Linear regression and LASSO and Ridge

In [43]:
```
par(family  = "Arial")
#install.packages("showtext")
library(showtext)
showtext_auto()
options(repr.plot.width=8, repr.plot.height=8)
```

In [23]:
```
attach(faithful)
```

In [24]:
```
 eruption.lm = lm(eruptions ~ waiting)
```

In [27]:
```
summary(eruption.lm)
```

```
Call:
lm(formula = eruptions ~ waiting)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.29917 -0.37689  0.03508  0.34909  1.19329 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.874016   0.160143  -11.70   <2e-16 ***
waiting      0.075628   0.002219   34.09   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4965 on 270 degrees of freedom
Multiple R-squared:  0.8115,	Adjusted R-squared:  0.8108 
F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16
```

In [28]:
```
newdata = data.frame(waiting=80)
```

In [29]:
```
newdata
```

waiting
80

In [30]:
```
fitted = predict(eruption.lm, newdata, interval="predict")
fitted
```

fit	lwr	upr
4.17622	3.196089	5.156351

In [31]:
```
fitted[,'fit']
```

4.17621984973828

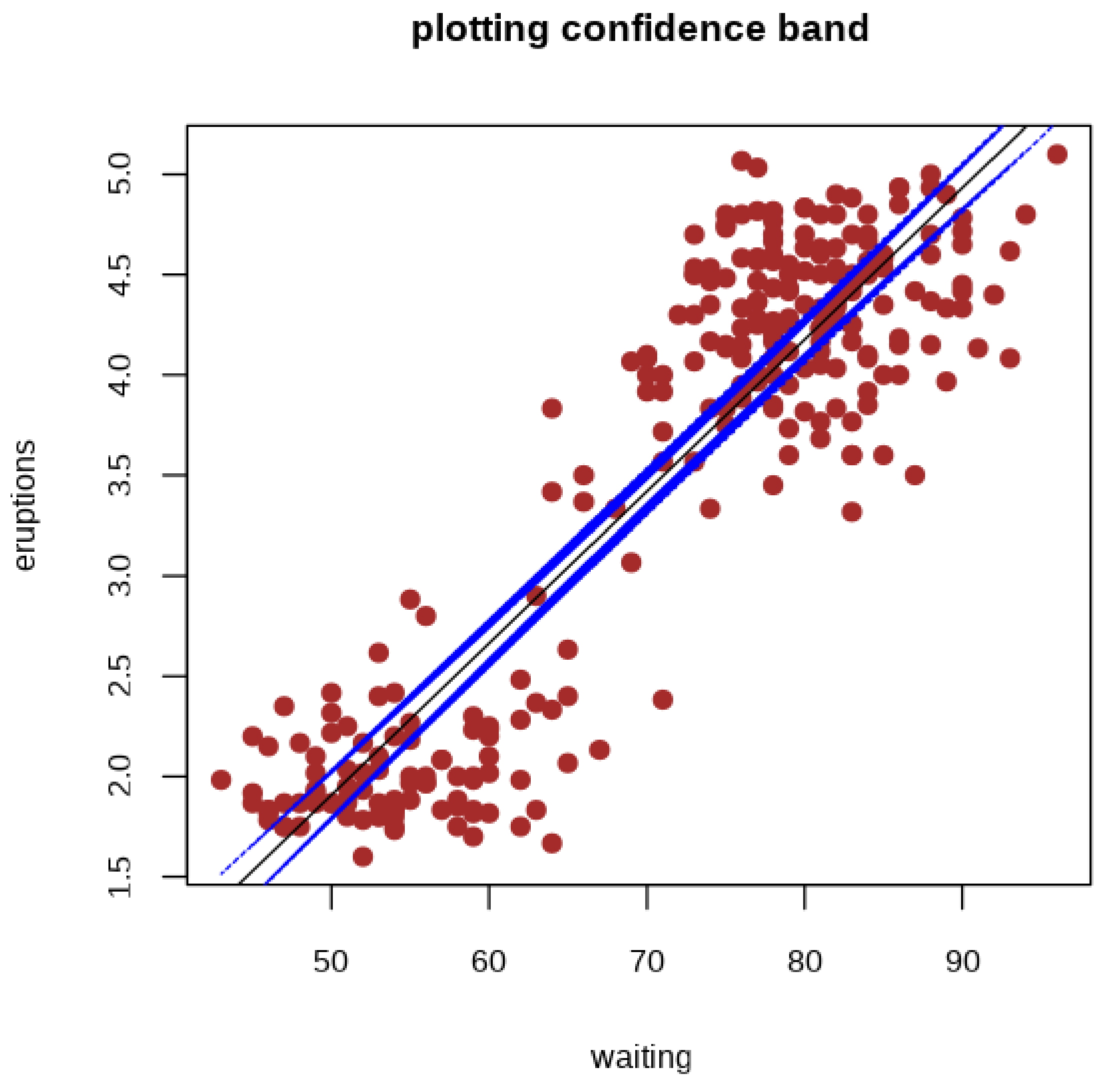
In [32]:
```
fitted[ ,'lwr']
```

3.19608856098821

plot the data and the fitted line

In [33]:

```
fitted <- predict(eruption.lm, interval = "confidence")
plot(y=eruptions,x=waiting,col='brown',pch=19,main = "plotting confidence band")
#plot confidence band
# plot the confidence bands
lines(waiting, eruption.lm$fitted.values)

lines(waiting, fitted[, "upr"], lty = "dotted",
col="blue")
lines(waiting, fitted[, "lwr"], lty = "dotted",
col="blue")
```



degree of freedom

Definition:

$$df(\hat{y}) = \frac{1}{\sigma^2}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i)$$

- Larger covariance → stronger connection → model can approximate data better→ model more flexible (complex)
- For linear smoothers $\hat{Y} = S(X)Y$

$$df = trace(S)$$

- For linear regression, degrees of freedom is

$$df = trace(P) = p$$

Ridge Regression

- **Idea**: Keep all predictors but shrink coefficients to make model less complex

minimize $-loglikelihood + \lambda_0 ||w||_2^2$

→ **l_2 regularization**

- Given that model is Gaussian, we get **Ridge regression:**

$$\hat{w}^{ridge} = \operatorname{argmin}\left\{\sum_{i=1}^{N}(y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2 + \lambda \sum_{j=1}^{p} w_j^2\right\}$$

- $\lambda > 0$ is **penalty factor**

Equivalent form

$$\hat{w}^{ridge} = \operatorname{argmin} \sum_{i=1}^{N} (y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2$$

$$\textbf{subject to } \sum_{j=1}^{p} w_j^2 \le s$$

Solution

$$\boxed{\hat{\boldsymbol{w}}^{ridge} = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{w} = \boldsymbol{X}(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

Hat matrix

How do we compute degrees of freedom here?

Properties

- Extreme cases:
  - $\lambda = 0$ usual linear regression (no shrinkage)
  - $\lambda = +\infty$ fitting a constant ($w = 0$ except of w_0)
- When input variables are ortogonal (not realistic), $X^T X = I \rightarrow$

  $\hat{\boldsymbol{w}}^{\mathrm{ridge}} = \frac{1}{1+\lambda} \mathbf{w}^{\mathbf{linreg}}$ → coefficients are equally shrunk
- **Ridge regression is particularly useful if the explanatory variables are strongly correlated to each other.**
  - Correlated variables often correspond large w → shrunk
- Degrees of freedom decrease when λ increases
  - $\lambda = 0 \rightarrow d.f. = p$

Ridge Regression works better than linear regression even if (P (# featues) > n (# datapoints)). becaues :

$$(X^T X + \lambda I \text{ is always nonsingular})$$

R code: use package glmnet with alpha=0 (Ridge regression)

Seeing how Ridge converges

In [ ]:
```
data=read.csv("machine.csv",covariates=scale(data[,3:8])
response=scale(data[, 9])
header=F)
model0=glmnet(as.matrix(covariates),
response, alpha=0,family="gaussian")
plot(model0, xvar="lambda", label=TRUE)
```

Choosing the best model by cross-validation

In [ ]:
```
model=cv.glmnet(as.matrix(covariates),
response, alpha=0,family="gaussian")
model$lambda.min
plot(model)
coef(model, s="lambda.min")
```

How good is this model in prediction?

