Tutorial 3: Probability & Distributions

Johannes Karreth

RPOS 517, Day 3

This tutorial shows you:

- how to simulate a random process
- how to plot the distribution of a variable
- how to assess the distribution of a variable

Saving your code (as usual)

Click on File -> New -> R Markdown. This will open a blank document above the console. As you go along you can copy and paste your code into R code chunks and save it. This is a good way to keep track of your code and be able to reuse it later. To run your code from this document, highlight the code and hit the Run button, or highlight the code and hit command+enter or a mac or control+enter on a PC.

You'll also want to save your R Markdown file. To do so click on the disk icon. The first time you hit save, RStudio will ask for a file name; you can name it anything you like. Once you hit save you'll see the file appear under the Files tab in the lower right panel. You can reopen this file anytime by simply clicking on it.

Simulations in R: Basics

Random processes are very easy to simulate in R. In a simulation, you set the ground rules of a random process and then the computer uses random numbers to generate an outcome that adheres to those rules. As a simple example, you can simulate flipping a fair coin with the following.

```r
outcomes <- c("heads", "tails")
sample(outcomes, size = 1, replace = TRUE)
```

```
## [1] "heads"
```

The vector `outcomes` can be thought of as a hat with two slips of paper in it: one slip says `heads` and the other says `tails`. The function `sample` draws one slip from the hat and tells us if it was a head or a tail.

Run the second command listed above several times. Just like when flipping a coin, sometimes you'll get a heads, sometimes you'll get a tails, but in the long run, you'd expect to get roughly equal numbers of each.

If you wanted to simulate flipping a fair coin 100 times, you could either run the function 100 times or, more simply, adjust the `size` argument, which governs how many samples to draw (the `replace = TRUE` argument indicates we put the slip of paper back in the hat before drawing again). Save the resulting vector of heads and tails in a new object called `sim_fair_coin`.

```r
sim_fair_coin <- sample(outcomes, size = 100, replace = TRUE)
```

To view the results of this simulation, type the name of the object and then use `table` to count up the number of heads and tails.

```
sim_fair_coin
```

```
##   [1] "tails" "tails" "tails" "heads" "heads" "heads" "heads" "tails"
##   [9] "heads" "heads" "heads" "heads" "heads" "heads" "tails" "heads"
##  [17] "tails" "tails" "tails" "heads" "heads" "tails" "tails" "tails"
##  [25] "tails" "tails" "tails" "tails" "tails" "tails" "tails" "heads"
##  [33] "heads" "tails" "tails" "tails" "tails" "heads" "heads" "tails"
##  [41] "heads" "tails" "heads" "tails" "tails" "tails" "heads" "tails"
##  [49] "tails" "heads" "tails" "tails" "heads" "heads" "heads" "tails"
##  [57] "heads" "heads" "heads" "heads" "tails" "heads" "heads" "heads"
##  [65] "heads" "heads" "heads" "tails" "heads" "heads" "tails" "heads"
##  [73] "tails" "tails" "tails" "tails" "heads" "tails" "tails" "heads"
##  [81] "heads" "heads" "heads" "tails" "tails" "heads" "tails" "tails"
##  [89] "heads" "tails" "heads" "heads" "tails" "heads" "tails" "heads"
##  [97] "tails" "tails" "heads" "tails"
```

```
table(sim_fair_coin)
```

```
## sim_fair_coin
## heads tails
##    49    51
```

Since there are only two elements in `outcomes`, the probability that we "flip" a coin and it lands heads is 0.5. Say we're trying to simulate an unfair coin that we know only lands heads 20% of the time. We can adjust for this by adding an argument called `prob`, which provides a vector of two probability weights.

```
sim_unfair_coin <- sample(outcomes, size = 100, replace = TRUE, prob = c(0.2, 0.8))
```

`prob = c(0.2, 0.8)` indicates that for the two elements in the `outcomes` vector, we want to select the first one, `heads`, with probability 0.2 and the second one, `tails` with probability 0.8. Another way of thinking about this is to think of the outcome space as a bag of 10 chips, where 2 chips are labeled "head" and 8 chips "tail". Therefore at each draw, the probability of drawing a chip that says "head"" is 20%, and "tail" is 80%.

