Tenth R Practice exercise merging annotation data into a gene expression analysis results data frame.Rmd

Alan E. Berger Feb 17, 2020

available at https://github.com/AlanBerger/Practice-programming-exercises-for-R

Introduction

This is the tenth in a sequence of programming exercises in "composing" an R function to carry out a particular task. Several of these "exercise files" likely will take several sessions to master the content. The material below practices composing a logical sequence of steps to program a function that will accomplish a specified task, and preparing a corresponding data frame.

The idea of this set of exercises is to practice correct use of R constructs and built in functions (functions that "come with" the basic R installation), while learning how to "put together" a correct sequence of blocks of commands that will obtain the desired result.
Note these exercises are quite cumulative - one should do them in order.

In these exercises, there will be a statement of what your function should do (what are the input variables and what the function should return) and a sequence of "hints". To get the most out of these exercises, try to write your function using as few hints as possible.
Note there are often several ways to write a function that will obtain the correct result. For these exercises the directions and hints may point toward a particular approach intended to practice particular constructs in R and a particular line of reasoning, even if there is a more efficent way to obtain the same result. There may also be an existing R function or package that will do what is stated for a given practice exercise, but here the point is to practice formulating a logical sequence of steps, with each step a section of code, to obtain a working function, not to find an existing solution or a quick solution using a more powerful R construct that is better addressed later on.

Motivation for this exercise

In some cases, such as with a gene expression data set, one will want to combine analysis results as obtained in the previous exercise with annotation information on the probes and on the genes that is in a separate file that can be read in as a data frame.

In the R code below we repeat the analysis, done in the previous exercise, of a small subset of gene expression data comparing expression levels in PBMC samples from patients with Wegener's granulomatosis (WG) with samples from normal controls (NC). We then also read in a small subset of the annotation file for the Illumina microarray platform used to measure these expression levels. The web site containing the full expression data set is: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE18885 and the web site containing the full annotation data for the microarray platform used in obtaining this data is: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL6104