In [ ]:
```
ind=sample(209, floor(209*0.5))
data1=scale(data[,3:9])
train=data1[ind,]
test=data1[-ind,]
covariates=train[,1:6]
response=train[, 7]
model=cv.glmnet(as.matrix(covariates), response, alpha=1,family="gaussian",
lambda=seq(0,1,0.001))
y=test[,7]
ynew=predict(model, newx=as.matrix(test[, 1:6]), type="response")
#Coefficient of determination
sum((ynew-mean(y))^2)/sum((y-mean(y))^2) #if 0 underfit, if 1 overfit
sum((ynew-y)^2)
```

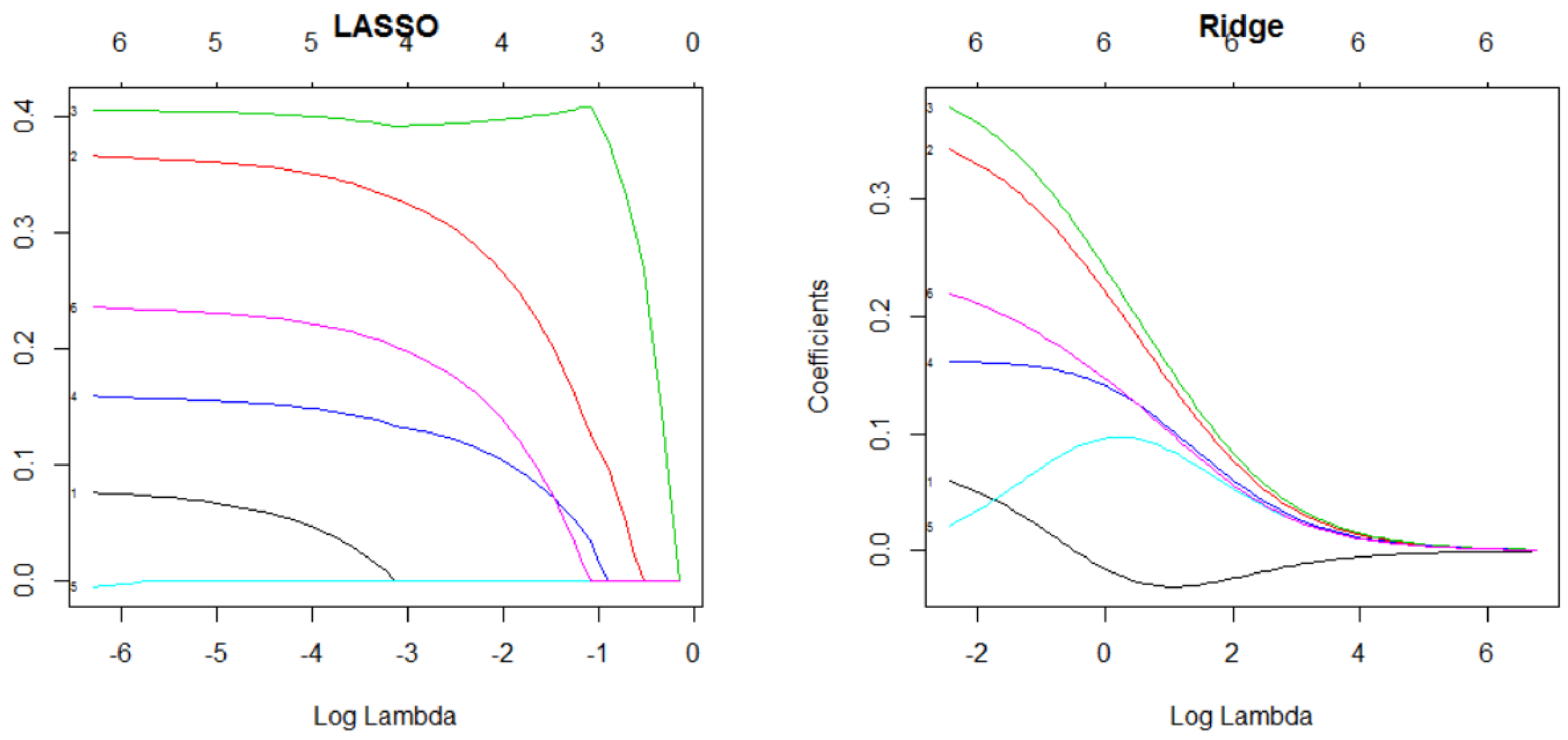
α=1 is lasso regression (default) and α=0 is ridge regression.

LASSO

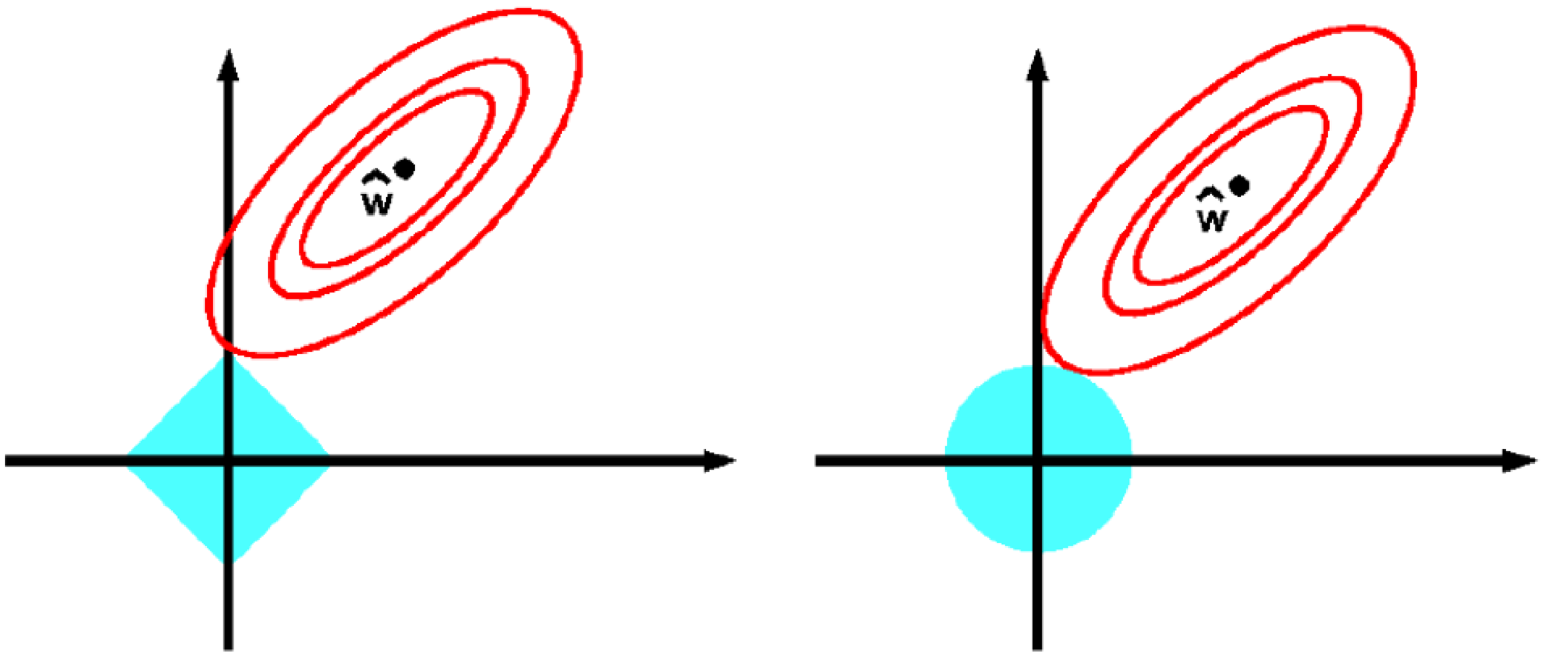
- Minimize minus loglikelihood plus **linear** penalty factor→ **l_1 regularization**
  - Given that model is Gaussian, we get **LASSO** (least absolute shrinkage and selection operator):

$$\hat{w}^{lasso} = \operatorname{argmin}\left\{\sum_{i=1}^{N}(y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2 + \lambda\sum_{j=1}^{p}|w_i|\right\}$$

- **LASSO yields sparse solutions!**



- Why Lasso leads to sparse solutions?
  - Feasible area for Ridge is a circle (2D)
  - Feasible area for LASSO is a polygon (2D)



In Lasso there is a higher chance that the polygen meet the ellipsoid at one of the corners and as soon as they met at the corners then then one of the coef(w1 in the pic) will be 0.

Lasso works when (P (# featues) > > n (# datapoints)). :e.g Genome data

- **Lasso is widely used when $p \gg n$**
  - Linear regression breaks down when $p > n$
  - Application: DNA sequence analysis, Text Prediction

- When inputs are orthonormal,

$$\hat{w}_i^{\text{lasso}} = sign(w_i^{linreg})\left(|w_i^{linreg}| - \frac{\lambda}{2}\right)_+$$

- No explicit formula for $\hat{w}^{lasso}$
  - Optimization algorithms used

Coding in R: use glmnet() with alpha=1

notes on variable selection

Often, we do not need all features available in the data to be in the model.

Alternative 1: Variable subset selection

• Best subset selection:

– Consider different subsets of the full set of features, fit models and evaluate their quality

• Problem: computationally difficult for p around 30 or more

• How to choose the best model size? Some measure of predictive performance normally used (ex. AIC).

• Forward and Backward stepwise selection

– Starts with 0 features (or full set) and then adds a feature (removes feature) that most improves the measure selected.

