# hw04

Zeng Geduo 2020302121048

2022/4/27

## 7.2

Refer to the law data (bootstrap). Use the jackknife-after-bootstrap method to estimate the standard error of the bootstrap estimate of se(R).

### Solution

```r
data(patch, package = "bootstrap")
n <- nrow(patch)
x <- rnorm(1000)
B <- 2000
theta.b <- numeric(B)
# set up storage for the sampled indices
indices <- matrix(0, nrow = B, ncol = n)
# jackknife-after-bootstrap step 1: run the bootstrap
for (b in 1:B) {
i <- sample(1:n, size = n, replace = TRUE)
x <- x[i]
theta.b[b] <- mean(x)
#save the indices for the jackknife
indices[b, ] <- i
}
#jackknife-after-bootstrap to est. se(se)
se.jack <- numeric(n)
for (i in 1:n) {
#in i-th replicate omit all samples with x[i]
keep <- (1:B)[apply(indices, MARGIN = 1,
FUN = function(k) {!any(k == i)})]
se.jack[i] <- sd(theta.b[keep])
}

print(sd(theta.b))
```

```
## [1] 0.02012235
```

```r
print(sqrt((n-1) * mean((se.jack - mean(se.jack))^2)))
```

```
## [1] 0.01323455
```

## 7.3

Obtain a bootstrap t confidence interval estimate for the correlation statistic in Example 7.2 (law data in bootstrap).

**Solution 1**

First compute the $\hat{R}$.

```
n <- 100
r_mu <- r(tbl_law)
r_mu
```

```
## [1] 0.7763745
```

Then use bootstrap to compute the $\widehat{\hat{se}R^{(b)}}$ for every sample in 7.2.

```
# bootstrap for every sample
boot_grp_boot <- list()
grp_boot_grp_boot <- list()
sd_r_boot <- numeric(0)

for (i in 1:n){
  boot_grp_boot[i] <- modelr::bootstrap(grp_boot[[i]] %>%
                                  as.data.frame(), n)

  grp_boot_grp_boot[[i]] <- boot_grp_boot[[i]] %>%
    map(as_tibble) # convert to tibble

  sd_r_boot[[i]] <- grp_boot_grp_boot[[i]] %>%
      map_dbl(r) %>%
      sd()
}
```

Then Compute the t-statistics for every $\widehat{\hat{se}R^{(b)}}$ and compute the quantile of them.

```
t_boot <- (r_boot - r_mu) / sd_r_boot

alpha <- 0.1

Qt <- quantile(t_boot, c(alpha/2, 1-alpha/2), type = 1)
```

In the end, compute the sample standard deviation $\hat{se}\hat{R}$ in the first resampling and compute the Bootstrap t CI.

```
se_boot <- sd(r_boot)

r_mu + Qt * se_boot
```

```
##        5%       95%
## 0.6294145 1.5907464
```

**Solution 2**

```

get_r <- function(data, indices, x, y) {

  d <- data[indices, ]
  r <- as.numeric(cor(d[x], d[y]))

  return(r)

}
```

```r
get_r_var <- function(x, y, data, indices, its) {

  d <- data[indices, ]
  r <- cor(d[x], d[y]) %>%
    as.numeric()

  n <- nrow(d)

  v <- boot::boot(
    x=x,
    y=y,
    R = its,
    data = d,
    statistic = get_r
  ) %>%
    pluck("t") %>%
    var(na.rm = TRUE)

  return(c(r, v))

}

boot_t_out <- boot::boot(
  x = "LSAT", y = "GPA", its = 200,
  R = 1000, data = tbl_law, statistic = get_r_var
)
boot::boot.ci(boot_t_out, type="stud")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot::boot.ci(boot.out = boot_t_out, type = "stud")
##
## Intervals :
## Level    Studentized
## 95%   (-0.4886,  0.9973 )
## Calculations and Intervals on Original Scale
```

## 7.4

Refer to the air-conditioning data set *aircondit* provided in the *boot* package. The 12 observations are the times in hours between failures of airconditioning equipment [63, Example 1.1]:

3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487.

