DS502- HW3

Mahdi Alouane and Rahul Pande

1. (10 points) Section 6.8, page 259, question 2

(a) iii. holds true. Lasso puts a budget constraint on the parameters which decreases the model variance and it reduces overfitting. However, when we put a constraint, the model bias increases. From the bias-variance trade-off concept we can say that the lasso regression will give better prediction when its increase in bias is less that its decrease in variance.

(b) iii. holds true. Like Lasso above, Ridge also puts a budget constraint on the parameters which decreases the model variance and it reduces overfitting. However, when we put a constraint, the model bias increases. Similarly as above, from the bias-variance trade-off concept we can say that the Ridge regression will give better prediction when its increase in bias is less that its decrease in variance.

(c) ii. holds true. Since non-linear methods are more flexible, they have higher variance than least squares regression but lower bias. Again, from the bias variance trade-off, if the increase on model variance is less than the decrease in bias, then non-linear model will have better prediction accuracy.

2. (20 points) Section 6.8, page264, question 11

(a)

```
library(MASS)
library(leaps)
library(glmnet)
```

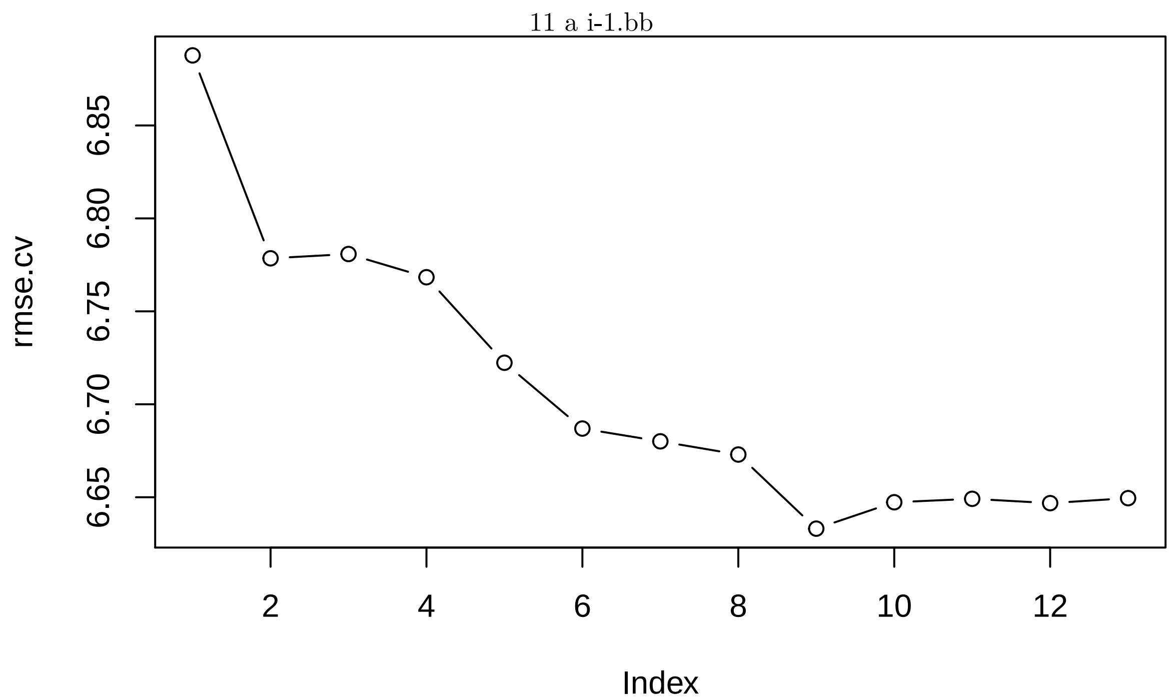
```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-16
```

```
attach(Boston)

colSums(sapply(Boston, is.na))
```

```
##    crim      zn   indus    chas     nox      rm     age     dis     rad
##       0       0       0       0       0       0       0       0       0
##     tax ptratio   black   lstat    medv
##       0       0       0       0       0
```

```
# No NAs in the dataset

# k-fold cross validation
k = 10

n = dim(Boston)[1]
p = dim(Boston)[2]-1

set.seed(123)
folds = sample(rep(1:k, length = nrow(Boston)), replace = T)
```

```r
form.subset = as.formula("crim ~ .")

cv.errors = matrix(NA, k, p)

for (i in 1:k) {
  subset.fit <- regsubsets(form.subset, Boston[folds!=i, ], nvmax = p)
  for (n.subset in 1:p) {
    m.mat <- model.matrix(form.subset, Boston[folds==i, ])
    best.coef <- coef(subset.fit, id=n.subset)
    pred <- m.mat[, names(best.coef)] %*% best.coef
    # mean squared error
    error = mean((Boston[folds==i, ]$crim - pred)^2)
    cv.errors[i,n.subset] = error
  }

}
# root mean squared values
rmse.cv = sqrt(apply(cv.errors, 2, mean))
plot(rmse.cv, type = "b")
```




```r
which(rmse.cv == min(rmse.cv))
```

```
## [1] 9
```