• Can handle large p quickly • Does not examine all possible subsets (not the “best”)

Variable selection in R

Use stepAIC() in MASS

In [ ]:
```
library(MASS)
fit <- lm(V9~.,data=data.frame(data1))
step <- stepAIC(fit, direction="both")
step$anova
summary(step)
model.matrix(step) #list of variable
```

In [5]:
```
library(MASS)
data(mtcars)
anova(lm(mpg~disp+wt+am,mtcars),lm(mpg~disp+wt,mtcars))
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
28	246.5563	NA	NA	NA	NA
29	246.6825	-1	-0.1262404	0.0143364	0.9055483

In [9]:
```
library(MASS)
res.lm <- lm(Fertility ~., data = swiss)
step <- stepAIC(res.lm, direction = "both", trace = FALSE)
```

In [15]:
```
head(swiss,1)
```

	Fertility	Agriculture	Examination	Education	Catholic	Infant.Mortality
Courtelary	80.2	17	15	12	9.96	22.2

In [18]:
```
model.matrix(step)[1:10,]  #the chosen model
```

	(Intercept)	Agriculture	Education	Catholic	Infant.Mortality
Courtelary	1	17.0	12	9.96	22.2
Delemont	1	45.1	9	84.84	22.2
Franches-Mnt	1	39.7	5	93.40	20.2
Moutier	1	36.5	7	33.77	20.3
Neuveville	1	43.5	15	5.16	20.6
Porrentruy	1	35.3	7	90.57	26.6
Broye	1	70.2	7	92.85	23.6
Glane	1	67.8	8	97.16	24.9
Gruyere	1	53.3	7	97.67	21.0
Sarine	1	45.2	13	91.38	24.4

AIC is a single number score that can be used to determine which of multiple models is most likely to be the best model for a given dataset.

Assignment 3. Linear regression and LASSO

The **tecator.csv** contains the results of study aimed to investigate whether a near infrared absorbance spectrum can be used to predict the fat content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry.

Divide data randomly into train and test (50/50) by using the codes from the lectures.

1. Assume that Fat can be modeled as a linear regression in which absorbance characteristics (Channels) are used as features. Report the underlying probabilistic model, fit the linear regression to the training data and estimate the training and test errors. Comment on the quality of fit and prediction and therefore on the quality of model.

3.1

The underlying problistic model is a linear regression model with the distribution

$$y \sim N(w^T \mathcal{X}, \sigma^2)$$

- y is a vector of the response variable "Fat"
- w^T is a vector of the regression parameters
- $\mathcal{X}$ is a matrix of the observed values for the independent variables used in the model which are the fat channels
- σ^2 is a scalar that denotes the variance of the features.

In [1]:
```
tecator = read.csv('tecator.csv')
n = dim(tecator)[1]
```

In [2]:
```
head(tecator)
```

Sample	Channel1	Channel2	Channel3	Channel4	Channel5	Channel6	Channel7	Channel8	Channel9	...	Channel94	Channel95	Channel96	Channel97	Cl
1	2.61776	2.61814	2.61859	2.61912	2.61981	2.62071	2.62186	2.62334	2.62511	...	2.94013	2.91978	2.89966	2.87964	
2	2.83454	2.83871	2.84283	2.84705	2.85138	2.85587	2.86060	2.86566	2.87093	...	3.26655	3.25369	3.24045	3.22659	
3	2.58284	2.58458	2.58629	2.58808	2.58996	2.59192	2.59401	2.59627	2.59873	...	2.65112	2.63262	2.61461	2.59718	
4	2.82286	2.82460	2.82630	2.82814	2.83001	2.83192	2.83392	2.83606	2.83842	...	2.92576	2.90251	2.87988	2.85794	
5	2.78813	2.78989	2.79167	2.79350	2.79538	2.79746	2.79984	2.80254	2.80553	...	3.25831	3.23784	3.21765	3.19766	
6	3.00993	3.01540	3.02086	3.02634	3.03190	3.03756	3.04341	3.04955	3.05599	...	3.53442	3.52221	3.50972	3.49682	

In [3]:
```
ind = sample(n,floor(n*0.5),replace = FALSE)
```

In [61]:
```
train_data =  tecator[ind,]
test_data = tecator[-ind,]
```

In [62]:
```
train = train_data[,-c(1,103,104)]
test = test_data[,-c(1,103,104)]
ytrain = train$Fat
ytest = test$Fat
```

In [63]:
```
X = train_data[,-c(1,102,103,104)]
head(X,1)
```

	Channel1	Channel2	Channel3	Channel4	Channel5	Channel6	Channel7	Channel8	Channel9	Channel10	...	Channel91	Channel92	Channel93	Chan
166	3.28923	3.29435	3.29952	3.30482	3.31024	3.31584	3.32175	3.32799	3.33462	3.34161	...	3.82507	3.80137	3.77789	3.7

In [15]:
```
fit_lm = lm( Fat ~., data = train)
```

In [16]:
```
summary(fit_lm)
```

```
Channel87    4.127e+04  2.956e+03  13.960 8.42e-06 ***
Channel88   -5.906e+04  2.722e+03 -21.696 6.26e-07 ***
Channel89    1.071e+05  3.251e+03  32.933 5.22e-08 ***
Channel90   -1.042e+05  4.742e+03 -21.963 5.82e-07 ***
Channel91    7.379e+04  5.233e+03  14.100 7.94e-06 ***
Channel92    6.075e+03  4.902e+03   1.239 0.261543
Channel93   -4.843e+04  3.469e+03 -13.960 8.42e-06 ***
Channel94    2.240e+04  2.534e+03   8.841 0.000116 ***
Channel95   -5.914e+04  2.676e+03 -22.102 5.61e-07 ***
Channel96    8.869e+04  3.759e+03  23.596 3.80e-07 ***
Channel97   -3.065e+04  3.036e+03 -10.098 5.48e-05 ***
Channel98   -2.431e+04  1.231e+03 -19.752 1.09e-06 ***
Channel99    2.583e+04  1.717e+03  15.038 5.45e-06 ***
Channel100  -7.481e+03  8.641e+02  -8.658 0.000131 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04523 on 6 degrees of freedom
Multiple R-squared:      1,     Adjusted R-squared:      1
F-statistic: 7.925e+04 on 100 and 6 DF,  p-value: 9.59e-15
```

Estimating the error on trainig set:

In [17]:
```
pred_ytrain = predict(fit_lm)
Mse_train = mean((ytrain - pred_ytrain)**2)
Mse_train
```

0.000114689794806587

In [110]:
```
head(test,1)
```

Channel1	Channel2	Channel3	Channel4	Channel5	Channel6	Channel7	Channel8	Channel9	Channel10	...	Channel92	Channel93	Channel94	Channel95
2.61776	2.61814	2.61859	2.61912	2.61981	2.62071	2.62186	2.62334	2.62511	2.62722	...	2.98145	2.96072	2.94013	2.91978

Estimating the error on test set:

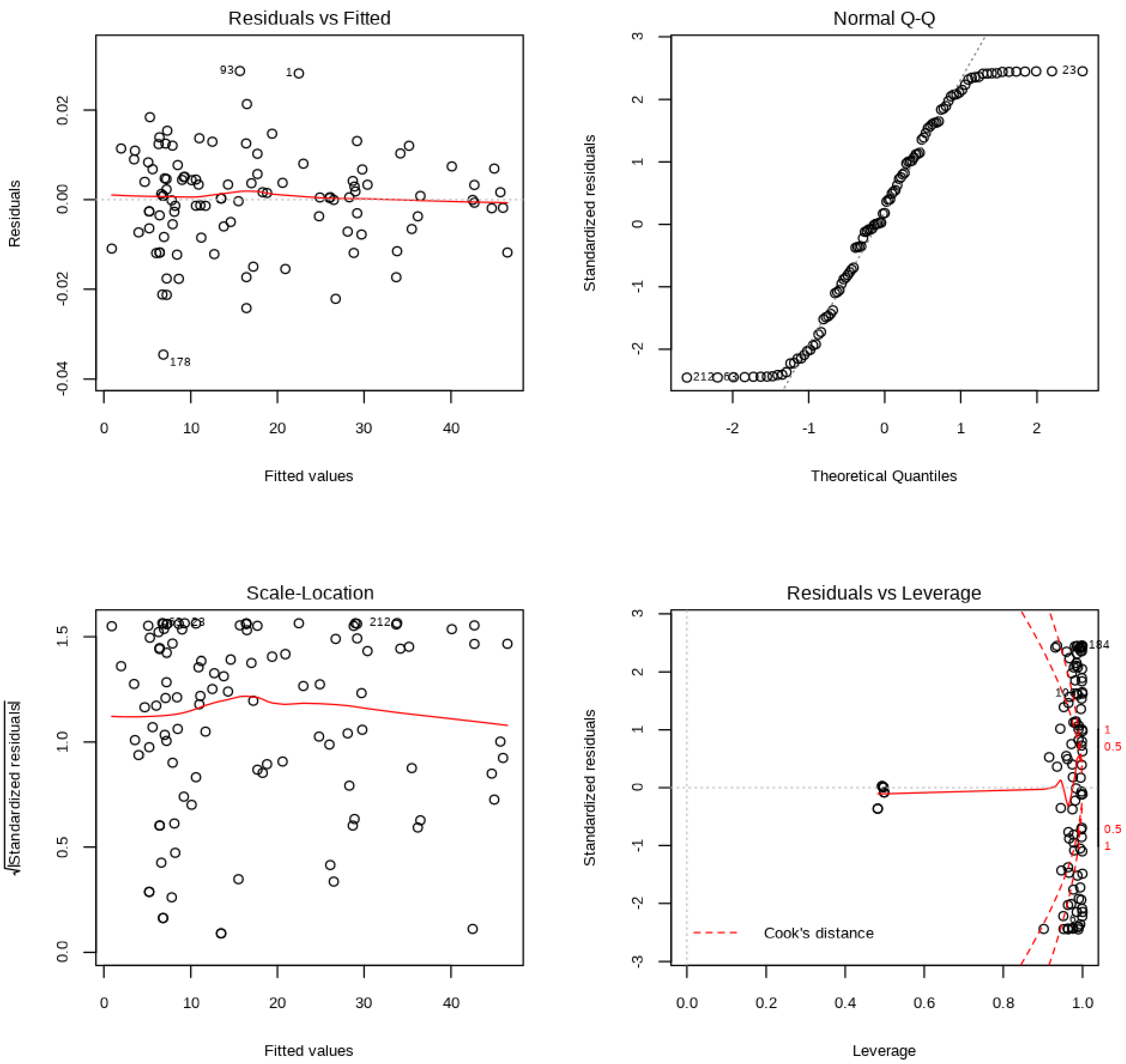
In [20]:
```
pred_ytest = predict(fit_lm,test)
Mse_test = mean((ytest - pred_ytest)**2)
Mse_test
```

647.12077301104

With the very low training MSE and comparably high test MSE we have reason to believe that there is overfitting occurring with the model. From viewing the summary for the training model it was found to give a adjusted R-square of 9.59e-15 which gives an clear indication that the model is overfit.

In [44]:
```
# Plotting the linear regression model

