Solution for Data Assignment on Re-sampling (Francesca Chiaromonte)

> chicken_toy <- read.table("chicken_toy.txt", header=TRUE)  #load data
> library(boot)  #load boot package

Part 1
Hypothesis system to be tested:
Ho: median log-length-ratio(mic+med) = median log-length-ratio(macro) i.e. diff=0
Ha: median log-length-ratio(mic+med) > median log-length-ratio(macro) i.e. diff>0

> # add a column to the data with a recoding of “ch” in “class” (1=mic+med; 2=macro)
> class_clmn <- 1*(chicken_toy[,"ch"]!="macro") + 2*(chicken_toy[,"ch"]=="macro")
> chicken_toy_ext <- data.frame(chicken_toy, "class"=class_clmn)

> # FIRST OPTION: bootstrap ALL the data
> # Define a function that computes the difference btw medians of y for class 1 and 2.
> # This will be reusable for any data set with a quantitative column called “y”
> # and a column called “class” expressing a binary classification (1,2).
> med_diff_fun <- function(data, indices){
>   m1 <- median(data[indices,"y"]*[data[indices,"class"]==1])
>   m2 <- median(data[indices,"y"]*[data[indices,"class"]==2])
>   diff = m1-m2
> }
> # perform the bootstrap
> boot_med_diff <- boot(chicken_toy_ext,med_diff_fun, R=999,sim="ordinary")
> boot_med_diff
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = chicken_toy_ext, statistic = med_diff_fun, R = 999, sim = "ordinary")
Bootstrap Statistics :
          original bias     std. error
t1* 0.104925 0.0003004104  0.0454823
> # produce histogram as a plot of the simulated sampling distribution
> plot(boot_med_diff)
> # construct an approximate one-sided (right) p-value counting what fraction of the
> # differences in medians on bootstrap samples is ≤ 0 (left tail) -- “parallel construction”
> sum(boot_med_diff$t <=0)/999 #bootstrap-based p-value, first option
> [1] 0.01001001
> # very small, strong evidence against Ho.

> # SECOND OPTION: bootstrap each sub-sample separately.
> # Define a function that computes the median of a set of numbers
> med_fun <- function(data,indices){
> median(data[indices])
> }
> # perform the bootstrap within class 1 (mic+med)
> boot_med_1 <- boot(chicken_toy_ext[,"y"][chicken_toy_ext[,"class"]==1],med_fun,
> R=999,sim="ordinary")
> # perform the bootstrap within class 2 (macro)
> boot_med_2 <- boot(chicken_toy_ext[,"y"][chicken_toy_ext[,"class"]==2],med_fun,
> R=999,sim="ordinary")
> # form a vector of R=999 differences
> diff_boot_med <- boot_med_1$t - boot_med_2$t
> # produce histogram with 30 intervals (breaks) and densities (freq=false)
> # as a plot of the simulated sampling distribution
> hist(diff_boot_med,breaks=30,freq=FALSE,plot=TRUE)

> # construct an approximate one-sided (right) p-value counting what fraction of these
> # differences is ≤ 0 (left tail) -- “parallel construction”
> sum(diff_boot_med <=0)/999 #bootstrap-based p-value, second option
> [1] 0.004004004
> # very small, strong evidence against Ho.

