Introduction to R

There are many software programs designed to store, summarize, visualize, or analyze data:

* Spreadsheets: Microsoft Excel, Google Sheets, etc.
* Data Analysis and Statistical programs: SAS, SPSS, Minitab
* Programming languages: Python, C, C++, Fortran, Java
* Database software: Microsoft Access, SAP, SQL databases

R is statistical software which can read data, manipulate data, summarize data, fit statistical models, create quality plots, and more. It is used by companies like Google, Twitter, and Microsoft, as well as in government agencies and at national labs.

# Setting Up

This guide will help set up R and RStudio on your machine. It is included for reference.

## Installing R:

On Windows or OS X:

* Go to <http://www.r-project.org/>
* Click the CRAN link on the left, and pick a download site (0-Cloud is a good choice)
* Choose link based on your OS
* On Windows, choose the "base" subdirectory to install R.
* On OS X, choose the .pkg file to install R.

## Installing RStudio:

* Browse to <https://www.rstudio.com/>
* Mouse over Products and click RStudio
* Choose RStudio Desktop
* Click Download RStudio Desktop
* Choose the installer appropriate for your platform

# Example: Tips data

## Goals

* Explore a real dataset using R
* Get the "flavor" of R for data management and exploration
* Don't focus on the code - it will be explained later and in much more detail

## Tips Dataset

A server recorded the tips they received over about 10 weeks, including several variables:

* Amount they were tipped
* Cost of the total bill
* Characteristics about the party (# people, gender, etc.)

Primary Question: How do these variable influence the amount being tipped?

Follow along using [Tips-Example.R](code/Tips-Example.R)

## First look: Reading in the data

Lets use R to look at the top few rows of the tips data set. First, we load tip using read.csv:

tips <- read.csv("https://srvanderplas.github.io/NPPD-Analytics-Workshop/01.Introduction/data/tips.csv")

Now, we use the head function to look at the first 6 rows of the data:

head(tips)  
## total\_bill tip sex smoker day time size  
## 1 16.99 1.01 Female No Sun Dinner 2  
## 2 10.34 1.66 Male No Sun Dinner 3  
## 3 21.01 3.50 Male No Sun Dinner 3  
## 4 23.68 3.31 Male No Sun Dinner 2  
## 5 24.59 3.61 Female No Sun Dinner 4  
## 6 25.29 4.71 Male No Sun Dinner 4

How big is this data set and what types of variables are in each column?

str(tips)  
## 'data.frame': 244 obs. of 7 variables:  
## $ total\_bill: num 17 10.3 21 23.7 24.6 ...  
## $ tip : num 1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...  
## $ sex : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...  
## $ smoker : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ time : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...  
## $ size : int 2 3 3 2 4 4 2 4 2 2 ...

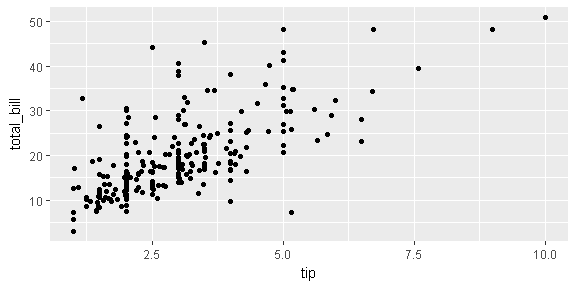
We can get a summary of the values for each variable in tips:

summary(tips)  
## total\_bill tip sex smoker day   
## Min. : 3.07 Min. : 1.000 Female: 87 No :151 Fri :19   
## 1st Qu.:13.35 1st Qu.: 2.000 Male :157 Yes: 93 Sat :87   
## Median :17.80 Median : 2.900 Sun :76   
## Mean :19.79 Mean : 2.998 Thur:62   
## 3rd Qu.:24.13 3rd Qu.: 3.562   
## Max. :50.81 Max. :10.000   
## time size   
## Dinner:176 Min. :1.00   
## Lunch : 68 1st Qu.:2.00   
## Median :2.00   
## Mean :2.57   
## 3rd Qu.:3.00   
## Max. :6.00

What is the relationship between total bill and tip value? First, we need to install and load ggplot2 (a graphics library):

