```
  |
|   0%

| In this lesson, you'll learn how to use lapply() and sapply(), the two most
important members of
| R's *apply family of functions, also known as loop functions.

...

  |==
|   2%
| These powerful functions, along with their close relatives (vapply() and ta
pply(), among others)
| offer a concise and convenient means of implementing the Split-Apply-Combin
e strategy for data
| analysis.

...

  |====
|   4%
| Each of the *apply functions will SPLIT up some data into smaller pieces, A
PPLY a function to each
| piece, then COMBINE the results. A more detailed discussion of this strateg
y is found in Hadley
| Wickham's Journal of Statistical Software paper titled 'The Split-Apply-Com
bine Strategy for Data
| Analysis'.

...

  |======
|   6%
| Throughout this lesson, we'll use the Flags dataset from the UCI Machine Le
arning Repository. This
| dataset contains details of various nations and their flags. More informati
on may be found here:
| http://archive.ics.uci.edu/ml/datasets/Flags

...

  |=======
|   8%
| Let's jump right in so you can get a feel for how these special functions w
ork!

...

  |=========
|  10%
| I've stored the dataset in a variable called flags. Type head(flags) to pre
view the first six lines
| (i.e. the 'head') of the dataset.

> head(flags)
            name landmass zone area population language religion bars stripes
colours red green blue
1    Afghanistan        5    1  648         16       10        2    0       3
5   1     1    0
2        Albania        3    1   29          3        6        6    0       0
3   1     0    0
3        Algeria        4    1 2388         20        8        2    2       0
3   1     1    0
```

```
4 American-Samoa        6    3    0          0        1        1    0       0
5   1     0    1
5        Andorra        3    1    0          0        6        0    3       0
3   1     0    1
6         Angola        4    2 1247          7       10        5    0       2
3   1     0    0
  gold white black orange mainhue circles crosses saltires quarters sunstars
crescent triangle icon
1    1     1     1      0   green       0       0        0        0        1
0        0    1
2    1     0     1      0     red       0       0        0        0        1
0        0    0
3    0     1     0      0   green       0       0        0        0        1
1        0    0
4    1     1     0      1    blue       0       0        0        0        0
0        1    1
5    1     0     0      0    gold       0       0        0        0        0
0        0    0
6    1     0     1      0     red       0       0        0        0        1
0        0    1
  animate text topleft botright
1       0    0   black    green
2       1    0     red      red
3       0    0   green    white
4       1    0    blue      red
5       0    0    blue      red
6       0    0     red    black

| Keep up the great work!

  |===========
|  12%
| You may need to scroll up to see all of the output. Now, let's check out th
e dimensions of the
| dataset using dim(flags).

> dim(flags)
[1] 194  30

| You nailed it! Good job!

  |=============
|  14%
| This tells us that there are 194 rows, or observations, and 30 columns, or
variables. Each
| observation is a country and each variable describes some characteristic of
that country or its
| flag. To open a more complete description of the dataset in a separate text
file, type viewinfo()
| when you are back at the prompt (>).

...

  |===============
|  16%
| As with any dataset, we'd like to know in what format the variables have be
en stored. In other
| words, what is the 'class' of each variable? What happens if we do class(fl
ags)? Try it out.

> class(flags)
[1] "data.frame"

| Keep up the great work!
```

```
  |=================
|  18%
| That just tells us that the entire dataset is stored as a 'data.frame', whi
ch doesn't answer our
| question. What we really need is to call the class() function on each indiv
idual column. While we
| could do this manually (i.e. one column at a time) it's much faster if we c
an automate the process.
| Sounds like a loop!

...

  |==================
|  20%
| The lapply() function takes a list as input, applies a function to each ele
ment of the list, then
| returns a list of the same length as the original one. Since a data frame i
s really just a list of
| vectors (you can see this with as.list(flags)), we can use lapply() to appl
y the class() function
| to each column of the flags dataset. Let's see it in action!

...

