Assignment 4 - QBS 177

Code

Spencer Bertsch
Jan. 2022

Some helpful commands:
* Insert a new code chunk: *Cmd+Option+I*
* Run a code cell: *Cmd+Shift+Enter*
* Preview the notebook HTML file: *Cmd+Shift+K*

Imports

Hide

```
library(glue)
library(qqman)
```

First we need to change our current working directory to the correct directory for Assignment 3

Hide

```
setwd('/Users/spencerbertsch/Desktop/dev/phd_courses/MATH177/assignment4')
getwd()
```

```
[1] "/Users/spencerbertsch/Desktop/dev/phd_courses/MATH177/assignment4"
```

Import the raw data

Hide

```
load('lab1.Rdata')
```

Here we can transform the subpopulation data into 21 countries instead of 25. We need this to be the case later on

Hide

```
chn <- (fulldat[,5] + fulldat[,6])/2
ind <- (fulldat[,11] + fulldat[,14])/2
nga <- (fulldat[,25] + fulldat[,8])/2
usa <- (0.777*fulldat[,4] + 0.132*fulldat[,2] + 0.053*(chn + ind + fulldat[,16] + fullda
t[,15])/4)/(0.777+0.132+0.053)
fulldat <-fulldat[,c(3,1,5,7,9,12,11,24,15,17,19,25,21,20,22,18,13,23,10,4,16)]
fulldat[,3] <- chn
fulldat[,7] <- ind
fulldat[,12] <- nga
fulldat[,20] <- usa
```

The fulldat data frame now has only 21 columns.

We can now read in our response variable: the smoking data set

Hide

```
yy <- read.delim("smoking_outcome.txt")
```

Hide

```
y <- yy[,2]

# here we want to calculate the correlation between yy and the first 10000 loci
# we also want to know the p-values for each of the linear regressions run
plong <- corlong <- NULL

proc.time()
```

```
   user  system elapsed 
  1.469   0.335   2.716 
```

Hide

```
for (i in 1:10000){

  # remove loci with all minor allele frequncy 0
  if(var(fulldat[i,])){

  # run a linear regression using the smoking data as the response and the i'th row of f
ulldat as the predictor
  fit <- lm(y~fulldat[i,])

  # find the correlation between the response variable and the i'th row of fulldat
  # (This is why they needed to be of the same dimension! 21 elements long)
  corlong[i] <- cor(fulldat[i,], y)

  #output slope from linear regression
  plong[i] <- summary(fit)$coef[2,4]
  }
}
proc.time()
```

```
   user  system elapsed 
  4.022   0.366   5.300 
```

This process took 2.54 seconds to finish, so if we assume that processing 10,000 records takes 2.54 seconds, then processing 1,099,146 samples would take 279.18 seconds, or about **4.65 minutes**.

We can now try to rewrite the above code without using a for loop so that we can improve the runtime of the algorithm.

Hide

```
# n is the sample size
n <- dim(yy)[1]

# initialize a variable to store p values.
pvalue <- matrix(0, length(chr), 1)

# initialize a variable to store correlation.
corvalue <- matrix(0, length(chr), 1)

colnames(pvalue) <- c("prevave")
colnames(corvalue) <- c("prevave")
```

And we can now run the vector operation to obtain the same information, but without using a for-loop. Let's see how much more quickly this process runs compared with the above process.

Hide

```
proc.time()
```

```
   user  system elapsed 
  4.031   0.373   5.315 
```

Hide

```
temp <- scale(t(fulldat))
y1 <- y/sqrt(var(y))

asd <- y1%*%temp/(n-1) # obtain correlation using apply
#asd <- suppressWarnings(apply( t(fulldat) , 2 , cor , y)) # alternative way

# mid step to calculate p value
asd1 <- 1-asd^2

# this is the T statistics
asd2 <- sqrt(n-2)*asd/sqrt(asd1)

# transform T statistics to p-value
pvalue[,1] <- 2*(1-pt(abs(asd2),(n-2)))

# pass correlation to corvalue.
corvalue[,1] <- asd
proc.time()
```

```
   user  system elapsed 
  7.250   0.651   8.812 
```

Only 3.351 seconds instead of an estimated 4.65 minutes! Pretty great. We want to double check that we got the same correlations and p-values using the same two methods:

Hide

```
abs(sum(plong-pvalue[1:10000], na.rm=T))
```

```
[1] 5.214536e-13
```

Hide

```
abs(sum(corlong-corvalue[1:10000], na.rm=T))
```

```
[1] 2.177679e-13
```

And indeed we see that these two values are essentially zero, meaning that we were able to obtain the same correlations and p-values.

