myHomework4

chenpeng

2024-07-05

目录

1. **1**

2. **2**

3. **3**

a. 3

b. 3

c. 4

4. **5**

a. 5

b. 6

c. 7

***** **8**

1.

```
library(tidyverse)
ckm_nodes <- read_csv("../../data/ckm_nodes.csv")
```

```
## Rows: 246 Columns: 13
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr (10): city, medical_school, attend_meetings, free_time_with, discuss_med...
## dbl  (3): adoption_date, medical_journals, drs_among_three_best_friends
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
noinfor <- which(is.na(ckm_nodes$adoption_date))
ckm_nodes <- ckm_nodes[-noinfor, ]
ckm_network <- read.table("../../data/ckm_network.dat")
ckm_network <- ckm_network[-noinfor, -noinfor]
```

2.

```
df <- data.frame(doctor = rownames(ckm_nodes)) %>%
    slice(rep(1:n(), each = 17)) %>%
    mutate(month = rep(1:17, length.out = n()))
df <- df %>%
    mutate(prescribe_that_month = as.numeric(ckm_nodes[doctor,
        2] == month), prescribe_before = as.numeric(ckm_nodes[doctor,
        2] < month))



f <- function(x) {
    node_index <- as.numeric(x[1])
    condition <- ckm_network[node_index, ] == 1
    relevant_nodes <- ckm_nodes[which(condition), 2]
    return(sum(relevant_nodes < as.numeric(x[2])))
}

df <- df %>%
    mutate(contacts_str_before = apply(df, 1, f))

f <- function(x) {
    node_index <- as.numeric(x[1])
    condition <- ckm_network[node_index, ] == 1
    relevant_nodes <- ckm_nodes[which(condition), 2]
```

```
    return(sum(relevant_nodes <= as.numeric(x[2])))
}

df <- df %>%
    mutate(contacts_in_before = apply(df, 1, f))
```

There are 6 variables, so it has 6 columns.

from lab6: 125 doctors * 17 months = 2125, so the datafram has 2125 rows.

3.

a.

```
max(apply(ckm_network, 1, sum))
```

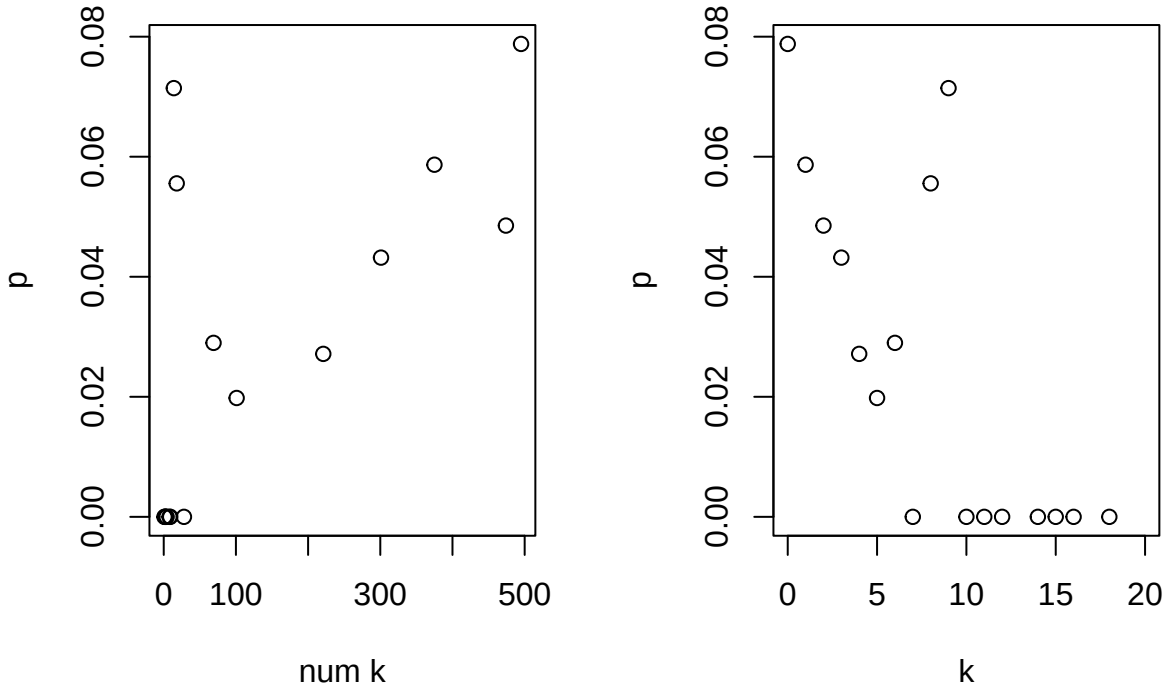
```
## [1] 20
```

Because the maximum of the contacts of a doctor is 20, so the possible values of k are from 0 to 20, the number of which is no more than 21.