In a sense, we've shrunken the size of the slip of paper that says "heads", making it less likely to be drawn and we've increased the size of the slip of paper saying "tails", making it more likely to be drawn. When we simulated the fair coin, both slips of paper were the same size. This happens by default if you don't provide a `prob` argument; all elements in the `outcomes` vector have an equal probability of being drawn.

If you want to learn more about `sample` or any other function, recall that you can always check out its help file.

```
?sample
```

Distributions of variables

In this tutorial, we'll investigate the probability distribution that is most central to statistics: the normal distribution. If we are confident that our data are nearly normal, that opens the door to many powerful statistical methods. Here we'll use the graphical tools of R to assess the normality of our data and also learn how to generate random numbers from a normal distribution.

The Data

As an example, we'll be working with measurements of body dimensions. This data set contains measurements from 247 men and 260 women, most of whom were considered healthy young adults.

```
bdims <- read.csv("http://www.jkarreth.net/files/bdims.csv")
```

Let's take a quick peek at the first few rows of the data.

```
head(bdims)
```

```
##   bia.di bii.di bit.di che.de che.di elb.di wri.di kne.di ank.di sho.gi
## 1   42.9   26.0   31.5   17.7   28.0   13.1   10.4   18.8   14.1  106.2
## 2   43.7   28.5   33.5   16.9   30.8   14.0   11.8   20.6   15.1  110.5
## 3   40.1   28.2   33.3   20.9   31.7   13.9   10.9   19.7   14.1  115.1
## 4   44.3   29.9   34.0   18.4   28.2   13.9   11.2   20.9   15.0  104.5
## 5   42.5   29.9   34.0   21.5   29.4   15.2   11.6   20.7   14.9  107.5
## 6   43.3   27.0   31.5   19.6   31.3   14.0   11.5   18.8   13.9  119.8
##   che.gi wai.gi nav.gi hip.gi thi.gi bic.gi for.gi kne.gi cal.gi ank.gi
## 1   89.5   71.5   74.5   93.5   51.5   32.5   26.0   34.5   36.5   23.5
## 2   97.0   79.0   86.5   94.8   51.5   34.4   28.0   36.5   37.5   24.5
## 3   97.5   83.2   82.9   95.0   57.3   33.4   28.8   37.0   37.3   21.9
## 4   97.0   77.8   78.8   94.0   53.0   31.0   26.2   37.0   34.8   23.0
## 5   97.5   80.0   82.5   98.5   55.4   32.0   28.4   37.7   38.6   24.4
## 6   99.9   82.5   80.1   95.3   57.5   33.0   28.0   36.6   36.1   23.5
##   wri.gi age  wgt   hgt sex
## 1   16.5  21 65.6 174.0   1
## 2   17.0  23 71.8 175.3   1
## 3   16.9  28 80.7 193.5   1
## 4   16.6  23 72.6 186.5   1
## 5   18.0  22 78.8 187.2   1
## 6   16.9  21 74.8 181.5   1
```

