Penalized Regression

Example: Credit

Can we predict the balance that someone will carry on their Credit card?

```
library(ISLR)
library(tidyverse)
data(Credit)
dim(Credit)
```

```
## [1] 400  12
```

```
head(Credit)
```

```
##   ID  Income Limit Rating Cards Age Education Gender Student Married
## 1  1  14.891  3606    283     2  34        11   Male      No     Yes
## 2  2 106.025  6645    483     3  82        15 Female     Yes     Yes
## 3  3 104.593  7075    514     4  71        11   Male      No      No
## 4  4 148.924  9504    681     3  36        11 Female      No      No
## 5  5  55.882  4897    357     2  68        16   Male      No     Yes
## 6  6  80.180  8047    569     4  77        10   Male      No      No
##   Ethnicity Balance
## 1 Caucasian     333
## 2     Asian     903
```

Least Squares

```
m1 <- lm(Balance ~ ., data = Credit)
summary(m1)$coef
```

```
##                      Estimate  Std. Error     t val
## (Intercept)      -487.07423743 36.73406787 -13.25946
## ID                  0.04104764  0.04342822   0.94518
## Income             -7.80739871  0.23430821 -33.32106
## Limit               0.19052127  0.03278566   5.81111
## Rating              1.14248766  0.49100242   2.32684
## Cards              17.83638753  4.34324353   4.10669
## Age                -0.62954679  0.29449493  -2.13771
## Education          -1.09830902  1.59817101  -0.68722
## GenderFemale       -9.54615446  9.98430546  -0.95611
## StudentYes        426.16715394 16.73077463  25.47205
## MarriedYes         -8.78055030 10.36758355  -0.84692
## EthnicityAsian     16.85751762 14.12111625   1.19378
## EthnicityCaucasian  9.29289272 12.24194143   0.75910
```

Ridge regression

```
X <- model.matrix(Balance ~ ., data = Credit)[, -1]
X[1:2, ]
```

```
##   ID  Income Limit Rating Cards Age Education GenderFemale StudentYes
## 1  1  14.891  3606    283     2  34        11            0          0
## 2  2 106.025  6645    483     3  82        15            1          1
##   MarriedYes EthnicityAsian EthnicityCaucasian
## 1          1              0                   1
## 2          1              1                   0
```

```
Y <- Credit$Balance
lambdas <- seq(from = 1e4, to = 1e-2, length.out = 100)
library(glmnet)
rm1 <- glmnet(x = X, y = Y, alpha = 0,
              lambda = lambdas, standardize = TRUE)
class(rm1)
```

```
## [1] "elnet"  "glmnet"
```

Digression: classes and methods in R

Certain functions in R have different *methods* (i.e. functionality) when used on objects of different *class*.

```
slice(iris, 1:3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

Methods for `numeric`

```
sepal_length <- iris$Sepal.Length
class(sepal_length)
```

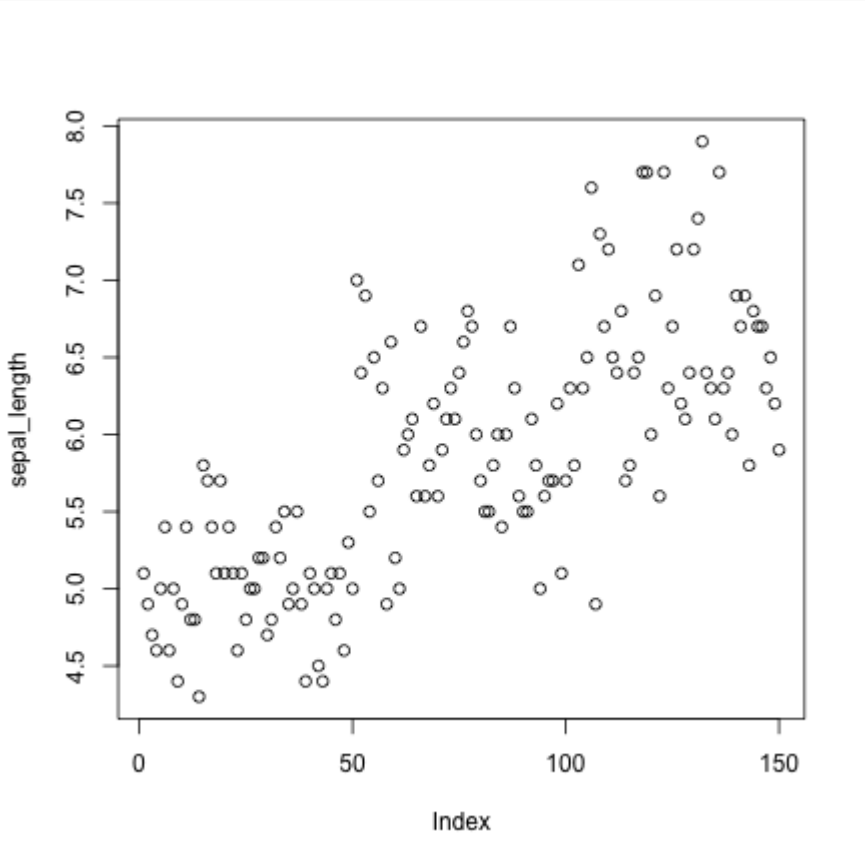
```
## [1] "numeric"
```