par(mfrow = c(2,2))

plot(fit_lm)
```

```
Warning message in sqrt(crit * p * (1 - hh)/hh):
“NaNs produced”Warning message in sqrt(crit * p * (1 - hh)/hh):
“NaNs produced”
```



Analyzing the plots we find that the assumption that the model residuals are i.i.d does seems to be fulfilled. From the residual vs fitted plot it can be observed that the assumption for constant variance is not fulfilled with a wider spread on the left hand side compared to the right hand side. It's clear that the residuals do not follow anything similar to a normal distribution with mean 0 and constant variance when observing the QQ-plot. Hence the assumptions for the model are not fulfilled so we should not draw any conclusions from the model.

2. Assume now that Fat can be modeled as a LASSO regression in which all Channels are used as features. Report the objective function that should be optimized in this scenario.

The function we are looking to optimize in this case is loss function of the LASSO model.

$$\sum_{i=1}^{N}(y_i - \mathbb{X}_i\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

Where N is the number of observations and $\mathbb{X}$ is the matrix of features and y is the dependent variable. β are the parameters that the shrinkage function will choose so that the loss function minimizes. With λ being the shrinkage parameter and P the number of features. Where the chosen λ will shrink the β parameters.

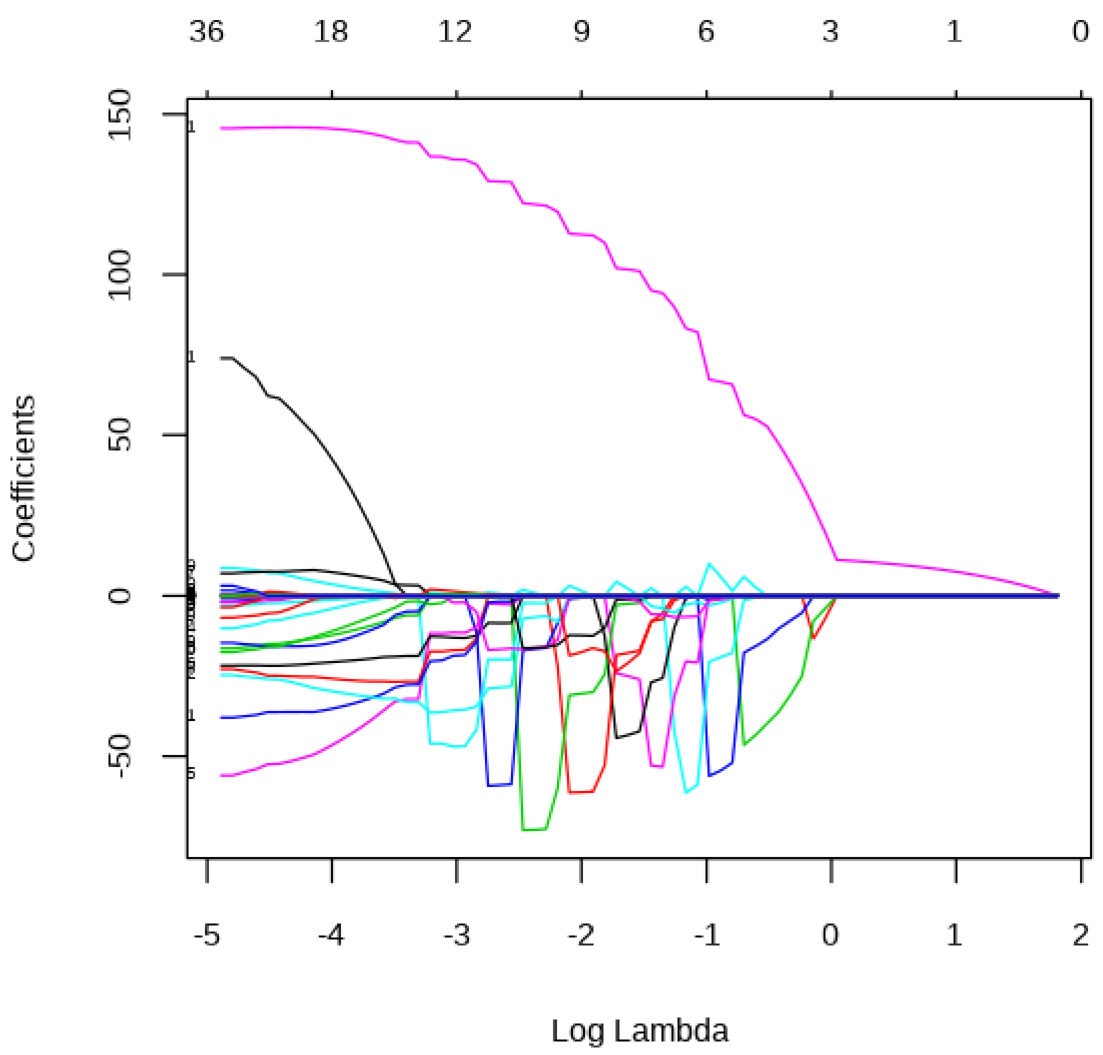
3. Fit the LASSO regression model to the training data. Present a plot illustrating how the regression coefficients depend on the log of penalty factor ($\log\lambda$) and interpret this plot. What value of the penalty factor can be chosen if we want to select a model with only three features?

In [153]:
```
library(glmnet)
```

In [154]:
```
set.seed(12345)
lasso_model = glmnet(as.matrix(X), ytrain, alpha=1,family="gaussian")
# TO  select the model with 3 feature the degree of freedom should be 3
lasso_model$lambda[which(lasso_model$df == 3)]
```

0.951175620716936 0.789682710834879 0.719529475323402 0.655608459898808

In [155]:
```
plot(lasso_model,xvar = "lambda",label = TRUE)
```



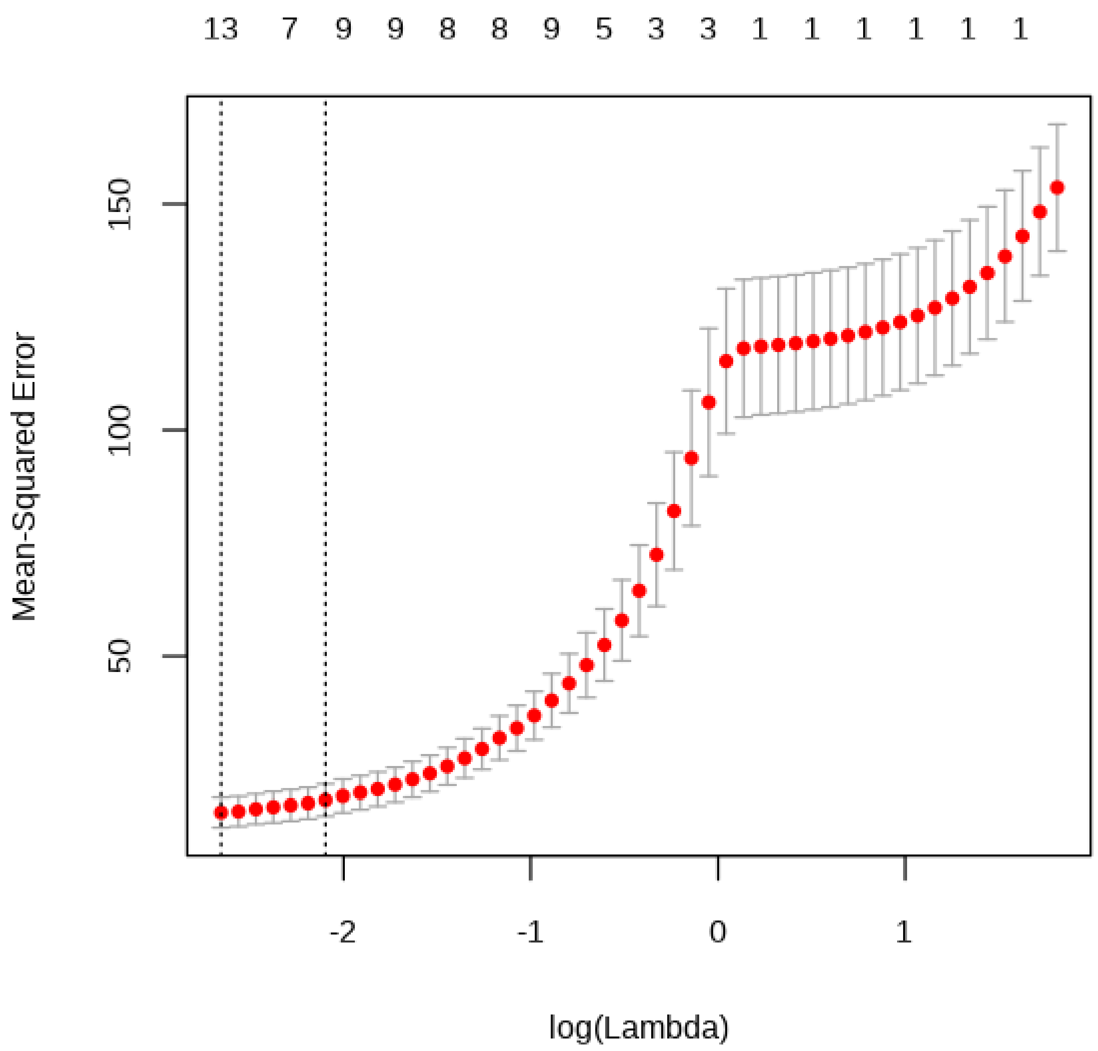
From the plot we can see that where the function converges to 3 features that have non zero coefficients is approximately at $\lambda = 0.72$ Each curve corresponds to a variable. It shows the path of its coefficient against the ℓ1-norm of the whole coefficient vector as λ varies. The axis above indicates the number of nonzero coefficients at the current λ, which is the effective degrees of freedom (df) for the lasso.

In [175]:
```
print(lasso_model)
```

