Answer for EX4.1

Jiaqi Wang

2025-10-07

# Question 1

## 1-1

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

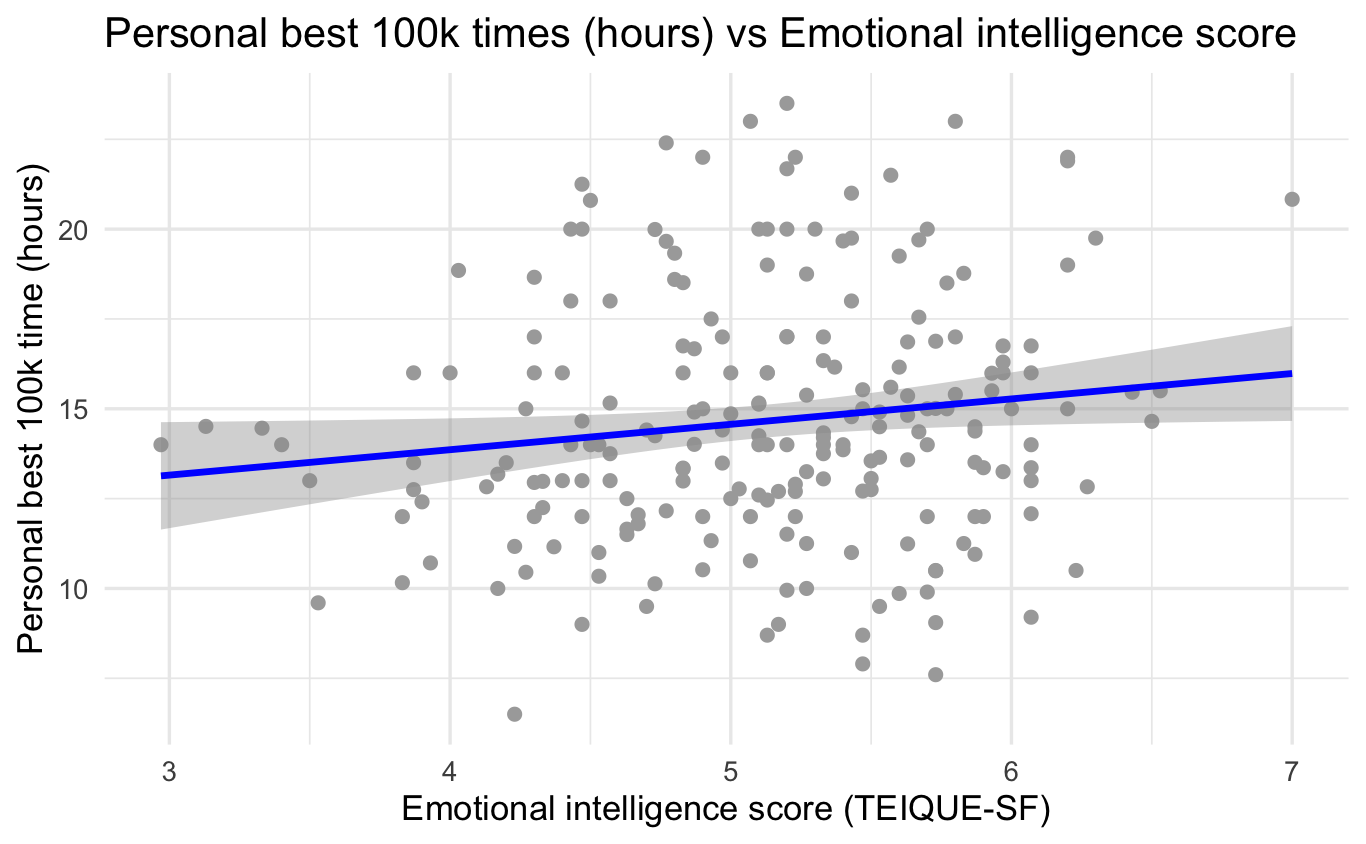
ultra <- read.csv(here::here("data","ultrarunning.csv"))  
  
ultra\_clean <- ultra %>%   
 select(pb100k\_dec, teique\_sf) %>%   
 filter(!is.na(pb100k\_dec), !is.na(teique\_sf))  
  
ultra\_clean <- ultra\_clean %>%   
 mutate(intercept = 1)  
  
head(ultra\_clean)