What we will want to do is, conceptually, for each row **r** of the analysis results data frame, "find" the row **ra** of the annotation data frame that has the same Illumina Probe_ID and in effect append selected columns of row ra from the annotation data frame to row r of the analysis results data frame. R has a function **merge** that will do this, but for the first exercise we will practice using basic R constructs to compose R code that will do this - the second exercise here will use the merge function. First read through the code below, that provides the data frames one will use.

```
##############  analyze the gene expression data

# the url for reading the little gene expression data file into an R data frame using
# read.delim  (for reading in tab delimited text files) is given in the next 3 lines
```

```
url.for.data.file <- "https://raw.githubusercontent.com/AlanBerger/
Practice-programming-exercises-for-R/master/tiny-subset-of-GSE18885-
gene-expression-data-9-genes-WG-5-samples-Normal-Control-4-samples.tab.txt"

# read in the data as a data frame
ma <- read.delim(url.for.data.file, nrows = 9, check.names = FALSE,
                 stringsAsFactors = FALSE)

# display ma
ma
```

```
##   Illumina PROBE_ID    gene   NC 1   NC 2    NC 3   NC 4    WG 1   WG 2    WG 3
## 1      ILMN_1730867    AZU1 7.5719 7.8004  9.2853 7.6631 10.1137 7.7436  9.2764
## 2      ILMN_1766736     BPI 7.6313 8.0540  8.5862 8.3428 10.2506 8.7722 10.3036
## 3      ILMN_1688580    CAMP 8.7349 8.4446  9.9021 8.6375 12.1782 9.1157 13.2256
## 4      ILMN_1806056 CEACAM8 8.3485 8.2501 10.2744 8.1821 11.8043 8.0849 12.3573
## 5      ILMN_1753347   DEFA4 7.9815 8.5601 10.7857 7.9922 12.5999 8.4461 11.2124
## 6      ILMN_1706635    ELA2 8.8103 8.8859 11.1437 8.5417 11.5893 8.8538 10.5037
## 7      ILMN_1796316    MMP9 7.5506 7.5359  7.6544 7.6399  9.8731 9.2065 12.5744
## 8      ILMN_1705183     MPO 8.4673 8.4729  9.8118 8.5687 10.6068 9.0168  9.2412
## 9      ILMN_1802867  RNASE3 7.8771 8.6170  9.2255 7.9219 11.4129 8.5187 10.3054
##      WG 4   WG 5
## 1 13.2686 7.5619
## 2 14.3789 8.2839
## 3 14.9976 9.5919
## 4 15.3588 8.3634
## 5 15.4463 8.6771
## 6 15.0403 8.2517
## 7 14.1953 8.0277
## 8 14.5349 8.2586
## 9 14.2301 8.6456
```

```
# now, in a for loop, get the p-values and fold changes
num.genes <- nrow(ma)  # the number of genes in this data frame

gene <- ma$gene  # the column of gene names
probe.vec <- ma[[1]]  # the column of Illumina Probe_IDs
# get vectors to hold the p-value and fold change values
p.value <- numeric(num.genes)
fold.change <- numeric(num.genes)

for (i in 1:num.genes) {
# get the vector for the WG expression values and the vector
# for the NC expression values for the ith gene
   NCvec <- unlist(ma[i, 3:6])
   WGvec <- unlist(ma[i, 7:11])

# calculate the p-value and fold change
   pval <- t.test(NCvec, WGvec)$p.value  # two-sided unequal variance (Welch) t-test
   p.value[i] <- pval
   WG.over.NC.fold.change <- 2^(mean(WGvec) - mean(NCvec))
   fold.change[i] <- WG.over.NC.fold.change
}
```

```
# Construct the desired data frame.
analysis.results <- data.frame(probe.vec, gene, p.value, fold.change,
                               stringsAsFactors = FALSE, check.names = FALSE)
colnames(analysis.results) <- c("Illumina PROBE_ID", "gene", "two-sided p-value",
                                "WG/NC fold change")
analysis.results
```

```
##   Illumina PROBE_ID    gene two-sided p-value WG/NC fold change
## 1      ILMN_1730867    AZU1        0.22993719          2.853366
## 2      ILMN_1766736     BPI        0.10421720          4.737957
## 3      ILMN_1688580    CAMP        0.05731959          7.423116
## 4      ILMN_1806056 CEACAM8        0.15314950          5.388804
## 5      ILMN_1753347   DEFA4        0.14680018          5.450864
## 6      ILMN_1706635    ELA2        0.30872616          2.833058
## 7      ILMN_1796316    MMP9        0.04866715          9.064328
## 8      ILMN_1705183     MPO        0.25730723          2.831340
## 9      ILMN_1802867  RNASE3        0.10343345          4.633701
```

```
###############  read in the annotation data file (a small subset of the full annotation)

# read in the short edited Illumina microarray annotation data file called
# GPL6104-Illumina-microarray-platform-annotation-from-GEO-repository-
# small-subset-edited-example-Feb12.tab.txt

url.for.annotation.file <-
"https://raw.githubusercontent.com/AlanBerger/Practice-programming-exercises-for-R/
master/GPL6104-Illumina-microarray-platform-annotation-from-GEO-repository-small-
subset-edited-example-Feb12.tab.txt"

annotation.df  <- read.delim(url.for.annotation.file, nrows = 15, check.names = FALSE,
                  stringsAsFactors = FALSE)

# Note the use of nrows = 15 since there is information on the
# source of this data in later rows that should not be read in as data.
# The choice check.names = FALSE "tells" R to leave the column headers as is

annotation.df
```

```