We can also have a quick look at the summary statistics of all variables:

```
summary(bdims)
```

```
##      bia.di          bii.di          bit.di          che.de
##  Min.   :32.40   Min.   :18.70   Min.   :24.70   Min.   :14.30
##  1st Qu.:36.20   1st Qu.:26.50   1st Qu.:30.60   1st Qu.:17.30
##  Median :38.70   Median :28.00   Median :32.00   Median :19.00
##  Mean   :38.81   Mean   :27.83   Mean   :31.98   Mean   :19.23
##  3rd Qu.:41.15   3rd Qu.:29.25   3rd Qu.:33.35   3rd Qu.:20.90
##  Max.   :47.40   Max.   :34.70   Max.   :38.00   Max.   :27.50
##      che.di          elb.di          wri.di          kne.di
##  Min.   :22.20   Min.   : 9.90   Min.   : 8.10   Min.   :15.70
##  1st Qu.:25.65   1st Qu.:12.40   1st Qu.: 9.80   1st Qu.:17.90
##  Median :27.80   Median :13.30   Median :10.50   Median :18.70
##  Mean   :27.97   Mean   :13.39   Mean   :10.54   Mean   :18.81
##  3rd Qu.:29.95   3rd Qu.:14.40   3rd Qu.:11.20   3rd Qu.:19.60
##  Max.   :35.60   Max.   :16.70   Max.   :13.30   Max.   :24.30
##      ank.di          sho.gi           che.gi           wai.gi
##  Min.   : 9.90   Min.   : 85.90   Min.   : 72.60   Min.   : 57.90  
##  1st Qu.:13.00   1st Qu.: 99.45   1st Qu.: 85.30   1st Qu.: 68.00  
##  Median :13.80   Median :108.20   Median : 91.60   Median : 75.80  
##  Mean   :13.86   Mean   :108.20   Mean   : 93.33   Mean   : 76.98  
##  3rd Qu.:14.80   3rd Qu.:116.55   3rd Qu.:101.15   3rd Qu.: 84.50  
##  Max.   :17.20   Max.   :134.80   Max.   :118.70   Max.   :113.20  
##      nav.gi           hip.gi           thi.gi          bic.gi     
##  Min.   : 64.00   Min.   : 78.80   Min.   :46.30   Min.   :22.40  
##  1st Qu.: 78.85   1st Qu.: 92.00   1st Qu.:53.70   1st Qu.:27.60  
##  Median : 84.60   Median : 96.00   Median :56.30   Median :31.00  
##  Mean   : 85.65   Mean   : 96.68   Mean   :56.86   Mean   :31.17  
##  3rd Qu.: 91.60   3rd Qu.:101.00   3rd Qu.:59.50   3rd Qu.:34.45  
##  Max.   :121.10   Max.   :128.30   Max.   :75.70   Max.   :42.40  
##      for.gi          kne.gi          cal.gi          ank.gi     
##  Min.   :19.60   Min.   :29.00   Min.   :28.40   Min.   :16.40  
##  1st Qu.:23.60   1st Qu.:34.40   1st Qu.:34.10   1st Qu.:21.00  
##  Median :25.80   Median :36.00   Median :36.00   Median :22.00  
##  Mean   :25.94   Mean   :36.20   Mean   :36.08   Mean   :22.16  
##  3rd Qu.:28.40   3rd Qu.:37.95   3rd Qu.:38.00   3rd Qu.:23.30  
##  Max.   :32.50   Max.   :49.00   Max.   :47.70   Max.   :29.30  
##      wri.gi          age             wgt              hgt       
##  Min.   :13.0   Min.   :18.00   Min.   : 42.00   Min.   :147.2  
##  1st Qu.:15.0   1st Qu.:23.00   1st Qu.: 58.40   1st Qu.:163.8  
##  Median :16.1   Median :27.00   Median : 68.20   Median :170.3  
##  Mean   :16.1   Mean   :30.18   Mean   : 69.15   Mean   :171.1  
##  3rd Qu.:17.1   3rd Qu.:36.00   3rd Qu.: 78.85   3rd Qu.:177.8  
##  Max.   :19.6   Max.   :67.00   Max.   :116.40   Max.   :198.1  
##       sex        
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.4872  
##  3rd Qu.:1.0000  
##  Max.   :1.0000
```

Notice here that there are again *no* missing observations for any of the variables, like in last week's tutorial. This means that we don't have to think about how to deal with missing data. In other (applied) cases, this will be different, and you need to give some thought to missing data in your R commands and your data analysis.