  |====================
|  22%
| Type cls_list <- lapply(flags, class) to apply the class() function to each
column of the flags
| dataset and store the result in a variable called cls_list. Note that you j
ust supply the name of
| the function you want to apply (i.e. class), without the usual parentheses
after it.

> cls_list <- lapply(flags, class)

| Keep working like that and you'll get there!

|======================
|  24%
| Type cls_list to view the result.

> cls_list
$name
[1] "factor"

$landmass
[1] "integer"

$zone
[1] "integer"

$area
[1] "integer"

$population
[1] "integer"

$language
[1] "integer"

$religion
[1] "integer"
```

```
$bars
[1] "integer"

$stripes
[1] "integer"

$colours
[1] "integer"

$red
[1] "integer"

$green
[1] "integer"

$blue
[1] "integer"

$gold
[1] "integer"

$white
[1] "integer"

$black
[1] "integer"

$orange
[1] "integer"

$mainhue
[1] "factor"

$circles
[1] "integer"

$crosses
[1] "integer"

$saltires
[1] "integer"

$quarters
[1] "integer"

$sunstars
[1] "integer"

$crescent
[1] "integer"

$triangle
[1] "integer"

$icon
[1] "integer"

$animate
[1] "integer"

$text
[1] "integer"

$topleft
```

```
[1] "factor"

$botright
[1] "factor"


| You are doing so well!

  |========================
|  26%
| The 'l' in 'lapply' stands for 'list'. Type class(cls_list) to confirm that
lapply() returned a
| list.

> class(cls_list)
[1] "list"

| You are really on a roll!

  |==========================
|  28%
| As expected, we got a list of length 30 -- one element for each variable/co
lumn. The output would
| be considerably more compact if we could represent it as a vector instead o
f a list.

...

  |============================
|  30%
| You may remember from a previous lesson that lists are most helpful for sto
ring multiple classes of
| data. In this case, since every element of the list returned by lapply() is
a character vector of
| length one (i.e. "integer" and "vector"), cls_list can be simplified to a c
haracter vector. To do
| this manually, type as.character(cls_list).

> as.character(cls_list)
 [1] "factor"  "integer" "integer" "integer" "integer" "integer" "integer" "i
nteger" "integer"
[10] "integer" "integer" "integer" "integer" "integer" "integer" "integer" "i
nteger" "factor"
[19] "integer" "integer" "integer" "integer" "integer" "integer" "integer" "i
nteger" "integer"
[28] "integer" "factor"  "factor"

| That's a job well done!

  |=============================
|  32%
| sapply() allows you to automate this process by calling lapply() behind the
scenes, but then
| attempting to simplify (hence the 's' in 'sapply') the result for you. Use
sapply() the same way
| you used lapply() to get the class of each column of the flags dataset and
store the result in
| cls_vect. If you need help, type ?sapply to bring up the documentation.

> clas_vect<-sapply(flags,class)

| Nice try, but that's not exactly what I was hoping for. Try again. Or, type
info() for more
| options.
| Type cls_vect <- sapply(flags, class) to store the column classes in a char
acter vector called
| cls_vect.

> cls_vect <- sapply(flags, class)

| All that hard work is paying off!

  |===============================
|  34%
| Use class(cls_vect) to confirm that sapply() simplified the result to a cha
racter vector.

> clas_vect(cls_vect)
Error in clas_vect(cls_vect) : could not find function "clas_vect"
> class(clas_vect)
[1] "character"

| That's not the answer I was looking for, but try again. Or, type info() for
more options.

| Type class(cls_vect) to confirm that sapply() returned a character vector.

> class(cls_vect)
[1] "character"

| You are doing so well!

  |=================================
|  36%
| In general, if the result is a list where every element is of length one, t
hen sapply() returns a
| vector. If the result is a list where every element is a vector of the same
length (> 1), sapply()
| returns a matrix. If sapply() can't figure things out, then it just returns
a list, no different
| from what lapply() would give you.

