Answer for EX4.1

Jiaqi Wang

2025-10-07

Question 1

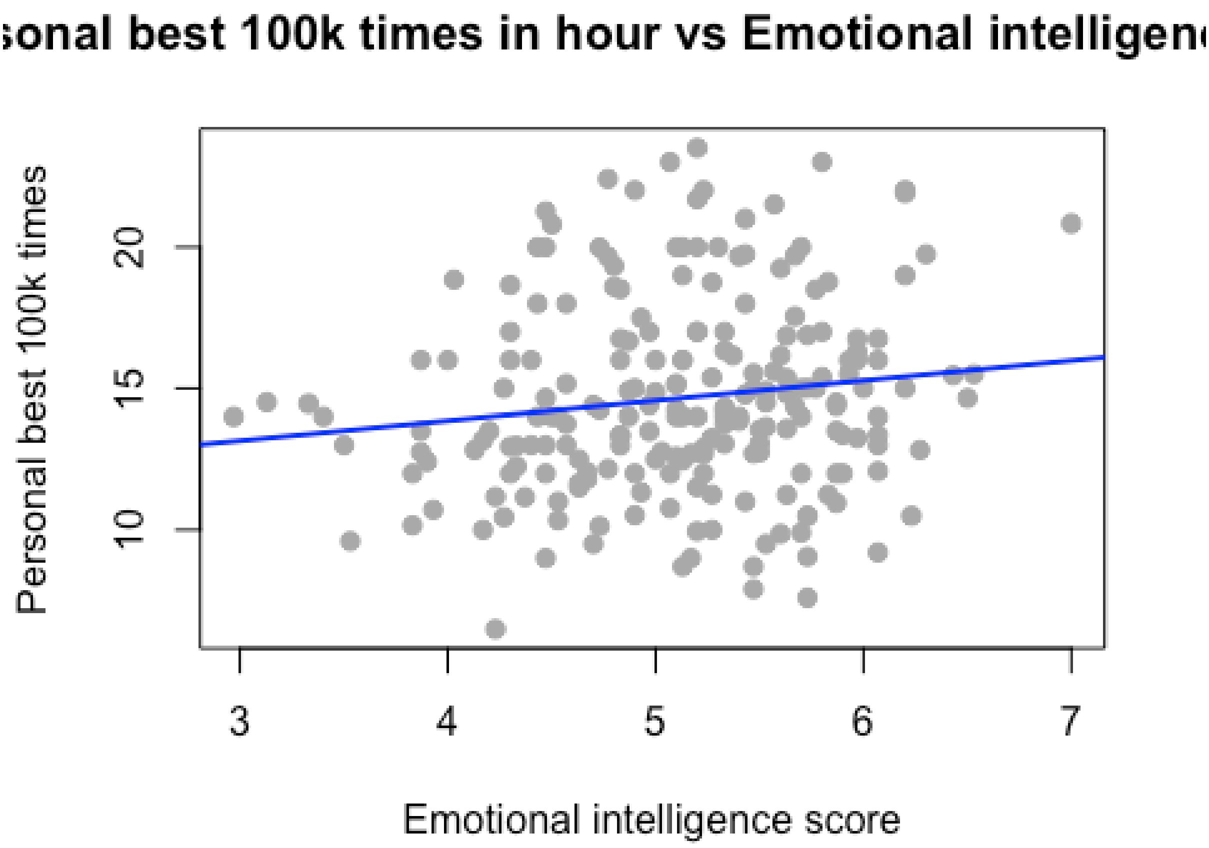
1-1

```
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

```
#scatterplot
plot(ultra_clean$teique_sf, ultra_clean$pb100k_dec,
```

```
    pch = 19, col = "darkgray",
    main = "Personal best 100k times in hour vs Emotional intelligence score",
    xlab = "Emotional intelligence score",
    ylab = "Personal best 100k times")
abline(lm(pb100k_dec ~ teique_sf,
        data = ultra_clean), col = "blue", lwd = 2)
```



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
```

$\hat{\beta}_0 = 11.03$: predicted 100k time when EI = 0 $\hat{\beta}_1 = 0.71$: 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 R^2. 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 $\hat{\beta}_0$ and $\hat{\beta}_1$.

2-4

```
lm_obj$coefficients["teique_sf"]
```

```
## teique_sf
##  0.706835
```

This retrieves the slope estimate for $\hat{\beta}_1$.

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

```r
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 $\bar{Y}$) 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 = $\frac{\text{MSR}}{\text{MSE}} = \frac{\text{SSR}/1}{\text{SSE}/(n-2)}$. Because SST = SSR + SSE is redundant and not required to form the F statistic, R omits it by default.

3-7

```r
# 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(β_0) = 2.99897 Var(β_1) = 0.11211 The off-diagonal elements are covariances between β_0 and β_1. 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)
```

```r
# 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(β_1) as vcov() and summary(lm_obj).

4-3

The numerator $\sum_i \left(Y_i - \hat{Y}_i\right)^2$ 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 $Y_i - \hat{Y}_i$ are small, $\sum_i \left(Y_i - \hat{Y}_i\right)^2$ is small.

4-4

$\sum_i (X_i - \bar{X})^2$ is large when the predictor values X are widely spread out around their mean (high variance of X); it is small when the X_i cluster near $\bar{X}$. Because $\mathrm{SE}\left(\hat{\beta}_1\right) = \sqrt{\frac{\mathrm{SSE}/(n-2)}{SSXX}}$, you want a small numerator (good fit / small residuals) and a large $SSXX$.

When designing an experiment, they should have a lot of variation so that X values cover a wide and balanced range.The denominator $\sum(X_i - \bar{X})^2$ 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 ($\hat{\beta}_1 = 0.71$) 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.