# Exercise 1

a) The pinball loss is :

$$L(y, f(x)) = \begin{cases} -(1-\alpha)(y - f(x)) & , y < f(x) \\ \alpha(y - f(x)) & , y \geqslant f(x) \end{cases}$$

$$r_{(i)} = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} = \begin{cases} \frac{\partial((1-\alpha) f(x))}{\partial f(x)} \\ \frac{\partial(-\alpha f(x))}{\partial f(x)} \end{cases}$$

$$\Rightarrow \begin{cases} 1-\alpha & , y_i < f(x_i) \\ -\alpha & , y_i \geqslant f(x_i) \end{cases}$$

b) Let, $f(x_i) = \theta_0 + \theta_1 x_i$

$r_i$ is defined as :

$$-r_i = \frac{\partial L}{\partial \theta_0} \quad ; \quad -r_i x_i = \frac{\partial L}{\partial \theta_1}$$

The updated rules are $\rightarrow$

$$\theta_0 \longrightarrow \theta_0 - \eta \frac{1}{n} \sum_{i=1}^{n} (-r_i) = \theta_0 + \eta \frac{1}{n} \sum_{i=1}^{n} r_i$$

$$\theta_1 \longrightarrow \theta_1 - \eta \frac{1}{n} \sum_{i=1}^{n} (-r_i x_i) = \theta_1 + \eta \frac{1}{n} \sum_{i=1}^{n} r_i x_i$$

# Homework 6

## C

```
data <- read.csv("gd.csv")

theta_0 <- 2
theta_1 <- 2
alpha <- 0.7
eta <- 0.8
n_iter <- 25
n <- nrow(data)

theta_0_values <- numeric(n_iter + 1)
theta_1_values <- numeric(n_iter + 1)
empirical_risk <- numeric(n_iter + 1)

#pinball loss
pinball_loss <- function(y, fx, alpha) {
  if (y < fx) {
    return((1 - alpha) * (fx - y))
  } else {
    return(alpha * (y - fx))
  }
}

#j = 0
theta_0_values[1] <- theta_0
theta_1_values[1] <- theta_1

loss_sum <- 0
for (i in 1:n) {
  x_i <- data$x[i]
  y_i <- data$y[i]
  fx <- theta_0 + theta_1 * x_i
  loss_sum <- loss_sum + pinball_loss(y_i, fx, alpha)
}
empirical_risk[1] <- loss_sum

# Gradient descent
for (j in 1:n_iter) {
  for (i in 1:n) {
    x_i <- data$x[i]
    y_i <- data$y[i]
    fx <- theta_0 + theta_1 * x_i

    grad <- if (y_i < fx) { 1 - alpha } else { -alpha }

    theta_0 <- theta_0 - eta * grad
```