Assume that the times between failures follow an exponential model Exp($\lambda$). Obtain the MLE of the hazard rate $\lambda$ and use bootstrap to estimate the bias and standard error of the estimate.

### Solution

```r
data_air <- boot::aircondit

vec_air_time_diff <- data_air %>%
  as_vector() %>%
  diff()

vec_air_time_diff
```

```
##  hours2  hours3  hours4  hours5  hours6  hours7  hours8  hours9 hours10 hours11 
##       2       2      11      25      42       6       7       2      30     100 
## hours12 
##     257 
```

Because the times between failures follow an exponential model Exp($\lambda$), so the likelihood function is

$$L(\lambda) = \prod \lambda e^{\lambda X_i}$$

.

Considering that

$$lnL(\lambda) = nln\lambda - \lambda \sum X_i$$

$$\frac{\partial lnL}{\partial \lambda} = \frac{n}{\lambda} - \sum X_i = 0$$

so that

$$MLE(\lambda) = \frac{1}{\overline{X_i}}$$

```
MLE_exp <- function(data, i){
  return(1 / mean(data[i]))
}
boot_obj <- boot::boot(vec_air_time_diff, MLE_exp, n)
boot_obj
```

```
## 
## ORDINARY NONPARAMETRIC BOOTSTRAP
## 
## 
## Call:
## boot::boot(data = vec_air_time_diff, statistic = MLE_exp, R = n)
## 
## 
## Bootstrap Statistics :
##       original     bias    std. error
## t1* 0.02272727 0.00710451   0.0177203
```

## 7.5

Refer to Exercise 7.4. Compute 95% bootstrap confidence intervals for the mean time between failures $1/\lambda$ by the standard normal, basic, percentile, and BCa methods. Compare the intervals and explain why they may differ.

### Solution

```
boot::boot.ci(boot_obj, type=c("norm", "basic", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 100 bootstrap replicates
## 
## CALL : 
```

```
## boot::boot.ci(boot.out = boot_obj, type = c("norm", "basic",
##     "perc", "bca"))
##
## Intervals :
## Level      Normal              Basic
## 95%   (-0.0191,  0.0504 )   (-0.0353,  0.0350 )
##
## Level     Percentile            BCa
## 95%   ( 0.0105,  0.0807 )   ( 0.0099,  0.0744 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
## Some percentile intervals may be unstable
## Some BCa intervals may be unstable
```

## 7.7

Refer to Exercise 7.6. Efron and Tibshirani discuss the following example [84, Ch. 7]. The five-dimensional scores data have a $5 \times 5$ covariance matrix $\Sigma$, with positive eigenvalues $\lambda_1 > \cdots > \lambda_5$. In principal components analysis,

$$\theta = \frac{\lambda_1}{\sum_{j=1}^{5} \lambda_j}$$

measures the proportion of variance explained by the first principal component. Let $\hat{\lambda}_1 > \cdots > \hat{\lambda}_5$ be the eigenvalues of $\hat{\Sigma}$, where $\hat{\Sigma}$ is the MLE of $\Sigma$. Compute the sample estimate

$$\hat{\theta} = \frac{\hat{\lambda}_1}{\sum_{j=1}^{5} \hat{\lambda}_j}$$

### Solution

```
tbl_scor <- bootstrap::scor
res_pca_scor <- prcomp(tbl_scor, scale = TRUE)

get_theta <- function(data, indices){
  d <- data[indices,]
  egn_vl <- prcomp(d, scale=TRUE)$sdev ** 2
  return(egn_vl[[1]] / sum(egn_vl))
}

boot_scor <- boot::boot(tbl_scor, get_theta, n)
boot_scor
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot::boot(data = tbl_scor, statistic = get_theta, R = n)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 0.636196 -0.00155686  0.04300418
```

### 7.8

Refer to Exercise 7.7. Obtain the jackknife estimates of bias and standard error of $\hat{\theta}$.