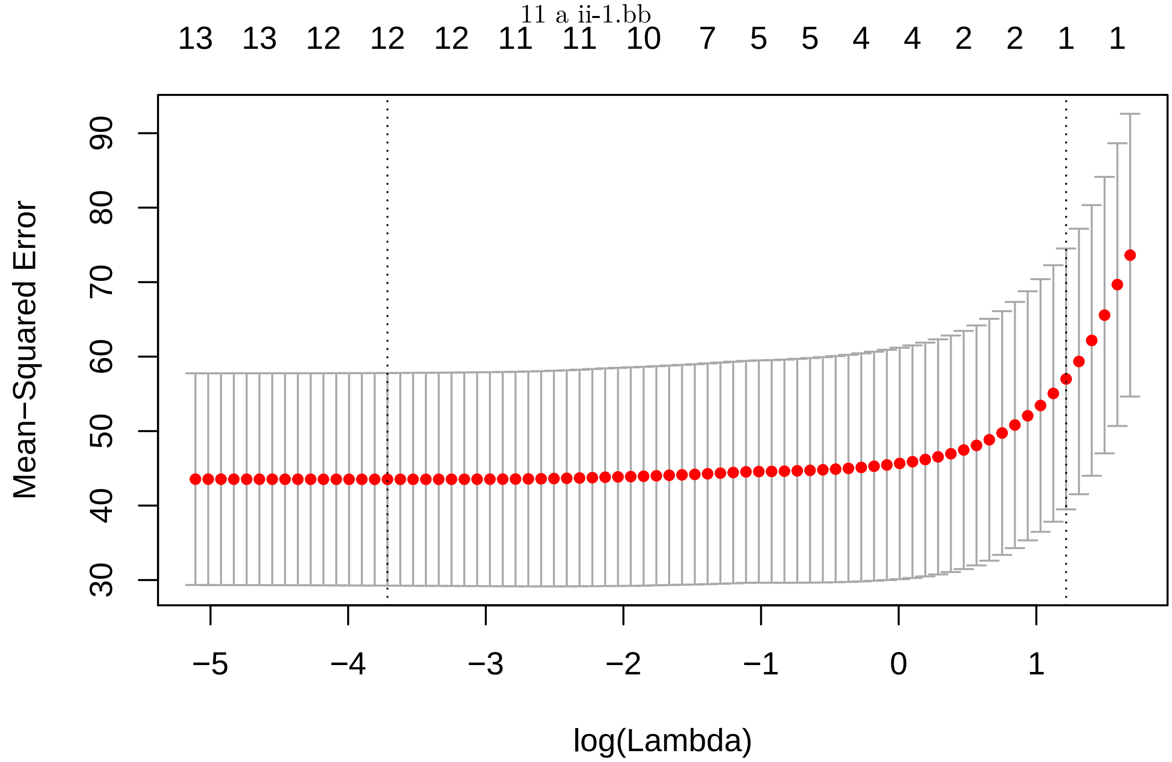
```r
rmse.cv[which(rmse.cv == min(rmse.cv))]
```

```
## [1] 6.633116
```

```r
# Lasso regression

X = model.matrix(crim ~ ., data = Boston)[, -1]
y = Boston$crim
cv.lasso = cv.glmnet(X, y, type.measure = "mse", nfolds = 10)
```

```r
plot(cv.lasso)
```