```
Call:  glmnet(x = as.matrix(X), y = ytrain, family = "gaussian", alpha = 1)

      Df    %Dev   Lambda
 [1,]  0 0.00000 6.114000
 [2,]  1 0.04190 5.571000
 [3,]  1 0.07668 5.076000
 [4,]  1 0.10560 4.625000
 [5,]  1 0.12950 4.214000
 [6,]  1 0.14940 3.840000
 [7,]  1 0.16600 3.499000
 [8,]  1 0.17970 3.188000
 [9,]  1 0.19110 2.905000
[10,]  1 0.20050 2.647000
[11,]  1 0.20840 2.412000
[12,]  1 0.21490 2.197000
[13,]  1 0.22030 2.002000
[14,]  1 0.22480 1.824000
[15,]  1 0.22850 1.662000
[16,]  1 0.23160 1.515000
[17,]  1 0.23420 1.380000
[18,]  1 0.23630 1.257000
[19,]  1 0.23810 1.146000
[20,]  1 0.23960 1.044000
[21,]  3 0.34100 0.951200
[22,]  4 0.42560 0.866700
[23,]  3 0.49690 0.789700
[24,]  3 0.55560 0.719500
[25,]  3 0.60430 0.655600
[26,]  4 0.64470 0.597400
[27,]  5 0.67860 0.544300
[28,]  5 0.70670 0.495900
[29,]  7 0.73610 0.451900
[30,]  9 0.76280 0.411700
[31,]  9 0.78540 0.375200
[32,]  6 0.80690 0.341800
[33,]  8 0.82250 0.311500
[34,]  7 0.83630 0.283800
[35,]  8 0.85080 0.258600
[36,]  8 0.86030 0.235600
[37,]  8 0.87040 0.214700
[38,]  9 0.87720 0.195600
[39,]  9 0.88290 0.178200
[40,]  7 0.89120 0.162400
[41,]  7 0.89560 0.148000
[42,]  9 0.89880 0.134800
[43,] 10 0.90150 0.122800
[44,]  9 0.90750 0.111900
[45,]  7 0.91030 0.102000
[46,]  9 0.91190 0.092930
[47,] 10 0.91330 0.084680
[48,]  8 0.91920 0.077150
[49,] 13 0.92010 0.070300
[50,] 14 0.92090 0.064050
[51,] 11 0.92530 0.058360
[52,] 11 0.92680 0.053180
[53,] 12 0.92730 0.048450
[54,] 13 0.92820 0.044150
[55,] 14 0.92860 0.040230
[56,] 11 0.93120 0.036650
[57,] 14 0.93130 0.033400
[58,] 15 0.93310 0.030430
[59,] 16 0.93650 0.027730
[60,] 15 0.93890 0.025260
[61,] 15 0.94100 0.023020
[62,] 16 0.94270 0.020970
[63,] 16 0.94420 0.019110
[64,] 18 0.94540 0.017410
[65,] 18 0.94650 0.015870
[66,] 19 0.94720 0.014460