b.

```
p.vec <- vector(mode = "numeric", length = 21)
k.vec <- p.vec
for (k in 0:20) {
    idx <- df$contacts_str_before == k
    k.vec[k + 1] <- sum(idx)
    if (k.vec[k + 1] == 0) {
        p.vec[k + 1] <- NA
        next
    }
    dfk <- df[idx, ]
    p1 <- sum(dfk$prescribe_that_month == 1)
    p.vec[k + 1] <- p1/k.vec[k + 1]
}
k <- c(0:20)
```

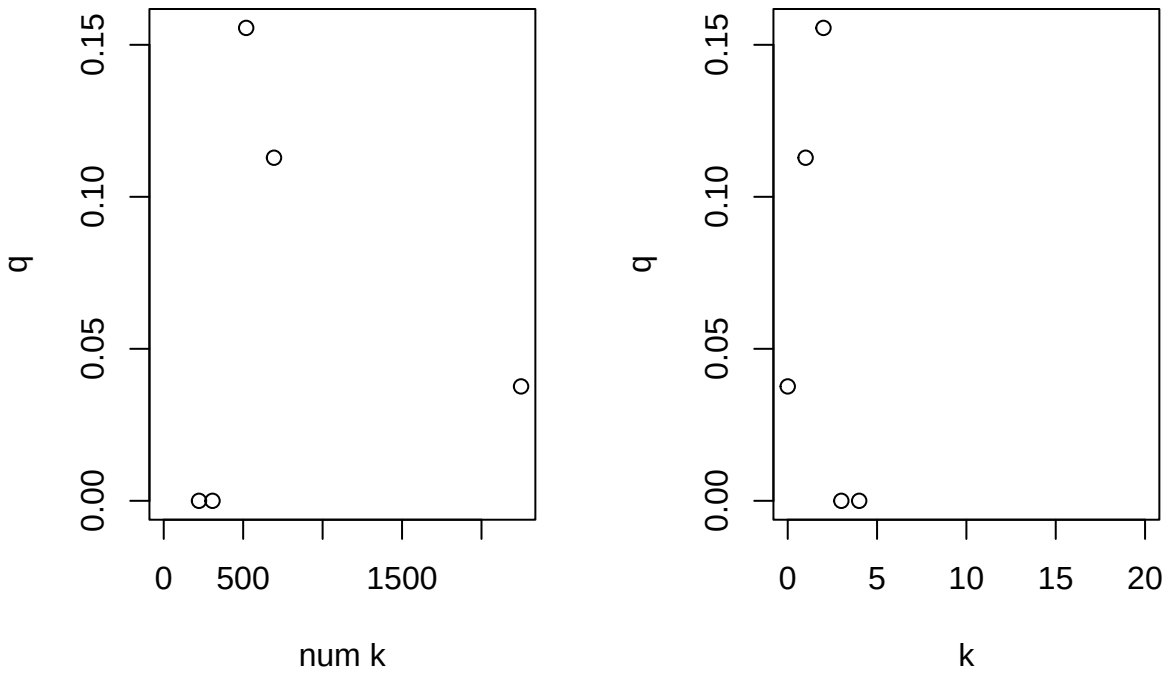
```r
par(mfrow = c(1, 2))
plot(k.vec, p.vec, xlab = "num k", ylab = "p")
plot(k, p.vec, xlab = "k", ylab = "p")
```



c.

```r
p.vec2 <- vector(mode = "numeric", length = 21)
k.vec2 <- p.vec2
for (k in 0:20) {
    idx <- (df$contacts_in_before - df$contacts_str_before) ==
        k
    k.vec2[k + 1] <- sum(idx)
    if (k.vec2[k + 1] == 0) {
        p.vec2[k + 1] <- NA
        next
    }
    dfk <- df[idx, ]
```

```
    p1 <- sum(dfk$prescribe_that_month == 1)
    p.vec2[k + 1] <- p1/k.vec2[k + 1]
}
k <- c(0:20)
par(mfrow = c(1, 2))
plot(k.vec2 + k.vec, p.vec2, xlab = "num k", ylab = "q")
plot(k, p.vec2, xlab = "k", ylab = "q")
```