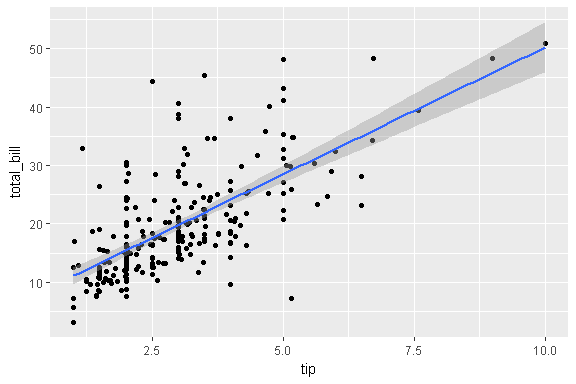
install.packages("ggplot2")  
library(ggplot2)

qplot(x = tip, y = total\_bill, geom = "point", data = tips)



We can add linear regression line to the plot:

qplot(x = tip, y = total\_bill, geom = "point", data = tips) +   
 geom\_smooth(method = "lm")

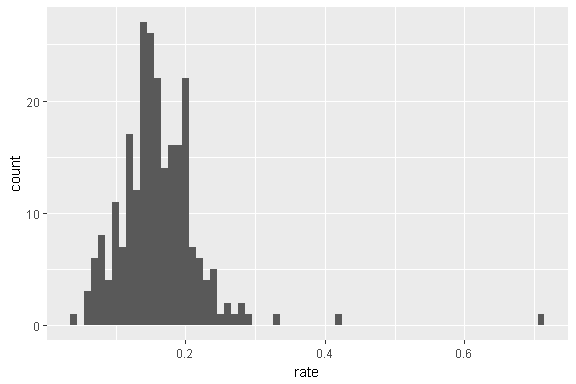


Tipping is generally done using a rule of thumb based on a percentage of the total bill. We will make a new variable in the data set for the tipping rate = tip / total bill

tips$rate <- tips$tip / tips$total\_bill  
  
summary(tips$rate)  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.03564 0.12910 0.15480 0.16080 0.19150 0.71030

What is the distribution of tip rate in this dataset? We can use R to create a histogram:

qplot(x = rate, data = tips, binwidth = .01)



One person tipped over 70%, who are they?

tips[which.max(tips$rate),]  
## total\_bill tip sex smoker day time size rate  
## 173 7.25 5.15 Male Yes Sun Dinner 2 0.7103448

We can also calculate the average tipping rate for men and women separately:

mean(tips$rate[tips$sex == "Male"])  
## [1] 0.1576505  
mean(tips$rate[tips$sex == "Female"])  
## [1] 0.1664907

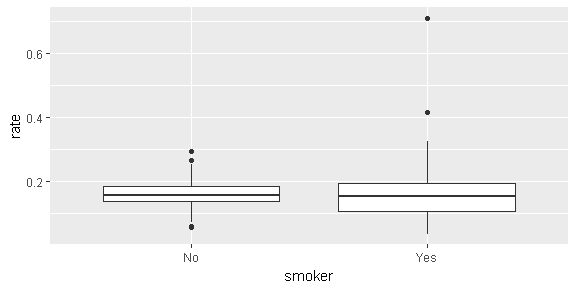
Is the difference between the two groups statistically significant? We can calculate a t-test to examine whether the average tip rate for men is the same as the average tip rate for women.

t.test(rate ~ sex, data = tips)  
##   
## Welch Two Sample t-test  
##   
## data: rate by sex  
## t = 1.1433, df = 206.76, p-value = 0.2542  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -0.006404119 0.024084498  
## sample estimates:  
## mean in group Female mean in group Male   
## 0.1664907 0.1576505

The p-value is greater than 0.05, and the confidence interval contains 0: there is no evidence of a significant difference in tipping rate between men and women.

We can compare two groups using a side-by-side boxplot. For instance, we can see whether smokers tip at a different rate than non-smokers:

qplot(x = smoker, y = rate, geom = "boxplot", data = tips)



## Your Turn!

Try playing with chunks of code shown above to further investigate the tips data:

1. Get a summary of the total bill values
2. Make side by side boxplots of tip rates for different days of the week
3. Find the average tip value for smokers

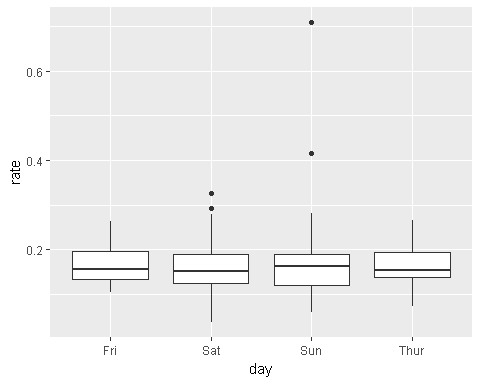
### Solutions

1. Get a summary of the total bill values

summary(tips$total\_bill)  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.07 13.35 17.80 19.79 24.13 50.81