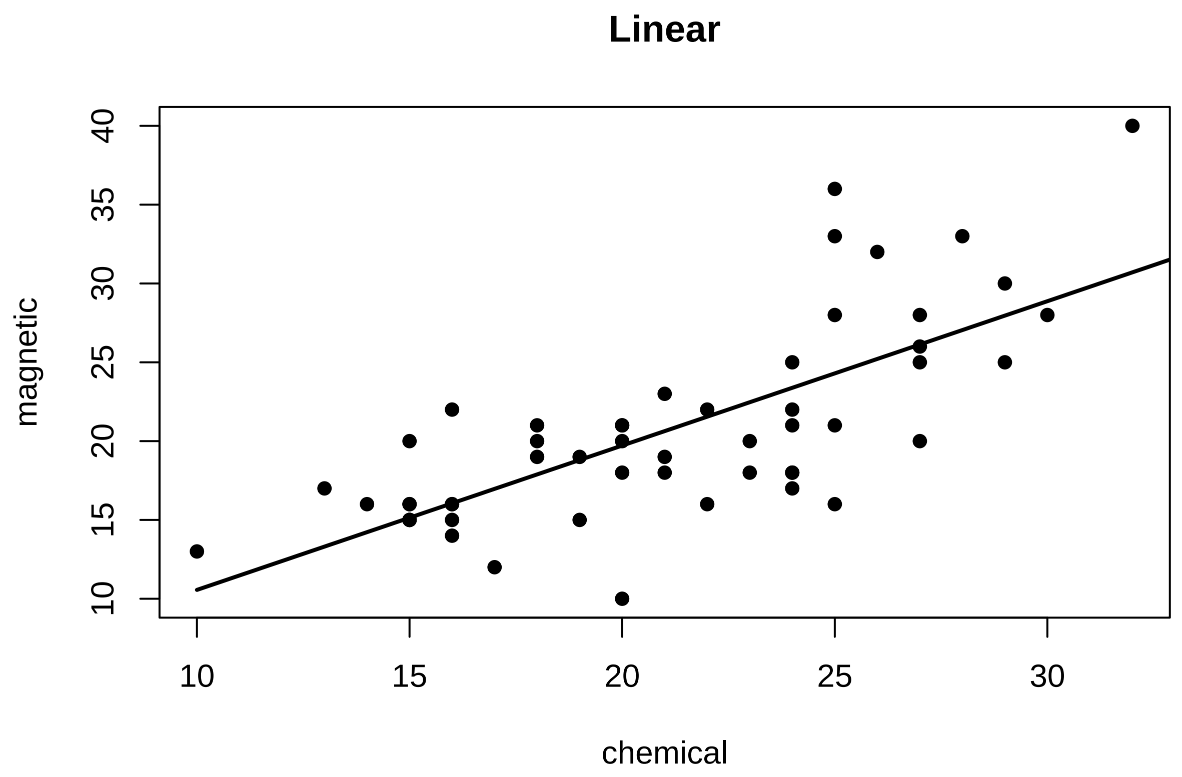
```
# Bootstrap
n <- 100
boot_ <- tbl_scor %>%
  modelr::bootstrap(n)

grp_boot <- boot_$strap %>% map(as_tibble)


# jackknife
r_boot <-  grp_boot %>% map_dbl(get_theta)

jknf <- r_boot %>% jackknife(mean)

#result
jknf$jack.se
```

```
## [1] 0.004482676
```