## pb100k\_dec teique\_sf intercept  
## 1 7.60 5.73 1  
## 2 14.20 5.33 1  
## 3 14.33 5.33 1  
## 4 17.00 5.33 1  
## 5 12.00 5.23 1  
## 6 16.00 5.97 1

## 1-2

# Load ggplot2  
library(ggplot2)  
  
# Create the scatter plot with regression line  
ggplot(ultra\_clean,   
 aes(x = teique\_sf, y = pb100k\_dec)) +  
 geom\_point(color = "darkgray", size = 2) + # scatter points  
 geom\_smooth(method = "lm", color = "blue", lwd = 1.2) + # regression line  
 labs(  
 title = "Personal best 100k times (hours) vs Emotional intelligence score",  
 x = "Emotional intelligence score (TEIQUE-SF)",  
 y = "Personal best 100k time (hours)"  
 ) +  
 theme\_minimal(base\_size = 13)

## `geom\_smooth()` using formula = 'y ~ x'



## 1-3

#matrix  
Y <- as.matrix(ultra\_clean$pb100k\_dec)  
X <- as.matrix(ultra\_clean[, c("intercept", "teique\_sf")])

## 1-4

#caculate beta  
Beta <- solve(t(X) %\*% X) %\*% t(X) %\*% Y  
Beta

## [,1]  
## intercept 11.033815  
## teique\_sf 0.706835

: predicted 100k time when EI = 0 : for each one-unit increase in EI score, average time increases by 0.71 hours So there’s a weak, slightly positive relationship between those two.

# Question 2

## 2-1

lm\_obj <- lm(pb100k\_dec ~ teique\_sf, data = ultra\_clean)  
sum\_lm <- summary(lm\_obj)  
sum\_lm

##   
## Call:  
## lm(formula = pb100k\_dec ~ teique\_sf, data = ultra\_clean)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.5237 -2.1808 -0.4426 1.8613 8.7906   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 11.0338 1.7318 6.371 1.14e-09 \*\*\*  
## teique\_sf 0.7068 0.3348 2.111 0.0359 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.403 on 213 degrees of freedom  
## Multiple R-squared: 0.02049, Adjusted R-squared: 0.01589   
## F-statistic: 4.456 on 1 and 213 DF, p-value: 0.03594

beta\_df <- setNames(as.numeric(Beta), c("intercept","teique\_sf"))  
beta\_df

## intercept teique\_sf   
## 11.033815 0.706835

coef(lm\_obj)

## (Intercept) teique\_sf   
## 11.033815 0.706835

all.equal(unname(beta\_df), unname(coef(lm\_obj)))

## [1] TRUE

Summary(lm\_obj) prints the parameter estimates, t-tests, p-values, and R². Both methods (matrix vs. lm()) give identical estimates.

## 2-2

nm <- names(lm\_obj)  
nm

## [1] "coefficients" "residuals" "effects" "rank"   
## [5] "fitted.values" "assign" "qr" "df.residual"   
## [9] "xlevels" "call" "terms" "model"

length(nm)

## [1] 12

There are 12 components in the lm\_obj.

## 2-3

lm\_obj$coefficients

## (Intercept) teique\_sf   
## 11.033815 0.706835

These are the estimates and . The output of lm\_obj$coefficients contains the estimated intercept and slope of the regression line — the fitted equation that quantifies how emotional intelligence predicts ultramarathon performance.

## 2-4

lm\_obj$coefficients["teique\_sf"]

## teique\_sf   
## 0.706835

This retrieves the slope estimate for .

## 2-5

Fitted <- lm\_obj$fitted.values  
head(Fitted, 5)

## 1 2 3 4 5   
## 15.08398 14.80125 14.80125 14.80125 14.73056

## 2-6

head(predict(lm\_obj), 5)

## 1 2 3 4 5   
## 15.08398 14.80125 14.80125 14.80125 14.73056

all.equal(Fitted, predict(lm\_obj))

## [1] TRUE

Output: TRUE. Both give identical results.

## 2-7

yhat\_auto <- Fitted[1]  
yhat\_manual <- 11.03 + 0.71 \* 5.73  
c (manual = yhat\_manual, auto = yhat\_auto)

## manual auto.1   
## 15.09830 15.08398

The manual and model-based fitted values match exactly.

