The basics: 06 functions

Ari Anisfeld

9/8/2020

Questions

Writing functions

Recall a function has the following form

```
name <- function(args) {
  # body
  do something (probably with args)
}
```

1. Write a function called `calc_quadratic` that takes an input `x` and calculates $f(x) = x^2 + 2x + 1$. For example:

   ```
   calc_quadratic(5)
   ```

   ```
   ## [1] 36
   ```

   a. What are the arguments to your function? What is the body of the function?
   b. This function is vectorized! (Since binary operators are vectorized). Show this is true by running `calc_quadratic` with an input vector that is -10 to 10.

2. You realize you want to be able to work with any quadratic. Update your functions so that it can work with any quadratic in standard form $f(x) = ax^2 + bx + c$.

   - Your new function will take arguments `x`, `a`, `b` and `c`.
   - Set the default arguments to `a=1`, `b=2` and `c=1`

3. Write a function called `solve_quadratic` that takes arguments `a`, `b` and `c` and provides the two roots using the quadratic formula.

   In our outline, we suggest you:

   - Calculate the determinant ($\sqrt{b^2 - 4ac}$) and store as an intermediate value.
   - Return two values by putting them in a vector. If you stored the roots as `root_1` and `root_2`, then the final line of code in the function should be `c(root_1, root_2)` or, if you prefer, `return(c(root_1, root_2))`.

   ```
   # fill in the ... with appropriate code
   solve_quadratic <- function(...){

     determinant <- ...
     root_1 <- ...
     root_2 <- ...

     c(root_1, root_2)
   ```

```
}
```

The code should work as follows:

```
solve_quadratic(a = -4, b = 0, c = 1)
```

```
## [1] -0.5  0.5
```

4. We "normalize" a variable by subtracting the mean and dividing by the standard deviation $\frac{x-\mu}{\sigma}$. Write a function called `normalize` that takes a vector as input and normalizes it.

   You should get the following output.

```
normalize(1:5)
```

```
## [1] -1.2649111 -0.6324555  0.0000000  0.6324555  1.2649111
```

   a. What output do you get when the input vector is `0:4`? How about `-100:-96`? Why?

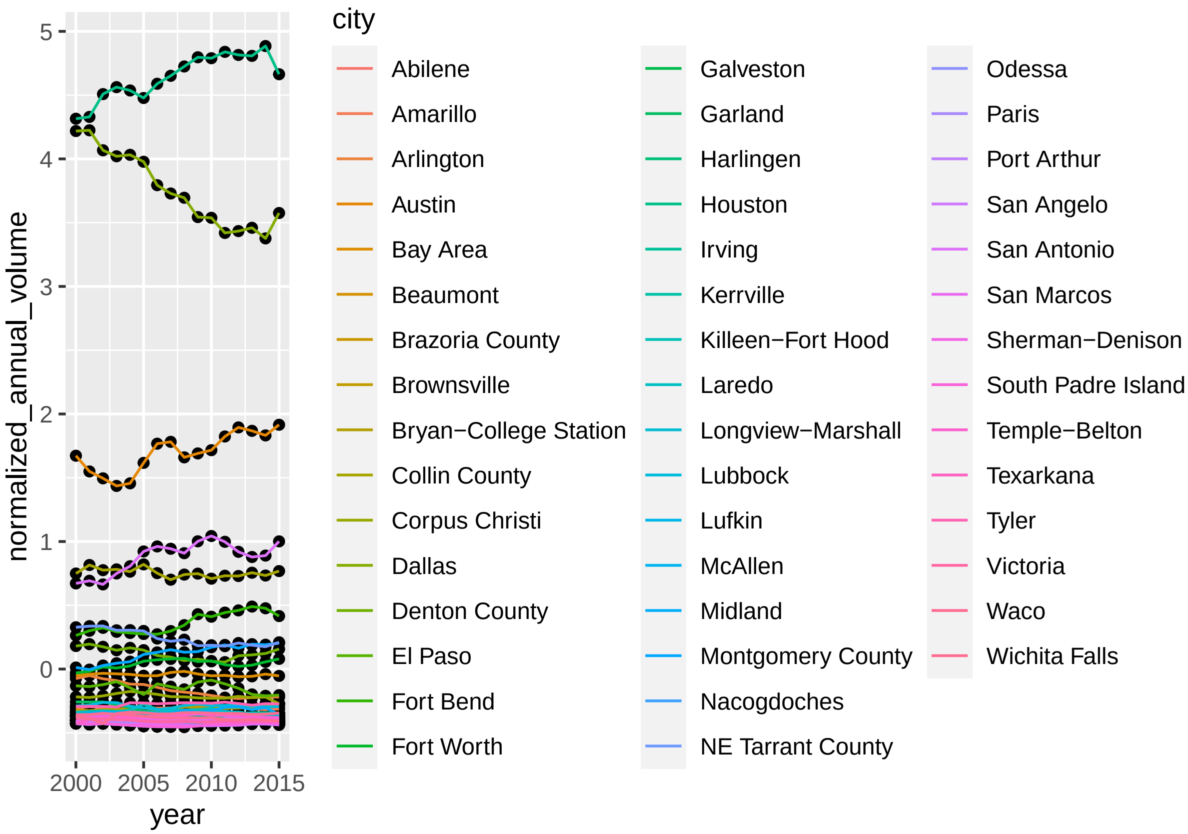
   b. What happens when your input vector is c(1,2,3,4,5, NA)? Rewrite the function so the result is:[1]

```
## [1] -1.2649111 -0.6324555  0.0000000  0.6324555  1.2649111         NA
```