11 a ii-1.bb




```r
coef(cv.lasso)[,1]
```

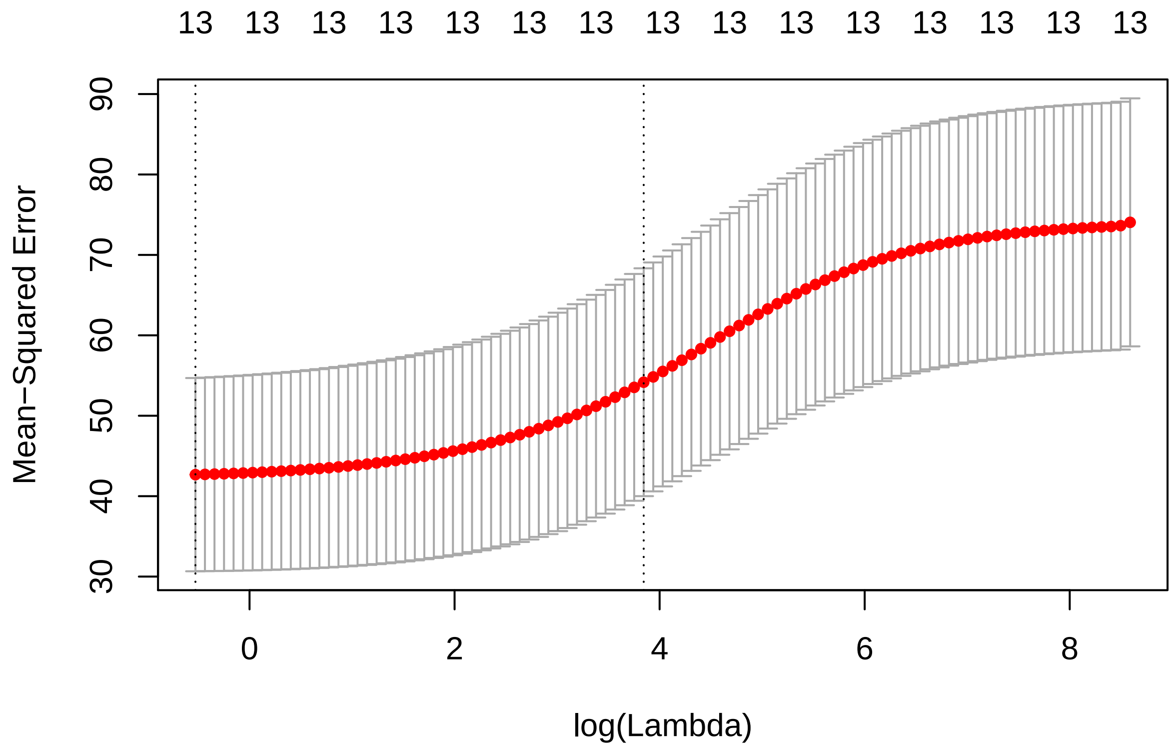
```
## (Intercept)          zn       indus        chas         nox          rm
##   1.4186415   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##         age         dis         rad         tax     ptratio       black
##   0.0000000   0.0000000   0.2298449   0.0000000   0.0000000   0.0000000
##       lstat        medv
##   0.0000000   0.0000000
```

```r
# One standard error lamda to avoid overfitting
chosen.lambda = cv.lasso$lambda.1se

# root mean square error for the chosen lamdba
sqrt(cv.lasso$cvm[cv.lasso$lambda == chosen.lambda])
```

```
## [1] 7.549995
```

```r
cv.ridge = cv.glmnet(X, y, type.measure = "mse", alpha = 0, nfolds = 10)
plot(cv.ridge)
```