##    Illumina Probe_ID ILMN_Gene Entrez_Gene_ID Chromosome
## 1       ILMN_1698220     PHTF2          57157          7
## 2       ILMN_1810835     SPRR3           6707          1
## 3       ILMN_1688580      CAMP            820          3
## 4       ILMN_1802867    RNASE3           6037         14
## 5       ILMN_1766736       BPI            671         20
## 6       ILMN_1753347     DEFA4           1669          8
## 7       ILMN_1749014      ACLY             47         17
## 8       ILMN_1785926    ZNF621         285268          3
## 9       ILMN_1796316      MMP9           4318         20
## 10      ILMN_1706635      ELA2           1991         19
## 11      ILMN_1705183       MPO           4353         17
## 12      ILMN_1806056   CEACAM8           1088         19
## 13      ILMN_1730867      AZU1            566         19
```

```
## 14      ILMN_1813399    ATP2B1            490         12
## 15      ILMN_1750599    ATP2B1            490         12
##           Probe location in chromosome
## 1                    77424374-77424423
## 2                  151242655-151242704
## 3  48241909-48241918:48241919-48241958
## 4                    20430090-20430139
## 5                    36399055-36399104
## 6      6781040-6781073:6781660-6781675
## 7                    37277254-37277303
## 8                    40555813-40555862
## 9                    44078320-44078369
## 10                       807179-807228
## 11                   53702640-53702689
## 12                   47776394-47776443
## 13                       781858-781907
## 14                   88506745-88506794
## 15                   88516580-88516629
##      Protein coded by gene - very short description
## 1                homeodomain transcription factor 2
## 2                      small proline-rich protein 3
## 3                cathelicidin antimicrobial peptide
## 4                   ribonuclease, RNase A family, 3
## 5      bactericidal/permeability-increasing protein
## 6                                 defensin, alpha 4
## 7                                 ATP citrate lyase
## 8                           zinc finger protein 621
## 9                         matrix metallopeptidase 9
## 10                                      elastase 2
## 11                                  myeloperoxidase
## 12 carcinoembryonic antigen-related cell adhesion 8
## 13               azurocidin 1 antimicrobial protein
## 14              ATPase, Ca++ transporting variant 2
## 15              ATPase, Ca++ transporting variant 1
```

```
# We see that this annotation data file has data for more Illumina probes than
# are in the analysis results data frame, and that the probe IDs are not in the
# same order as in the analysis results data frame.  For the purpose of the
# practice exercise below we will only append the columns containing the gene name,
# Chromosome number (which chromosome the gene is located on)
# and the short desciption of the protein encoded by the gene.
# Repeating the gene name gives a indicator to use to double check that the "merge"
# was done correctly.

# Note this data frame has an example of more than 1 probe for a given
# gene (ATP2B1), where different parts of the same gene are "queried".

# To keep things simpler with this example, for each probe ID in the analysis results,
# there is 1 row of the annotation data frame with the same probe ID.

# The merge function can handle the case where there is no matching probe ID in
# the annotation file for a probe ID in the analysis results data frame, in which case
# we would want to append NA's indicating that information is not available in the
# annotation file being used.
```

```
columns.to.keep <- c(1, 2, 4, 6)  # keep just these columns of the annotation data frame
# to have print outs easy to see
# we need to keep the probe IDs in column 1 to be able to match rows

annotation.df <- annotation.df[, columns.to.keep]
# from now on annotation.df will refer to this shortened version of the
# annotation data frame

annotation.df
```

```
##    Illumina Probe_ID ILMN_Gene Chromosome
## 1       ILMN_1698220     PHTF2          7
## 2       ILMN_1810835     SPRR3          1
## 3       ILMN_1688580      CAMP          3
## 4       ILMN_1802867    RNASE3         14
## 5       ILMN_1766736       BPI         20
## 6       ILMN_1753347     DEFA4          8
## 7       ILMN_1749014      ACLY         17
## 8       ILMN_1785926    ZNF621          3
## 9       ILMN_1796316      MMP9         20
## 10      ILMN_1706635      ELA2         19
## 11      ILMN_1705183       MPO         17
## 12      ILMN_1806056   CEACAM8         19
## 13      ILMN_1730867      AZU1         19
## 14      ILMN_1813399    ATP2B1         12
## 15      ILMN_1750599    ATP2B1         12
##      Protein coded by gene - very short description
## 1                 homeodomain transcription factor 2
## 2                       small proline-rich protein 3
## 3                 cathelicidin antimicrobial peptide
## 4                    ribonuclease, RNase A family, 3
## 5      bactericidal/permeability-increasing protein
## 6                                  defensin, alpha 4
## 7                                  ATP citrate lyase
## 8                            zinc finger protein 621
## 9                          matrix metallopeptidase 9
## 10                                      elastase 2
## 11                                   myeloperoxidase
## 12 carcinoembryonic antigen-related cell adhesion 8
## 13               azurocidin 1 antimicrobial protein
## 14              ATPase, Ca++ transporting variant 2
## 15              ATPase, Ca++ transporting variant 1
```