You'll see that for every observation we have 25 measurements, many of which are either diameters or girths. A key to the variable names can be found at http://www.openintro.org/stat/data/bdims.php, but we'll be focusing on just three columns to get started: weight in kg (`wgt`), height in cm (`hgt`), and `sex` (`1` indicates male, `0` indicates female).

Since males and females tend to have different body dimensions, it will be useful to create two additional data sets: one with only men and another with only women.

```
mdims <- subset(bdims, sex == 1)
fdims <- subset(bdims, sex == 0)
```

The normal distribution

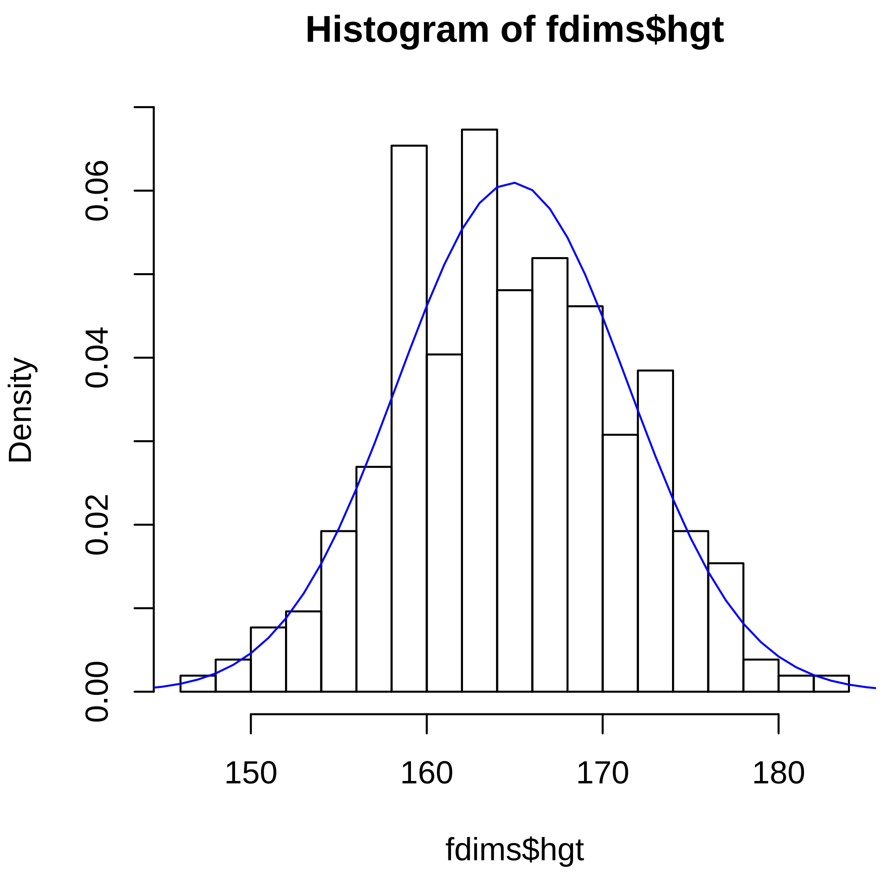
In your description of the distributions, would you use words like *bell-shaped* or *normal*? It's tempting to say so when faced with a unimodal symmetric distribution.

To see how accurate that description is, we can plot a normal distribution curve on top of a histogram to see how closely the data follow a normal distribution. This normal curve should have the same mean and standard deviation as the data. We'll be working with women's heights, so let's store them as a separate object and then calculate some statistics that will be referenced later.

```
fhgtmean <- mean(fdims$hgt)
fhgtsd   <- sd(fdims$hgt)
```