...

  |===================================
|  38%
| Let's practice using lapply() and sapply() some more!

...

  |=====================================
|  40%
| Columns 11 through 17 of our dataset are indicator variables, each represen
ting a different color.
| The value of the indicator variable is 1 if the color is present in a count
ry's flag and 0
| otherwise.

...

  |=======================================
|  42%
| Therefore, if we want to know the total number of countries (in our dataset
) with, for example, the
| color orange on their flag, we can just add up all of the 1s and 0s in the
'orange' column. Try
| sum(flags$orange) to see this.
```

```
> sum(flags$orange)
[1] 26

| You're the best!

  |========================================
|  44%
| Now we want to repeat this operation for each of the colors recorded in the
dataset.

...

  |==========================================
|  46%
| First, use flag_colors <- flags[, 11:17] to extract the columns containing
the color data and store
| them in a new data frame called flag_colors. (Note the comma before 11:17.
This subsetting command
| tells R that we want all rows, but only columns 11 through 17.)

> flag_colors <- flags[, 11:17]

| You are quite good my friend!

  |============================================
|  48%
| Use the head() function to look at the first 6 lines of flag_colors.

> head(flag_colors)
  red green blue gold white black orange
1   1     1    0    1     1     1      0
2   1     0    0    1     0     1      0
3   1     1    0    0     1     0      0
4   1     0    1    1     1     0      1
5   1     0    1    1     0     0      0
6   1     0    0    1     0     1      0

| Nice work!

  |==============================================
|  50%
| To get a list containing the sum of each column of flag_colors, call the la
pply() function with two
| arguments. The first argument is the object over which we are looping (i.e.
flag_colors) and the
| second argument is the name of the function we wish to apply to each column
(i.e. sum). Remember
| that the second argument is just the name of the function with no parenthes
es, etc.

> lapply(flag_colours, sum)
Error in lapply(flag_colours, sum) : object 'flag_colours' not found
> lapply(flag_colors, sum)
$red
[1] 153

$green
[1] 91

$blue
[1] 99

$gold
```

```
[1] 91

$white
[1] 146

$black
[1] 52

$orange
[1] 26


| That's correct!

  |================================================
|  52%
| This tells us that of the 194 flags in our dataset, 153 contain the color r
ed, 91 contain green, 99
| contain blue, and so on.

...

  |==================================================
|  54%
| The result is a list, since lapply() always returns a list. Each element of
this list is of length
| one, so the result can be simplified to a vector by calling sapply() instea
d of lapply(). Try it
| now.

> sapply(flag_colors,sum)
   red  green   blue   gold  white  black orange 
   153     91     99     91    146     52     26 

| All that practice is paying off!

  |====================================================
|  56%
| Perhaps it's more informative to find the proportion of flags (out of 194)
containing each color.
| Since each column is just a bunch of 1s and 0s, the arithmetic mean of each
column will give us the
| proportion of 1s. (If it's not clear why, think of a simpler situation wher
e you have three 1s and
| two 0s -- (1 + 1 + 1 + 0 + 0)/5 = 3/5 = 0.6).

...

  |=====================================================
|  58%
| Use sapply() to apply the mean() function to each column of flag_colors. Re
member that the second
| argument to sapply() should just specify the name of the function (i.e. mea
n) that you want to
| apply.

> sapply(flag_colors, mean)
      red     green      blue      gold     white     black    orange 
0.7886598 0.4690722 0.5103093 0.4690722 0.7525773 0.2680412 0.1340206 

| Perseverance, that's the answer.

  |=======================================================
|  60%
| In the examples we've looked at so far, sapply() has been able to simplify
the result to vector.
| That's because each element of the list returned by lapply() was a vector o
f length one. Recall
| that sapply() instead returns a matrix when each element of the list return
ed by lapply() is a
| vector of the same length (> 1).

...