Programming Exercise: Append to analysis.results information from the annotation data frame

Approach: Form a vector of row numbers, call it annot.rows, such that for each row r of the analysis.results data frame; annot.rows[r] will contain the row of the annotation.df data frame that has the same Illumina Probe_ID as does row r of analysis.results
Then column binding annotation.df[annot.rows,] to analysis.results (using **cbind**) will yield the desired result.

Hint: Use a for loop, and use the **which** function to find, for each row r of analysis.results the row number ra of annotation.df that has the same probe ID as does row r of analysis.results

A working version of R code which does this is given below.

```
# use the analysis.results and annotation.df obtained in the R code above.
nrows <- nrow(analysis.results)
# create the integer vector annot.rows of length nrows to hold the
# row numbers of annotation.df matching (with respect to the probe ID) the
# analysis.results rows
annot.rows <- vector(mode = "integer", length = nrows)

# get the Illumina probe IDs vector from the annotation data frame
annotation.df.probe.ids <- annotation.df[[1]]

for (r in 1:nrows) {
   probe.id <- analysis.results[r, 1]
#  find the row ra of annotation.df whose Illumina probe ID matches probe.id
ra <- which(annotation.df.probe.ids == probe.id)
if (length(ra) != 1) stop("did not find unique matching probe id row")
annot.rows[r] <- ra
}

# append the correct rows (correctly lined up) of annotation.df to analysis.results
analysis.results.with.annotation <- cbind(analysis.results, annotation.df[annot.rows, ])
analysis.results.with.annotation  # display it
```

```
##    Illumina PROBE_ID    gene two-sided p-value WG/NC fold change
## 13      ILMN_1730867    AZU1        0.22993719          2.853366
## 5       ILMN_1766736     BPI        0.10421720          4.737957
## 3       ILMN_1688580    CAMP        0.05731959          7.423116
## 12      ILMN_1806056 CEACAM8        0.15314950          5.388804
## 6       ILMN_1753347   DEFA4        0.14680018          5.450864
## 10      ILMN_1706635    ELA2        0.30872616          2.833058
## 9       ILMN_1796316    MMP9        0.04866715          9.064328
## 11      ILMN_1705183     MPO        0.25730723          2.831340
## 4       ILMN_1802867  RNASE3        0.10343345          4.633701
##    Illumina Probe_ID ILMN_Gene Chromosome
## 13      ILMN_1730867      AZU1         19
## 5       ILMN_1766736       BPI         20
## 3       ILMN_1688580      CAMP          3
## 12      ILMN_1806056   CEACAM8         19
## 6       ILMN_1753347     DEFA4          8
## 10      ILMN_1706635      ELA2         19
## 9       ILMN_1796316      MMP9         20
## 11      ILMN_1705183       MPO         17
## 4       ILMN_1802867    RNASE3         14
##      Protein coded by gene - very short description
## 13               azurocidin 1 antimicrobial protein
## 5      bactericidal/permeability-increasing protein
## 3                cathelicidin antimicrobial peptide
## 12 carcinoembryonic antigen-related cell adhesion 8
## 6                                 defensin, alpha 4
## 10                                       elastase 2
```

```
## 9                          matrix metallopeptidase 9
## 11                                    myeloperoxidase
## 4                   ribonuclease, RNase A family, 3
```

```
# Note the row numbers are from the rows of annotation.df whose probe IDs
# matched up with the those in analysis.results
```

Second exercise: use the R merge function to append matching annotation lines to analysis.results

The R merge function can combine two data frames in various ways. See for example the web page by Joachim Schork which is a page in https://statisticsglobe.com/ titled "Merge Data Frames by Column Names in R (3 Examples)": https://statisticsglobe.com/r-merging-data-frames-by-column-names-merge-function
See also the R help on the merge function (via ? merge).

