**Lab 10 – Exponential distribution, normality testing**

**To submit: answers to all numbered questions. When the question asks you to write code or create graphs, submit the code and/or graphs in the Word document as part of your answer. Also submit a single .R file that contains all of your code.**

# Exponential distribution: waiting for a more realistic bus

In our last lab, we imagined a person waiting for a bus that comes exactly every 20 minutes. Anyone who has waited for buses knows that no bus comes exactly every 20 minutes (or exactly every 5 minutes, or exactly every hour). More realistically, a bus may come *on average* every 20 minutes. The exponential distribution with mean 20 gives a fairly accurate model of waiting times **between** buses.

We use the \***exp()** family of commands to find exponential waiting times. For an exponential distribution with rate **r,**  the command

> qexp(p, rate = r)

returns the value such that there is a probability **p** that the time between occurrences will be **less** than that value.

> qexp(.1, rate = 1)

[1] 0.10536

> qexp(.5, rate = 1)

[1] 0.69315

> qexp(.9, rate = 1)

[1] 2.3026

For instance, at a mean rate of 1 occurrence per minute, there is a 50% chance the waiting time between occurrences will be less than 0.69315 minutes.

1. For the bus example, what is **r**? (Hint: the units of **r** are “buses per minute”.)
2. Find qexp(.5, rate = r) for that value of **r**. This should be the median of an exponentially-distributed random variable with mean 20. How does it compare to the mean? Based on the distribution of the exponential random variable, does this makes sense? (Hint: the word “skew” should appear in your answer.)

The **pexp()** command returns probabilities corresponding to exponential distributions. For an exponential distribution with rate **r**, the command

> pexp(q, rate = r)

returns the probability that the time between occurrences will be less than **q**.

> pexp(2, rate = 1)

[1] 0.86466 47

> pexp(1, rate = 1)

[1] 0.6321206

> pexp(0.5, rate = 1)

[1] 0.3934693

1. Find the probability that a person is waiting…
   1. Less than 15 minutes for a bus
   2. More than 25 minutes for a bus
   3. Between 10 and 20 minutes for a bus

You can compare your answers to ones you obtain with the formulas from class.

We can also do simulations with exponential probabilities. In order to model **n** waiting times, we need to get waiting times corresponding to **n** probabilities. Those **n** probabilities will be distributed uniformly. That way, approximately 10% of waiting times will be in the bottom 10% of the exponential distribution.

The command

> rexp(n, r)

gives **n** waiting times distributed exponentially with a mean rate of **r** per unit time.

1. Generate appropriately-labelled histograms that give the frequency of waits between buses that have exponentially-distributed waiting times with mean 20 minutes for n=100, 1000, and 10000. Use the same number of classes for each. Do the distributions look exponential?

# Distribution of sample means in an exponentially-distributed population

Previously, we simulated sampling from a uniformly-distributed population – die rolls – and investigated the distribution of the sample means. We then simulated sampling from a normally-distributed population. Now we will investigate the distributions of sample means of a very skewed population – one with an exponential distribution.

1. Create a function called **ExponentialMeans(n,m)** that simulates taking **n** samples of size **m** from exponentially-distributed populations with rate 1. As before, this new function will compute (and keep track of) the sample mean for each of those **n** samples.Your function should return the following:

* The mean of the **n** sample means
* The standard deviation of the **n** sample means
* An appropriately-named **histogram** of the **n** sample means (use the default bins)

Run your function for **n=10000** and for the following values of **m**: 1, 2, 10, 50, 100.

Submit five outputs (five histograms, each with the means and standard deviations listed below). How do the means and standard deviations compare as **m** increases? How do the shapes of the graphs compare to one another, and to the graphs you obtained earlier?

