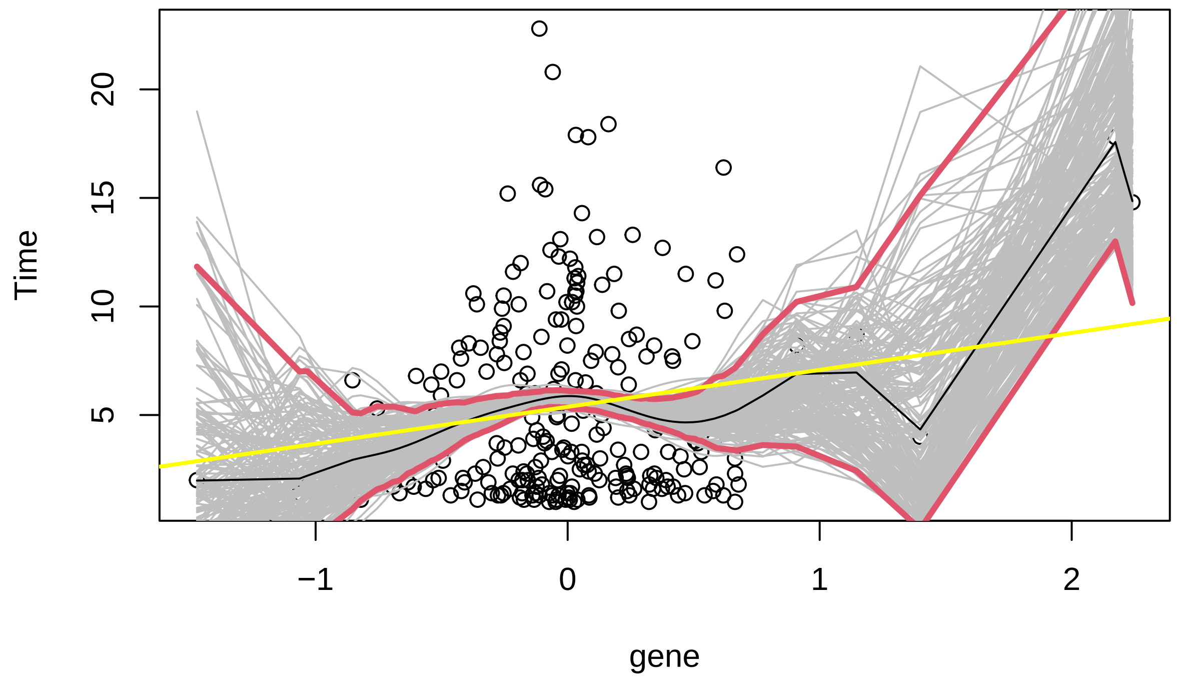
The smoothing method that you use in this Task may be the same or a different one as used in Task 5, but you are not allowed to make use of R function gam. If you use any built-in R functions to select the smoothing parameter or carry out the bootstrap, explain briefly what they do. Produce a plot which displays the fitted smoother with the bootstrapped confidence bands. Add to this plot the regression line of a simple linear model with the only predictor variable being the chosen gene (beside the intercept). Finally, report the values of R2 of both the nonparametric and the parametric model. Conclude with a statement on the usefulness of the nonparametric model.

```r
set.seed(124)
require(KernSmooth)
gene=training_data$G6014
h <- dpill(gene, Time,  truncate=FALSE)

smooth.loc<- function(x,y,xgrid=x, h){
  N<-length(xgrid)
  smooth.est<-rep(0,N)
  for (j in 1:N){
    smooth.est[j]<- lm(y~I(x-xgrid[j]),
    weights=dnorm(x,xgrid[j],h) )$coef[1]
  }
list(x= xgrid, fit=smooth.est)
}

gene.loc <- smooth.loc(gene, Time, h=h)


plot(gene, Time)
lines(gene[order(gene)], gene.loc$fit[order(gene)],
    col=3, lwd=2)
```



```
###bootstrap
Time.res <- Time- gene.loc$fit
boot.fit <- matrix(0,200,length(Time))
for (j in 1:200){
  boot.res <- sample(Time.res, size=length(Time))
  new.y    <-  gene.loc$fit +boot.res
  # h <- dpill(gene, new.y,  truncate=FALSE)
     # the line above would be used to account for uncertainty in selection of h
     # but numerically unstable
  boot.fit[j,] <-  smooth.loc(gene, new.y, h=h)$fit
```

```r
}

lower<-upper<- rep(0,length(Time))
for (i in 1:length(Time)){
  lower[i]<-quantile(boot.fit[,i],0.025)
  upper[i]<-quantile(boot.fit[,i],0.975)
}

plot(gene, Time)
for (j in 1:200){ lines(gene[order(gene)], boot.fit[j,][order(gene)], col="grey")}
lines(gene[order(gene)], gene.loc$fit[order(gene)])
lines(gene[order(gene)], upper[order(gene)], col=2, lwd=3)
lines(gene[order(gene)], lower[order(gene)], col=2, lwd=3)

parametric_model <- lm(Time ~ gene)

# Plot the regression line
abline(parametric_model, col = "yellow", lwd = 2)
```



```r
# Calculate R-squared for both models
R_squared_smooth <- cor(gene.loc$fit, Time)^2
R_squared_linear <- summary(parametric_model)$r.squared

cat("h:",h, "\n")
```

```
## h: 0.19631
```

```r
# Output R-squared values
cat("R-squared for smooth model:", R_squared_smooth, "\n")
```

```
## R-squared for smooth model: 0.09522959
```

```r
cat("R-squared for linear regression model:", R_squared_linear, "\n")
```

```
## R-squared for linear regression model: 0.02666934
```

```r
# Conclusion on the usefulness of the nonparametric model
cat("The nonparametric Smooth model captures the nonlinear relationship between the gene expression and
```

```
## The nonparametric Smooth model captures the nonlinear relationship between the gene expression and ti
```

answer

H=0.19631

R-squared for smooth model: 0.09522959

R-squared for linear regression model: 0.02666934

R-squared for linear regression model: 0.02666934

The nonparametric Smooth model captures the nonlinear relationship between the gene expression and time without assuming a specific functional form. This flexibility can be particularly useful when the relationship is complex and nonlinear, as it provides a more accurate representation compared to a simple linear regression model.