  |==========================================================
|  62%
| To illustrate this, let's extract columns 19 through 23 from the flags data
set and store the result
| in a new data frame called flag_shapes. flag_shapes <- flags[, 19:23] will
do it.

>
> flag_shapes <- flags[, 19:23]

| You got it!

  |============================================================
|  64%
| Each of these columns (i.e. variables) represents the number of times a par
ticular shape or design
| appears on a country's flag. We are interested in the minimum and maximum n
umber of times each
| shape or design appears.

...

  |==============================================================
|  66%
| The range() function returns the minimum and maximum of its first argument,
which should be a
| numeric vector. Use lapply() to apply the range function to each column of
flag_shapes. Don't worry
| about storing the result in a new variable. By now, we know that lapply() a
lways returns a list.

> lapply(flag_shapes,range)
$circles
[1] 0 4

$crosses
[1] 0 2

$saltires
[1] 0 1

$quarters
[1] 0 4

$sunstars
[1]  0 50


| You are doing so well!

  |================================================================
|  68%
| Do the same operation, but using sapply() and store the result in a variabl
e called shape_mat.
```

```
> shape_mat<-sapply(flag_shapes, range)

| You got it right!

  |================================================================
|  70%
| View the contents of shape_mat.

> shape_mat
     circles crosses saltires quarters sunstars
[1,]       0       0        0        0        0
[2,]       4       2        1        4       50

| You're the best!

  |==================================================================
|  72%
| Each column of shape_mat gives the minimum (row 1) and maximum (row 2) numb
er of times its
| respective shape appears in different flags.

...

  |====================================================================
|  74%
| Use the class() function to confirm that shape_mat is a matrix.

> class(shape_mat)
[1] "matrix"

| All that practice is paying off!

  |======================================================================
|  76%
| As we've seen, sapply() always attempts to simplify the result given by lap
ply(). It has been
| successful in doing so for each of the examples we've looked at so far. Let
's look at an example
| where sapply() can't figure out how to simplify the result and thus returns
a list, no different
| from lapply().

...

  |========================================================================
|  78%
| When given a vector, the unique() function returns a vector with all duplic
ate elements removed. In
| other words, unique() returns a vector of only the 'unique' elements. To se
e how it works, try
| unique(c(3, 4, 5, 5, 5, 6, 6)).

> unique(c(3, 4, 5, 5, 5, 6, 6))
[1] 3 4 5 6

| Your dedication is inspiring!

  |==========================================================================
|  80%
| We want to know the unique values for each variable in the flags dataset. T
o accomplish this, use
| lapply() to apply the unique() function to each column in the flags dataset
, storing the result in
```

```
| a variable called unique_vals.

> unique_vals<-lapply(flags, unique)

| You got it right!

  |===========================================================================
=                 |  82%
| Print the value of unique_vals to the console.

