Volpe R Course: Session 2

Instructors: Don Fisher, Dan Flynn, Jessie Yang
Course webpage: `http://bit.ly/volpeR`

9/13/2017

Review

Last session we covered essential material on

- Basic R operations
- Summarizing data; writing functions
- Getting data in
- Indexing

Today we will review these steps, and introduce two powerful features of R, loops and functions. In addition, we will introduce some basic data analysis in R.

Homework: Essential steps

Working in R

- Use a script!
- Save your work, make comments
- Try new things
  - Look at the Resources document in the course shared drive for ideas
  - The more different things you try, the more useful feedback we can provide
- Spaces around <-, =, and ~, please!
- There are many ways to solve problems in R

Homework: Common pitfalls

- Review how to set the working directory:

```
setwd("H:/R")
cars <- read.csv("Motor Trend Car Road Tests.csv")
```

- Pay attention to capitalization, spaces, and other details

```
hist(cars$mpg)
hist(cars$MPG)
```

- Know what the variables are that you want to work with:

```
names(cars)
summary(cars)
head(cars)
View(cars)
```

Essential concepts in R: functions and loops

Functions are sets of commands that you bundle together to perform a specific task. Often you write a function to perform a set of commands that you know you will have to carry out repeatedly, and want to make R do all the repetition for you.

Loops are ways of repeatedly doing the same set of commands over all elements of an object. For instance, you can carry out some processing of your data across all rows in a data frame, or make the same type of plot for multiple subsets of your data.

Both functions and loops are important tools in R, and can be very handy when dealing with a large data set or a complex set of operations. Sometimes it is not worth writing a function or loop, it depends on how many times you might repeat a task!

Functions:

Let's use some hypothetical data to illustrate how to write a function in R, and by doing so also review basic statistics. These steps also review how to operate on vectors in R.

Make up some data:

Or generate random values from a Normal distribution:

Firsts, try summing the using `sum(x)`. Ok, now try `mean(x)`. Another: `length(x)` tells you how many numbers there are in the vector `x`; you can also think of this as the n number of samples in your data set.

Now, let's combine them to start doing some statistics. Remember that the sum of squares an important value in statistics, which is the sum of the squared differences between each data point and the mean. In symbols:

$SS = \sum(x - \overline{x})^2$

Functions: DIY Statistics

We can make R do this by typing a series of commands nested inside each other, adding the parentheses to keep things straight. First, the difference between each x and the mean value for all x's:

```
x - mean(x)
```

Now square it:

```
(x - mean(x))^2
```

And sum them all:

```
sum((x - mean(x))^2)
```

Functions: DIY Statistics

Great. Now what can we do with this number? We might want to know the variance of the data, which is the sum of squares divided by n-1:

$$s^2 = \frac{\sum (x - \overline{x})^2}{n-1}$$

We can tell R to do this by adding one more bit to our last command (hint: use the up arrow to restore the last command without having to type it in):

```
(sum((x-mean(x))^2))/(length(x)-1)
```

What if we want to calculate this for a lot of variables, and don't enjoy cutting and pasting all that much? We can write a function to make our lives easier. It's much easier to do this in an Editor window than in the R console. Go to File > New File > R Script.

Functions: DIY Statistics

In the new window that opens up, type this:

```
variance <- function(x){
    sum(((x-mean(x))^2)/(length(x)-1))
}
```

And test it out:

```
variance(x)
```

Congratulations. You just wrote your first R function. In fact, your work has already been done by others: R has a built-in function to calculate variances, sensibly called `var()`. Check `var(x)` and `var(y)` to see that your function works perfectly.

Functions: DIY Statistics

What if you are already confused about what all variables you have to work with? Look at the list of objects in your workspace using `ls()` (without typing anything in the parentheses). You'll see `variance`, `x`, `y`, and `example` in there. Remove `variance` by the command

```
rm(variance)
```

Functions: Picking random colors

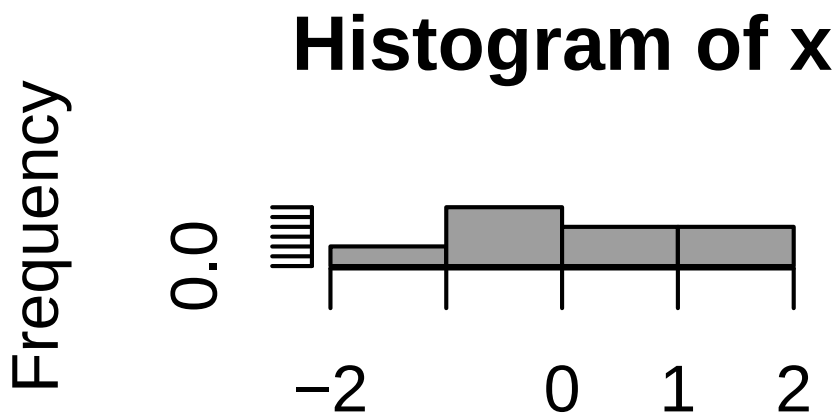
Here is another kind of function, which returns random colors for plotting:

```
pickacolor <- function(x = 1){
  col <- sample(colors(), size = x)
  col
  }
```

Now use this function to make some plots:

```
pickacolor(1)
pickacolor()
pickacolor(3)
```

```
hist(x, col = pickacolor())
```



Functions: Exercise

Write a function which calculates the mean and median of a vector of data!

