# loops-after

Azul Carrillo

2022-11-23

#Exercise 1: The Data Set

##1. Read the UHURU tree data available for download in a tab delimited (“"̂)

```
tree_data <- read.csv("https://ndownloader.figshare.com/files/5629536",
                  sep = '\t',
                  na.strings = c("dead", "missing", "MISSING", "NA", "?", "3.3."))
```

##2. What is the code doing? Explain the meaning of each argument and how the values used for each argument afect the outcome. - the first line of code is reading the UHURU tree data from a link and creating an object named tree_data - then this is followed by sep = ‘̂, this means that the file is tab deliminated meaning that the data in different columns are separated with a ’tab’ character - finally the na.strings argument is saying that matching strings with the inputc (“dead”, “missing”, “MISSING”, “NA”, “?”, “3.3.”) words should be replaced with NA.

#Exercise 2: Tree Volumes

##1. Write a function called tree_volume_calc() that calculates the canopy volume for the Acacia species in the dataset. To do so, use an if statement in combination with the str_detect() function from the stringr R package. The code str_detect(SPECIES, “Acacia”) will return TRUE if the string stored in this variable contains the word “Acacia” and FALSE if it does not. This function will have to take the following arguments as input: SPECIES, HEIGHT, AXIS_1, AXIS_2. Then run the following line:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stringr)
tree_volume_calc <- function(SPECIES, HEIGHT, AXIS_1, AXIS_2){
   if (str_detect(SPECIES, "Acacia")) {
      volume <- 0.16 * HEIGHT^0.8 * pi * AXIS_1 * AXIS_2
   }
```

```
    return(volume)
}

tree_volume_calc("Acacia_brevispica", 2.2, 3.5, 1.12) # [1] 3.702491
```

```
## [1] 3.702491
```

##2. Expand this function to additionally calculate canopy volumes for other types of trees in this dataset by adding if/else statements and including the volume equations for the Balanites genus and other genera. Then run the following lines:

```
tree_volume_calc <- function(SPECIES, HEIGHT, AXIS_1, AXIS_2){
   if (str_detect(SPECIES, "Acacia")) {
  volume <- 0.16 * HEIGHT^0.8 * pi * AXIS_1 * AXIS_2
   } else if (str_detect(SPECIES,"Balanites")) {
    volume  <- 1.2 * HEIGHT^0.26 * pi * AXIS_1 * AXIS_2
 } else {
  volume = 0.5 * HEIGHT^0.6 * pi * AXIS_1 * AXIS_2
  }
    return(volume)
}
tree_volume_calc("Balanites", 2.2, 3.5, 1.12) # 18.14041
```

```
## [1] 18.14041
```

```
tree_volume_calc("Croton", 2.2, 3.5, 1.12)# 9.882335
```

```
## [1] 9.882335
```

##3. Now get the canopy volumes for all the trees in the tree_data dataframe and add them as a new column to the data frame. You can do this using tree_volume_calc() and either mapply() or using dplyr with rowwise and mutate.

```
colnames(tree_data)
```

```
##  [1] "SURVEY"       "YEAR"         "SITE"         "TREATMENT"    "BLOCK"
##  [6] "PLOT"         "SPECIES"      "ORIGINAL_TAG" "NEW_TAG"      "DEAD"
## [11] "HEIGHT"       "AXIS_1"       "AXIS_2"       "CIRC"         "MEASUREMENT"
## [16] "STEMS"
```

```
tree_volume <- tree_data %>%
  rowwise() %>%
 mutate(volumes = tree_volume_calc(SPECIES, HEIGHT, AXIS_1, AXIS_2))
tree_volume
```

```
## # A tibble: 7,508 x 17
## # Rowwise:
##    SURVEY  YEAR SITE  TREATMENT BLOCK PLOT  SPECIES ORIGI~1 NEW_TAG DEAD  HEIGHT
##     <int> <int> <chr> <chr>     <int> <chr> <chr>     <int>   <int> <chr>  <dbl>
##  1      1  2009 SOUTH TOTAL         2 S2TO~ Acacia~       1      NA N       3.4
```