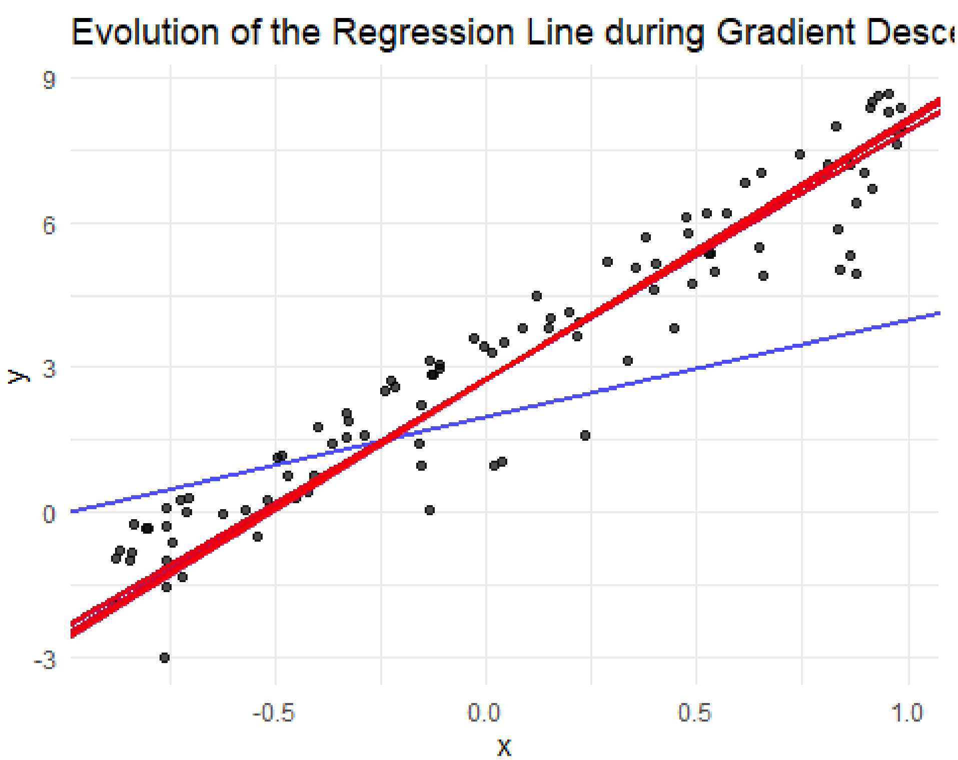
```r
    theta_1 <- theta_1 - eta * grad * x_i
  }

  theta_0_values[j + 1] <- theta_0
  theta_1_values[j + 1] <- theta_1


  loss_sum <- 0
  for (i in 1:n) {
    x_i <- data$x[i]
    y_i <- data$y[i]
    fx <- theta_0 + theta_1 * x_i
    loss_sum <- loss_sum + pinball_loss(y_i, fx, alpha)
  }
  empirical_risk[j + 1] <- loss_sum
}

results <- data.frame(
  Iteration = 0:n_iter,
  theta_0 = theta_0_values,
  theta_1 = theta_1_values,
  Empirical_Risk = empirical_risk
)

print(results)
```

```
   Iteration theta_0  theta_1 Empirical_Risk
1          0     2.0 2.000000      113.60258
2          1     2.8 5.306699       41.73446
3          2     2.8 5.146467       40.98632
4          3     2.8 5.396298       42.40155
5          4     2.8 5.304486       41.71798
6          5     2.8 5.144254       40.97840
7          6     2.8 5.394084       42.38507
8          7     2.8 5.302272       41.70150
9          8     2.8 5.142040       40.97047
10         9     2.8 5.307286       41.73883
11        10     2.8 5.147054       40.98843
12        11     2.8 5.396884       42.40592
13        12     2.8 5.305072       41.72235
14        13     2.8 5.144841       40.98050
15        14     2.8 5.394671       42.38944
16        15     2.8 5.302859       41.70587
17        16     2.8 5.142627       40.97257
18        17     2.8 5.307873       41.74320
19        18     2.8 5.147641       40.99053
20        19     2.8 5.397471       42.41029
21        20     2.8 5.305659       41.72672
22        21     2.8 5.145427       40.98260
23        22     2.8 5.395258       42.39381
24        23     2.8 5.303446       41.71024
```

```
25          24      2.8 5.143214        40.97467
26          25      2.8 5.393045        42.37733
```

