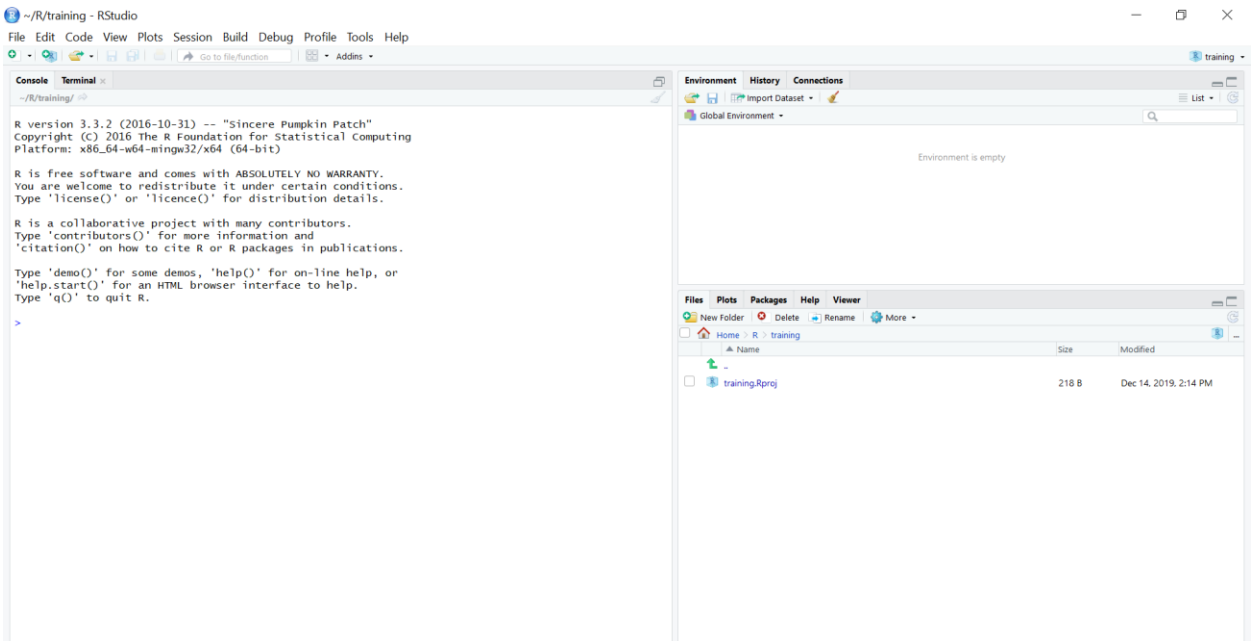
Mini-Tutorials for Creating Graphs in R

**Dotplots**

The examples below are intended to instruct you to create statistical graphs in R with minimal initial training in R. You should be able to follow the example codes to obtain graphs by modifying the included code. Some examples (and a key) will be included at the end of the document for practice. The screenshots I will show of the environment use R Studio, which is a free program you can find online. Other R environments will look different, but probably have similar functionality.

When you open up a new project environment in R Studio, it looks like this.




The command line environment is on the left. Images when we construct them will appear on the bottom right. As we add variables, they will appear in the list at the top right (name, dimensions and samples will display, which is useful for checking that you didn't skip entries when entering data by hand).

We are going to create a simple dotplot from raw data. There are other ways to create dotplots with other packages, but this is the simplest. It has its limitations, but it mirrors what we would do by hand with the least amount of behind-the-scenes preprocessing.

Copy the commands shown into the command line.

**Step 1.** Enter the data to be plotted in vector form.

```
InflationRate=c(1.59, 3.01, 2.78, 4.27, 5.46, 5.84, 4.30, 3.27, 6.16, 11.03, 9.20, 5.75, 6.50,
7.62, 11.22, 13.58, 10.35, 6.16, 3.22, 4.30, 3.55, 1.91, 3.66, 4.08, 4.83, 5.39, 4.25, 3.03, 2
.96, 2.61, 2.81, 2.93, 2.34, 1.55, 2.19, 3.38, 2.83, 1.59, 2.27, 2.68, 3.39, 3.24, 2.85, 3.85, -
0.34, 1.64, 3.16, 2.07, 1.47, 1.62, 0.12, 1.26, 2.13)
```