While it is good practice to use basic R constructs until they are easy for you to use, using an available R function can greatly simplify code which then makes it easier to keep free of bugs. Code that uses the merge function is given below. The merge function is capable of a number of types of merging in addition to the example below.

```
# Use the R merge function to append annotation to the analysis results

# recall that annotation.df is referring to the shortened version of the annotation
# merged.df <- merge(x = analysis.results, y = annotation.df,
#               by.x = "Illumina PROBE_ID", by.y = "Illumina Probe_ID",
#               all.x = TRUE, all.y = FALSE, sort = FALSE)

# What the above call to merge will do (once the comment symbols # are removed) is:
# use the by.x = "Illumina PROBE_ID" column of analysis.results as the "guide"
# and for each row r of analysis.results, the merge function will in effect search
# to find the row ra of annotation.df such that the entry of row ra in column
# by.y = "Illumina Probe_ID" of annotation.df  matches the entry of row r in
# the column by.x = "Illumina PROBE_ID" of analysis.results
# Note these 2 column names are not exactly the same so we need to specify
# the column names in x and y to be used to do matching of rows,
# using the arguments by.x and by.y
# The merge function will, in effect, append row ra of annotation.df to row r
# of analysis.results
# The choice all.x = TRUE means: if there is no match for the entry of row r in
# the column "Illumina PROBE_ID" of analysis.results anywhere in the column
# "Illumina Probe_ID" of annotation.df, then a row of NA's is appended to row r
# of analysis.results
# The choice all.y = FALSE means don't include rows of annotation.df other than
# those appended to analysis.results as desribed above.
# The choice sort = FALSE means do not sort the resulting data frame
# (any sorting would have been done for this call to merge using
# the "Illumina PROBE_ID" column).

merged.df <- merge(x = analysis.results, y = annotation.df,
              by.x = "Illumina PROBE_ID", by.y = "Illumina Probe_ID",
              all.x = TRUE, all.y = FALSE, sort = FALSE)
# display it
merged.df
```

```
##   Illumina PROBE_ID    gene two-sided p-value WG/NC fold change ILMN_Gene
## 1      ILMN_1730867    AZU1        0.22993719          2.853366      AZU1
## 2      ILMN_1766736     BPI        0.10421720          4.737957       BPI
## 3      ILMN_1688580    CAMP        0.05731959          7.423116      CAMP
## 4      ILMN_1806056 CEACAM8        0.15314950          5.388804   CEACAM8
## 5      ILMN_1753347   DEFA4        0.14680018          5.450864     DEFA4
## 6      ILMN_1706635    ELA2        0.30872616          2.833058      ELA2
## 7      ILMN_1796316    MMP9        0.04866715          9.064328      MMP9
## 8      ILMN_1705183     MPO        0.25730723          2.831340       MPO
## 9      ILMN_1802867  RNASE3        0.10343345          4.633701    RNASE3
##   Chromosome   Protein coded by gene - very short description
## 1         19               azurocidin 1 antimicrobial protein
## 2         20     bactericidal/permeability-increasing protein
## 3          3               cathelicidin antimicrobial peptide
## 4         19 carcinoembryonic antigen-related cell adhesion 8
## 5          8                                 defensin, alpha 4
## 6         19                                       elastase 2
## 7         20                         matrix metallopeptidase 9
## 8         17                                    myeloperoxidase
## 9         14                 ribonuclease, RNase A family, 3
```

```
# Note the Illumina Probe_ID column of annotation.df is NOT included in merge.df
# Let's check that merged.df is the same as analysis.results.with.annotation obtained
# above. First we need to remove the Illumina Probe_ID column from annotation.df
# that is included in analysis.results.with.annotation before we check.
analysis.results.with.annotation <- analysis.results.with.annotation[, -5]

# check if they are the same
identical(analysis.results.with.annotation, merged.df)
```

```
## [1] FALSE
```

```
# What happened? they looked the same -- so now a little "adventure"
# in finding out what happened -- this sort of thing "comes with the territory"
# when programming in any language (they each have their own quirks).

# Let's look closer
attributes(analysis.results.with.annotation)
```

```
## $names
## [1] "Illumina PROBE_ID"
## [2] "gene"
## [3] "two-sided p-value"
## [4] "WG/NC fold change"
## [5] "ILMN_Gene"
## [6] "Chromosome"
## [7] "Protein coded by gene - very short description"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] 13  5  3 12  6 10  9 11  4
```

```r
attributes(merged.df)
```

```
## $names
## [1] "Illumina PROBE_ID"
## [2] "gene"
## [3] "two-sided p-value"
## [4] "WG/NC fold change"
## [5] "ILMN_Gene"
## [6] "Chromosome"
## [7] "Protein coded by gene - very short description"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3 4 5 6 7 8 9
```

```r
# So the row names were different

# Looks like we can fix this by setting the row names of
# analysis.results.with.annotation to be those for merged.df
row.names(analysis.results.with.annotation) <- row.names(merged.df)
identical(analysis.results.with.annotation, merged.df)
```

```
## [1] FALSE
```

```r
# Now what ????   Let's look at the attributes again

attributes(analysis.results.with.annotation)
```

```
## $names
## [1] "Illumina PROBE_ID"
## [2] "gene"
## [3] "two-sided p-value"
## [4] "WG/NC fold change"
## [5] "ILMN_Gene"
## [6] "Chromosome"
## [7] "Protein coded by gene - very short description"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