> unique_vals
$name
  [1] Afghanistan              Albania                  Algeria
  [4] American-Samoa           Andorra                  Angola
  [7] Anguilla                 Antigua-Barbuda          Argentina
 [10] Argentine                Australia                Austria
 [13] Bahamas                  Bahrain                  Bangladesh
 [16] Barbados                 Belgium                  Belize
 [19] Benin                    Bermuda                  Bhutan
 [22] Bolivia                  Botswana                 Brazil
 [25] British-Virgin-Isles     Brunei                   Bulgaria
 [28] Burkina                  Burma                    Burundi
 [31] Cameroon                 Canada                   Cape-Verde-Islands
 [34] Cayman-Islands           Central-African-Republic Chad
 [37] Chile                    China                    Colombia
 [40] Comorro-Islands          Congo                    Cook-Islands
 [43] Costa-Rica               Cuba                     Cyprus
 [46] Czechoslovakia           Denmark                  Djibouti
 [49] Dominica                 Dominican-Republic       Ecuador
 [52] Egypt                    El-Salvador              Equatorial-Guinea
 [55] Ethiopia                 Faeroes                  Falklands-Malvinas
 [58] Fiji                     Finland                  France
 [61] French-Guiana            French-Polynesia         Gabon
 [64] Gambia                   Germany-DDR              Germany-FRG
 [67] Ghana                    Gibraltar                Greece
 [70] Greenland                Grenada                  Guam
 [73] Guatemala                Guinea                   Guinea-Bissau
 [76] Guyana                   Haiti                    Honduras
 [79] Hong-Kong                Hungary                  Iceland
 [82] India                    Indonesia                Iran
 [85] Iraq                     Ireland                  Israel
 [88] Italy                    Ivory-Coast              Jamaica
 [91] Japan                    Jordan                   Kampuchea
 [94] Kenya                    Kiribati                 Kuwait
 [97] Laos                     Lebanon                  Lesotho
[100] Liberia                  Libya                    Liechtenstein
[103] Luxembourg               Malagasy                 Malawi
[106] Malaysia                 Maldive-Islands          Mali
[109] Malta                    Marianas                 Mauritania
[112] Mauritius                Mexico                   Micronesia
[115] Monaco                   Mongolia                 Montserrat
[118] Morocco                  Mozambique               Nauru
[121] Nepal                    Netherlands              Netherlands-Antilles
[124] New-Zealand              Nicaragua                Niger
[127] Nigeria                  Niue                     North-Korea
[130] North-Yemen              Norway                   Oman
[133] Pakistan                 Panama                   Papua-New-Guinea
[136] Parguay                  Peru                     Philippines
[139] Poland                   Portugal                 Puerto-Rico
[142] Qatar                    Romania                  Rwanda
[145] San-Marino               Sao-Tome                 Saudi-Arabia
[148] Senegal                  Seychelles               Sierra-Leone
[151] Singapore                Soloman-Islands          Somalia
[154] South-Africa             South-Korea              South-Yemen
```

```
[157] Spain                    Sri-Lanka                St-Helena
[160] St-Kitts-Nevis           St-Lucia                 St-Vincent
[163] Sudan                    Surinam                  Swaziland
[166] Sweden                   Switzerland              Syria
[169] Taiwan                   Tanzania                 Thailand
[172] Togo                     Tonga                    Trinidad-Tobago
[175] Tunisia                  Turkey                   Turks-Cocos-Islands
[178] Tuvalu                   UAE                      Uganda
[181] UK                       Uruguay                  US-Virgin-Isles
[184] USA                      USSR                     Vanuatu
[187] Vatican-City             Venezuela                Vietnam
[190] Western-Samoa            Yugoslavia               Zaire
[193] Zambia                   Zimbabwe
194 Levels: Afghanistan Albania Algeria American-Samoa Andorra Angola Anguill
a ... Zimbabwe

$landmass
[1] 5 3 4 6 1 2

$zone
[1] 1 3 2 4

$area
  [1]   648    29  2388     0  1247  2777  7690    84    19     1   143    31