We now want to generate a manhattan plot so we can visualize our data. In order to do this, we need to have a dataframe that has 4 columns with the following names: 1. CHR 2. BP 3. SNP 4. P These columns allows the plotting function to create the manhattan plot. In order to make our Manhattan plot below, we need to create a data frame with the charactersitics mentioned above:

Hide

```
# create dataframe with thefollwong chracteristics
index <- c(1:dim(pvalue)[1])
df1 <- data.frame(index, chr, pvalue, location)
```

We can use the below code to fill the NA values in the p-value vector in order to fix the bug in the plot

Hide

```
# pvalue[is.na(pvalue[,1]), 1] <- mean(pvalue[,1], na.rm = TRUE)
# sum(is.na(pvalue))
# mu_new = colMeans(pvalue)
# print(mu_new)
# df1 <- data.frame(chr, pvalue, location)
```

Another option here instead of filling the p-values with the mean p-value is to simply drop the rows that have NAs in the position of p-values. It's probably not a best practice to fill NAs for the results of statistical tests, so better to throw out the records that are missing the results so we can ignore them.

Hide

```
print('We can find out how many NAs there are in our p-values:')
```

```
[1] "We can find out how many NAs there are in our p-values:"
```

Hide

```
dim(df1)[1] - dim(df1[!is.na(df1$prevave),])[1]
```

```
[1] 24410
```

Hide

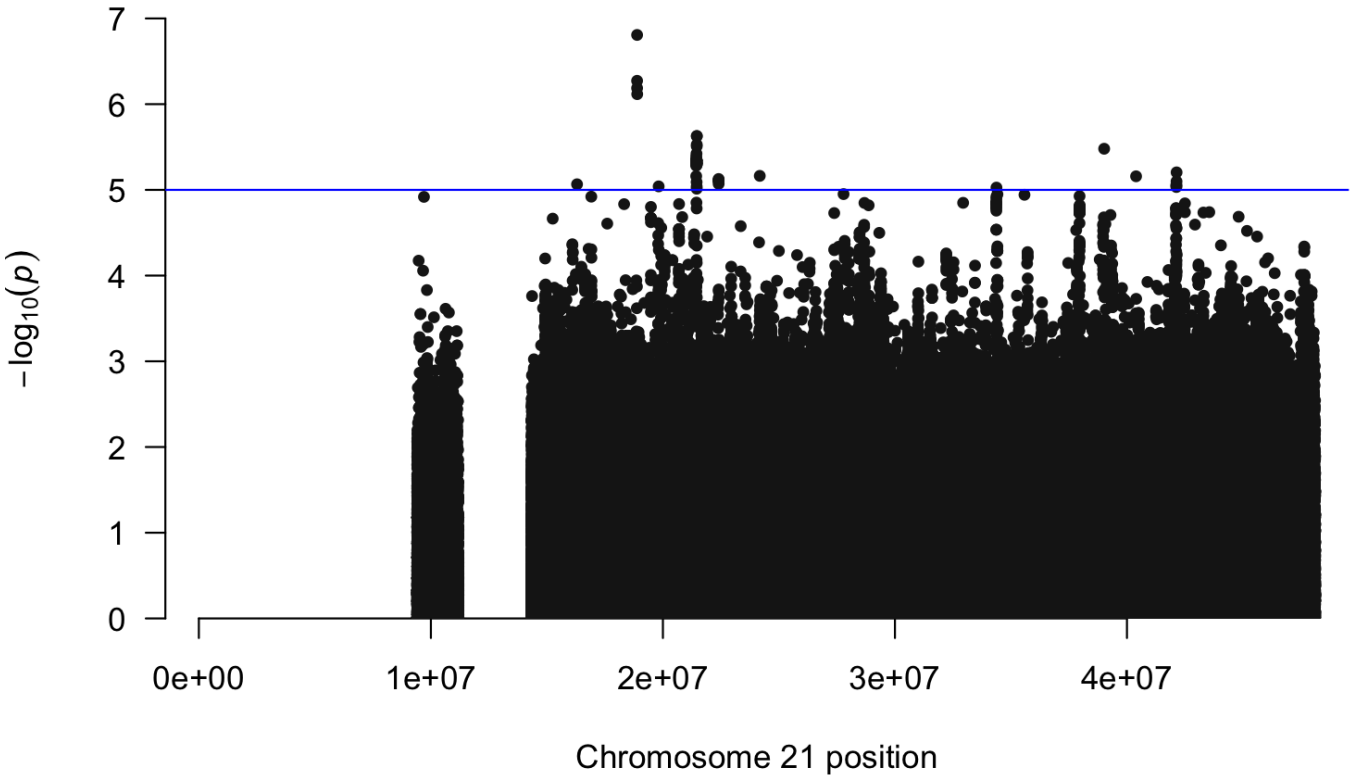
```
df2 <- df1[!is.na(df1$prevave),]
```