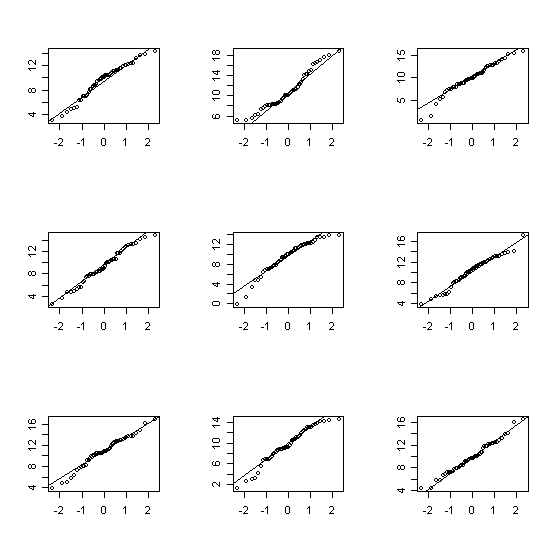
# Normality testing

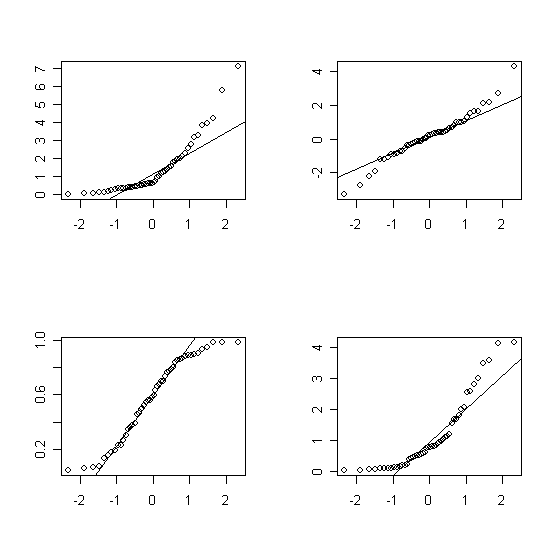
We saw in an earlier that sample means are distributed normally if the population follows a normal distribution, or if the sample is “large enough”. Soon we will see that we can estimate population means based on sample means, and that some of our methods only work if the population is distributed normally. It follows that we often need to know whether a sample appears to have come from a normally-distributed population. One way to do that is by producing a **quantile-quantile plot**, which we can do easily in R.

A quantile-quantile plot, or **QQ plot**, is a graph specifically designed to check for normality. The plot usually requires a minimum of 15 – 20 data points in order to accurately reveal normality.

If the data comes from a normal distribution the points should form a line with positive slope.

Examples of QQ plots of normal-ish data:



Examples of QQ plots of non-normal data:

We will do a few examples before delving into the theory.

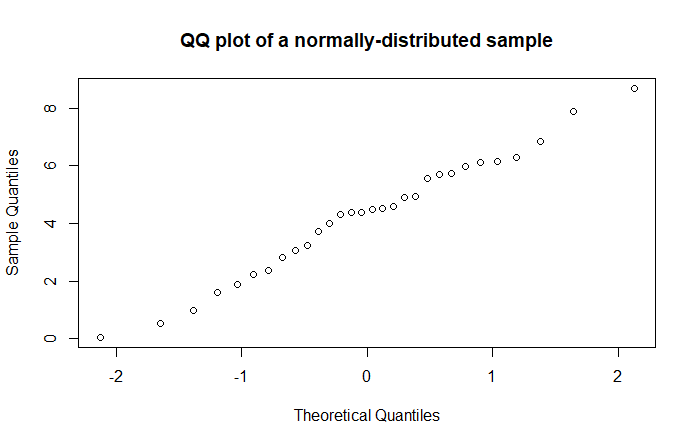
# Quantile-quantile plots of normal data

We will begin by generating a sample that comes from a normal distribution. We have generated similar samples before, using the **rnorm** function. Here are thirty data points that come from a normally-distributed dataset with mean 4 and standard deviation 2, though any mean and standard deviation will do:

> normaldata=rnorm(30,4,2)

We then produce the QQ plot with the **qqnorm** function:

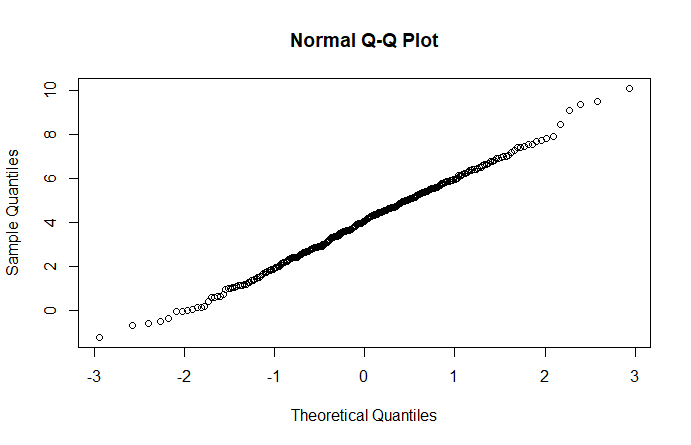
> qqnorm(normaldata, main="QQ plot of a normally-distributed sample")



The points roughly follow a line. If our sample were larger, we’d see a clearer pattern:

> normaldata=rnorm(300,4,2)

> qqnorm(normaldata)