Next we make a density histogram to use as the backdrop and use the `lines` function to overlay a normal probability curve. The difference between a frequency histogram and a density histogram is that while in a frequency histogram the *heights* of the bars add up to the total number of observations, in a density histogram the *areas* of the bars add up to 1. The area of each bar can be calculated as simply the height *times* the width of the bar. Using a density histogram allows us to properly overlay a normal distribution curve over the histogram since the curve is a normal probability density function. Frequency and density histograms both display the same exact shape; they only differ in their y-axis. You can verify this by comparing the frequency histogram you constructed earlier and the density histogram created by the commands below.

```
hist(fdims$hgt, probability = TRUE, breaks = 20)
x <- 140:190
y <- dnorm(x = x, mean = fhgtmean, sd = fhgtsd)
lines(x = x, y = y, col = "blue")
```



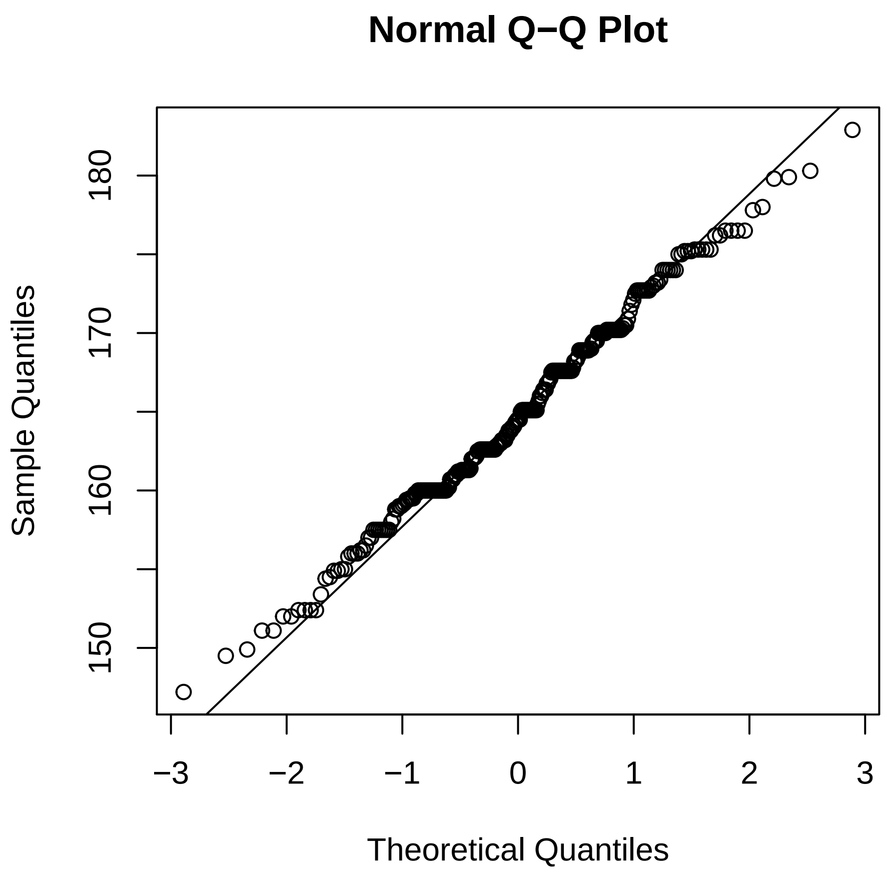
After plotting the density histogram with the first command, we create the x- and y-coordinates for the normal curve. We chose the `x` range as 140 to 190 in order to span the entire range of `fheight`. To create `y`, we use `dnorm` to calculate the density of each of those x-values in a distribution that is normal with mean `fhgtmean` and standard deviation `fhgtsd`. The final command draws a curve on the existing plot (the density histogram) by connecting each of the points specified by `x` and `y`. The argument `col` simply sets the color for the line to be drawn. If we left it out, the line would be drawn in black.

The top of the curve may be cut off because the limits of the x- and y-axes are set to best fit the histogram. To adjust the y-axis you can add a third argument to the histogram function: `ylim = c(0, 0.06)`.