Question 1

Create a manhattan chart showing the resulting p-values and the ‘chr’ vector, and the ‘locaiton’ vector presented in the lab1.RData file:

Hide

```
# Make the Manhattan plot on the gwasResults dataset
manhattan(df2, chr="chr", bp="location", p="prevave", snp="location", col = c("gray10", "gray60"))
```



Question 2

Identify top 9 locations on chromosome 21 based on the association p-values.

In order to answer this question we can take our newly created dataframe **df1** and sort it by p-value. Then we can take the top 9 rows and store the locations of those rows and use them to perform the online lookups to find the ‘rs’ number for the top 9 locations.

By “top locations’ here we mean the smallest p-values. If the model does a good job of fitting the data, then the p-value for the fit will be quite small. If there is no reasonable fit, then the p-value will be larger (closer to 1). We can then add an index column to our data and sort it by p-value in order to get the locations that correspond to the smallest 9 p-values.

Hide

```
top_p_value_df <- df1[order(df1$prevave, decreasing=FALSE), ]
top_p_value_df
```

	index <int>	chr <int>	prevave <dbl>	location <int>
171220	171220	21	1.557500e-07	18887601
171210	171210	21	5.341077e-07	18887201
171218	171218	21	6.498068e-07	18887483
171381	171381	21	7.643547e-07	18892725
251471	251471	21	2.356621e-06	21460976
251473	251473	21	2.356621e-06	21460979
251399	251399	21	2.963624e-06	21458577
251385	251385	21	3.099801e-06	21458112
795183	795183	21	3.312984e-06	39016910
251062	251062	21	3.785508e-06	21447536

1-10 of 1,099,164 rows Previous **1** 2 3 4 5 6 ... 100 Next

This is the result of looking up each of the top 9 p-values on https://www.ncbi.nlm.nih.gov (https://www.ncbi.nlm.nih.gov)

Chromosome	p-value	location	rs-number
21	1.557500e-07	18887601	rs7279313
21	5.341077e-07	18887201	rs8129365
21	6.498068e-07	18887483	rs8129817
21	7.643547e-07	18892725	rs59660381
21	2.356621e-06	21460976	rs78892776
21	2.356621e-06	21460979	rs79009469
21	2.963624e-06	21458577	rs78552707
21	3.099801e-06	21458112	rs77095203
21	3.312984e-06	39016910	rs1787423

These rs-numbers are the results of lookups for the locaitons that correspond with the smallest 9 p-values found above.

Question 3

Draw scatter plot between smoking prevalence (y) and minor allele frequencies of each loci's in 2. Add a regression line to the scatter plot. Add the correlation value to the plots as a legend. Use par(mfrow=c(3,3)) to plot all 9 figures in one plot.

We can examine our response variable here:

Hide

```
# we have the response variable y:
yy
```

Country <chr>	**smoking.prevalence.in..** <dbl>
Bangladesh	20.55
Barbados	6.42
China	22.99
Columbia	10.77
Finland	17.58
Gambia	10.34
India	11.74
Italy	24.19
Japan	22.15
Kenya	8.88

1-10 of 21 rows Previous **1** 2 3 Next

Now we need to get the minor allele frequencies for each of the top 9 records seen in question 2. We can do this by adding an index column into the dataframe and using that to index the correct rows out of the initial fulldat matrix.

