# HW2 Solution

Dante Wu

2023-01-27

## Problem 1

```
nutri=read.table("nutrient.txt")
nutri=nutri[,-1] #Removing index column

names(nutri)=c("calcium","iron","protein","vitaminA","vitaminC" )
```

### 1.1

```
apply(nutri,2,mean)
```

```
##   calcium      iron   protein  vitaminA  vitaminC
## 624.04925  11.12990  65.80344 839.63535  78.92845
```

```
apply(nutri,2,sd)
```

```
##    calcium       iron    protein   vitaminA   vitaminC
##  397.27754    5.98419   30.57576 1633.53983   73.59527
```

### 1.2

```
n=737 #sample size

(tcrit=abs(qt(.025,736))) #Critical value at 5% level
```

```
## [1] 1.963192
```

```
attach(nutri)
(mean(calcium)-1000)/(sd(calcium)/sqrt(n)) #Signifincantly different
```

```
## [1] -25.69039
```

```
(mean(iron)-15)/(sd(iron)/sqrt(n))#Signifincantly different
```

```
## [1] -17.55701
```

```
(mean(protein)-60)/(sd(protein)/sqrt(n))#Signifincantly different
```

```
## [1] 5.152786
```

```
(mean(vitaminA)-800)/(sd(vitaminA)/sqrt(n))#NOT
```

```
## [1] 0.6586985
```

```
(mean(vitaminC)-75)/(sd(vitaminC)/sqrt(n))#NOT
```

```
## [1] 1.449121
```

We reject the null hypothesis for the first 3 variables since its t-statistic is greater than the 5% critical value.

### 1.3

US Women have a significant deficiency in Calcium and Iron, and are signifi- cantly above the recommended amount of protein. Suggest Calcium and Iron supplement.

## Problem 2

```
multi=read.table("multiple.txt")
```

### 2.1

```
mean.vector = apply(multi,2,mean)
sd.vector = apply(multi,2,sd)
t.vector = mean.vector/(sd.vector/sqrt(100))
round(t.vector,4)
```

```
##      V1      V2      V3      V4      V5      V6      V7      V8      V9     V10
## 18.4753 17.8513 18.2911 20.2730 19.1680 18.0646 21.7395 18.7614 18.5835 23.1711
##     V11     V12     V13     V14     V15     V16     V17     V18     V19     V20
##  0.1894  0.9613  0.1981 -1.1941 -0.2093  0.9568  0.7592  0.1274  0.7001 -1.7513
##     V21     V22     V23     V24     V25     V26     V27     V28     V29     V30
##  0.3280 -0.7848 -0.5965  0.7130  0.3098  1.5053  0.0091  1.0540 -0.4160  1.3200
##     V31     V32     V33     V34     V35     V36     V37     V38     V39     V40
## -0.7225 -1.6960 -0.7193 -0.1020  1.2511 -0.4524  0.9571  1.2313  0.1506 -1.8111
##     V41     V42     V43     V44     V45     V46     V47     V48     V49     V50
##  0.3633  0.4589 -0.3980 -2.2800  1.2367 -0.2961  0.9411  1.5002  0.1403 -0.3439
```

```
rej = abs(t.vector) > abs(qt(.05,df=99)) #Checking if t-statistic is larger than the critical value
print(rej)
```

```
##    V1    V2    V3    V4    V5    V6    V7    V8    V9   V10   V11   V12   V13
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
##   V14   V15   V16   V17   V18   V19   V20   V21   V22   V23   V24   V25   V26
## FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##   V27   V28   V29   V30   V31   V32   V33   V34   V35   V36   V37   V38   V39
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   V40   V41   V42   V43   V44   V45   V46   V47   V48   V49   V50
##  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

The variables which T-statistic is larger than critical value had the null hypothesis rejected.

### 2.2

```
alpha = 0.1
FWER = 1 - (1 - alpha)^50
sprintf("FWER: %f", FWER)
```

```
## [1] "FWER: 0.994846"
```

```r
num_type1 = sum(rej[11:50])
num_type2 = sum(!rej[1:10])
fdp = num_type1 /sum(rej)
sprintf("number of type 1 errors: %i", num_type1)
```