```
##  2      2  2010 SOUTH TOTAL         2 S2TO~ Acacia~       1      NA N       3.32
##  3      3  2011 SOUTH TOTAL         2 S2TO~ Acacia~       1      NA N       3.65
##  4      4  2012 SOUTH TOTAL         2 S2TO~ Acacia~       1      NA N       3.74
##  5      5  2013 SOUTH TOTAL         2 S2TO~ Acacia~       1      NA N       3.59
##  6      1  2009 SOUTH TOTAL         2 S2TO~ Acacia~       2      NA N       2.3
##  7      2  2010 SOUTH TOTAL         2 S2TO~ Acacia~       2      NA N       2.32
##  8      3  2011 SOUTH TOTAL         2 S2TO~ Acacia~       2      NA N       2.75
##  9      4  2012 SOUTH TOTAL         2 S2TO~ Acacia~       2      NA Y      NA
## 10      5  2013 SOUTH TOTAL         2 S2TO~ Acacia~       2      NA N       2.86
## # ... with 7,498 more rows, 6 more variables: AXIS_1 <dbl>, AXIS_2 <dbl>,
## #   CIRC <dbl>, MEASUREMENT <chr>, STEMS <chr>, volumes <dbl>, and abbreviated
## #   variable name 1: ORIGINAL_TAG
```

#Exercise 3: Tree growth

##1. Write a function named get_growth() that takes two inputs, a vector of sizes and a vector of years, and calculates the average annual growth rate. Pseudo-code for calculating this rate is (size_in_last_year - size_in_first_year) / (last_year - first_year). Test this function by running get_growth(c(40.2, 42.6, 46.0), c(2020, 2021, 2022))

```
library(dplyr)
1 == 3
```

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

```
years <- c(1887, 1950, 2022, 1983)
years == 2022
```

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

```
max(years)
```

```
## [1] 2022
```

```
min(years)
```

```
## [1] 1887
```

```
i <- years == min(years)
i
```

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

```
get_growth<- function(sizes , years){
  size_in_first_year <- sizes[years == min(years)]
  size_in_last_year <- sizes[years == max(years)]
  last_year <- max(years)
  first_year <- min(years)
  growth <- (size_in_last_year - size_in_first_year) / (last_year - first_year)
  return(growth)
}

get_growth(c(40.2, 42.6, 46.0), c(2020, 2021, 2022)) #2.9
```

```
## [1] 2.9
```

##2. Use dplyr and the function you created above to get the growth for each individual tree along with information about the TREATMENT for that tree. Trees are identified by a unique value in the ORIGINAL_TAG column. Don't include information for cases where a TREATMENT is not known (e.g., where it is NA).

```
library(dplyr)

tree_growth <- tree_volume %>%
  group_by(ORIGINAL_TAG) %>%
  filter(!is.na(TREATMENT)) %>%
  filter(TREATMENT != "" ) %>%
  summarise(growth = get_growth(volumes, YEAR), treatment = unique(TREATMENT))
```

```
## `summarise()` has grouped output by 'ORIGINAL_TAG'. You can override using the
## `.groups` argument.
```

```
tree_volume$TREATMENT[519]
```

```
## [1] ""
```

```
tree_growth
```

```
## # A tibble: 1,529 x 3
## # Groups:   ORIGINAL_TAG [1,528]
##    ORIGINAL_TAG growth treatment
##           <int>  <dbl> <chr>
##  1            1   4.03 TOTAL
##  2            2   1.29 TOTAL
##  3            3  10.1  TOTAL
##  4            4   6.37 TOTAL
##  5            5   5.42 TOTAL
##  6            6   3.29 TOTAL
##  7            7   5.18 TOTAL
##  8            8   2.51 TOTAL
##  9            9   6.14 TOTAL
## 10           10  12.4  TOTAL
## # ... with 1,519 more rows
```

##3. Using ggplot and the output from (2) make a histogram of growth rates for each TREATMENT, whith each TREATMENT in it's own facet. Use geom_vline() to add a vertical line at 0 to help indicate which trees are getting bigger vs. smaller. Include good axis labels.

```
library(ggplot2)
tree_growth %>%
  ggplot() +
  geom_histogram(mapping = aes(x = growth)) +
  geom_vline(xintercept = 0 ) +
  facet_wrap(~treatment, scales = "free_x")+
  labs(x = "Anual Growth Rate", y = "Number of individuals", title = " Anual growth from Tree Data")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 112 rows containing non-finite values (`stat_bin()`).
```