21bcs022-ex6

Nagajothi.R

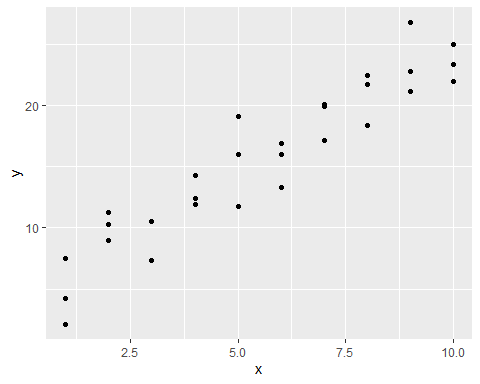
2024-03-14

#Common Question:  
library(tidyverse)

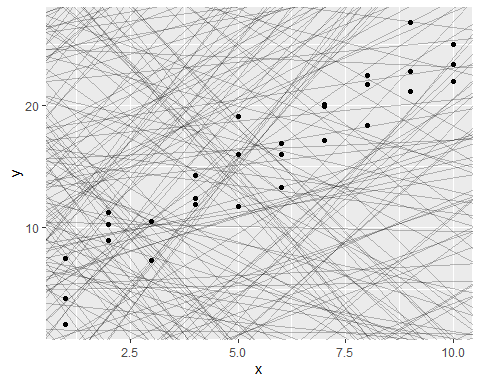
## Warning: package 'ggplot2' was built under R version 4.3.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.0 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(modelr)  
#simple model  
ggplot(sim1, aes(x, y)) +  
 geom\_point()



models <- tibble(  
a1 = runif(250, -20, 40),  
a2 = runif(250, -5, 5)  
)  
ggplot(sim1, aes(x, y)) +  
geom\_abline(  
aes(intercept = a1, slope = a2),  
data = models, alpha = 1/4  
) + geom\_point()



model1 <- function(a, data)   
{   
 a[1] + data$x \* a[2]   
}  
model1(c(7, 1.5), sim1)

## [1] 8.5 8.5 8.5 10.0 10.0 10.0 11.5 11.5 11.5 13.0 13.0 13.0 14.5 14.5 14.5  
## [16] 16.0 16.0 16.0 17.5 17.5 17.5 19.0 19.0 19.0 20.5 20.5 20.5 22.0 22.0 22.0

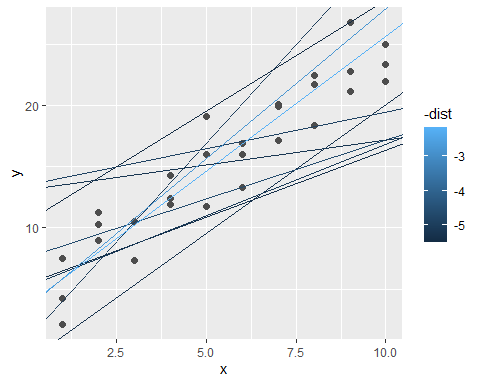
#Root Mean Squared Derivation:  
measure\_distance <- function(mod, data)   
{  
 diff <- data$y - model1(mod, data)   
 sqrt(mean(diff ^ 2))   
}   
measure\_distance(c(7, 1.5), sim1)

## [1] 2.665212

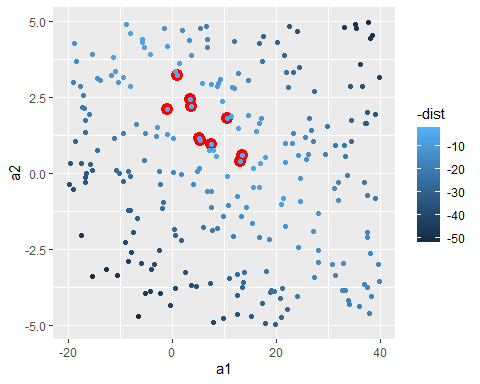
sim1\_dist <- function(a1, a2)   
{   
 measure\_distance(c(a1, a2), sim1)   
}   
models <- models %>%   
mutate(dist = purrr::map2\_dbl(a1, a2, sim1\_dist))   
models

## # A tibble: 250 × 3  
## a1 a2 dist  
## <dbl> <dbl> <dbl>  
## 1 7.54 -1.10 16.8   
## 2 -16.0 1.95 20.9   
## 3 15.5 -1.63 14.1   
## 4 -12.7 -3.14 47.9   
## 5 -8.85 4.91 8.88  
## 6 0.564 -2.02 28.6   
## 7 26.4 1.81 20.9   
## 8 12.7 1.50 6.07  
## 9 8.82 2.95 10.1   
## 10 -18.9 4.29 12.8   
## # ℹ 240 more rows

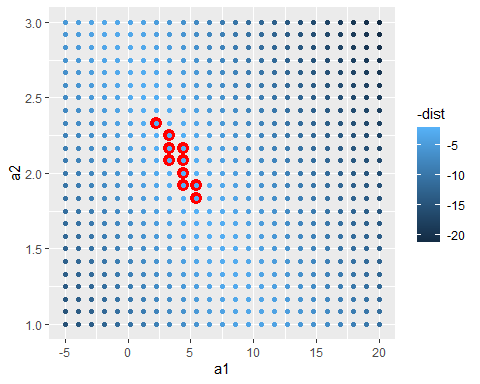
ggplot(sim1, aes(x, y)) +   
 geom\_point(size = 2, color = "grey30") +   
 geom\_abline(   
 aes(intercept = a1, slope = a2, color = -dist),  
 data = filter(models, rank(dist) <= 10) )



