STSCI 4520 Lab 4

Nick Gembs

2/14/2023

Lab 04 - printing output to console

Our objective in this lab is to reproduce the output produced by printing the summary object of an lm object.

```
dat = read.csv("C:/Users/Nick/Documents/GitHub/statcomp2023/datasets/voice_heights.csv")
m1 <- lm( height ~ voice, data = dat )
s1 <- summary(m1)
```

This will teach you more about the information stored in the lm object and in the summary object of the lm object, as well as about sprintf.

sprintf

Read the documentation for sprintf and answer the following questions.

1. The formatting syntax uses %, followed by optional specifications, and ending with a single character. How many different ending characters are supported?

```
nchar("aAdifeEgGosxX%")
```

```
## [1] 14
```

14 endings are supported.

2. Which format most closely resembles how R prints double precision values in the console?

```
sprintf("Test %1.1f", 3.4)
```

```
## [1] "Test 3.4"
```

```
sprintf("Test %1.0f", 3)
```

```
## [1] "Test 3"
```

%f most closely resembles R's double precision printing with setting (# of characters used).(# of digits after decimal)

3. Create a character vector containing two spaces and then π to 18 decimal places. Is it correct?

```
sprintf("%22.18f", pi)
```

```
## [1] "  3.141592653589793116"
```

It is not correct, pi to 18 decimal places is 3.141592653589793238. The last 3 decimal places are incorrect.

4. Create a character vector containing the codes "001", "002", "003", . . . , "099", "100"

```
x <- 1:100

sprintf("%03.0f", x)
```

```
##   [1] "001" "002" "003" "004" "005" "006" "007" "008" "009" "010" "011" "012"
##  [13] "013" "014" "015" "016" "017" "018" "019" "020" "021" "022" "023" "024"
##  [25] "025" "026" "027" "028" "029" "030" "031" "032" "033" "034" "035" "036"
##  [37] "037" "038" "039" "040" "041" "042" "043" "044" "045" "046" "047" "048"
##  [49] "049" "050" "051" "052" "053" "054" "055" "056" "057" "058" "059" "060"
##  [61] "061" "062" "063" "064" "065" "066" "067" "068" "069" "070" "071" "072"
##  [73] "073" "074" "075" "076" "077" "078" "079" "080" "081" "082" "083" "084"
##  [85] "085" "086" "087" "088" "089" "090" "091" "092" "093" "094" "095" "096"
##  [97] "097" "098" "099" "100"
```

lm and summary.lm objects

Both `m1` and `s1` (defined in code above) are "list-like" objects, meaning that they have several named fields that can be accessed in the same way any list's elements can be accessed. Familiarize yourself with the contents of `m1` and `s1` by printing each object's elements to the console (you don't need to include that output here).

The lab exercise will go more smoothly if you spend some time here inspecting the contents of these objects.

Reproduce `print(s1)`

```
{
cat("Call:\n")
cat(sprintf("%s", s1[1]),
    '\n\n')

cat("Residuals:\n")
cat("     Min      1Q  Median      3Q     Max \n")

minres = min(s1$residuals)
q1res = quantile(s1$residuals, .25)
medres = median(s1$residuals)
q3res = quantile(s1$residuals, .75)
maxres = max(s1$residuals)

for (i in c(minres,q1res,medres,q3res,maxres)){
  cat(sprintf("%8.4f", i))
```

```
}

cat("\n\nCoefficients:\n             Estimate Std. Error t value Pr(>|t|) \n")


coefficients = s1$coefficients
lencol1 = max(nchar((variable.names(m1))))
index = 1
index1 = 1


for (elem in variable.names(m1)){

  index = index1

  spaces = (lencol1) - nchar(elem)
  spaces = rep(" ", spaces)
  cat(elem, spaces, sep = "")

  for (i in 1:4) {

    if (index > length(coefficients)*.75) {

      if (coefficients[index] <= 2* 10 ^ (-16)) {
        cat("  < 2e-16")
      } else if ( coefficients[index] < .001){
        cat(sprintf("%9.2e", coefficients[index]))
      } else {
        cat(sprintf("%9.3f", coefficients[index]))
      }

      if(coefficients[index] <= .001) {
        cat(" ***")
      } else if (coefficients[index] <= .01) {
        cat(" **")
      } else if (coefficients[index] <= .05) {
        cat(" *")
      } else if (coefficients[index] <= .1) {
        cat(" .")
      }


    } else if (index > length(coefficients)*.5) {
      cat(sprintf("%8.3f", coefficients[index])) }

   else if (index > length(coefficients)*.25) {
      cat(sprintf("%11.4f", coefficients[index])) }

    else {
      cat(sprintf("%9.4f", coefficients[index]))
    }

    index = index + 4
```

```
  }


  index1 = index1+1
  cat("\n")
}

cat("---\nSignif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1\n\n")

cat("Residual standard error: ")
cat(sprintf("%5.3f", s1$sigma), "on" , sprintf("%d", s1$df[2]),
    "degrees of freedom\n")
cat("Multiple R-squared: ")
cat(sprintf("%7.4f,", s1$r.squared),"   Adjusted R-squared:",
    sprintf("%7.4f", s1$adj.r.squared), "\n")
cat("F-statistic: ")

pval = pf(s1$fstatistic[1],s1$fstatistic[2],s1$fstatistic[3],lower.tail=FALSE)


cat(sprintf("%4.1f", s1$fstatistic[1]), "on", sprintf("%d", s1$fstatistic[2]),
    "and", sprintf("%d", s1$fstatistic[3]),
    "DF,  p-value: ")

if (pval <= 2* 10 ^ (-16)) {
  cat("< 2.2e-16") } else if (pval <= .001) {
    cat(sprintf("%9.2e", pval))
  } else {
      cat(sprintf("%9.3f", pval))
    }

}
```

```
## Call:
## lm(formula = height ~ voice, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
##  -5.1500 -1.8857  0.2821  1.2821  7.1143
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   64.8857     0.4250 152.655  < 2e-16 ***
## voicebass      5.8322     0.5855   9.961  < 2e-16 ***
## voicesoprano  -0.6357     0.5969  -1.065    0.289
## voicetenor     4.2643     0.7049   6.050 1.54e-08 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 2.515 on 126 degrees of freedom
## Multiple R-squared:  0.5706,    Adjusted R-squared:  0.5603
## F-statistic: 55.8 on 3 and 126 DF,  p-value: < 2.2e-16
```