23   113    47  1099
 [17]   600  8512     6   111   274   678    28   474  9976     4   623  1284
757  9561  1139     2
 [33]   342    51   115     9   128    43    22    49   284  1001    21  1222
12    18   337   547
 [49]    91   268    10   108   249   239   132  2176   109   246    36   215
112    93   103  3268
 [65]  1904  1648   435    70   301   323    11   372    98   181   583   236
30  1760     3   587
 [81]   118   333  1240  1031  1973  1566   447   783   140    41  1267   925
121   195   324   212
 [97]   804    76   463   407  1285   300   313    92   237    26  2150   196
72   637  1221    99
[113]   288   505    66  2506    63    17   450   185   945   514    57     5
164   781   245   178
[129]  9363 22402    15   912   256   905   753   391

$population
 [1]   16    3   20    0    7   28   15    8   90   10    1    6  119    9
35    4   24    2   11
[20] 1008    5   47   31   54   17   61   14  684  157   39   57  118   13
77   12   56   18   84
[39]   48   36   22   29   38   49   45  231  274   60

$language
 [1] 10  6  8  1  2  4  3  5  7  9

$religion
[1] 2 6 1 0 5 3 4 7

$bars
[1] 0 2 3 1 5

$stripes
 [1]  3  0  2  1  5  9 11 14  4  6 13  7

$colours
[1] 5 3 2 8 6 4 7 1

$red
```

```
[1] 1 0

$green
[1] 1 0

$blue
[1] 0 1

$gold
[1] 1 0

$white
[1] 1 0

$black
[1] 1 0

$orange
[1] 0 1

$mainhue
[1] green  red    blue   gold   white  orange black  brown 
Levels: black blue brown gold green orange red white

$circles
[1] 0 1 4 2

$crosses
[1] 0 1 2

$saltires
[1] 0 1

$quarters
[1] 0 1 4

$sunstars
 [1]  1  0  6 22 14  3  4  5 15 10  7  2  9 50

$crescent
[1] 0 1

$triangle
[1] 0 1

$icon
[1] 1 0

$animate
[1] 0 1

$text
[1] 0 1

$topleft
[1] black  red    green  blue   white  orange gold  
Levels: black blue gold green orange red white

$botright
[1] green  red    white  black  blue   gold   orange brown 
Levels: black blue brown gold green orange red white


| Keep up the great work!
```

```
  |=============================================================================
===                |  84%
| Since unique_vals is a list, you can use what you've learned to determine t
he length of each
| element of unique_vals (i.e. the number of unique values for each variable)
. Simplify the result,
| if possible. Hint: Apply the length() function to each element of unique_va
ls.

> length(unique_vals)
[1] 30

| Give it another try. Or, type info() for more options.

| Apply the length() function to each element of the unique_vals list using s
apply(). Remember, no
| parentheses after the name of the function you are applying (i.e. length).

> sapply(unique_vals,length)
      name   landmass       zone       area population   language   religion 
bars    stripes 
       194          6          4        136         48         10          8 
5         12 
   colours        red      green       blue       gold      white      black 
orange    mainhue 
         8          2          2          2          2          2          2 
2          8 
   circles    crosses   saltires   quarters   sunstars   crescent   triangle 
icon    animate 
         4          3          2          3         14          2          2 
2          2 
      text    topleft   botright 
         2          7          8 

| You got it!

  |=============================================================================
=====              |  86%
| The fact that the elements of the unique_vals list are all vectors of *diff
erent* length poses a
| problem for sapply(), since there's no obvious way of simplifying the resul
t.

...

  |=============================================================================
=======            |  88%
| Use sapply() to apply the unique() function to each column of the flags dat
aset to see that you get
| the same unsimplified list that you got from lapply().

> sapply(flags, unique)
$name
  [1] Afghanistan              Albania                  Algeria                 
  [4] American-Samoa           Andorra                  Angola                  
  [7] Anguilla                 Antigua-Barbuda          Argentina               
 [10] Argentine                Australia                Austria                 
 [13] Bahamas                  Bahrain                  Bangladesh              
 [16] Barbados                 Belgium                  Belize                  
 [19] Benin                    Bermuda                  Bhutan                  