Evaluating the normal distribution

Eyeballing the shape of the histogram is one way to determine if the data appear to be nearly normally distributed, but it can be frustrating to decide just how close the histogram is to the curve. An alternative approach involves constructing a normal probability plot, also called a normal Q-Q plot for "quantile-quantile".

```
qqnorm(fdims$hgt)
qqline(fdims$hgt)
```



A data set that is nearly normal will result in a probability plot where the points closely follow the line. Any deviations from normality leads to deviations of these points from the line. The plot for female heights shows points that tend to follow the line but with some errant points towards the tails. We're left with the same problem that we encountered with the histogram above: how close is close enough?

A useful way to address this question is to rephrase it as: what do probability plots look like for data that I *know* came from a normal distribution? We can answer this by simulating data from a normal distribution using `rnorm`.

```
sim_norm <- rnorm(n = length(fdims$hgt), mean = fhgtmean, sd = fhgtsd)
```

The first argument indicates how many numbers you'd like to generate, which we specify to be the same number of heights in the `fdims` data set using the `length` function. The last two arguments determine the mean and standard deviation of the normal distribution from which the simulated sample will be generated. We can take a look at the shape of our simulated data set, `sim_norm`, as well as its normal probability plot.

Even better than comparing the original plot to a single plot generated from a normal distribution is to compare it to many more plots using the following function. This function "repeats" the `qqnorm()` command 9 times. We will talk more about function writing later in the seminar; for now, just copy & paste this function into your R script and run it.

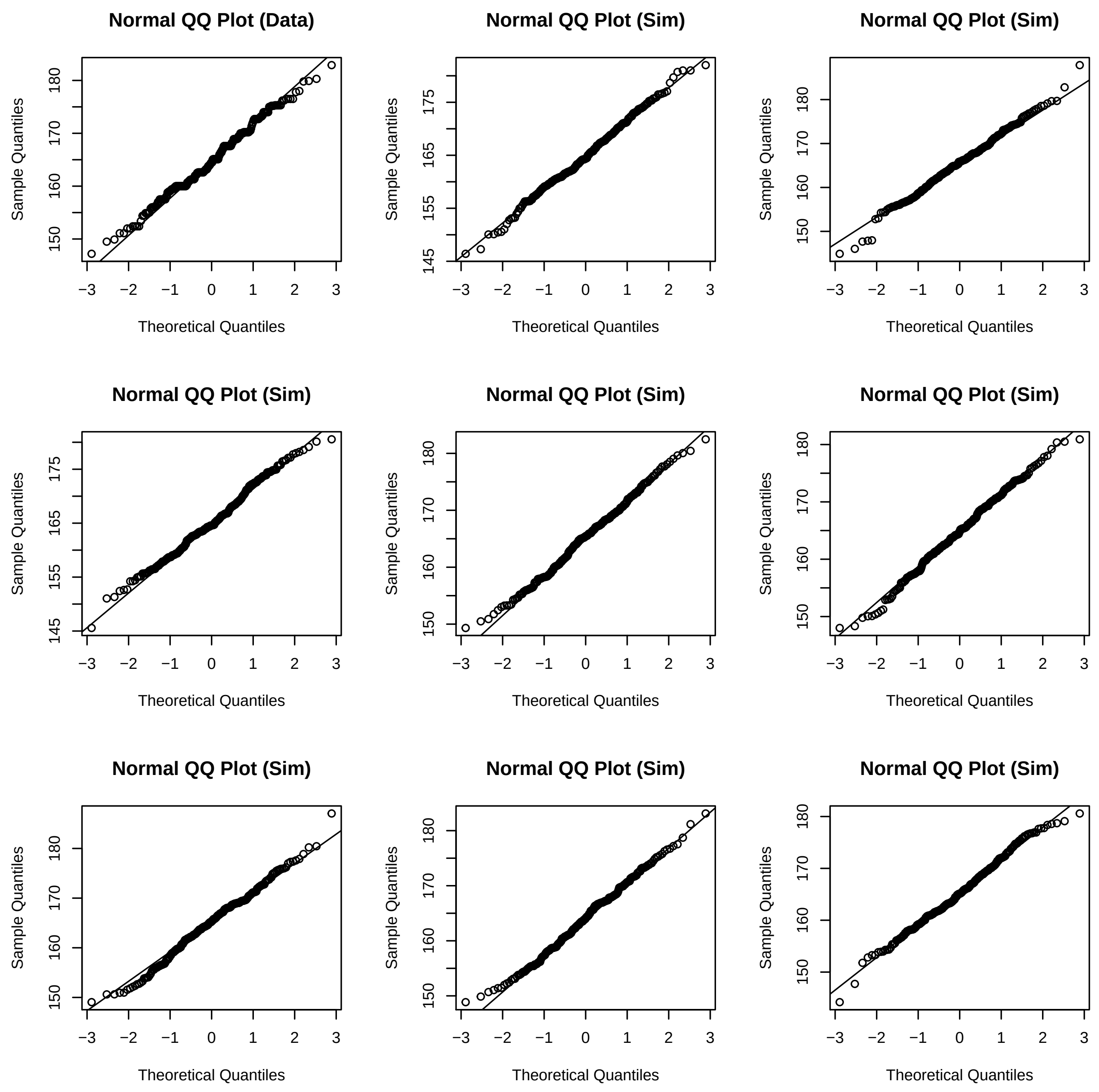
```r
qqnormsim <- function (dat)
{
    par(mfrow = c(3, 3))
    qqnorm(dat, main = "Normal QQ Plot (Data)")
    qqline(dat)
    for (i in 1:8) {
        simnorm <- rnorm(n = length(dat), mean = mean(dat), sd = sd(dat))
        qqnorm(simnorm, main = "Normal QQ Plot (Sim)")
        qqline(simnorm)
    }
    par(mfrow = c(1, 1))
}
```