11 a iii-1.bb




```r
coef(cv.ridge)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  0.757566248
## zn          -0.002519179
## indus        0.036530724
## chas        -0.259774554
## nox          2.423435801
## rm          -0.169106780
## age          0.007845252
## dis         -0.125063596
## rad          0.067739853
## tax          0.002972047
## ptratio      0.093676787
## black       -0.003748456
## lstat        0.049092196
## medv        -0.032058198
```

```r
# One standard error lamda to avoid overfitting
chosen.lambda = cv.ridge$lambda.1se

# root mean square error for the chosen lambda
sqrt(cv.ridge$cvm[cv.ridge$lambda == chosen.lambda])
```

```
## [1] 7.359946
```

```r
library(pls)
```

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

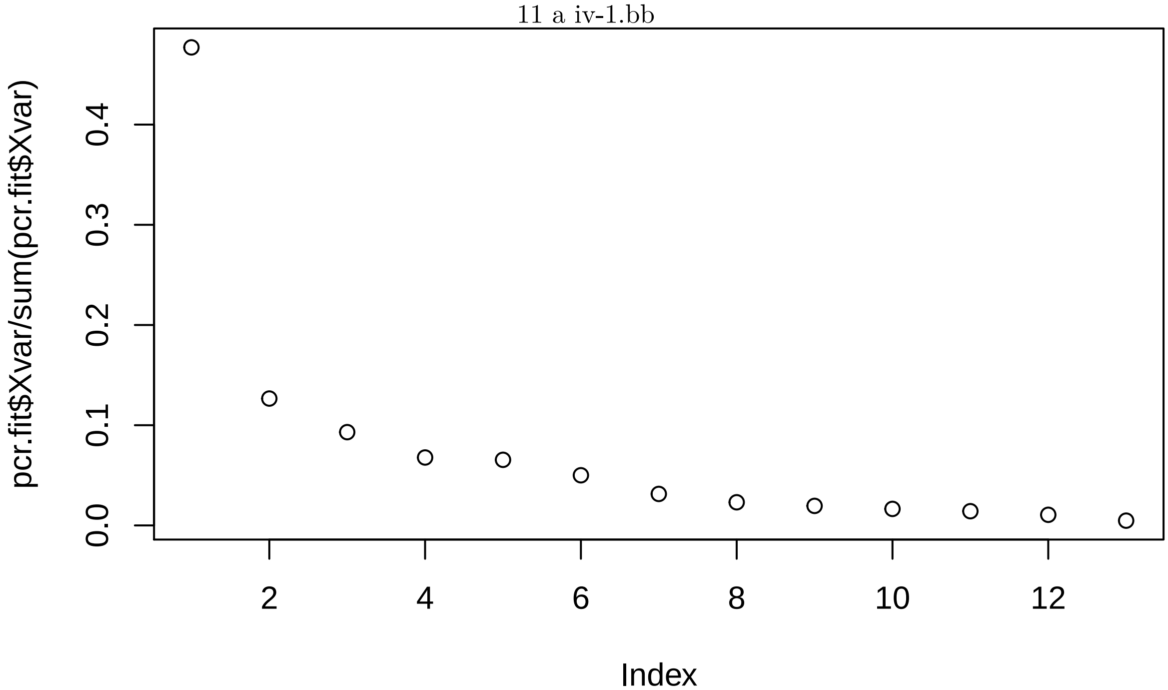
```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
pcr.fit = pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV", segments= 10)
summary(pcr.fit)
```

```
## Data:    X dimension: 506 13
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8.61    7.189    7.185    6.762    6.770    6.769    6.784
## adjCV         8.61    7.187    7.183    6.757    6.764    6.764    6.778
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.789    6.653    6.696     6.680     6.673     6.626     6.563
## adjCV    6.782    6.645    6.686     6.671     6.663     6.615     6.551
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       47.70    60.36    69.67    76.45    82.99    88.00    91.14
## crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       93.45    95.40     97.04     98.46     99.52     100.0
## crim    42.47    42.55     42.78     43.04     44.13      45.4
```

