Programming Fundamentals 3

Rupinder Sandhu

April 10, 2017

Choice operators

```
w <- 10.2
x <- 1.3
y <- 2.8
z <- 17.5
dna1 <- "attattaggaccaca"
dna2 <- "attattaggaacaca"

w > 10
```

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

```
w + x < 15
```

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

```
x > y
```

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

```
2*x + 0.2 == y
```

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

```
dna1 == dna2
```

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

```
dna1 != dna2
```

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

```
str_count(dna1, "t") == str_count(dna2, "t")
```

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

```
w > x & y> z
```

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

```
(x * w > 13.2) & (x * w < 13.5)
```

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

```
str_count(dna1) > 5
```

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

```
str_length(dna1) + str_length(dna2) >= 30
```

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

```
(w + x + y)/ log10(100) == 7.15
```

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

```
gc_content1 <- (str_count(dna1, "g") + str_count(dna1, "c"))/str_length(dna1)*100
gc_content2 <- (str_count(dna2, "g") + str_count(dna2, "c"))/str_length(dna2)*100
gc_content1 != gc_content2
```

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

Check if two geographic points are near each other

```
#1.
near <- function(lat1, long1, lat2, long2){
  # Check if two geographic points are near each other
  if ((abs(lat1 - lat2) < 1) & (abs(long1 - long2) < 1)){
    near <- TRUE}
  else {
    near <- FALSE
  }
  return(near)
}

#2.
Pt1_closeto_Pt2 <- function(lat1, long1, lat2, long2){
  # Check if two geographic points are near each other
  if ((abs(lat1 - lat2) < 1) & (abs(long1 - long2) < 1)){
    Pt1_closeto_Pt2 <- TRUE}
  else {
    Pt1_near_Pt2 <- FALSE
  }
  return(Pt1_closeto_Pt2)
}

#3.
Pt1_closeto_Pt2 (29.65, -82.33, 41.74, -111.83)
```

```
## function(lat1, long1, lat2, long2){
##   # Check if two geographic points are near each other
##   if ((abs(lat1 - lat2) < 1) & (abs(long1 - long2) < 1)){
##     Pt1_closeto_Pt2 <- TRUE}
##   else {
##     Pt1_near_Pt2 <- FALSE
##   }
##   return(Pt1_closeto_Pt2)
## }
```

```
#4.
Pt1_closeto_Pt2 (29.65, -82.33, 30.5, -82.8)
```

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

```
#5,6
Pt1_nearby_Pt2_default <- function(lat1, long1, lat2, long2, nearvalue=1){
  # Check if two geographic points are near each other
  if ((abs(lat1 - lat2) < nearvalue) & (abs(long1 - long2) < nearvalue)){
    Pt1_nearby_Pt2_default  <- TRUE}
  else {
    Pt1_nearby_Pt2_default  <- FALSE
  }
  return(Pt1_nearby_Pt2_default )
}
#7.
Pt1_nearby_Pt2 <- function(lat1, long1, lat2, long2, nearvalue){
  # Check if two geographic points are near each other
  if ((abs(lat1 - lat2) < nearvalue) & (abs(long1 - long2) < nearvalue)){
    Pt1_nearby_Pt2 <- TRUE}
  else {
    Pt1_nearby_Pt2 <- FALSE
  }
  return(Pt1_nearby_Pt2)
}
Pt1_nearby_Pt2 (48.86, 2.35, 41.89, 2.5, 7)
```

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

Choice with functions

```
UHURU <- read.csv("ACACIA_DREPANOLOBIUM_SURVEY.txt", sep = "\t")
report_rsquared <- function(data, species, formula){
  subset <- dplyr::filter(data, ANT == species)
  test <- lm(formula, data = subset)
  rsquared <- round(summary(test)$r.squared, 3)
  output <- data.frame(species = species, r2 = rsquared)
  return(output)
}
#1.
report_rsquared (UHURU, "CM", AXIS1~CIRC)
```

```
##   species    r2
## 1      CM 0.866
```

```
#2.
report_rsquared_significance <- function(data, species, formula, threshold){
  subset <- dplyr::filter(data, ANT == species)
  test <- lm(formula, data = subset)
  rsquared <- round(summary(test)$r.squared, 3)
  if(rsquared > threshold)
    significance <- "s"
  else {significance <- "NS"}
    output <- data.frame(species = species, r2 = rsquared, significance = significance)
  return(output)
}
#3.
report_rsquared_significance(UHURU, "CM", AXIS1~CIRC, 0.667)
```

```
##   species    r2 significance
## 1      CM 0.866            s
```

```
report_rsquared_significance(UHURU, "CS", AXIS1~CIRC, 0.667)
```

```
##   species    r2 significance
## 1      CS 0.437           NS
```

```
report_rsquared_significance(UHURU, "TP", AXIS1~CIRC, 0.667)
```

```
##   species    r2 significance
## 1      TP 0.701            s
```