## D

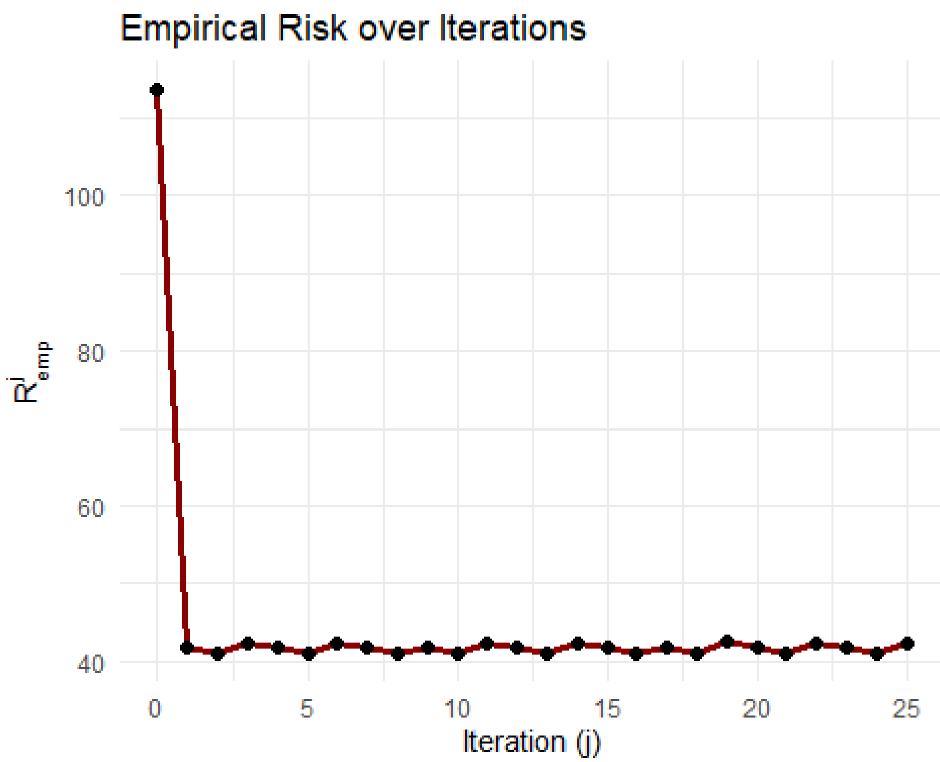
```
library(ggplot2)

ggplot(data, aes(x = x, y = y)) +
  geom_point(color = "black", alpha = 0.7) +
  lapply(0:n_iter, function(j) {
    theta0 <- theta_0_values[j + 1]
    theta1 <- theta_1_values[j + 1]
    color <- rgb(j / n_iter, 0, 1 - j / n_iter)
    geom_abline(intercept = theta0, slope = theta1, color = color, size = 0.8,
alpha = 0.7)
  }) +
  labs(title = "Evolution of the Regression Line during Gradient Descent",
       x = "x", y = "y") +
  theme_minimal()

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
ℹ Please use `linewidth` instead.
```




## E

```
library(ggplot2)
```

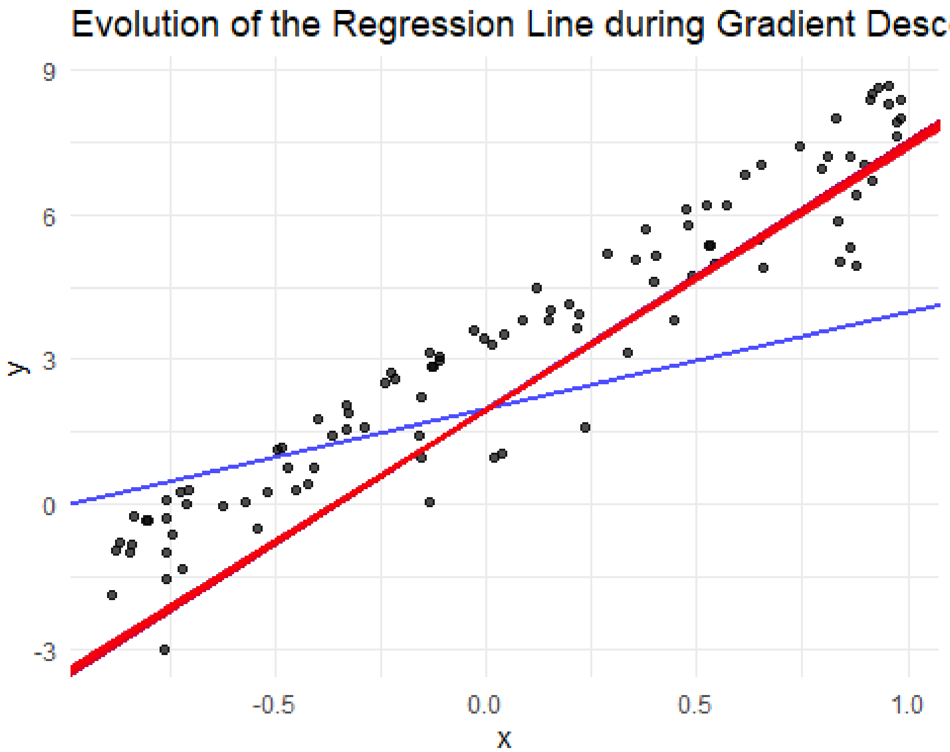
```r
ggplot(results, aes(x = Iteration, y = Empirical_Risk)) +
  geom_line(color = "darkred", linewidth = 1.2) +
  geom_point(color = "black", size = 2) +
  labs(title = "Empirical Risk over Iterations",
       x = "Iteration (j)",
       y = expression(R[emp]^j)) +
  theme_minimal()
```