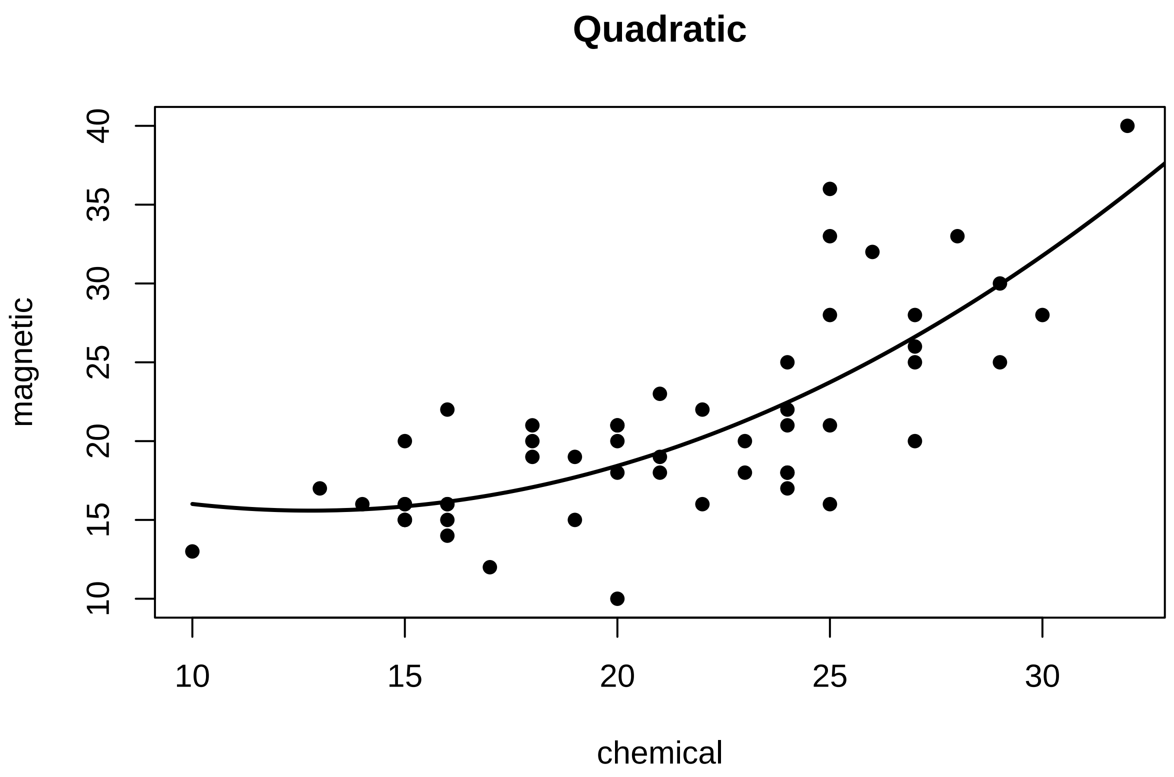
### 7.10

In Example 7.18, leave-one-out (n-fold) cross validation was used to select the best fitting model. Repeat the analysis replacing the Log-Log model with a cubic polynomial model. Which of the four models is selected by the cross validation procedure? Which model is selected according to maximum adjusted $R^2$?