4.

a.

```
df.p <- data.frame(k = 0:20, p = p.vec)
m.1 <- lm(p ~ k, data = df.p)
summary(m.1)
```

```
##
```

```
## Call:
## lm(formula = p ~ k, data = df.p)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.030334 -0.014584 -0.002344  0.005534  0.048694
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0569324  0.0090507   6.290 1.45e-05 ***
## k           -0.0037997  0.0009184  -4.137 0.000877 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02015 on 15 degrees of freedom
##   (因为不存在，4个观察量被删除了)
## Multiple R-squared:  0.533,  Adjusted R-squared:  0.5018
## F-statistic: 17.12 on 1 and 15 DF,  p-value: 0.0008773
```

```
          Estimate Std. Error t value Pr(>|t|)
```

a 0.0569324 0.0090507 6.290 1.45e-05 ***b -0.0037997 0.0009184 -4.137 0.000877***

b.

```
# logistic.nls
f <- function(k, a, b) {
    return(exp(a + b * k)/(1 + exp(a + b * k)))
}
m.2 <- nls(p ~ f(k, a, b), data = df.p, start = list(a = 0,
    b = -0.2))

# logistic.lm --convert to ak+b = f(p)
m.3 <- lm(p.log ~ k, df.p %>%
    mutate(p.log = ifelse(p == 0, log(1e-04/(1 - 1e-04)),
        log(p/(1 - p)))))

m.4 <- glm(p ~ k, data = df.p, family = binomial)
```

```r
summary(m.2)
```

```
##
## Formula: p ~ f(k, a, b)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a -2.56508    0.20610 -12.446 2.62e-09 ***
## b -0.17051    0.05371  -3.174  0.00628 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01957 on 15 degrees of freedom
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 1.127e-07
##   (因为不存在，4个观察量被删除了)
```

c.

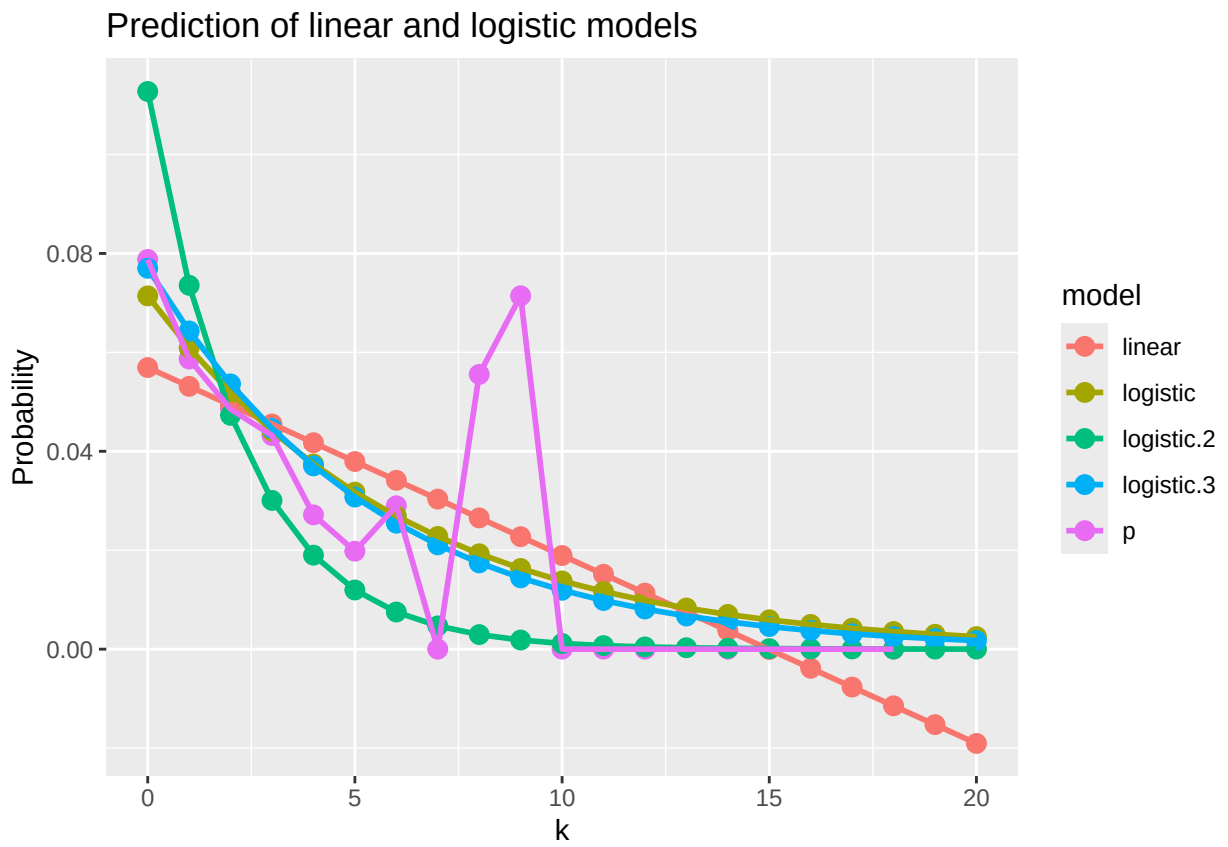
```r
m1 = predict(m.1, newdata = data.frame(k = c(0:20)))
m2 = predict(m.2, newdata = data.frame(k = c(0:20)))
df.p <- mutate(df.p, linear = m1, logistic = m2)
t = predict(m.3, newdata = data.frame(k = c(0:20)))
m3 = exp(t)/(1 + exp(t))

r = predict(m.4, newdata = data.frame(k = c(0:20)))
m4 = exp(r)/(1 + exp(r))

df.p <- mutate(df.p, linear = m1, logistic = m2, logistic.2 = m3,
    logistic.3 = m4)


df.tidy <- df.p %>%
    gather(model, res, -k) %>%
    na.omit()
df.tidy %>%
    ggplot() + geom_point(aes(x = k, y = res, color = model),
```

```
    size = 3) + geom_line(aes(x = k, y = res, color = model),
    size = 1) + labs(y = "Probability", title = "Prediction of linear and logistic models")
```