1. Make side by side boxplots of tip rates for different days of the week

qplot(day, rate, geom = "boxplot", data = tips)



1. Find the average tip value for smokers

mean(tips$tip[tips$smoker == "Yes"])  
## [1] 3.00871

# R Basics

## Getting Help in R

The help() function is useful for getting help with a function:

help(head)

The ? function also works:

?head

When searching for results online, it is helpful to use R + CRAN + to get good results.

## R Reference Card

A copy of the R reference card is available at:

<http://cran.r-project.org/doc/contrib/Short-refcard.pdf>

This card contains short versions of the most common functions used in R.

## R as an Overgrown Calculator

R can perform simple mathematical operations.

# Addition and Subtraction  
2 + 5 - 1  
## [1] 6  
  
# Multiplication  
109\*23452  
## [1] 2556268  
  
# Division  
3/7  
## [1] 0.4285714

Here are a few more complex operations:

# Integer division  
7 %/% 2  
## [1] 3  
  
# Modulo operator (Remainder)  
7 %% 2  
## [1] 1  
  
# Powers  
1.5 ^ 3  
## [1] 3.375

# Exponentiation  
exp(3)  
## [1] 20.08554  
  
# Logarithms  
log(3)  
## [1] 1.098612  
log(3, base = 10)  
## [1] 0.4771213  
  
# Trig functions  
sin(0)  
## [1] 0  
cos(0)  
## [1] 1  
tan(pi/4)  
## [1] 1  
  
asin(0)  
## [1] 0  
acos(1)  
## [1] 0  
atan(1)  
## [1] 0.7853982

## Variables in R

Variables in R are created using the assignment operator, <-:

x <- 5  
R\_awesomeness <- Inf  
MyAge <- 21 #Haha

These variables can then be used in computation:

log(x)  
## [1] 1.609438  
MyAge ^ 2  
## [1] 441

## Rules for Variable Names

* Variable names can't start with a number
* Names are case-sensitive
* Common letters are used internally by R and should be avoided as variable names  
  c, q, t, C, D, F, T, I
* There are reserved words that R won't let you use for variable names.  
  for, in, while, if, else, repeat, break, next
* R will let you use the name of a predefined function.  
  Try not to overwrite those!

### Error messages:

# Variable starts with a number  
1age <- 3  
## Error: <text>:2:2: unexpected symbol  
## 1: # Variable starts with a number  
## 2: 1age  
## ^

# Case Sensitive  
Age <- 3  
age  
## Error in eval(expr, envir, enclos): object 'age' not found

# Special Words can't be variable names  
for <- 3  
## Error: <text>:2:5: unexpected assignment  
## 1: # Special Words can't be variable names  
## 2: for <-  
## ^

### Cautionary Tale

# This is a VERY bad idea:  
  
T <- FALSE  
F <- TRUE  
  
T == FALSE  
## [1] TRUE  
F == TRUE  
## [1] TRUE  
  
rm(T, F) # Fix it!  
  
T == FALSE  
## [1] FALSE

Note: In R, T and F are shorthand for TRUE and FALSE respectively.

## Vectors

A variable can contain more than one value.  
A *vector* is a variable which contains a set of values of the same type.  
The c() (combine) function is used to create vectors:

y <- c(1, 5, 3, 2)  
z <- c(y, y)

R performs operations on the entire vector at once:

y / 2  
## [1] 0.5 2.5 1.5 1.0  
z + 3  
## [1] 4 8 6 5 4 8 6 5

## Your Turn

Using the R Reference Card (and the Help pages, if needed), do the following:

1. Find out how many rows and columns the `iris' data set has. Figure out at least 2 ways to do this.  
   *Hint: "Variable Information" section on the first page of the reference card!*
2. Use the rep function to construct the following vector: 1 1 2 2 3 3 4 4 5 5  
   *Hint: "Data Creation" section of the reference card*
3. Use rep to construct this vector: 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

### Solutions

1. Find out how many rows and columns the `iris' data set has. Figure out at least 2 ways to do this.

data(iris)  
  