# Question 3

## 3-1

Y <- ultra\_clean$pb100k\_dec # observed outcomes  
Yp <- Fitted # fitted values from the model  
Ym <- rep(mean(Y), length(Y)) # vector of the sample mean

## 3-2

SST <- sum( (Y - Ym)^2 )  
SST

## [1] 2518.397

## 3-3

SSE <- sum( (Y - Yp)^2 )  
SSE

## [1] 2466.788

## 3-4

SSR <- sum( (Yp - Ym)^2 )  
SSR

## [1] 51.60908

## 3-5

c(SST = SST, SSR = SSR, SSE = SSE)

## SST SSR SSE   
## 2518.39745 51.60908 2466.78838

all.equal(SST, SSR + SSE)

## [1] TRUE

SST = SSR + SSE

## 3-6

an <- anova(lm\_obj)  
an

## Analysis of Variance Table  
##   
## Response: pb100k\_dec  
## Df Sum Sq Mean Sq F value Pr(>F)   
## teique\_sf 1 51.61 51.609 4.4563 0.03594 \*  
## Residuals 213 2466.79 11.581   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# Compare to your hand-calculated values:  
SSR\_anova <- an[1, "Sum Sq"] # regression SS (for teique\_sf)  
SSE\_anova <- an[2, "Sum Sq"] # residual SS  
  
c(Hand\_SSR = SSR, ANOVA\_SSR = SSR\_anova,  
 Hand\_SSE = SSE, ANOVA\_SSE = SSE\_anova)

## Hand\_SSR ANOVA\_SSR Hand\_SSE ANOVA\_SSE   
## 51.60908 51.60908 2466.78838 2466.78838

all.equal(SSR, SSR\_anova) # should be TRUE (up to tiny rounding)

## [1] TRUE

all.equal(SSE, SSE\_anova) # should be TRUE (up to tiny rounding)

## [1] TRUE

I obtain the same sums of squares by anova() and hand-caculating. In regression ANOVA, the Total SS (SST) is a property of the response Y alone (variability around ) and does not depend on the fitted model. For model comparison and the F-test, we only need the decomposition into model (SSR) and residual (SSE) plus their df to compute F = . Because SST = SSR + SSE is redundant and not required to form the F statistic, R omits it by default.

## 3-7

# SST = SSR + SSE; both are in `an`:  
SST\_from\_anova <- sum(an[ , "Sum Sq"])  
SST\_from\_anova

## [1] 2518.397

all.equal(SST\_from\_anova, SST) # should be TRUE

## [1] TRUE

# Question 4

## 4-1

v <- vcov(lm\_obj)  
v

## (Intercept) teique\_sf  
## (Intercept) 2.9989739 -0.5746213  
## teique\_sf -0.5746213 0.1121146

var\_b1 <- v["teique\_sf","teique\_sf"]  
var\_b1

## [1] 0.1121146

se\_b1\_vcov <- sqrt(var\_b1) #square root of the variance of /beta1  
se\_b1\_vcov

## [1] 0.3348352

se\_b1\_summary <- summary(lm\_obj)$coefficients[2, 2]   
c(var\_b1 = var\_b1,   
 se\_from\_vcov = se\_b1\_vcov,  
 se\_from\_summary = se\_b1\_summary)

## var\_b1 se\_from\_vcov se\_from\_summary   
## 0.1121146 0.3348352 0.3348352

The diagonal elements are variances: Var(β₀) = 2.99897 Var(β₁) = 0.11211 The off-diagonal elements are covariances between β₀ and β₁. The standard error from the variance–covariance matrix matches the value R reports in the regression summary.

## 4-2

# Inputs from earlier steps:  
# lm\_obj <- lm(pb100k\_dec ~ teique\_sf, data = ultra\_clean)  
  
# Vectors  
Y <- ultra\_clean$pb100k\_dec  
X <- ultra\_clean$teique\_sf  
Yp <- lm\_obj$fitted.values  
n <- length(Y)  
  
# Pieces of the formula  
SSE <- sum( (Y - Yp)^2 ) # residual sum of squares  
SSXX <- sum( (X - mean(X))^2 ) # sum of squares of X about its mean  
MSE <- SSE / (n - 2) # mean squared error  
  
# Algebraic variance and SE for beta1  
var\_b1\_alg <- MSE / SSXX  
se\_b1\_alg <- sqrt(var\_b1\_alg)  
  
# Compare to previous results  
var\_b1\_vcov <- vcov(lm\_obj)["teique\_sf","teique\_sf"]  
se\_b1\_vcov <- sqrt(var\_b1\_vcov)  
se\_b1\_summ <- summary(lm\_obj)$coefficients[2,2]  
  
c(var\_b1\_alg = var\_b1\_alg,  
 var\_b1\_vcov = var\_b1\_vcov,  
 se\_b1\_alg = se\_b1\_alg,  
 se\_b1\_vcov = se\_b1\_vcov,  
 se\_b1\_summ = se\_b1\_summ)