```
summary(sepal_length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.300   5.100   5.800   5.843   6.400   7.900
```

Methods for `numeric`, cont.

```
plot(sepal_length)
```



Methods for `lm`

```
lm1 <- lm(Sepal.Length ~ Sepal.Width + Petal.Length,
          data = iris)
class(lm1)
```

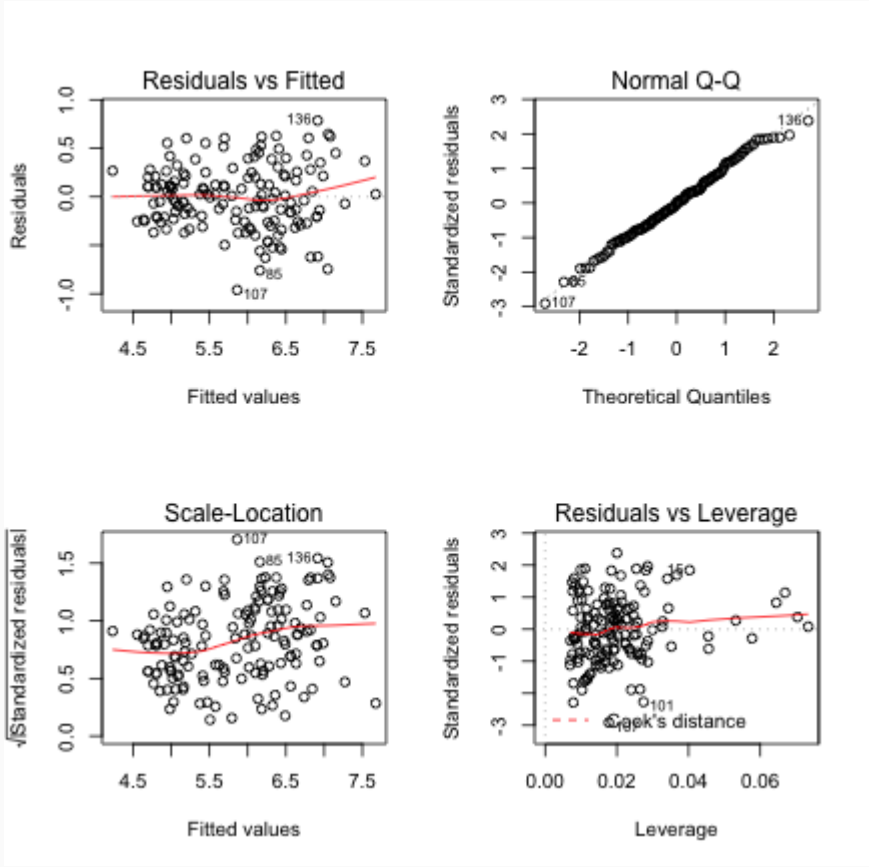
```
## [1] "lm"
```

```
summary(lm1)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96159 -0.23489  0.00077  0.21453  0.78557
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.24914    0.24797    9.07 7.04e-16 ***
## Sepal.Width   0.59552    0.06933    8.59 1.16e-14 ***
## Petal.Length  0.47192    0.01712   27.57  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Methods for `lm`, cont.

```
par(mfrow = c(2, 2))
plot(lm1)
```



Why does this happen?