   c. The `txhousing` data set is comes with `ggplot`. Use your normalize function in `mutate` to create `normalized_annual_volume` to make the following graph.

```
# replace the ... with the appropriate code.
txhousing %>%
  group_by(year, city) %>%
  summarize(annual_volume = sum(volume, na.rm = TRUE)) %>%
  group_by(year) %>%
  mutate(...) %>%
  ggplot(aes(x = year, y = normalized_annual_volume)) +
  geom_point() +
  geom_line(aes(color = city))
```

[1] Hint: take advantage of `mean` and `sd` NA handling.



Want to improve this tutorial? Report any suggestions/bugs/improvements on here! We're interested in learning from you how we can make this tutorial better.

Solutions

Writing functions

1. Write a function called `calc_quadratic` that takes an input `x` and calculates $f(x) = x^2 + 2x + 1$. For example:

```
calc_quadratic <- function(x) {
  x ^ 2 + 2 * x + 1
}
```

   a. What are the arguments to your function? What is the body of the function?

   arguments are `x`; the body is `x ^ 2 + 2 * x + 1`

   a. This function is vectorized! (Since binary operators are vectorized). Show this is true by running `calc_quadratic` with an input vector that is -10 to 10.

```
calc_quadratic(-10:10)
```

```
##  [1]  81  64  49  36  25  16   9   4   1   0   1   4   9  16  25  36  49  64  81
## [20] 100 121
```

2. You realize you want to be able to work with any quadratic. Update your functions so that it can work with any quadratic in standard form $f(x) = ax^2 + bx + c$.

- Your new function will take arguments `x`, `a`, `b` and `c`.
- Set the default arguments to `a = 1`, `b = 2` and `c = 1`

```
calc_quadratic <- function(x, a = 1, b = 2, c = 1) {
 a * x ^ 2 + b * x + c
}

calc_quadratic(5)
```

```
## [1] 36
```

3. Write a function called `solve_quadratic` that takes arguments `a`, `b` and `c` and provides the two roots using the quadratic formula.

```
solve_quadratic <- function(a, b, c){

  determinant <- sqrt(b ^ 2 - 4 * a * c)
  root_1 <- (-b + determinant) / (2 * a)
  root_2 <- (-b - determinant) / (2 * a)

  c(root_1, root_2)

  }
```

The code should work as follows:

```
solve_quadratic(a = -4, b = 0, c = 1)
```

```
## [1] -0.5  0.5
```

Notice, the code doesn't deal with functions with no roots. It returns `NaN`. If there is a single root (such as when a = 1, b = 0 and c = 0), it returns the same number twice. We could use `if()` statments in the function to have it explicitly deal with these issues.

4. We "normalize" a variable by subtracting the mean and dividing by the standard deviation $\frac{x-\mu}{\sigma}$. Write a function called `normalize` that takes a vector as input and normalizes it.

You should get the following output.

```
normalize(1:5)
```

```
## [1] -1.2649111 -0.6324555  0.0000000  0.6324555  1.2649111
```

a. What output do you get when the input vector is `0:4`? How about `-100:-96`? Why?

You get the same results as `1:5`. This is because when you demean all the vectors are identical.

a. What happens when your input vector is c(1,2,3,4,5, NA)? Rewrite the function so the result is:[2]

see above

a. The `txhousing` data set is comes with `ggplot`. Use your normalize function in `mutate` to create `normalized_annual_volume` to make the following graph.

```
txhousing %>%
  group_by(year, city) %>%
  summarize(annual_volume = sum(volume, na.rm = TRUE)) %>%
  group_by(year) %>%
  mutate(normalized_annual_volume = normalize(annual_volume)) %>%
```

[2] Hint: take advantage of `mean` and `sd` NA handling.

```r
  ggplot(aes(x = year, y = normalized_annual_volume)) +
  geom_point() +
  geom_line(aes(color = city))
```