```
## [1] "number of type 1 errors: 4"
```

```r
sprintf("number of type 2 errors: %i", num_type2)
```

```
## [1] "number of type 2 errors: 0"
```

```r
sprintf("fdp: %f", fdp)
```

```
## [1] "fdp: 0.285714"
```

No Type II errors since we reject that the mean equals 0 for the fist 10 variables, which have mean = 2. We have 4 Type I errors : `(V20,V32,V40,V44)`. So FDP = 4/14.

## 2.3

```r
pv <- numeric(50)
for(j in 1:50){
  t_test <- t.test(multi[,j])
  pv[j] <- t_test$p.value
}
print(pv) # Unadjusted p-values
```

```
##  [1] 7.477468e-34 1.030644e-32 1.613953e-33 5.107798e-37 4.299681e-35
##  [6] 4.181053e-33 1.782424e-39 2.281744e-34 4.767758e-34 9.020768e-42
## [11] 8.501800e-01 3.387655e-01 8.433591e-01 2.353081e-01 8.346443e-01
## [16] 3.410084e-01 4.495285e-01 8.988573e-01 4.855360e-01 8.298508e-02
## [21] 7.435778e-01 4.344605e-01 5.521855e-01 4.775194e-01 7.573900e-01
## [26] 1.354197e-01 9.927262e-01 2.944327e-01 6.783256e-01 1.898939e-01
## [31] 4.716828e-01 9.302681e-02 4.736660e-01 9.189993e-01 2.138515e-01
## [36] 6.519921e-01 3.408716e-01 2.211123e-01 8.806238e-01 7.315268e-02
## [41] 7.171567e-01 6.472996e-01 6.915055e-01 2.475067e-02 2.191321e-01
## [46] 7.678119e-01 3.489369e-01 1.367531e-01 8.886784e-01 7.316609e-01
```

```r
rej.bon = pv < alpha/50
print(rej.bon)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE
```

Or they can use `p.adjust`:

```r
p.adjust(pv,method=c("bonferroni"))
```

```
##  [1] 3.738734e-32 5.153220e-31 8.069765e-32 2.553899e-35 2.149840e-33
##  [6] 2.090526e-31 8.912120e-38 1.140872e-32 2.383879e-32 4.510384e-40
## [11] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
## [16] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
## [21] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
## [26] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
## [31] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
## [36] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
## [41] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
## [46] 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
```

According to the adjusted p-values, the null hypothesis is rejected only for the first 10 variables.

```
alpha.bon = alpha/50
FWER.bon = 1 - (1 - alpha.bon)^50
sprintf("FWER.bon: %f", FWER.bon)
```

```
## [1] "FWER.bon: 0.095253"
```

## 2.4

```
bh_rej <- function(pv, alpha)
{
  m <- length(pv)
  l <- alpha*c(1:m)/m
  sort_p <- sort(pv)
  set <- which(l>=sort_p)

  if(length(set)==0){
    rej <- set
    pvalue <- set
  } else{
    threshold <- sort_p[max(set)]
    rej <- which(pv <= threshold)
    pvalue <- pv[rej]
  }
  outlist<-list(pvalue=pvalue, rej=rej)
  return(outlist)
}

bh_test <- bh_rej(pv, 0.1)
print(bh_test$rej)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Therefore, we reject only the first 10 hypotheses.

They can also use `p.adjust`:

```
p.adjust(pv, method = c("BH")) < 0.1
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE
```

For FWER, we use $p_{(k^*)}$ as the threshold, where $k^*$ is the largest $k$ such that $p_{(k)} \leq \alpha \cdot k/m$. Then the FWER should be $1 - \{1 - p_{(k^*)}\}^m$.

```
p_k_star = max(bh_test$pvalue)
print(p_k_star)
```

```
## [1] 1.030644e-32
```

```
FWER.bh = 1 - (1 - p_k_star)^50
print(FWER.bh)
```

```
## [1] 0
```

```
# Examine conclusions
R <- length(bh_test$rej)                # total number of rejections
V <- length(which(bh_test$rej>10))      # number of false rejections
fdp.bh <- V/max(1,R)
sprintf("fdp.bh: %f", fdp.bh)
```

```
## [1] "fdp.bh: 0.000000"
```

Just like step 3, we removed all false rejections that occurred in step 1.