# F

```r
alpha <- 0.3
eta <- 0.8
n_iter <- 25
n <- nrow(data)

theta_0 <- 2
theta_1 <- 2

theta_0_values_03 <- numeric(n_iter + 1)
theta_1_values_03 <- numeric(n_iter + 1)
empirical_risk_03 <- numeric(n_iter + 1)

pinball_loss <- function(y, fx, alpha) {
  if (y < fx) {
    return((1 - alpha) * (fx - y))
  } else {
    return(alpha * (y - fx))
```

```
  }
}

theta_0_values_03[1] <- theta_0
theta_1_values_03[1] <- theta_1

loss_sum <- 0
for (i in 1:n) {
  x_i <- data$x[i]
  y_i <- data$y[i]
  fx <- theta_0 + theta_1 * x_i
  loss_sum <- loss_sum + pinball_loss(y_i, fx, alpha)
}
empirical_risk_03[1] <- loss_sum

for (j in 1:n_iter) {
  for (i in 1:n) {
    x_i <- data$x[i]
    y_i <- data$y[i]
    fx <- theta_0 + theta_1 * x_i
    grad <- if (y_i < fx) { 1 - alpha } else { -alpha }
    theta_0 <- theta_0 - eta * grad
    theta_1 <- theta_1 - eta * grad * x_i
  }

  theta_0_values_03[j + 1] <- theta_0
  theta_1_values_03[j + 1] <- theta_1

  loss_sum <- 0
  for (i in 1:n) {
    x_i <- data$x[i]
    y_i <- data$y[i]
    fx <- theta_0 + theta_1 * x_i
    loss_sum <- loss_sum + pinball_loss(y_i, fx, alpha)
  }
  empirical_risk_03[j + 1] <- loss_sum
}

ggplot(data, aes(x = x, y = y)) +
  geom_point(color = "black", alpha = 0.7) +
  lapply(0:n_iter, function(j) {
    theta0 <- theta_0_values_03[j + 1]
    theta1 <- theta_1_values_03[j + 1]
    color <- rgb(j / n_iter, 0, 1 - j / n_iter)
    geom_abline(intercept = theta0, slope = theta1, color = color, linewidth =
0.8, alpha = 0.7)
  }) +
  labs(title = "Evolution of the Regression Line during Gradient Descent (α =
0.3)",
       x = "x", y = "y") +
  theme_minimal()
```




When repeating the estimation with α = 0.3, the resulting regression line after 25 iterations differs significantly from the one obtained with α = 0.7.

Since α = 0.3 penalizes over-predictions more heavily (y < f(x)), the gradient descent updates aim to reduce the model's tendency to overshoot. As a result, the final model has a **lower slope** and lies **below** the α = 0.7 line.