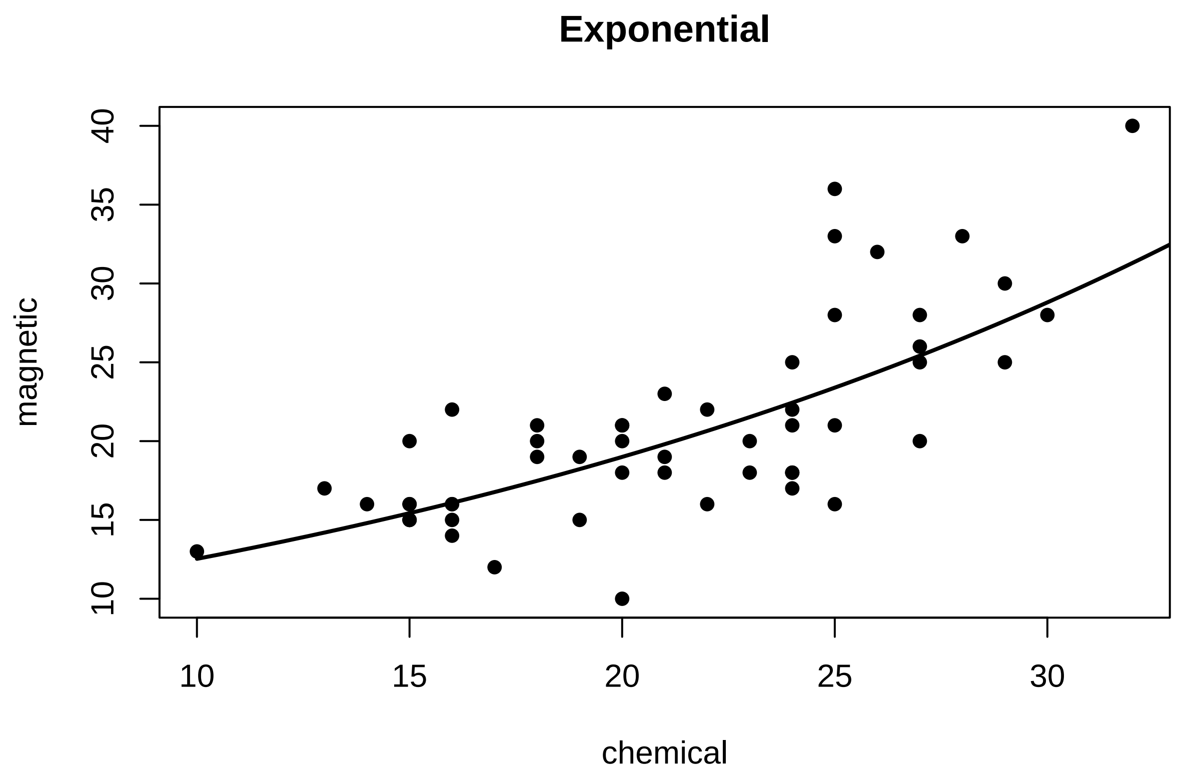
```
library(DAAG); attach(ironslag)
a <- seq(10, 40, .1) #sequence for plotting fits

L1 <- lm(magnetic ~ chemical)
plot(chemical, magnetic, main="Linear", pch=16)
yhat1 <- L1$coef[1] + L1$coef[2] * a
lines(a, yhat1, lwd=2)
```