Hide

```
top_indices <- top_p_value_df[1:9, ]$index
print('These are the rows in the fulldat dataframe we want to plot against the response
 variable:')
```

```
[1] "These are the rows in the fulldat dataframe we want to plot against the response va
riable:"
```

Hide

```
top_indices
```

```
[1] 171220 171210 171218 171381 251471 251473 251399 251385 795183
```

And finally we can get the correct rows out of the fulldat matrix that correspond to all of the p-values obtained from the processing above:

Hide

```
# we can prepare our data here:
response <- yy[, 2]

maf_loci1 <- fulldat[top_indices[1],]
maf_loci2 <- fulldat[top_indices[2],]
maf_loci3 <- fulldat[top_indices[3],]
maf_loci4 <- fulldat[top_indices[4],]
maf_loci5 <- fulldat[top_indices[5],]
maf_loci6 <- fulldat[top_indices[6],]
maf_loci7 <- fulldat[top_indices[7],]
maf_loci8 <- fulldat[top_indices[8],]
maf_loci9 <- fulldat[top_indices[9],]
```

Now that we have our predictors and response variables defined, we can create the matrix of scatter plots:

Hide

```
par(mfrow = c(3, 3))

# 1st plot
plot(maf_loci1, response, pch=16, col='red', cex=1.2, main='First plot')
abline(lm(response ~ maf_loci1), col='blue')
```

Hide

```
c1 <- round(cor(maf_loci1, response, method = "pearson"), 3)
legend(0.12, 25, legend=c(c1),
       fill = c("blue","red")
)

# 2nd plot
plot(maf_loci2, response, pch=16, col='red', cex=1.2, main='Second plot')
```

Hide

```
abline(lm(response ~ maf_loci2), col='blue')
c1 <- round(cor(maf_loci2, response, method = "pearson"), 3)
legend(0.125, 25, legend=c(c1),
       fill = c("blue","red")
)
```

Hide

```


# 3rd plot
plot(maf_loci3, response, pch=16, col='red', cex=1.2, main='Third plot')
abline(lm(response ~ maf_loci3), col='blue')
```

Hide

```
c1 <- round(cor(maf_loci3, response, method = "pearson"), 3)
legend(0.125, 25, legend=c(c1),
       fill = c("blue","red")
)

# 4th plot
plot(maf_loci4, response, pch=16, col='red', cex=1.2, main='Fourth plot')
```

Hide

```
abline(lm(response ~ maf_loci4), col='blue')
c1 <- round(cor(maf_loci4, response, method = "pearson"), 3)
legend(0.125, 25, legend=c(c1),
       fill = c("blue","red")
)
```

Hide

```


# 5th plot
plot(maf_loci5, response, pch=16, col='red', cex=1.2, main='Fifth plot')
abline(lm(response ~ maf_loci5), col='blue')
```

Hide

```
c1 <- round(cor(maf_loci5, response, method = "pearson"), 3)
legend(0.09, 25, legend=c(c1),
       fill = c("blue","red")
)

# 6th plot
plot(maf_loci6, response, pch=16, col='red', cex=1.2, main='Sixth plot')
```

Hide

```
abline(lm(response ~ maf_loci6), col='blue')
c1 <- round(cor(maf_loci6, response, method = "pearson"), 3)
legend(0.09, 25, legend=c(c1),
       fill = c("blue","red")
)
```

Hide

```


# 7th plot
plot(maf_loci7, response, pch=16, col='red', cex=1.2, main='Seventh plot')
abline(lm(response ~ maf_loci7), col='blue')
```

Hide

```
c1 <- round(cor(maf_loci7, response, method = "pearson"), 3)
legend(0.09, 25, legend=c(c1),
       fill = c("blue","red")
)

# 8th plot
plot(maf_loci8, response, pch=16, col='red', cex=1.2, main='Eighth plot')
```

Hide

```
abline(lm(response ~ maf_loci8), col='blue')
c1 <- round(cor(maf_loci8, response, method = "pearson"), 3)
legend(0.09, 25, legend=c(c1),
       fill = c("blue","red")
)
```