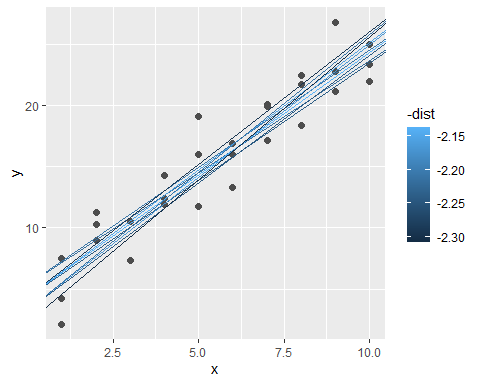
ggplot(models, aes(a1, a2)) +   
 geom\_point(   
 data = filter(models, rank(dist) <= 10),   
 size = 4, color = "red" ) +   
geom\_point(aes(color = -dist))



#Grid Search:  
grid <- expand.grid(   
 a1 = seq(-5, 20, length = 25),   
 a2 = seq(1, 3, length = 25)  
 ) %>%   
mutate(dist = purrr::map2\_dbl(a1, a2, sim1\_dist))  
  
grid %>%   
 ggplot(aes(a1, a2)) +   
 geom\_point(   
 data = filter(grid, rank(dist) <= 10),   
 size = 4, colour = "red" ) +   
 geom\_point(aes(color = -dist))



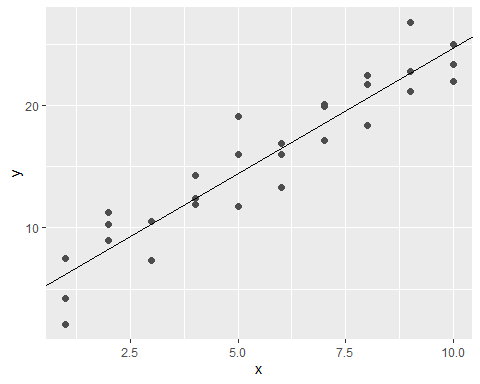
ggplot(sim1, aes(x, y)) +   
 geom\_point(size = 2, color = "grey30") +   
 geom\_abline(   
 aes(intercept = a1, slope = a2, color = -dist),   
 data = filter(grid, rank(dist) <= 10) )



#Newton–Raphson search  
best <- optim(c(0, 0), measure\_distance, data = sim1)   
best$par #best set of parameters found

## [1] 4.222248 2.051204

#> [1] 4.22 2.05  
 ggplot(sim1, aes(x, y)) +   
 geom\_point(size = 2, color = "grey30")+  
 geom\_abline(intercept = best$par[1], slope =best$par[2])



sim1\_mod <- lm(y ~ x, data = sim1)   
coef(sim1\_mod)

## (Intercept) x   
## 4.220822 2.051533

#Batch Questions:  
#1.Create the synthetic data.  
#exam score(out of 100) vs time (hours) for 120 students   
#by taking binomial distributed values for time [1:12].  
#and compute exam score using score = time \* 7 +R;   
#where R is the random value in [-7:16]  
#2.Use simple linear regression and find the model.  
#3.Do performance evaluation for the model.  
#4.Visualize the actual vs. predicted values.  
  
set.seed(0)  
n\_students <- 120  
x<- rbinom(n\_students,size=12,prob=0.5)  
R\_values <- sample(-7:16, n\_students, replace = TRUE)  
y<- sapply(1:n\_students, function(i) {  
 time <- x[i]  
 R <- R\_values[i]  
 score = time \* 7 +R;   
 return(score)  
})  
data=data.frame(x,y)  
print(head(data))

## x y  
## 1 8 58  
## 2 5 34  
## 3 5 46  
## 4 6 36  
## 5 8 58  
## 6 5 28

train\_index <- sample(1:length(x), 0.7 \* length(x))  
x\_train <- x[train\_index]  
y\_train <- y[train\_index]  
x\_test <- x[-train\_index]  
y\_test <- y[-train\_index]  
df\_train <- data.frame(x = x\_train, y = y\_train)  
df\_test <- data.frame(x = x\_test, y = y\_test)  
  
lm\_model <- function(df\_train) {  
 mean\_x <- mean(df\_train$x)  
 mean\_y <- mean(df\_train$y)  
 beta1 <- sum((df\_train$x - mean\_x) \* (df\_train$y - mean\_y)) /   
 sum((df\_train$x - mean\_x)^2)  
 beta0 <- mean\_y - beta1 \* mean\_x  
   
 # Return the coefficients  
 return(c(beta0 = beta0, beta1 = beta1))  
}  
  
model <- lm\_model(df\_train)  
  
l\_predict <- function(model, x\_test) {  
 y <- (model[2] \* x\_test) + model[1]  
 return(y)  
}  
  
y\_predict <- l\_predict(model, x\_test)  
  
# Calculate Evaluation metrics  
mse <- mean((df\_test$y - y\_predict)^2)  
rmse <- sqrt(mse)  
  
# Print evaluation metrics  
cat("Mean Squared Error (MSE):", mse, "\n")

## Mean Squared Error (MSE): 44.75253

cat("Root Mean Squared Error (RMSE):", rmse, "\n")

## Root Mean Squared Error (RMSE): 6.689733

# Plot actual vs. predicted values  
plot(x\_test, y\_test, col = "blue", pch = 16,   
 xlab = "x", ylab = "y", main = "Actual vs Predicted")  
points(x\_test, y\_predict, col = "red", pch = 16)  
legend("topleft", legend = c("Actual", "Predicted"),   
 col = c("blue", "red"), pch = 16)