[67,] 20 0.94800 0.013170
[68,] 23 0.94860 0.012000
[69,] 28 0.94880 0.010940
[70,] 30 0.94960 0.009965
[71,] 32 0.95000 0.009079
[72,] 33 0.95040 0.008273
[73,] 36 0.95040 0.007538
```

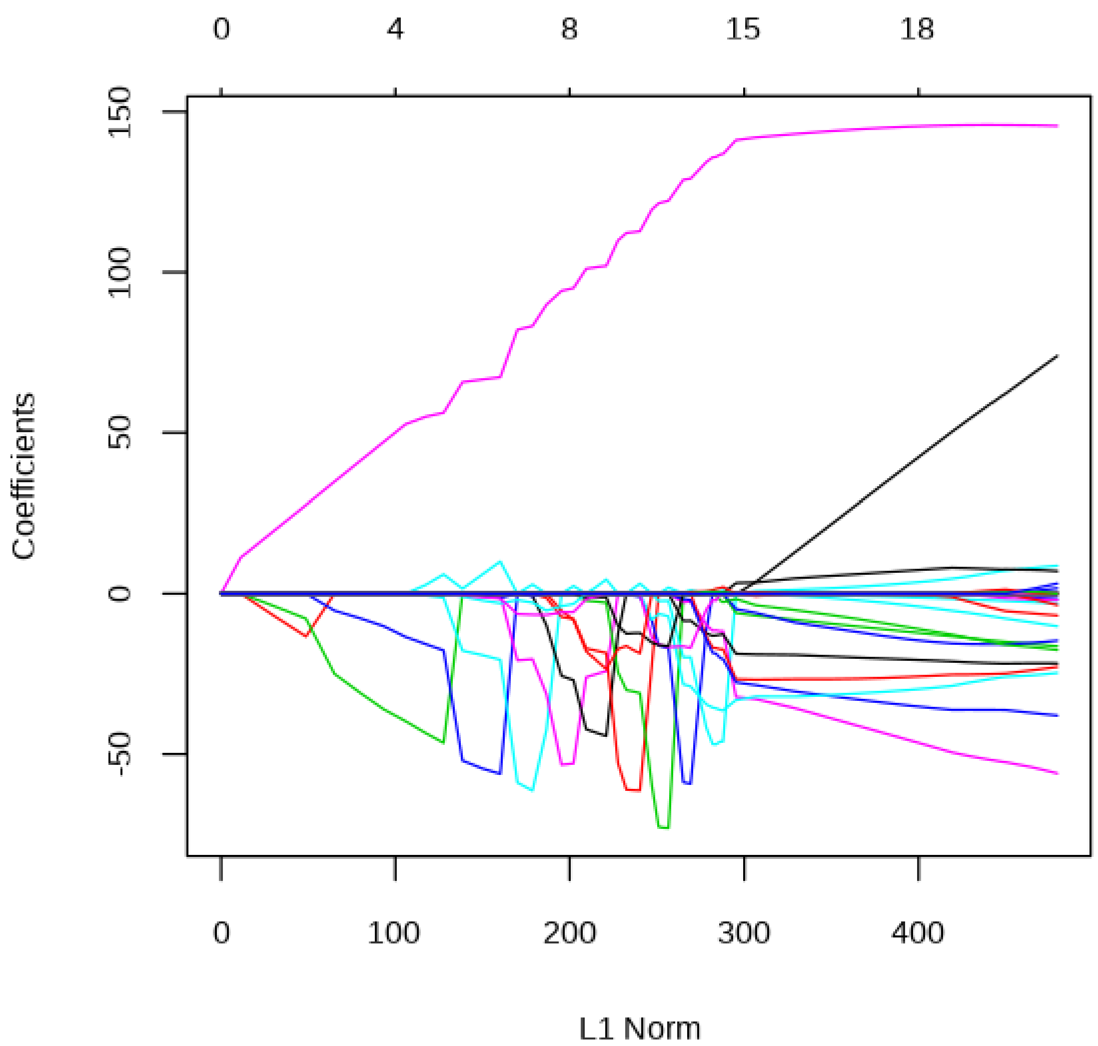
Present a plot of how degrees of freedom depend on the penalty parameter. Is the observed trend expected?

In [178]:
```
options(repr.plot.width=5, repr.plot.height=5)
lasso_model2=cv.glmnet(x = as.matrix(X),y = ytrain)
plot(lasso_model2)
```



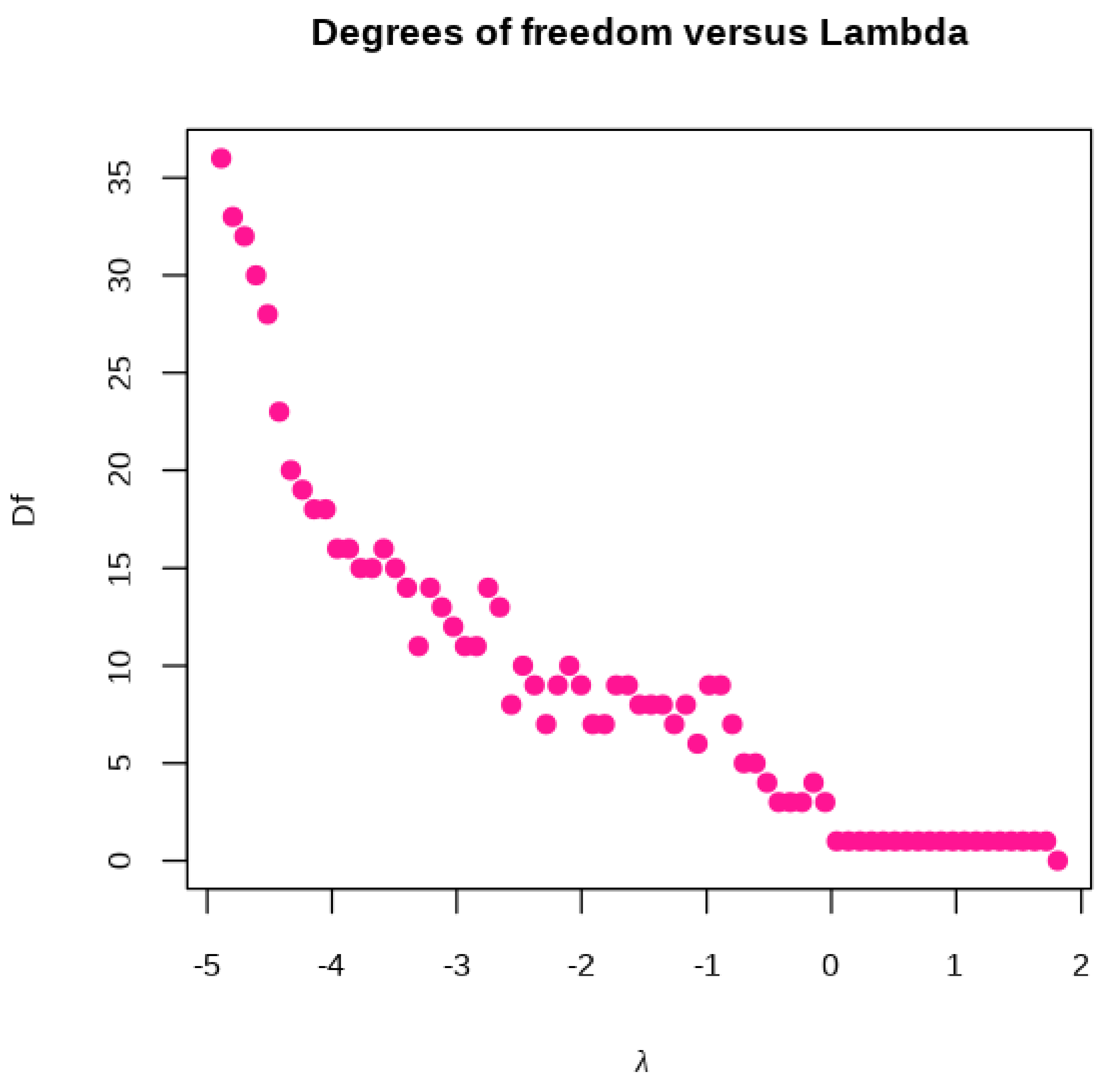
4. Present a plot of how degrees of freedom depend on the penalty parameter. Is the observed trend expected?

In [151]:
```
plot(lasso_model)
```



The axis above shows the degree of freedom. The graph shows that as the value of the shrinkage parameter lambda increases it reduces the number of features in the model. Which is exactly as expected from the given function with an sufficiently large λ it will reduce the number of features included to 0 as such decreasing the degrees of freedom.

In [149]:
```
options(repr.plot.width=5, repr.plot.height=5)
plot(x=log(lasso_model$lambda),y=lasso_model$df,xlab = expression(lambda),
ylab = "Df",main = "Degrees of freedom versus Lambda",pch=19,col='deeppink')
```



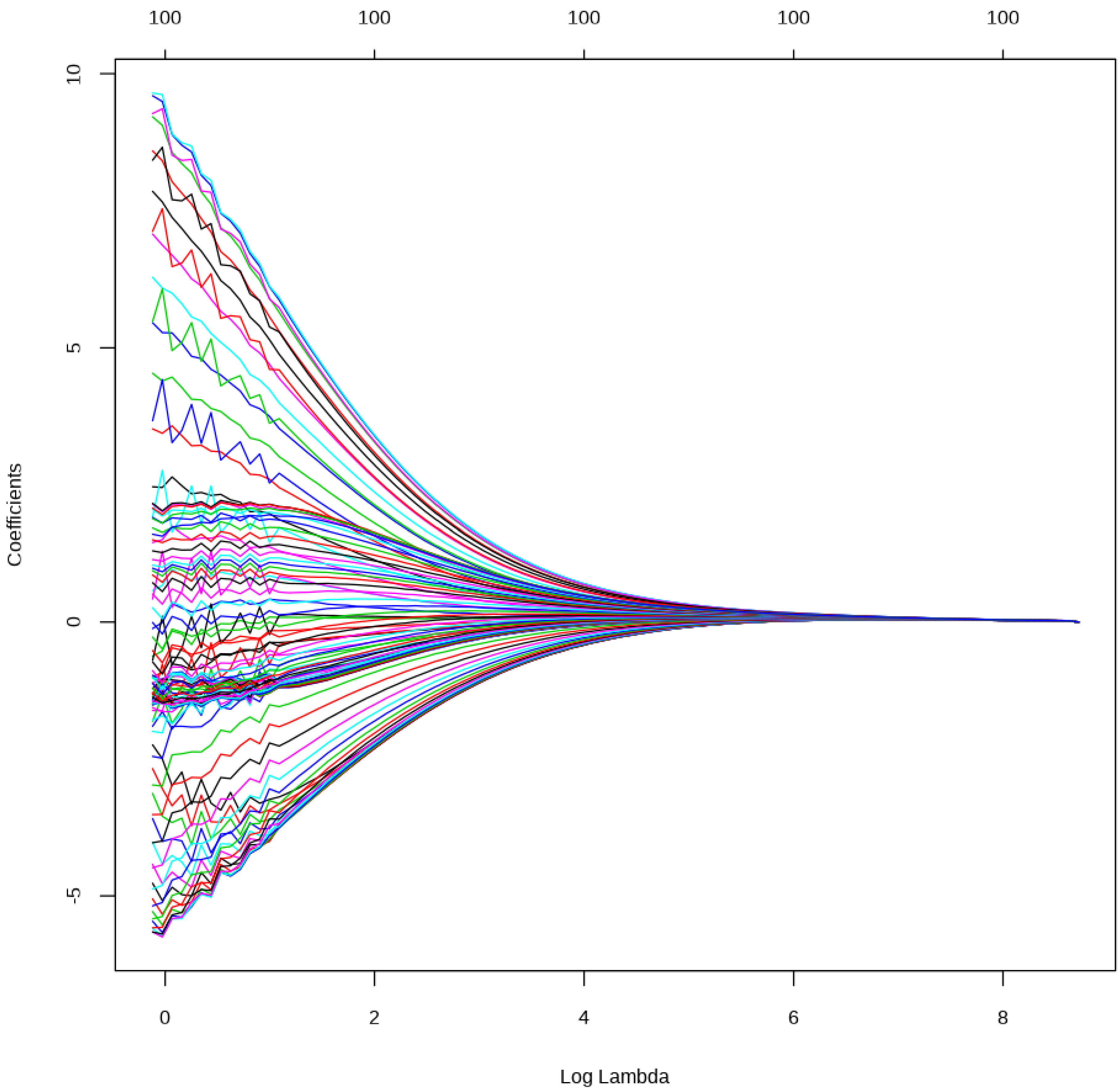
5. Repeat step 3 but fit Ridge instead of the LASSO regression and compare the plots from steps 3 and 5. Conclusions?

In [166]:
```
set.seed(12345)
ridge_model = glmnet(as.matrix(X), ytrain, alpha=0,family="gaussian")
# TO  select the model with 3 feature the degree of freedom should be 3
```

In [167]:
```
#the  ridge regression always uses all the features
ridge_model$df
```

100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100

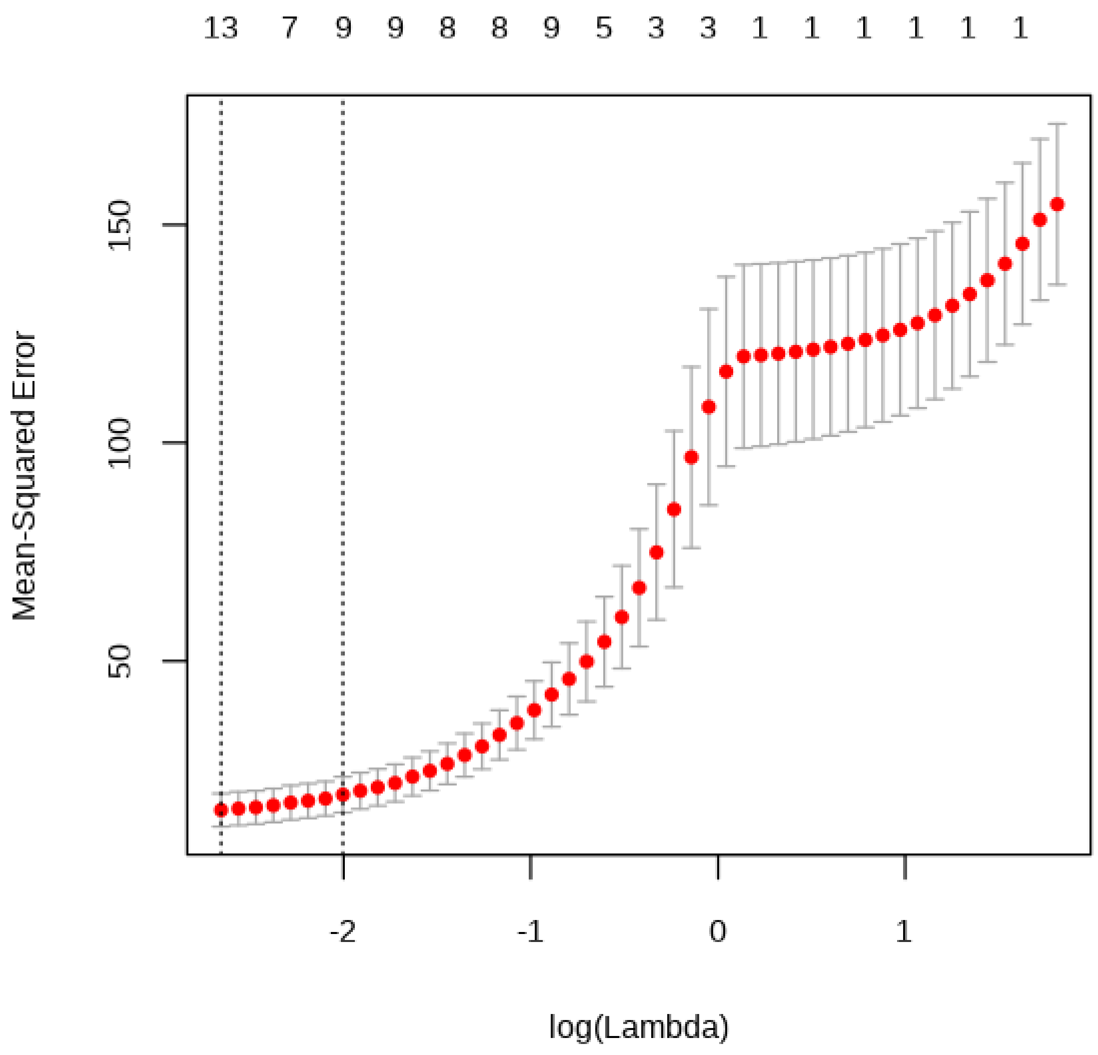
In [170]:
```
options(repr.plot.width=8, repr.plot.height=8)