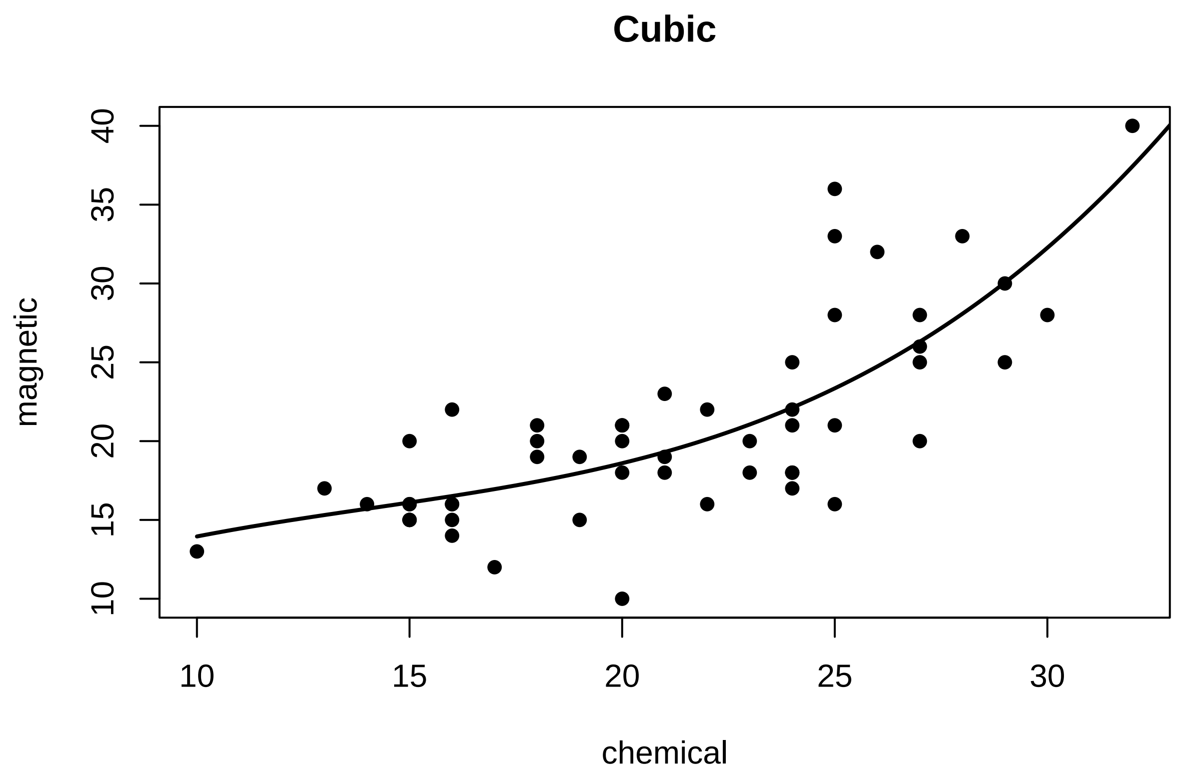
```
L2 <- lm(magnetic ~ poly(chemical, 2, raw=TRUE))
plot(chemical, magnetic, main="Quadratic", pch=16)
yhat2 <- L2$coef[1] + L2$coef[2] * a + L2$coef[3] * a^2
lines(a, yhat2, lwd=2)
```



```
L3 <- lm(log(magnetic) ~ chemical)
plot(chemical, magnetic, main="Exponential", pch=16)
logyhat3 <- L3$coef[1] + L3$coef[2] * a
yhat3 <- exp(logyhat3)
lines(a, yhat3, lwd=2)
```



```
L4 <- lm(magnetic ~ poly(chemical, 3, raw=TRUE))
plot(chemical, magnetic, main="Cubic", pch=16)
hat4 <- L4$coef[1] + L4$coef[2] * a + L4$coef[3] * a^2 + L4$coef[4] * a^3
lines(a, hat4, lwd=2)
```



```r
n <- length(magnetic) #in DAAG ironslag
e1 <- e2 <- e3 <- e4 <- numeric(n)
# for n-fold cross validation
# fit models on leave-one-out samples
for (k in 1:n) {
y <- magnetic[-k]
x <- chemical[-k]
J1 <- lm(y ~ x)
yhat1 <- J1$coef[1] + J1$coef[2] * chemical[k]
e1[k] <- magnetic[k] - yhat1

J2 <- lm(y ~ poly(x, 2, raw=TRUE))
yhat2 <- J2$coef[1] + J2$coef[2] * chemical[k] + J2$coef[3] * chemical[k]^2
e2[k] <- magnetic[k] - yhat2

J3 <- lm(log(y) ~ x)
logyhat3 <- J3$coef[1] + J3$coef[2] * chemical[k]
yhat3 <- exp(logyhat3)
e3[k] <- magnetic[k] - yhat3

J4 <- lm(y ~ poly(x, 3, raw=TRUE))
yhat4 <- J4$coef[1] + J4$coef[2] * chemical[k] + J4$coef[3] * chemical[k]^2 + J4$coef[4] * chemical[k]^
e4[k] <- magnetic[k] - yhat4
}

c(mean(e1^2), mean(e2^2), mean(e3^2), mean(e4^2))
```

```
## [1] 19.55644 17.85248 18.44188 18.17756
```

```
adj_rsq <- function(mdl) summary(mdl)$adj.r.squared
c(adj_rsq(L1), adj_rsq(L2), adj_rsq(L3), adj_rsq(L4))
```

```
## [1] 0.5281545 0.5768151 0.5280556 0.5740396
```

Choose the quadratic model according to the prediction error criterion.

Choose the quadratic model according to the adjusted r-squared.