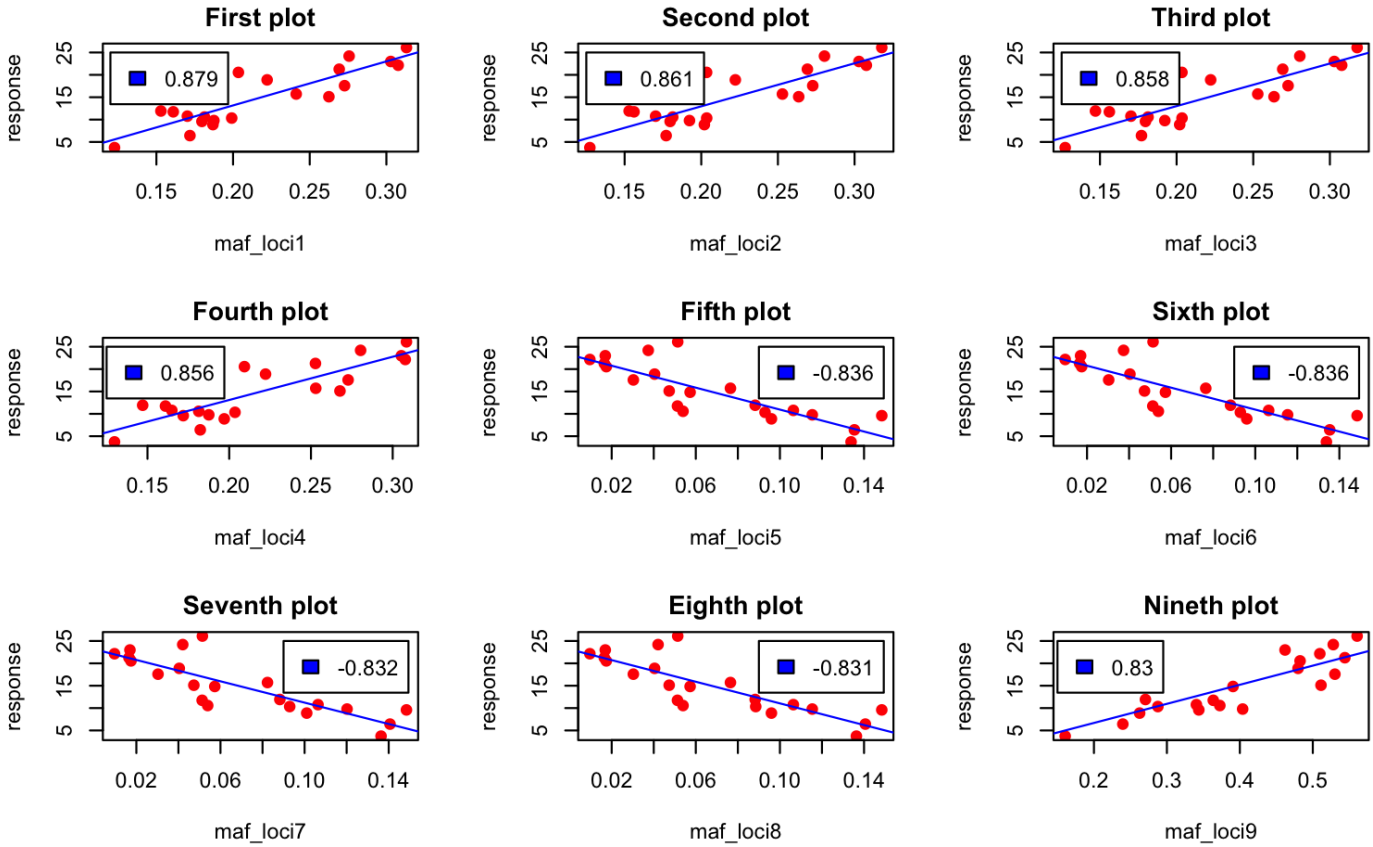
Hide

```


# 9th plot
plot(maf_loci9, response, pch=16, col='red', cex=1.2, main='Nineth plot')
abline(lm(response ~ maf_loci9), col='blue')
```

Hide

```
c1 <- round(cor(maf_loci9, response, method = "pearson"), 3)
legend(0.15, 25, legend=c(c1),
       fill = c("blue","red")
)
```


First plot
0.879
response
maf_loci1
Second plot
0.861
response
maf_loci2
Third plot
0.858
response
maf_loci3
Fourth plot
0.856
response
maf_loci4
Fifth plot
-0.836
response
maf_loci5
Sixth plot
-0.836
response
maf_loci6
Seventh plot
-0.832
response
maf_loci7
Eighth plot
-0.831
response
maf_loci8
Nineth plot
0.83
response
maf_loci9