```r
attributes(merged.df)
```

```
## $names
## [1] "Illumina PROBE_ID"
## [2] "gene"
## [3] "two-sided p-value"
```

```
## [4] "WG/NC fold change"
## [5] "ILMN_Gene"
## [6] "Chromosome"
## [7] "Protein coded by gene - very short description"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3 4 5 6 7 8 9
```

```
# So the row names for analysis.results.with.annotation are 1:9 as characters
# and the row names for merged.df are 1:9 as integers -
# As I said, every language has its quirks
row.names(analysis.results.with.annotation) <- 1:9
attributes(analysis.results.with.annotation)
```

```
## $names
## [1] "Illumina PROBE_ID"
## [2] "gene"
## [3] "two-sided p-value"
## [4] "WG/NC fold change"
## [5] "ILMN_Gene"
## [6] "Chromosome"
## [7] "Protein coded by gene - very short description"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3 4 5 6 7 8 9
```

```
# Now if they aren't identical we really do have problems
identical(analysis.results.with.annotation, merged.df)
```

```
## [1] TRUE
```

```
# So some semblance of order is restored.  The problem was
# row.names(merged.df) returned a character vector
str(row.names(merged.df))
```

```
##  chr [1:9] "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

```
# One final verification: I'm going to remove the row of the annotation
# data frame corresponding the probe ID for the BPI gene
# and then use the merge function
annotation.df <- annotation.df[-5, ]

merged.df <- merge(x = analysis.results, y = annotation.df,
              by.x = "Illumina PROBE_ID", by.y = "Illumina Probe_ID",
              all.x = TRUE, all.y = FALSE, sort = FALSE)
# display it
merged.df
```

```
##   Illumina PROBE_ID    gene two-sided p-value WG/NC fold change ILMN_Gene
## 1      ILMN_1730867    AZU1        0.22993719          2.853366      AZU1
## 2      ILMN_1688580    CAMP        0.05731959          7.423116      CAMP
## 3      ILMN_1806056 CEACAM8        0.15314950          5.388804   CEACAM8
## 4      ILMN_1753347   DEFA4        0.14680018          5.450864     DEFA4
## 5      ILMN_1706635    ELA2        0.30872616          2.833058      ELA2
## 6      ILMN_1796316    MMP9        0.04866715          9.064328      MMP9
## 7      ILMN_1705183     MPO        0.25730723          2.831340       MPO
## 8      ILMN_1802867  RNASE3        0.10343345          4.633701    RNASE3
## 9      ILMN_1766736     BPI        0.10421720          4.737957      <NA>
##   Chromosome   Protein coded by gene - very short description
## 1         19               azurocidin 1 antimicrobial protein
## 2          3               cathelicidin antimicrobial peptide
## 3         19 carcinoembryonic antigen-related cell adhesion 8
## 4          8                                 defensin, alpha 4
## 5         19                                       elastase 2
## 6         20                       matrix metallopeptidase 9
## 7         17                                  myeloperoxidase
## 8         14                 ribonuclease, RNase A family, 3
## 9         NA                                             <NA>
```

```
# Note merge filled in NA's for the annotation columns for the row for the probe ID
# corresponding to BPI as expected.
# The merge function also placed the row for which there was no match for the
# probe ID  in the annotation file at the bottom of the merged data frame.

# This illustrates the kind of "exploring" one should do when using a new R function,
# particularly one that has a somewhat complex range of options and for which the
# output has a range of possibilities, in order to be confident about what it will
# do when called a certain way
```

Hope this was informative and good practice. The next exercise will contain further practice in using data frames, and point out some types of logical mistakes that may result in actual output that, however, is incorrect, rather than an error message. This is the most dangerous type of mistake, in that if the incorrect output is not obviously wrong, the mistake might not be recognized until it causes serious consequences. That is why it is always wise to do, whenever possible, test runs for cases where one knows or can independently calculate the true result.

= =