 [22] Bolivia                  Botswana                 Brazil                  
 [25] British-Virgin-Isles     Brunei                   Bulgaria                
 [28] Burkina                  Burma                    Burundi                 
 [31] Cameroon                 Canada                   Cape-Verde-Islands
 [34] Cayman-Islands           Central-African-Republic Chad
 [37] Chile                    China                    Colombia
 [40] Comorro-Islands          Congo                    Cook-Islands
 [43] Costa-Rica               Cuba                     Cyprus
 [46] Czechoslovakia           Denmark                  Djibouti
 [49] Dominica                 Dominican-Republic       Ecuador
 [52] Egypt                    El-Salvador              Equatorial-Guinea
 [55] Ethiopia                 Faeroes                  Falklands-Malvinas
 [58] Fiji                     Finland                  France
 [61] French-Guiana            French-Polynesia         Gabon
 [64] Gambia                   Germany-DDR              Germany-FRG
 [67] Ghana                    Gibraltar                Greece
 [70] Greenland                Grenada                  Guam
 [73] Guatemala                Guinea                   Guinea-Bissau
 [76] Guyana                   Haiti                    Honduras
 [79] Hong-Kong                Hungary                  Iceland
 [82] India                    Indonesia                Iran
 [85] Iraq                     Ireland                  Israel
 [88] Italy                    Ivory-Coast              Jamaica
 [91] Japan                    Jordan                   Kampuchea
 [94] Kenya                    Kiribati                 Kuwait
 [97] Laos                     Lebanon                  Lesotho
[100] Liberia                  Libya                    Liechtenstein
[103] Luxembourg               Malagasy                 Malawi
[106] Malaysia                 Maldive-Islands          Mali
[109] Malta                    Marianas                 Mauritania
[112] Mauritius                Mexico                   Micronesia
[115] Monaco                   Mongolia                 Montserrat
[118] Morocco                  Mozambique               Nauru
[121] Nepal                    Netherlands              Netherlands-Antilles
[124] New-Zealand              Nicaragua                Niger
[127] Nigeria                  Niue                     North-Korea
[130] North-Yemen              Norway                   Oman
[133] Pakistan                 Panama                   Papua-New-Guinea
[136] Parguay                  Peru                     Philippines
[139] Poland                   Portugal                 Puerto-Rico
[142] Qatar                    Romania                  Rwanda
[145] San-Marino               Sao-Tome                 Saudi-Arabia
[148] Senegal                  Seychelles               Sierra-Leone
[151] Singapore                Soloman-Islands          Somalia
[154] South-Africa             South-Korea              South-Yemen
[157] Spain                    Sri-Lanka                St-Helena
[160] St-Kitts-Nevis           St-Lucia                 St-Vincent
[163] Sudan                    Surinam                  Swaziland
[166] Sweden                   Switzerland              Syria
[169] Taiwan                   Tanzania                 Thailand
[172] Togo                     Tonga                    Trinidad-Tobago
[175] Tunisia                  Turkey                   Turks-Cocos-Islands
[178] Tuvalu                   UAE                      Uganda
[181] UK                       Uruguay                  US-Virgin-Isles
[184] USA                      USSR                     Vanuatu
[187] Vatican-City             Venezuela                Vietnam
[190] Western-Samoa            Yugoslavia               Zaire
[193] Zambia                   Zimbabwe
194 Levels: Afghanistan Albania Algeria American-Samoa Andorra Angola Anguill
a ... Zimbabwe

$landmass
[1] 5 3 4 6 1 2

$zone
[1] 1 3 2 4
```

```
$area
  [1]   648    29  2388     0  1247  2777  7690    84    19     1   143    31
23   113    47  1099
 [17]   600  8512     6   111   274   678    28   474  9976     4   623  1284
757  9561  1139     2
 [33]   342    51   115     9   128    43    22    49   284  1001    21  1222
12    18   337   547
 [49]    91   268    10   108   249   239   132  2176   109   246    36   215
112    93   103  3268
 [65]  1904  1648   435    70   301   323    11   372    98   181   583   236
30  1760     3   587