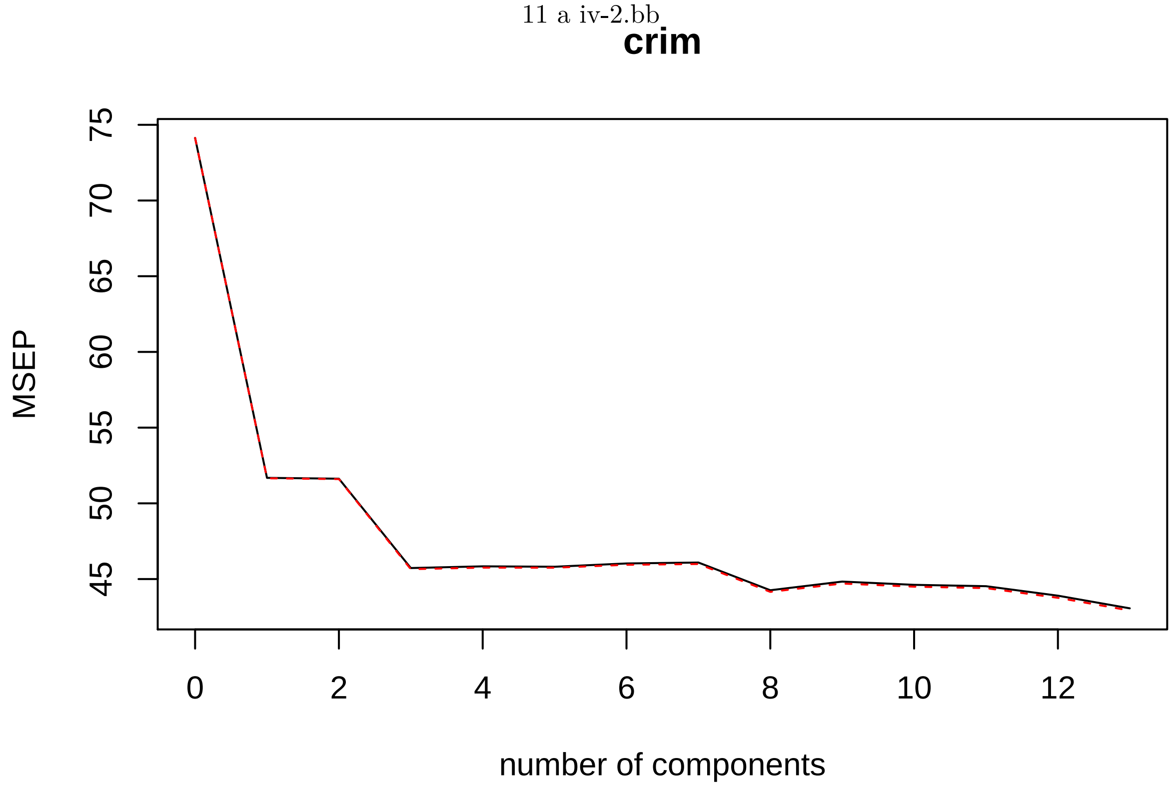
```
# variance explanation
plot(pcr.fit$Xvar/sum(pcr.fit$Xvar))
```

11 a iv-1.bb



```r
# validation plot
validationplot(pcr.fit,val.type='MSEP')
```

11 a iv-2.bb




```r
ncomp = 4

set.seed(123)
folds = sample(rep(1:k, length = nrow(Boston)), replace = T)

cv.errors = rep(0, k)

for (i in 1:k) {
  pcr.fit <- pcr(crim ~ ., data = Boston[folds !=i, ], scale = TRUE)
  pred = predict( pcr.fit, Boston[folds==i, ], ncomp=ncomp)
  error = mean((Boston[folds==i, ]$crim - pred)^2)
  cv.errors[i] = error
}
sqrt(mean(cv.errors))
```

```
## [1] 6.875975
```

(b)

```r
results <- rbind(
  c("Best Subset", 6.633116, 9),
  c("Lasso Regression", 7.549995, 1),
  c("Ridge Regression", 7.359946, 13),
  c("PCR", 6.875975, 4)
)
colnames(results) <- c("Method", "MSE", "# predictors")
```

```r
knitr::kable(results)
```

Method	MSE	# predictors
Best Subset	6.633116	9
Lasso Regression	7.549995	1
Ridge Regression	7.359946	13
PCR	6.875975	4

From the above table, we chose PCR model with 4 predictors as it has the mse very close to the lowest mse and is simpler model than the best subset model, since it has 4 predictors against the 9 predictors and thus has lower chances of overfitting the data. Second choice would be Best Subset model with 9 predictors as it has the lowest cross validation mse and has less number of predictors than Ridge Regression. Lasso Regression here seems to be underfit in this case.