What is the right option? The bootstrap should mimic the sampling process that produced the data – in our case, first option (sampling was NOT performed fixing how many windows to draw from each chromosome class, and then sampling separately).
> # Next we pass to **RANDOM PERMUTATIONS**. Define a function for the difference
> # btw medians of y for class 1 and 2 – reusable for any data set with a quantitative
> # clmn called “y” and a clmn called “class” expressing a binary classification (1,2).
> median_diff_perm_fun <- function(data,m) {
>     md <- median(data[,"y"][(data[,"class"]==1)] –
>          median(data[,"y"][(data[,"class"]==2)])
>     pdata <- data.frame("y"=data[,"y"],
>                          "class"=sample(data[,"class"],length(data[,"class"]), replace = FALSE))
>     pmd <- median(pdata[,"y"][(pdata[,"class"]==1)] –
>            median(pdata[,"y"][(pdata[,"class"]==2)])
>     count <- (pmd >= md) #for one-sided (right) p-value
>     for (i in 1:m-1) {
>         newpdata <- data.frame("y"=data[,"y"],
>                                "class"=sample(data[,"class"],length(data[,"class"]), replace = FALSE))
>         newpmd <- median(newpdata[,"y"][(newpdata[,"class"]==1)] –
>                  median(newpdata[,"y"][(newpdata[,"class"]==2)])
>         count <- count + (newpmd >= md)
>         pmd <- c(pmd, newpmd)
>     }
>     pval <- count/m
>     list("pmd"=pmd, "md"=md,"pval"=pval)
> }
> # perform the random permutations
> median_diff_perm_output <- median_diff_perm_fun(chicken_toy_ext,999)
> # display the observed difference
> median_diff_perm_output$md
> [1] 0.104925
> # display the one-sided (right) p-value
> median_diff_perm_output$pval
> [1] 0 #no difference obtained through permutations was > 0.104925
> # plot the simulated null distribution
> hist(median_diff_perm_output$pmd,breaks=30,freq=FALSE,plot=TRUE)
Part 2
Hypothesis system to be tested:
Ho: corr(log-length-ratio, log-insrt-ratio) = 0
Ha: corr(log-length-ratio, log-insrt-ratio) > 0

> # Define a function that computes the correlation btw y and x – reusable for any data
> # set with two quantitative columns called “y” and “x”.
> cor_fun <- function(data, indices){
> cor(data[indices,"y"],data[indices,"x"])
> }
> # perform the BOOTSTRAP
> boot_cor <- boot(chicken_toy,cor_fun, R=999,sim="ordinary")
> boot_cor
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = chicken_toy, statistic = cor_fun, R = 999, sim = "ordinary")
Bootstrap Statistics :
 original     bias        std. error
 t1*  0.3935879  0.003542511   0.08806583
> # produce histogram as a plot of the simulated sampling distribution
> plot(boot_cor)

> # construct an approximate one-sided (right) p-value counting what fraction of these
> # correlations is ≤ 0 (left tail) -- “parallel construction”
> sum(boot_cor$t <=0)/999 #bootstrap-based p-value
[1] 0 #no correlation obtained through bootstrap was ≤ 0
> # very small, strong evidence against Ho.
> # Next we pass to **RANDOM PERMUTATIONS**. Define a function for correlation
> # btw y and x – reusable for any data set with two quant clmns called “y” and “x”.
> cor_perm_fun <- function(data,m) {
#m=number of permutations
  c <- cor(data[,"y"],data[,"x"])
  pdata <- data.frame("
    "y”=data[,"y"],
    "x”=sample(data[,"x"],length(data[,"x"])), replace = FALSE))
  pc <- cor(pdata[,"y"],pdata[,"x"])
  count <- (pc >= c)   #for one-sided (right) p-value
  for (i in 1:m-1) {
    newpdata <- data.frame("
      "y”=data[,"y"],
      "x”=sample(data[,"x"],length(data[,"x"])), replace = FALSE))
    newpc <- cor(newpdata[,"y"],newpdata[,"x"])
    count <- count + (newpc >= c)
  }
  pc <- c(pc, newpc)
  pval <- count/m
  list("pc”=pc, "c”=c,"pval”=pval)
}
> # perform the random permutations
> cor_perm_output <- cor_perm_fun(chicken_toy,999)
> # display the observed difference
> cor_perm_output$c
[1] 0.3935879
> # display the one-sided (right) p-value
> cor_perm_output$pval
[1] 0   #no difference obtained through permutations was > 0. 3935879
> # plot the simulated null distribution
> hist(cor_perm_output$pc,breaks=30,freq=FALSE,plot=TRUE)