Compare this:

```
iris
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 15           5.8         4.0          1.2         0.2     setosa
## 16           5.7         4.4          1.5         0.4     setosa
## 17           5.4         3.9          1.3         0.4     setosa
## 18           5.1         3.5          1.4         0.3     setosa
```

Why does this happen? cont.

. . . to this:

```
tibble(iris)
```

```
## # A tibble: 150 x 1
##    iris$Sepal.Length $Sepal.Width $Petal.Length $Petal.Width $Species
##                <dbl>        <dbl>          <dbl>         <dbl> <fct>
##  1               5.1          3.5            1.4           0.2 setosa
##  2               4.9          3              1.4           0.2 setosa
##  3               4.7          3.2            1.3           0.2 setosa
##  4               4.6          3.1            1.5           0.2 setosa
##  5               5            3.6            1.4           0.2 setosa
##  6               5.4          3.9            1.7           0.4 setosa
##  7               4.6          3.4            1.4           0.3 setosa
##  8               5            3.4            1.5           0.2 setosa
##  9               4.4          2.9            1.4           0.2 setosa
## 10               4.9          3.1            1.5           0.1 setosa
## # … with 140 more rows
```

Different classes, different methods

```
class(iris)
```

```
## [1] "data.frame"
```

```
class(tibble(iris))
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
?print.data.frame
```

print.data.frame {base} R Documentation

Printing Data Frames

Description

Print a data frame.

Usage

```
## S3 method for class 'data.frame'
print(x, ..., digits = NULL,
      quote = FALSE, right = TRUE, row.names = TRUE, max = NULL)
```

Arguments

x	object of class data.frame.
...	optional arguments to print or plot methods.
digits	the minimum number of significant digits to be used: see print.default.
quote	logical, indicating whether or not entries should be printed with surrounding quotes.
right	logical, indicating whether or not strings should be right-aligned. The default is right-alignment.
row.names	logical (or character vector), indicating whether (or what) row names

```
?print.tbl
```

formatting {tibble} R Documentation

Printing tibbles

Description

lifecycle maturing

One of the main features of the `tbl_df` class is the printing:

- Tibbles only print as many rows and columns as fit on one screen, supplemented by a summary of the remaining rows and columns.
- Tibble reveals the type of each column, which keeps the user informed about whether a variable is, e.g., <chr> or <fct> (character versus factor).

Printing can be tweaked for a one-off call by calling `print()` explicitly and setting arguments like `n` and `width`. More persistent control is available by setting the options described below.

Usage

```
## S3 method for class 'tbl'
print(x, ..., n = NULL, width = NULL, n_extra = NULL)

## S3 method for class 'tbl'
format(x, ..., n = NULL, width = NULL, n_extra = NULL)
```

Back to ridge regression...

```
rm1 <- glmnet(x = X, y = Y, alpha = 0,
              lambda = lambdas, standardize = TRUE)
class(rm1)
```

```
## [1] "elnet"  "glmnet"
```

```
summary(rm1)
```

```
##           Length Class     Mode
## a0         100   -none-    numeric
## beta      1200   dgCMatrix S4
## df         100   -none-    numeric
## dim          2   -none-    numeric
## lambda     100   -none-    numeric
## dev.ratio  100   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         6   -none-    call
## nobs         1   -none-    numeric
```

```
str(rm1)
```

```
## List of 12
##  $ a0       : Named num [1:100] 434 433 432 431 430 ...
##   ..- attr(*, "names")= chr [1:100] "s0" "s1" "s2" "s3" ...
##  $ beta     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:1200] 0 1 2 3 4 5 6 7 8 9 ...
##   .. ..@ p       : int [1:101] 0 12 24 36 48 60 72 84 96 108 ...
##   .. ..@ Dim     : int [1:2] 12 100
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : chr [1:12] "ID" "Income" "Limit" "Rating" ...
##   .. .. ..$ : chr [1:100] "s0" "s1" "s2" "s3" ...
##   .. ..@ x       : num [1:1200] 0.000781 0.233122 0.0071 0.106163 1.24
##   .. ..@ factors : list()
##  $ df       : int [1:100] 12 12 12 12 12 12 12 12 12 12 ...
##  $ dim      : int [1:2] 12 100
##  $ lambda   : num [1:100] 10000 9899 9798 9697 9596 ...
##  $ dev.ratio: num [1:100] 0.139 0.14 0.141 0.142 0.144 ...
##  $ nulldev  : num 84339912
##  $ npasses  : int 396
##  $ jerr     : int 0
##  $ offset   : logi FALSE
##  $ call     : language glmnet(x = X, y = Y, alpha = 0, lambda = lambda
##  $ nobs     : int 400
##  - attr(*, "class")= chr [1:2] "elnet" "glmnet"
```

```
rm1$lambda[100]
```