plot(ridge_model,xvar = "lambda")
```



As seen in the graphs the lasso function will converge the coefficients of the model to 0 whereas the ridge regression model will reduce the coefficients close to zero but never to zero.

6. Use cross-validation to compute the optimal LASSO model. Present a plot showing the dependence of the CV score on $\log\lambda$ and comment how the CV score changes with $\log\lambda$. Report the optimal λ and how many variables were chosen in this model. Comment whether the selected λ value is statistically significantly better than $\log\lambda = -2$. Finally, create a scatter plot of the original test versus predicted test values for the model corresponding to optimal lambda and comment whether the model predictions are good.

In [180]:
```
fat_lasso_cv <- cv.glmnet(as.matrix(X),ytrain,alpha = 1)
plot(fat_lasso_cv)
```



As the $log(\lambda)$ increases the MSE will increase with a non-linear increase until approximately $log(\lambda)$ reaches 0 and then where it will slowdown the increasing rate of MSE.

This plots the cross-validation curve (red dotted line) along with upper and lower standard deviation curves along the λ sequence (error bars). Two special values along the λ sequence are indicated by the vertical dotted lines. lambda.min is the value of λ that gives minimum mean cross-validated error, while lambda.1se is the value of λ that gives the most regularized model such that the cross-validated error is within one standard error of the minimum.From the cross-validation (CV) plot with the help of the black horizontal line where it goes from the upper bound of the confidence interval of the MSE for optimal lambda. Since the black line crosses within the confidence interval of the $log(\lambda) = -2$, which means that it's not possible to conclude that the optimal lambda is significantly better than the $log(\lambda) = -2$.

In [181]:
```
print(fat_lasso_cv)
```

```
[63,] 16 0.94420 0.019110
[64,] 18 0.94540 0.017410
[65,] 18 0.94650 0.015870
[66,] 19 0.94720 0.014460
[67,] 20 0.94800 0.013170
[68,] 23 0.94860 0.012000
[69,] 28 0.94880 0.010940
[70,] 30 0.94960 0.009965
[71,] 32 0.95000 0.009079
[72,] 33 0.95040 0.008273
[73,] 36 0.95040 0.007538