 [81]   118   333  1240  1031  1973  1566   447   783   140    41  1267   925
121   195   324   212
 [97]   804    76   463   407  1285   300   313    92   237    26  2150   196
72   637  1221    99
[113]   288   505    66  2506    63    17   450   185   945   514    57     5
164   781   245   178
[129]  9363 22402    15   912   256   905   753   391

$population
 [1]   16    3   20    0    7   28   15    8   90   10    1    6  119    9
35    4   24    2   11
[20] 1008    5   47   31   54   17   61   14  684  157   39   57  118   13
77   12   56   18   84
[39]   48   36   22   29   38   49   45  231  274   60

$language
 [1] 10  6  8  1  2  4  3  5  7  9

$religion
[1] 2 6 1 0 5 3 4 7

$bars
[1] 0 2 3 1 5

$stripes
 [1]  3  0  2  1  5  9 11 14  4  6 13  7

$colours
[1] 5 3 2 8 6 4 7 1

$red
[1] 1 0

$green
[1] 1 0

$blue
[1] 0 1

$gold
[1] 1 0

$white
[1] 1 0

$black
[1] 1 0

$orange
[1] 0 1

$mainhue
[1] green  red    blue   gold   white  orange black  brown
```

```
Levels: black blue brown gold green orange red white

$circles
[1] 0 1 4 2

$crosses
[1] 0 1 2

$saltires
[1] 0 1

$quarters
[1] 0 1 4

$sunstars
 [1]  1  0  6 22 14  3  4  5 15 10  7  2  9 50

$crescent
[1] 0 1

$triangle
[1] 0 1

$icon
[1] 1 0

$animate
[1] 0 1

$text
[1] 0 1

$topleft
[1] black  red    green  blue   white  orange gold  
Levels: black blue gold green orange red white

$botright
[1] green  red    white  black  blue   gold   orange brown 
Levels: black blue brown gold green orange red white


| You are amazing!

  |===========================================================================
=========         |  90%
| Occasionally, you may need to apply a function that is not yet defined, thu
s requiring you to write
| your own. Writing functions in R is beyond the scope of this lesson, but le
t's look at a quick
| example of how you might do so in the context of loop functions.

...

  |===========================================================================
===========       |  92%
| Pretend you are interested in only the second item from each element of the
unique_vals list that
| you just created. Since each element of the unique_vals list is a vector an
d we're not aware of any
| built-in function in R that returns the second element of a vector, we will
construct our own
| function.

...
```

```
  |==========================================================================
============      |  94%
| lapply(unique_vals, function(elem) elem[2]) will return a list containing t
he second item from each
| element of the unique_vals list. Note that our function takes one argument,
elem, which is just a
| 'dummy variable' that takes on the value of each element of unique_vals, in
turn.

> lapply(unique_vals, function(elem) elem[2])
$name
[1] Albania
194 Levels: Afghanistan Albania Algeria American-Samoa Andorra Angola Anguill
a ... Zimbabwe

$landmass
[1] 3

$zone
[1] 3

$area
[1] 29

$population
[1] 3

$language
[1] 6

$religion
[1] 6

$bars
[1] 2

$stripes
[1] 0

$colours
[1] 3

$red
[1] 0

$green
[1] 0

$blue
[1] 1

$gold
[1] 0

$white
[1] 0

$black
[1] 0

$orange
[1] 1
```

```
$mainhue
[1] red
Levels: black blue brown gold green orange red white

$circles
[1] 1

$crosses
[1] 1

$saltires
[1] 1

$quarters
[1] 1

$sunstars
[1] 0

$crescent
[1] 1

$triangle
[1] 1

$icon
[1] 0

$animate
[1] 1

$text
[1] 1

$topleft
[1] red
Levels: black blue gold green orange red white

$botright
[1] red
Levels: black blue brown gold green orange red white


| You are quite good my friend!

  |==========================================================================
==============    |  96%
| The only difference between previous examples and this one is that we are d
efining and using our
| own function right in the call to lapply(). Our function has no name and di
sappears as soon as
| lapply() is done using it. So-called 'anonymous functions' can be very usef
ul when one of R's
| built-in functions isn't an option.

...

  |==========================================================================
================  |  98%
| In this lesson, you learned how to use the powerful lapply() and sapply() f
unctions to apply an
| operation over the elements of a list. In the next lesson, we'll take a loo
k at some close
| relatives of lapply() and sapply().
```

```
...

  |==========================================================================================
==================| 100%
| Would you like to receive credit for completing this course on Coursera.org
?
```