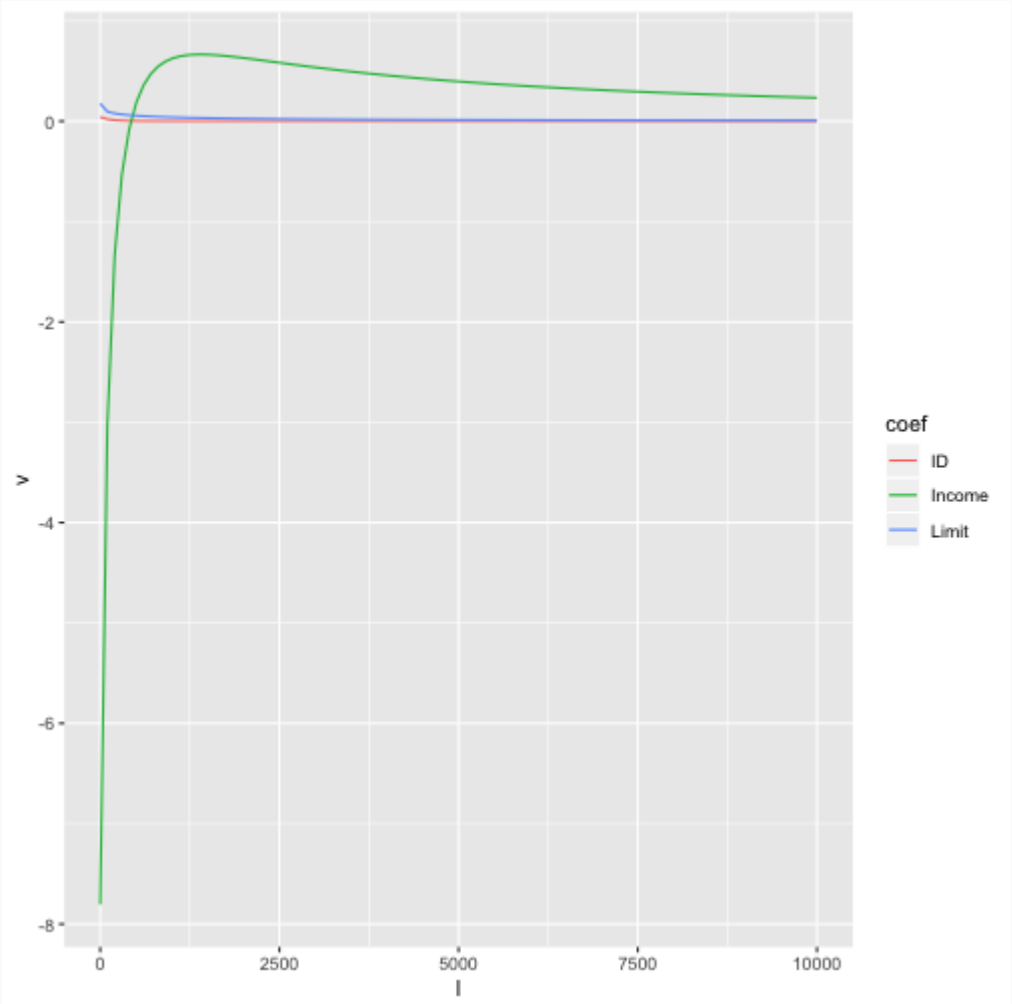
```
## [1] 0.01
```

```
coef(rm1)[1:4, 100]
```

```
##   (Intercept)            ID        Income         Limit
## -492.59495341    0.04129168   -7.80441341    0.17554032
```

```
coef(m1)[1:4]
```

```
##   (Intercept)            ID        Income         Limit
## -487.07423743    0.04104764   -7.80739871    0.19052127
```


0
-2
-4
-6
-8
v
0
2500
5000
7500
10000
l
coef
ID
Income
Limit