$lambda.min
[1] 0.07029875

$lambda.1se
[1] 0.1348267

attr(,"class")
[1] "cv.glmnet"
```

In [188]:
```
lasso3_6_opt <- glmnet(x = as.matrix(X), y = ytrain, alpha = 1, lambda = fat_lasso_cv$lambda.min)
```

In [189]:
```
lasso3_6_opt
```

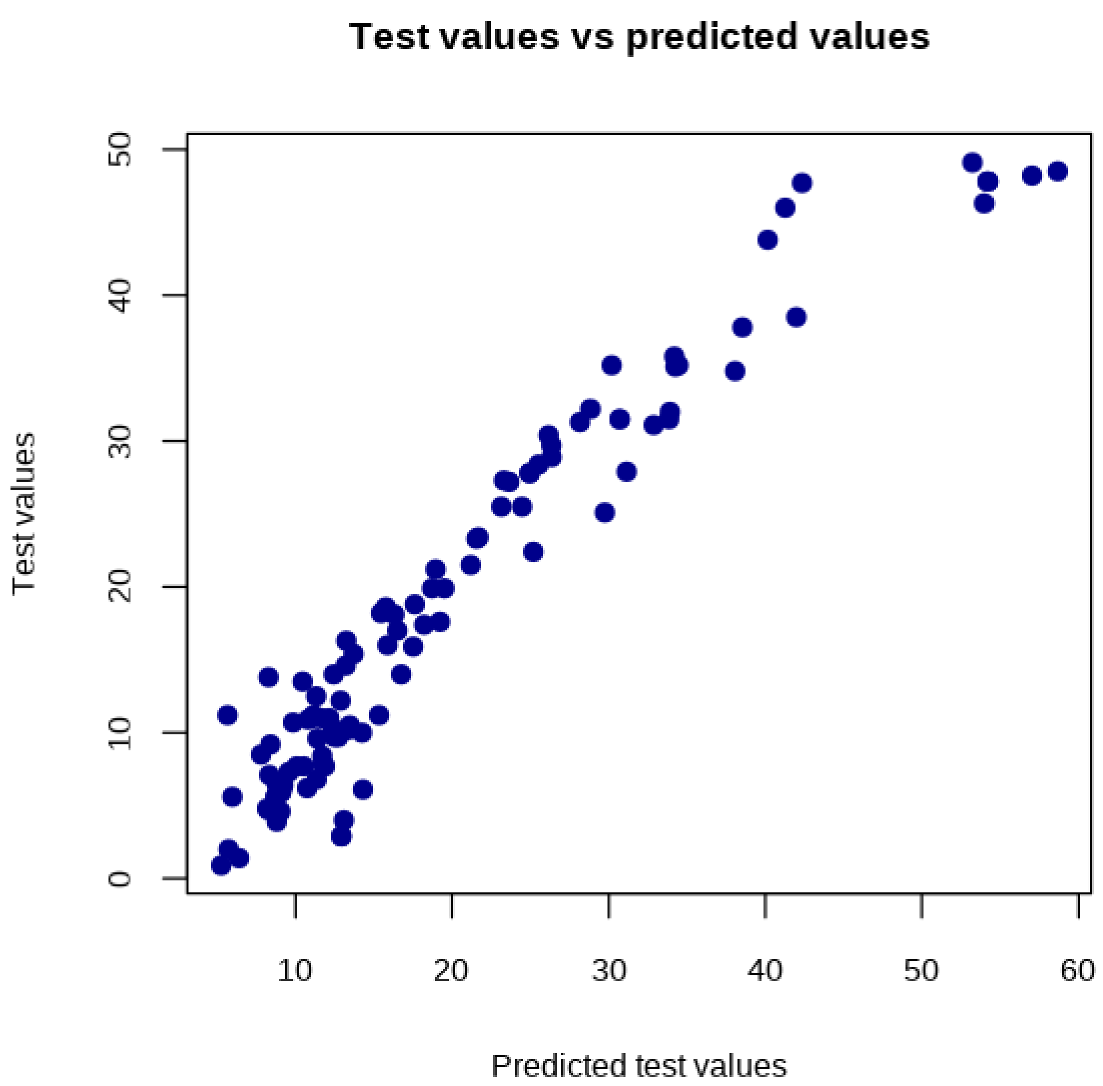
```
Call:  glmnet(x = as.matrix(X), y = ytrain, alpha = 1, lambda = fat_lasso_cv$lambda.min)

     Df   %Dev Lambda
[1,] 11 0.9235 0.0703
```

since $df = 11$,

11 features have been used in the optimal model

original test vs MSE test

In [190]:
```
fat_lasso_cv_preds <- predict(fat_lasso_cv,
newx = as.matrix(test[,1:100]),
s = "lambda.min")
plot(x = fat_lasso_cv_preds,y = test$Fat,
xlab = "Predicted test values",
ylab = "Test values",
main = "Test values vs predicted values",pch=19,col='darkblue')
```



From observing the plot we can say that the predictions are a decent estimate(at least for lower levels of Fat) as a perfect fit would results in a diagonal line over the graph. There seems to be a approximately linear trend between the predictions and true value. The correlation between the predicted values and the true values is 0.96. It can be seen that as Fat levels increase there is a slight curve to the slope of the points, as the model predicts too high levels of fat. As an example of this there are 5 points predicted to have a fat level between 50 and 60 while in reality these observations all had levels below 50.

7. Use the feature values from test data (the portion of test data with Channel columns) and the optimal LASSO model from step 6 to generate new target values. (Hint: use rnorm() and compute σ as standard deviation of residuals from train data predictions). Make a scatter plot of original Fat in test data versus newly generated ones. Comment on the quality of the data generation.

In [198]:
```
res = predict(lasso3_6_opt,newx = as.matrix(X))
sig = sd(res - ytrain)
sig
```

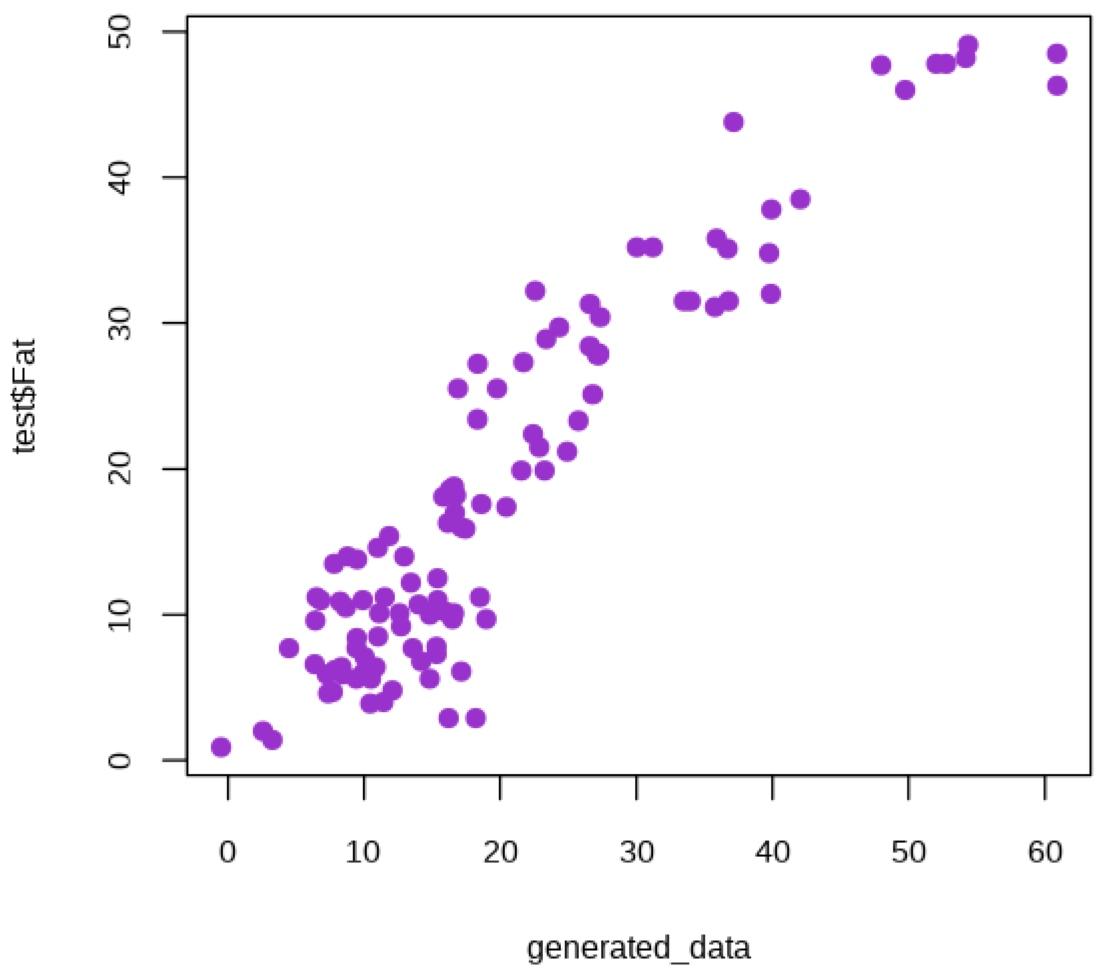
3.42007946146182

In [202]:
```
head(test[,1:100])
```

	Channel1	Channel2	Channel3	Channel4	Channel5	Channel6	Channel7	Channel8	Channel9	Channel10	...	Channel91	Channel92	Channel93	Chann
3	2.58284	2.58458	2.58629	2.58808	2.58996	2.59192	2.59401	2.59627	2.59873	2.60131	...	2.70934	2.68951	2.67009	2.65
4	2.82286	2.82460	2.82630	2.82814	2.83001	2.83192	2.83392	2.83606	2.83842	2.84097	...	2.99820	2.97367	2.94951	2.92
5	2.78813	2.78989	2.79167	2.79350	2.79538	2.79746	2.79984	2.80254	2.80553	2.80890	...	3.32201	3.30025	3.27907	3.25
9	3.27336	3.27996	3.28646	3.29300	3.29956	3.30627	3.31310	3.32006	3.32727	3.33472	...	3.84916	3.82620	3.80355	3.78
10	3.39805	3.40539	3.41271	3.42001	3.42735	3.43479	3.44245	3.45035	3.45838	3.46665	...	3.98173	3.95779	3.93407	3.91
11	3.00881	3.01401	3.01934	3.02469	3.03022	3.03601	3.04208	3.04849	3.05534	3.06260	...	3.54178	3.53029	3.51872	3.50

In [209]:
```
mu = predict(lasso3_6_opt,newx=as.matrix(test[,1:100]))
generated_data=rnorm(n=108,mean = mu,sd = sig)
```

In [218]:
```
plot(y=test$Fat,x=generated_data,pch=19,col = 'darkorchid')
```



The generated data has a larger spread than the predicted data which is reasonable with the noise included from the residuals. The generated data it self is not good as it comes from a discriminative model instead of a generative model which would produce more random outcomes for the target values. Which is why the generated data closely follows the same distribution as the observed with increased variance.