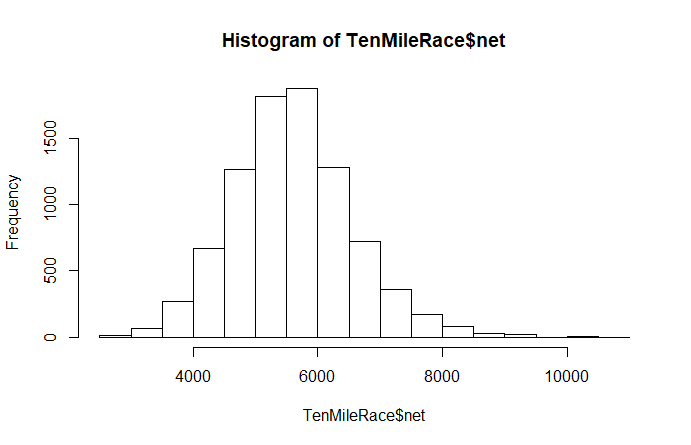


Note that the graph gets a bit weird at the edges. This is typical.

# QQ plots – a bit of theory

A QQ plot plots sample data against the z-scores that correspond to the percentiles of our data. If our data is normally distributed, then we would expect the same proportion of z-scores corresponding to the data as generated z-scores to be between *a* and *b*, for any numbers *a* and *b*. Graphically, this means that if our sample data came from a normally-distributed population, then the horizontal spacing between any two data points will be proportional to the vertical spacing, and our plot will be a straight line with positive slope. The less “normal” our sample data is, the more our plot will differ from being a straight line.

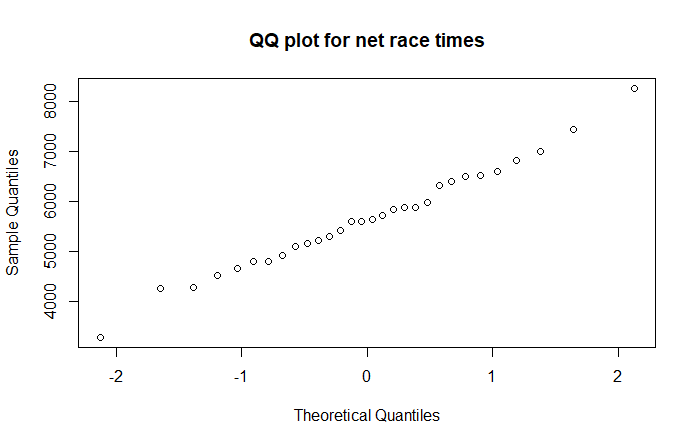
Let’s try this with some real data. Recall the dataset **TenMileRace** from Lab 1. Load this dataset (you will have to load the **mosaicData** library). Here is a histogram of net race times, which certainly appear to follow a normal distribution:



In real life, we often don’t have access to entire populations, or even large samples. When our samples are small, we need to check if they are normal before finding t-values to generate confidence intervals. For small samples, a QQ plot gives us a clearer picture of the distribution than a histogram. Since the population of net times is normally distributed, a sample of times will probably be close to normally distributed as well. Eg, here are 30 randomly-selected race times (the **sample** function gives us 30 indices, and then we retrieve the corresponding race times):

> racetimes30=TenMileRace$net[sample(1:8636, 30)]

> qqnorm(racetimes30, main="QQ plot for net race times")



It certainly appears that the sample of race times, like the population, is normally distributed.

(NOTE: your QQ plot will be different, and it might not look like it reflects a normal distribution. Unfortunately, this sort of thing happens occasionally: samples from normally distributed populations aren’t always normally distributed. Conversely, on occasion we have normally distributed samples that come from populations that are not normally distributed. You may have seen examples of both in Question 1. This sort of thing is why, in statistics, we never “prove” anything with complete certainty – we are only 90%, or 95%, or 99% confident of our conclusion. The other 10% or 5% or 1%, we just happened to get a sample that was not representative of the population. For instance, if the population is large and normally distributed, it’s possible (but not very likely) that through no fault of our own, it turned out that every number in our sample is more than two standard deviations from the mean. That said, if your QQ plot looks very different from a straight line, generate another sample of size 30 and get a QQ plot for it. It will probably look better than your last one.)

Load the **genotype** dataframe, which is part of the MASS library. Take a minute to read over its help file.

1. Filter your data to obtain four data sets, grouped by litter genotype. Note that these data sets are small, so we require them to be normally distributed if we wish to find a confidence interval. Create QQ plots for weight for each genotype. Which of the four appears to be the most normal?