The basics: 02 Vectors and data types

Ari Anisfeld

9/8/2020

Questions

Create vectors

1. In the lecture, we covered `c()`, `:`, `rep()`, `seq()` among other ways to create vectors.

```
dolly <- c(9, 10, 11, 12, 13, 14, 15, 16, 17)
bees <- c("b", "b", "b", "b", "b")
```

- Recreate `dolly` using `:`.
- Create the same vector using `seq()`.
- Recreate `bees` using `rep()`.

Random vectors

1. In the lecture, we also created vectors using `rnorm()` and `runif()`.

```
random_norm <- rnorm(100)
random_unif <- runif(1000)
```

   - How long are the vectors `random_norm` and `random_unif`? Use `length()` to verify.
   - What are the largest and smallest values in `random_norm` and `random_unif`? Use `min()` and `max()`.
   - Use `mean()` and `sd()` to calculate the mean and standard deviation of the two distributions.
   - Create a new vector with `10000` draws from the standard normal distribution.
   - `rnorm()` by default sets `mean = 0` (see `?rnorm`). Create a vector of `10000` draws from the normal distribution with `mean = 1`. Use `mean()` to verify.

Notice the functions `min()`, `max()`, `mean()` and `sd()` all take a vector with many values and summarize them as one value. These are good to use with `summarize()` when doing data analysis on tibbles.

data types

- Use `typeof()` to verify the data types of `dolly`, `bees`, `random_unif`
- Coerce `dolly` to a character vector. Recall we have functions `as.<type>()` for this kind of coercion.
- Try to coerce `bees` to type `numeric`. What does R do when you ask it to turn "b" into a number?

vectorized math

`a` and `b` are vectors of length 10. Look at them in the console.

```
a <- 1:10
b <- rep(c(2, 4), 5)
```

1. Add `a` and `b` element by element.
2. Subtract `a` and `b` element by element.
3. Divide `a` by `b` element by element.
4. Multiply `a` and `b` element by element.
5. Raise the element of `a` to the power of `b` element by element.
6. Multiply each element of `a` by 3 then subtract `b`
7. Raise each element of `b` to the third power.
8. Take the square root of each element of `a`.

vectorized comparison

1. Run the following code and make sure you understand the output.

```
a > b
```

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

```
a == b
```

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

creating tibbles with vectors

1. Create a tibble with columns called `a` and `b` where `a` is the numbers 1 to 100 and `b` is 100 random numbers from the standard normal distribution.

```
my_tibble<-
tibble(
  a = ...,
  b = ...
)
```

subsetting

`midwest` is a data set that comes with `ggplot`.

```
# try this to see midwest data
library(tidyverse)
midwest %>% head()
```

1. Use `pull()` to get the vector of `state` names.
2. Use `[[` to get a vector of `state` names. This is baseR and requires (normal) quotes around column names.
3. Use `select()` to get a tibble with `state` as the only column.
4. Use `[` to get a tibble with `state` as the only column

Solutions

Create vectors

```
dolly_colon <- 9:17
dolly_seq <- seq(9:17)
bees_rep <- rep("b", 5)
```

Random vectors

```
# lengths
length(random_norm)
```

```
## [1] 100
```

```
length(random_unif)
```

```
## [1] 1000
```

```
# largest and smallest values (repeat with random_unif)
max(random_norm)
```

```
## [1] 2.19117
```

```
min(random_norm)
```

```
## [1] -2.073629
```

```
# mean and sd  (repeat with random_unif)
mean(random_norm)
```

```
## [1] 0.05655819
```

```
sd(random_norm)
```

```
## [1] 1.027318
```

```
# rnorm with length 10000
longer_rnorm <- rnorm(10000)

# mean = 1
rnorm_centered_on_one <- rnorm(10000, mean = 1)
```

typeof

```
typeof(dolly)
```

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

```
typeof(bees)
```

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

```
typeof(random_unif)
```

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

```
# notice dolly is int if it's created by : or seq
typeof(dolly_seq)
```

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

```
typeof(dolly_colon)
```

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

```
# coercion
as.character(dolly)
```

```
## [1] "9"  "10" "11" "12" "13" "14" "15" "16" "17"
```

```
# R coerces "b" to NA because there is
# not a natural number to replace "b" with
as.numeric(bees)
```

```
## Warning: NAs introduced by coercion
```

```
## [1] NA NA NA NA NA
```

vectorized math

```
a + b
```

```
##  [1]  3  6  5  8  7 10  9 12 11 14
```

```
a - b
```

```
##  [1] -1 -2  1  0  3  2  5  4  7  6
```

```
a / b
```

```
##  [1] 0.5 0.5 1.5 1.0 2.5 1.5 3.5 2.0 4.5 2.5
```

```
a * b
```

```
##  [1]  2  8  6 16 10 24 14 32 18 40
```

```
a ^ b
```

```
##  [1]     1    16     9   256    25  1296    49  4096    81 10000
```

```
2 * a - b
```

```
##  [1]  0  0  4  4  8  8 12 12 16 16
```

```
b ^ 3
```

```
##  [1]  8 64  8 64  8 64  8 64  8 64
```

```
sqrt(a)
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
##  [9] 3.000000 3.162278
```

Creating tibbles

```
my_tibble<-
tibble(
```

```
  a = seq(1, 100),
  b = rnorm(100)
)
```

```
midwest %>% pull(state)
```

```
##   [1] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [16] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [31] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [46] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [61] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [76] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [91] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IN" "IN"
## [106] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [121] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [136] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [151] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [166] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [181] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "MI"
## [196] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [211] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [226] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [241] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [256] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [271] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [286] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [301] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [316] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [331] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [346] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [361] "OH" "OH" "OH" "OH" "OH" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [376] "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [391] "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [406] "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [421] "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [436] "WI" "WI"
```

```
midwest[["state"]]
```

```
##   [1] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [16] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [31] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [46] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [61] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [76] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL"
##  [91] "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IL" "IN" "IN"
## [106] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [121] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [136] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [151] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [166] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN"
## [181] "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "IN" "MI"
## [196] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [211] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [226] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
```

```
## [241] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [256] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI" "MI"
## [271] "MI" "MI" "MI" "MI" "MI" "MI" "MI" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [286] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [301] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [316] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [331] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [346] "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH" "OH"
## [361] "OH" "OH" "OH" "OH" "OH" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [376] "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [391] "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [406] "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [421] "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI" "WI"
## [436] "WI" "WI"
```

```
midwest %>% select(state)
```

```
## # A tibble: 437 x 1
##    state
##    <chr>
##  1 IL
##  2 IL
##  3 IL
##  4 IL
##  5 IL
##  6 IL
##  7 IL
##  8 IL
##  9 IL
## 10 IL
## # ... with 427 more rows
```

```
midwest[,"state"]
```

```
## # A tibble: 437 x 1
##    state
##    <chr>
##  1 IL
##  2 IL
##  3 IL
##  4 IL
##  5 IL
##  6 IL
##  7 IL
##  8 IL
##  9 IL
## 10 IL
## # ... with 427 more rows
```

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.