(c) No. PCR has only 4 predictors since from the graph, after 4 predictors, adding another predictor does not increase the explained variance a lot. Hence we have taken only 4 predictors. We can also have criterion like taking n components which explain at least x% of the variance.

3. (10 points) Section 7.9, Page 298, question 3

We have $b_1(X) = X, b_2(X) = (X-1)^2 * I(X \geq 1)$

For $X \geq 1, I(X \geq 1) = 1$, and $X < 1, I(X \geq 1) = 0$

Substituting $\hat{\beta}_0 = 1, \hat{\beta}_1 = 1, \hat{\beta}_2 = 2$ in

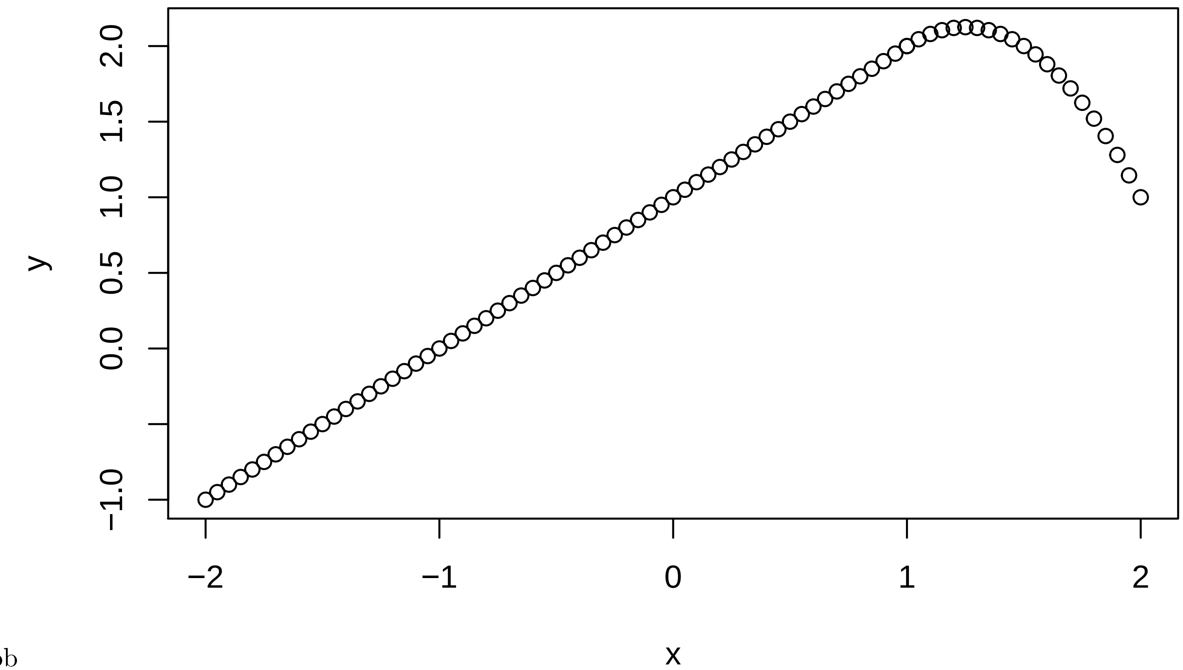
$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \epsilon$

We get,

$\hat{Y} = 1 + b_1(X) - 2b_2(X)$

For $X \geq 1, \hat{Y} = 1 + X - 2(X-1)^2 = -1 + 5X - 2X^2$, and for $X < 1, \hat{Y} = 1 + X$

```r
x_lower = seq(-2, 1, by = 0.05)
x_upper = seq(1, 2, by = 0.05)

y_lower = 1 + x_lower
y_upper = -1 + 5 * x_upper - 2 * (x_upper ^ 2)

x <- c(x_lower, x_upper)
y <- c(y_lower, y_upper)

plot(x,y)
```