When you want to run a command in R that is encompassed by braces `{}`, it is best to highlight the whole command rather than go line by line. Alternatively, in this case, rather than copying and pasting the function from this tutorial, you may also read it into R from my website, using the `source()` command:

```r
source("http://www.jkarreth.net/files/qqnormsim.R")
```

It may be helpful to click the zoom button in the plot window.

```r
qqnormsim(fdims$hgt)
```



Normal probabilities

Okay, so now you have a slew of tools to judge whether or not a variable is normally distributed. Why should we care?

It turns out that statisticians know a lot about the normal distribution. Once we decide that a random variable is approximately normal, we can answer all sorts of questions about that variable related to probability. Take, for example, the question of, "What is the probability that a randomly chosen young adult female is taller than 6 feet (about 182 cm)?" (The study that published this data set is clear to point out that the sample was not random and therefore inference to a general population is not suggested. We do so here only as an exercise.)

If we assume that female heights are normally distributed (a very close approximation is also okay), we can find this probability by calculating a Z score and consulting a Z table (also called a normal probability table). In R, this is done in one step with the function `pnorm`.

```r
1 - pnorm(q = 182, mean = fhgtmean, sd = fhgtsd)
```

```
## [1] 0.004434387
```

Note that the function `pnorm` gives the area under the normal curve below a given value, `q`, with a given mean and standard deviation. Since we're interested in the probability that someone is taller than 182 cm, we have to take one minus that probability.

Assuming a normal distribution has allowed us to calculate a theoretical probability. If we want to calculate the probability empirically, we simply need to determine how many observations fall above 182 then divide this number by the total sample size.

```r
sum(fdims$hgt > 182) / length(fdims$hgt)
```

```
## [1] 0.003846154
```

Although the probabilities are not exactly the same, they are reasonably close. The closer that your distribution is to being normal, the more accurate the theoretical probabilities will be.