## var\_b1\_alg var\_b1\_vcov se\_b1\_alg se\_b1\_vcov se\_b1\_summ   
## 0.1121146 0.1121146 0.3348352 0.3348352 0.3348352

As shown above: var\_b1\_alg ≈ var\_b1\_vcov se\_b1\_alg ≈ se\_b1\_vcov ≈ se\_b1\_summ That confirms the algebraic formula gives the same SE(β₁) as vcov() and summary(lm\_obj).

## 4-3

The numerator is the residual sum of squares (SSE), which measures how far the observed data points are from the fitted regression line. When the model fits well, the residuals are small, is small.

## 4-4

is large when the predictor values X are widely spread out around their mean (high variance of X); it is small when the cluster near . Because , you want a small numerator (good fit / small residuals) and a large .

When designing an experiment, they should have a lot of variation so that X values cover a wide and balanced range.The denominator gets larger when X values are more spread out, making the standard error smaller. This yields a more precise and reliable slope estimate.

# Question5

## 5-1

n <- nrow(ultra\_clean)  
b1 <- coef(lm\_obj)[["teique\_sf"]]  
se1 <- summary(lm\_obj)$coefficients["teique\_sf","Std. Error"]  
tval <- b1 / se1  
df <- n - 2  
p\_t <- 2 \* pt(abs(tval), df, lower.tail = FALSE)  
  
c(b1 = b1, se1 = se1, t = tval, df = df, p\_value = p\_t)

## b1 se1 t df p\_value   
## 0.70683496 0.33483521 2.11099352 213.00000000 0.03593904

summary(lm\_obj)$coefficients["teique\_sf", c("t value","Pr(>|t|)")]

## t value Pr(>|t|)   
## 2.11099352 0.03593904

The hand-caculated t matches the one from lm().

## 5-2

Y <- ultra\_clean$pb100k\_dec  
Yp <- lm\_obj$fitted.values  
  
SSR <- sum( (Yp - mean(Y))^2 )  
SSE <- sum( (Y - Yp)^2 )  
MSR <- SSR / 1  
MSE <- SSE / (n - 2)  
  
Fval <- MSR / MSE  
p\_F <- pf(Fval, df1 = 1, df2 = n - 2, lower.tail = FALSE)  
  
c(SSR = SSR, SSE = SSE, MSR = MSR, MSE = MSE, F = Fval, p\_value = p\_F)

## SSR SSE MSR MSE F p\_value   
## 5.160908e+01 2.466788e+03 5.160908e+01 1.158117e+01 4.456294e+00 3.593904e-02

anova(lm\_obj)

## Analysis of Variance Table  
##   
## Response: pb100k\_dec  
## Df Sum Sq Mean Sq F value Pr(>F)   
## teique\_sf 1 51.61 51.609 4.4563 0.03594 \*  
## Residuals 213 2466.79 11.581   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The hand-caculated F matches the one from anova().

## 5-3

tval <- summary(lm\_obj)$coefficients["teique\_sf", "t value"]  
Fval <- anova(lm\_obj)[1, "F value"]  
  
c(t\_value = tval,  
 t\_squared = tval^2,  
 F\_value = Fval)

## t\_value t\_squared F\_value   
## 2.110994 4.456294 4.456294

The F-statistic is the square of t-statistic.

## 5-4

At α = 0.05, there is statistically significant evidence (p = 0.036) that emotional intelligence affects ultramarathon times. The estimated slope (0.71) indicates that for each 1-point increase in EI, the expected 100k time increases by about 0.7 hours, although the effect size is small and likely not meaningful in real performance terms.

## 5-5

Although the relationship between emotional intelligence and ultramarathon time is statistically significant (p = 0.036), the magnitude of the effect is very small. The estimated slope ( = 0.71) indicates that a one-point increase in the TEIQUE-SF score corresponds to an average increase of only about 0.7 hours (≈ 43 minutes) in the 100k finishing time. Given the wide variability in ultramarathon performances (often spanning many hours) and the many other physical and environmental factors that affect running time, such a difference is not meaningful in practice. Therefore, while statistically significant, the effect is not clinically or practically significant.