# first way:   
nrow(iris)  
## [1] 150  
ncol(iris)  
## [1] 5  
  
# second way:   
dim(iris)  
## [1] 150 5  
  
# third way:   
str(iris) # look at the top line  
## 'data.frame': 150 obs. of 5 variables:  
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

1. Use the rep function to construct the following vector: 1 1 2 2 3 3 4 4 5 5

rep(c(1:5), each = 2)  
## [1] 1 1 2 2 3 3 4 4 5 5

1. Use rep to construct this vector: 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

rep(c(1:5), times = 3)  
## [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

## Indexing Vectors

A vector is a list of values that are all the same type. Vectors can be created using the c() or rep() function. To create a vector of consecutive values, use the : function:

a <- 10:15  
a  
## [1] 10 11 12 13 14 15

Different elements of a vector can be extracted using brackets:

a[1]  
## [1] 10  
a[2]  
## [1] 11  
a[5]  
## [1] 14

Indexes can also be more complicated:

a[c(1, 3, 5)]  
## [1] 10 12 14  
a[1:5]  
## [1] 10 11 12 13 14

## Logical Values

* R has built in support for logical values
* TRUE and FALSE are built in. T (for TRUE) and F (for FALSE) are supported but can be modified
* Logicals can result from a comparison using

## Indexing with Logicals

Logical vectors can be used for indexing as well:

x <- c(2, 3, 5, 7)  
x[c(TRUE, FALSE, FALSE, TRUE)]  
## [1] 2 7  
x > 3.5  
## [1] FALSE FALSE TRUE TRUE  
x[x > 3.5]  
## [1] 5 7

### Logical Examples

# Get the rate variable out of the tips dataset  
rate <- tips$rate   
  
head(rate)  
## [1] 0.05944673 0.16054159 0.16658734 0.13978041 0.14680765 0.18623962  
  
sad\_tip <- rate < 0.10  
  
rate[sad\_tip]  
## [1] 0.05944673 0.07180385 0.07892660 0.05679667 0.09935739 0.05643341  
## [7] 0.09553024 0.07861635 0.07296137 0.08146640 0.09984301 0.09452888  
## [13] 0.07717751 0.07398274 0.06565988 0.09560229 0.09001406 0.07745933  
## [19] 0.08364236 0.06653360 0.08527132 0.08329863 0.07936508 0.03563814  
## [25] 0.07358352 0.08822232 0.09820426

## Data Frames

A collection of vectors, similar to a table in an Excel spreadsheet

* A data set is stored in a data frame
* Each column is a vector of the same length
* Each column can be a different type of data
* Each element in the vector/column has to have the same type of data
* columns can be accessed using $

tips is a data frame:

head(tips)  
## total\_bill tip sex smoker day time size rate  
## 1 16.99 1.01 Female No Sun Dinner 2 0.05944673  
## 2 10.34 1.66 Male No Sun Dinner 3 0.16054159  
## 3 21.01 3.50 Male No Sun Dinner 3 0.16658734  
## 4 23.68 3.31 Male No Sun Dinner 2 0.13978041  
## 5 24.59 3.61 Female No Sun Dinner 4 0.14680765  
## 6 25.29 4.71 Male No Sun Dinner 4 0.18623962

tips$sex shows the sex column of tips

tips$sex[1:20]  
## [1] Female Male Male Male Female Male Male Male Male Male   
## [11] Male Female Male Male Female Male Female Male Female Male   
## Levels: Female Male  
# Show the first 20 items in the sex column of tips

## Your Turn

1. Find out how many people tipped over 20%.  
   *Hint: use the sum function on a logical vector to calculate how many TRUEs are in the vector:*

sum(c(TRUE, TRUE, FALSE, TRUE, FALSE))  
## [1] 3

1. **More Challenging**: Calculate the sum of the total bills of anyone who tipped over 20%

## Modifying Vectors

Vectors can be modified using indexing:

x <- bill[1:5]  
## Error in eval(expr, envir, enclos): object 'bill' not found  
x  
## [1] 2 3 5 7  
x[1] <- 20  
x  
## [1] 20 3 5 7

### Vector Elements

Elements of a vector must all be the same type:

head(rate)  
## [1] 0.05944673 0.16054159 0.16658734 0.13978041 0.14680765 0.18623962  
rate[sad\_tip] <- ":-("  
head(rate)  
## [1] ":-(" "0.160541586073501" "0.166587339362208"  
## [4] "0.139780405405405" "0.146807645384303" "0.186239620403321"

By changing a value to a string, all the other values were changed to strings as well.