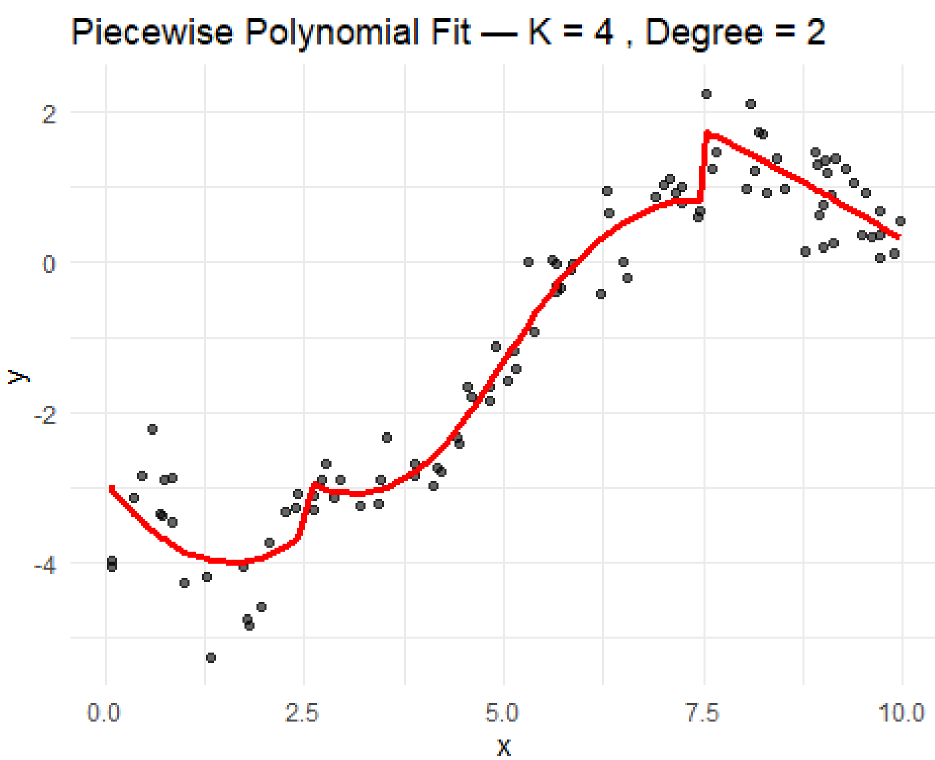
In the graph, we observe that the blue line (α = 0.3) is flatter and closer to the lower quantiles of the data, whereas the red line (α = 0.7) follows a steeper trend, capturing higher quantiles.

This behavior is expected, as the quantile loss shifts the fitted model depending on α: smaller α values lead to more conservative models, favoring lower quantiles.

# 2

```
data <- read.csv("pw.csv")

fit_piecewise_poly <- function(data, K, degree) {
  breaks <- seq(0, 10, length.out = K + 1)
  data$interval <- cut(data$x, breaks = breaks, include.lowest = TRUE, right =
FALSE)

  models <- list()
  fitted_data <- data

  for (k in levels(data$interval)) {
    subset_k <- subset(data, interval == k)
```

```r
    if (nrow(subset_k) >= degree + 1) {
      model <- lm(y ~ poly(x, degree, raw = TRUE), data = subset_k)
      models[[k]] <- model
      fitted_data$y_hat[data$interval == k] <- predict(model, newdata = subset_k)
    } else {
      models[[k]] <- NA
      fitted_data$y_hat[data$interval == k] <- NA
    }
  }

  p <- ggplot(fitted_data, aes(x = x)) +
    geom_point(aes(y = y), color = "black", size = 1.5, alpha = 0.6) +
    geom_line(aes(y = y_hat), color = "red", size = 1.2) +
    labs(title = paste("Piecewise Polynomial Fit — K =", K, ", Degree =", degree),
         x = "x", y = "y") +
    theme_minimal()

  return(p)
}

print(fit_piecewise_poly(data, K = 4, degree = 2))
```




In this piecewise polynomial regression model with K=4 intervals and polynomials of degree 2, the data range is divided into four equally sized sections over which separate quadratic models are fitted.

The resulting curve successfully captures local variations in the data, showing smooth curvature within each segment. The fit adapts well to the general trend and local fluctuations, thanks to the flexibility of quadratic polynomials.

However, since the polynomials are fitted independently in each segment, **there are discontinuities at the boundaries** between intervals. This is a common issue in piecewise models without continuity constraints.

Overall, the model balances flexibility and interpretability. Increasing K further or using higher-degree polynomials could improve the local fit, but also risks overfitting or creating even sharper jumps between segments.