The linear model is the worst. I use 3 methods to establish the logistic model and I think logistic3 using function glm() looks better on the whole. Because logistic.2 seems a little overfitting in the tail part and logistic.3 is more fitting than logistic.

*

```
idx <- !(is.na(p.vec) | p.vec == 0)
wt <- p.vec * (1 - p.vec)/k.vec
# We can only take data!=0 because we just can't get
# p=0 more precisely and thus the variance of that is
# meaningless.
m.w1 <- lm(p ~ k, data = df.p[idx, ], weight = 1/wt[idx])
summary(m.w1)
```

##

```
## Call:
## lm(formula = p ~ k, data = df.p[idx, ], weights = 1/wt[idx])
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67043 -0.30912  0.01392  0.83562  1.15775
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.068669   0.006248  10.991 1.14e-05 ***
## k           -0.008548   0.001922  -4.448  0.00298 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.772 on 7 degrees of freedom
## Multiple R-squared:  0.7387, Adjusted R-squared:  0.7013
## F-statistic: 19.79 on 1 and 7 DF,  p-value: 0.002978
```

```
m.w2 <- nls(p ~ f(k, a, b), data = df.p[idx, ], start = list(a = 0,
    b = -0.2), weight = 1/wt[idx])
summary(m.w2)
```

```
##
## Formula: p ~ f(k, a, b)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a -2.48943    0.07804 -31.901 7.69e-09 ***
## b -0.23754    0.03743  -6.346 0.000387 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5513 on 7 degrees of freedom
##
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 7.434e-07
```

```
m.w4 <- glm(p ~ k, data = df.p[idx, ], family = binomial,
    weight = 1/wt[idx])
summary(m.w4)
```

```
##
## Call:
## glm(formula = p ~ k, family = binomial, data = df.p[idx, ], weights = 1/wt[idx])
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.53520    0.03482  -72.82   <2e-16 ***
## k           -0.21195    0.01300  -16.30   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 368.02  on 8  degrees of freedom
## Residual deviance:  84.11  on 7  degrees of freedom
## AIC: 148.09
##
## Number of Fisher Scoring iterations: 4
```