Loops

Like functions, loops are powerful ways to take advantage of the fact that R is a scripting language. Let R handle any repetitious tasks for you!

The basic idea is to write a line of R code with one object replaced by a 'wildcard', often the character *i*.

Here are two loops:

```
for(i in 1:10){
  print(LETTERS[i])
  }
```

```
container <- vector()
for(indexvalue in 1:10){
  container <- c(container, LETTERS[rnorm(1, mean = 13, sd = 4)])
  }
container
```

```
##  [1] "K" "I" "F" "K" "Q" "N" "F" "P" "K" "K"
```

Exercise 2.1

Make a loop which selects 5 random colors and create a plot using these colors!

T-test: compare means of two groups

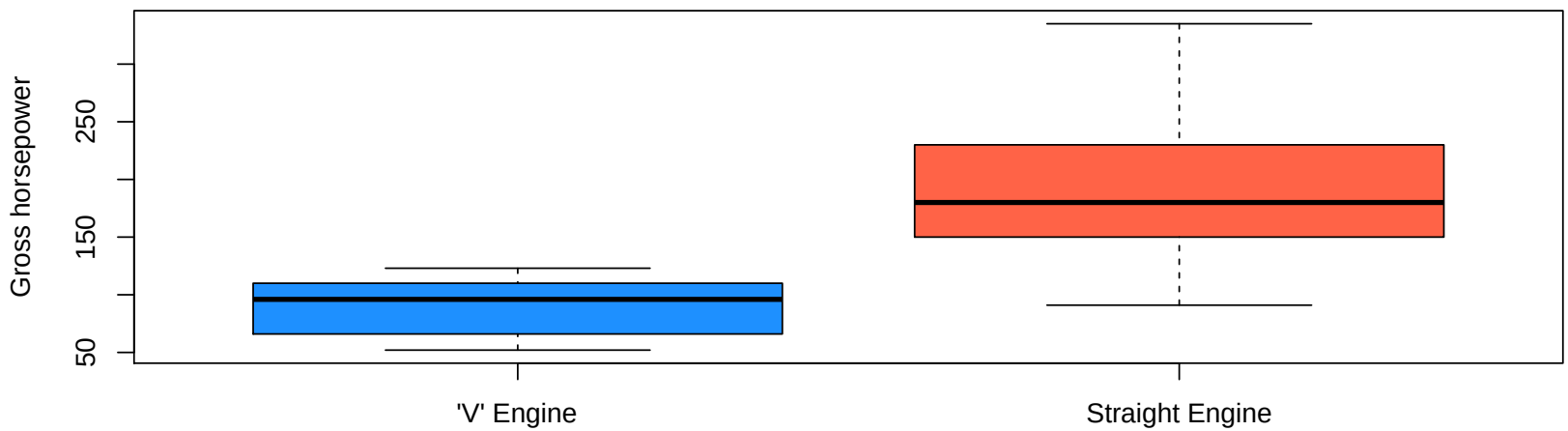
Comparing two groups is done in R with the function `t.test`. This tests if two groups have the same mean value. Here we are looking at horsepower for cars with a "V" engine (`vs == 1`) or or a straight engine (`vs == 0`):

```
with(cars, t.test(hp[vs=="1"], hp[vs=="0"]))
```

```
##
##  Welch Two Sample t-test
##
## data:  hp[vs == "1"] and hp[vs == "0"]
## t = -6.2908, df = 23.561, p-value = 1.82e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -130.66854  -66.06161
## sample estimates:
## mean of x mean of y
##  91.35714 189.72222
```

T-test: compare means of two groups

```
with(cars, boxplot(hp[vs=="1"], hp[vs=="0"],
                   names = c("'V' Engine",
                             "Straight Engine"),
                   col = c("dodgerblue", "tomato"),
                   ylab = "Gross horsepower"
                   )
     )
```



Analysis of Variance (ANOVA): compare means of more than two groups

Analysis of variance is used for comparing means of more than two groups. Analysis of covariance (ANCOVA) combines features of ANOVA and regression. In fact, all of these models are related, in that they are linear models. A *generalized linear model* is the broadest category of these models.

```
m1 <- aov(hp ~ cyl, data = cars)
summary(m1)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## cyl           1 100984  100984   67.71 3.48e-09 ***
## Residuals    30  44743    1491
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Homework

Take your dataset you used for Homework 1, or another dataset if you decide to change, and try to do the following:

1. Write a loop to perform some task on your data.
2. Write a function to perform some task on your data.

Document your work in a script as before, and upload to the Homework 2 folder on the course Google Drive.

The more different things you try, the more feedback you will get from us!