## Data Types in R

* Can use mode or class to find out information about variables
* str is useful to find information about the structure of your data
* Many data types: numeric, integer, character, Date, and factor most common

str(tips)  
## 'data.frame': 244 obs. of 8 variables:  
## $ total\_bill: num 17 10.3 21 23.7 24.6 ...  
## $ tip : num 1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...  
## $ sex : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...  
## $ smoker : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ time : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...  
## $ size : int 2 3 3 2 4 4 2 4 2 2 ...  
## $ rate : num 0.0594 0.1605 0.1666 0.1398 0.1468 ...

### Converting Between Types

Convert variables to a different type using the as series of functions:

size <- head(tips$size)  
size  
## [1] 2 3 3 2 4 4  
as.character(size)  
## [1] "2" "3" "3" "2" "4" "4"  
as.numeric("2")  
## [1] 2

## Some useful functions

There are a whole variety of useful functions to operate on vectors.

tip <- tips$tip  
x <- tip[1:5]  
length(x) # Number of elements of a vector  
## [1] 5  
sum(x) # Sum of elements in a vector  
## [1] 13.09

### Statistical Functions

Using the basic functions it wouldn't be hard to compute some basic statistics.

(n <- length(tip))  
## [1] 244  
(meantip <- sum(tip) / n)  
## [1] 2.998279  
(standdev <- sqrt(sum((tip - meantip)^2) / (n - 1)))  
## [1] 1.383638

But these functions are already built in to R.

### Built-in Statistical Functions

mean(tip)  
## [1] 2.998279  
sd(tip)  
## [1] 1.383638  
summary(tip)  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 2.000 2.900 2.998 3.562 10.000  
quantile(tip, c(.025, .975))  
## 2.5% 97.5%   
## 1.1760 6.4625

## Element-wise Logical Operators

* & (elementwise AND)
* | (elementwise OR)

c(T, T, F, F) & c(T, F, T, F)  
## [1] TRUE FALSE FALSE FALSE  
c(T, T, F, F) | c(T, F, T, F)  
## [1] TRUE TRUE TRUE FALSE  
# Which are big bills with a poor tip rate?  
id <- (bill > 40 & rate < .10)  
## Error in eval(expr, envir, enclos): object 'bill' not found  
tips[id,]  
## Error in `[.data.frame`(tips, id, ): object 'id' not found

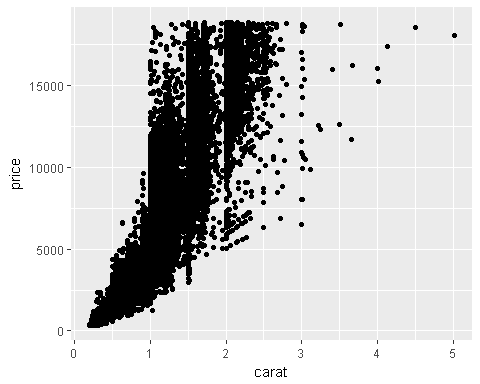
## Your Turn

1. Read up on the dataset (?diamonds)
2. Plot price by carat (use qplot - go back to the motivating example for help with the syntax)
3. Create a variable ppc for price/carat. Store this variable as a column in the diamonds data
4. Make a histogram of all ppc values that exceed $10000 per carat.
5. Explore any other interesting relationships you find

### Solutions

1. Plot price by carat (use qplot - go back to the motivating example for help with the syntax)

qplot(carat, price, data = diamonds)

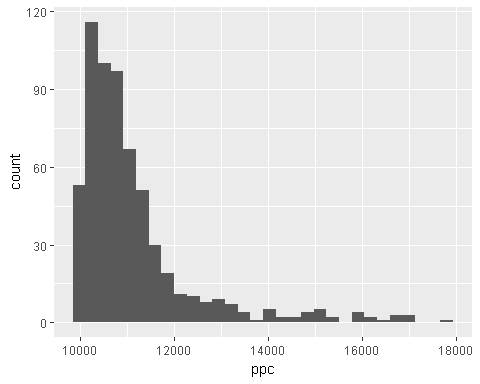


1. Create a variable ppc for price/carat. Store this variable as a column in the diamonds data

diamonds$ppc <- diamonds$price / diamonds$carat

1. Make a histogram of all ppc values that exceed $10000 per carat.

qplot(ppc, geom = "histogram", data = diamonds[diamonds$ppc > 10000,])  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



1. Explore any other interesting relationships you find