2-1.bb

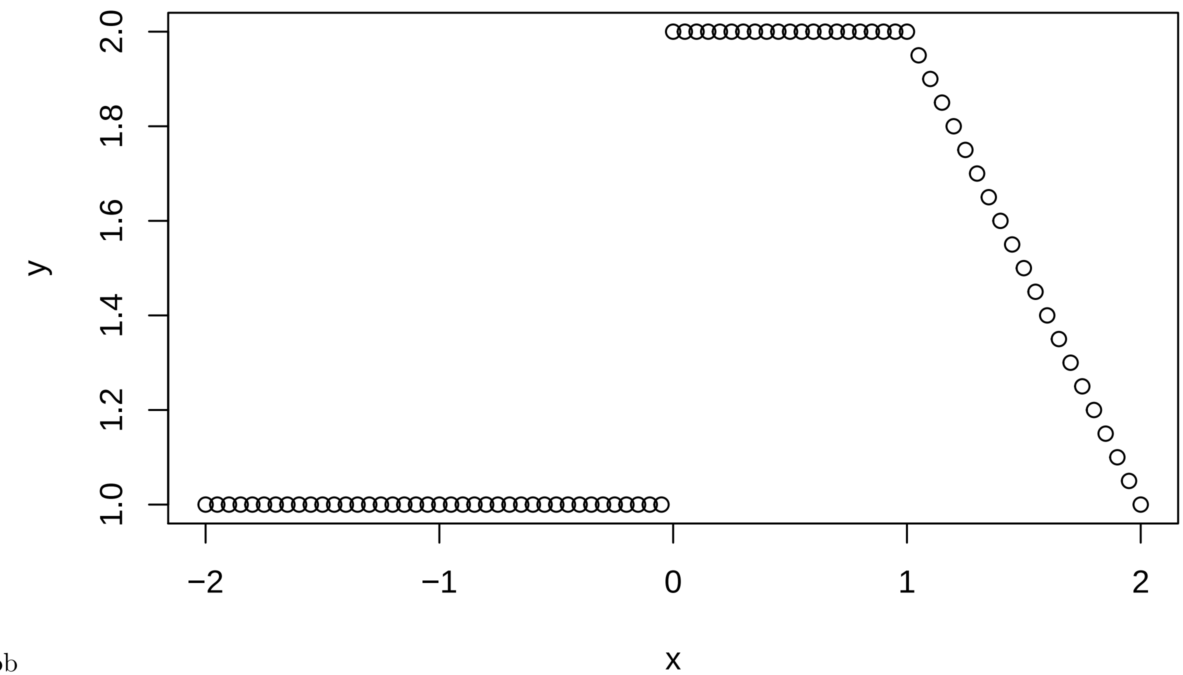
$Y = 1 + X, for X < 1 \; Now, for X = 0, Y = 1$

Therefore y-intercept is 1. Slope is 1 when $X < 1$ and by taking derivative for Y where $X \geq 1$, we get slope as $5 - 4X$

4. (10 points) Section 7.9, Page 298, question 4

Similary from above, we can split the function into multiple domains. Since there are a lot of cuts in this, we use the `I` function in R to enforce the conditions on x. It is as below.

```
x = seq(-2, 2, 0.05)
y = 1 + 1 * I(x <= 2 & x >= 0) - (x-1) * I(x <= 2 & x >= 1)  + 3 * (x-3) * I (x <= 4 & x >= 3) + I(x <=

plot(x, y)
```



4-1.bb

```r
# y-intercept
y[which(x==0)]
```

```
## [1] 2
```

The y-intercept is 2 (`y[which(x==0)]`). Slope is -1 for $1 \leq X \leq 2$ and 0 for $-2 \leq X < 0$ and $0 < X \leq 1$. The function is discontinuos at `x=0`

5. (10 points) Section 7.9, Page 299, question 6

6. (20 points) Section 7.9, Page 299, question 7

7. (10 points) Section 8.4, Page 332, question 1

8. (10 points) Section 8.4, Page 333-334, question 8