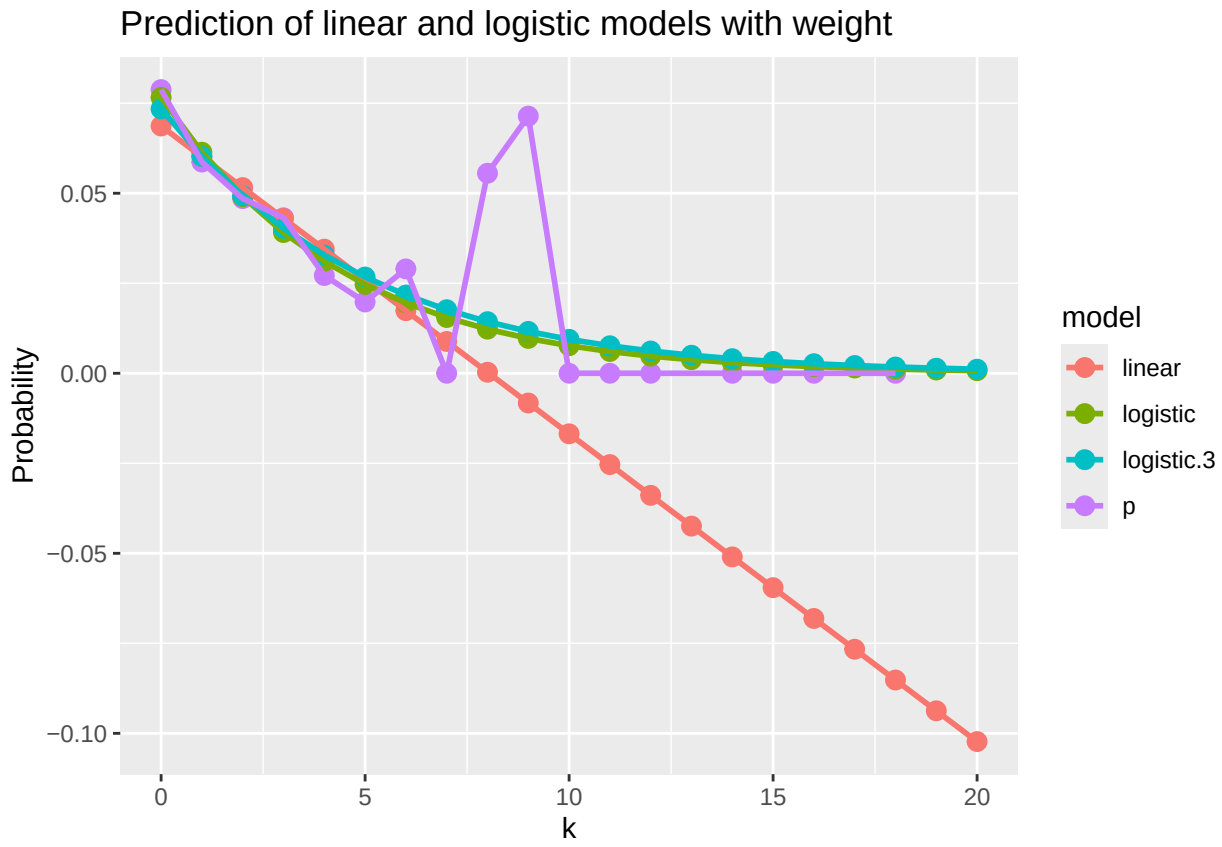
The change of parameters:

Parameters	Linear	Linear.weight	Logistic	Logistic.weight	Logistic.3	Logistic.3.weight
a	0.0569324	0.068669	-2.56508	-2.48943	-2.4838	-2.53520
b	-0.0037997	-0.008548	-0.17051	-0.23754	-0.1934	-0.21195

```
w1 = predict(m.w1, newdata = data.frame(k = c(0:20)))
w2 = predict(m.w2, newdata = data.frame(k = c(0:20)))
wr = predict(m.w4, newdata = data.frame(k = c(0:20)))
w4 = exp(wr)/(1 + exp(wr))


df.w <- data.frame(k = 0:20, p = p.vec)
df.w <- mutate(df.w, linear = w1, logistic = w2, logistic.3 = w4)
dft <- df.w %>%
    gather(model, res, -k) %>%
    na.omit()
dft %>%
    ggplot() + geom_point(aes(x = k, y = res, color = model),
    size = 3) + geom_line(aes(x = k, y = res, color = model),
```

```
    size = 1) + labs(y = "Probability", title = "Prediction of linear and logistic models with wei
```



The change of plots: Both curves are more fitting when k is small and seem to somehow neglect the dots in the middle-top. From my analysis, the reason is that smaller samples cause larger estimated variances, which reduce the weight of these points. And there are more observations when k is small, which adds to the weight.