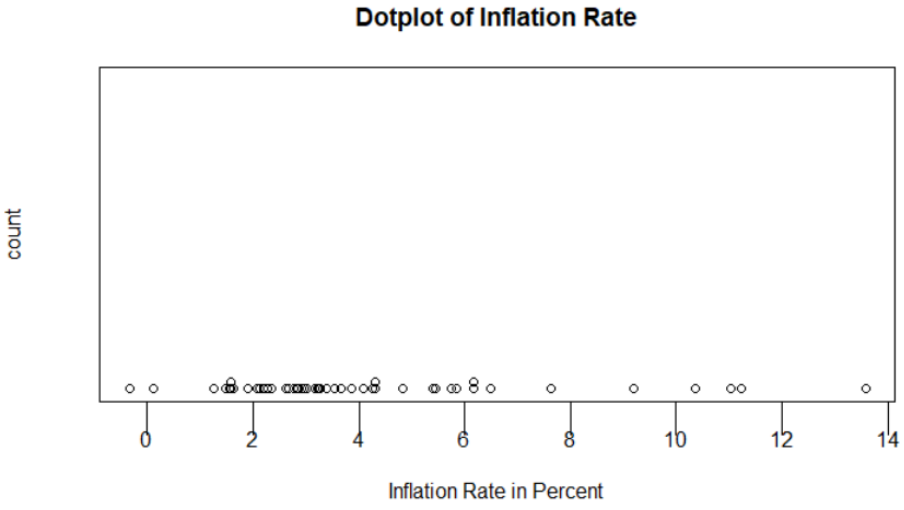
The difficult thing about creating a dotplot in R by this method is that the elements in the list need to be rounded to an appropriate level of accuracy so that they stack up on top of each other rather than overlapping just off to the side. For instance, we might want 2.95 and 2.96 to fall into the same bin, but if you graph the raw data as is, they won't stack. They will be treated as being in different bins. To know what size bin is needed, though, it may be necessary to graph the data and experiment with the required number of bins, which will depend on a number of factors like how much data there is. Too few bins might make the stacks too tall that they get cut off on the top. Too many bins and the stacks will overlap.

**Step 2.** Try graphing the data as is.

```
stripchart(InflationRate, method = "stack", offset = .275, at = 0, pch = 1, col="black",
main = "Dotplot of Inflation Rate", xlab = "Inflation Rate in Percent", ylab="count", tck=-
0.1)
```

The syntax shown for the stripchart command need the name of the variable to be graphed, the method ("stack" is for the dotplot), the offset you may need to experiment with and is related to the amount of space between the center of subsequent dots in the stack. You can adjust the color as you like. Main is the graph title, and for good graphs, be sure to include axis labels. "at" is above the bottom of graph, pch is dot type (there are 25 options), "tck" repositions tickmarks up if positive, down if negative. Feel free to experiment with different settings to see what they do and how you like the features. (We'll do this later.)

The graph we get is shown.



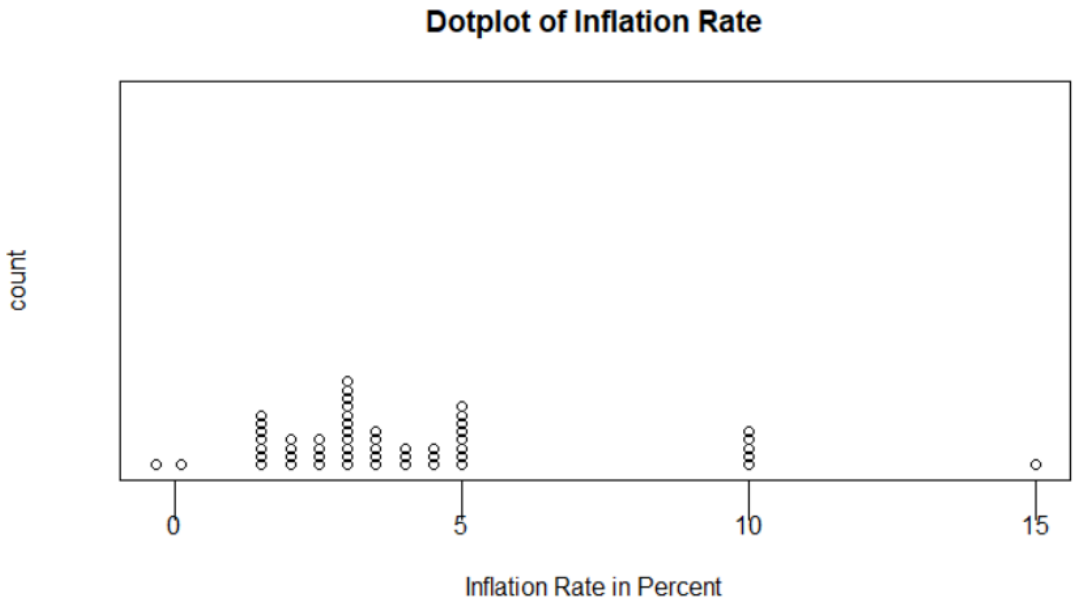
You can see what happens here with the data rounded as it is now. Data only stacks when two values are exactly the same. When they are off by just a little, they sit slightly to one side. Each "bin" is therefore 0.01 units wide, so there are 100 bins between each whole number. We need to round the values so that there are fewer bins. 4-5 bins per tickmark is generally a good rule of thumb to start with. Since each tick is spaced 2 apart, 2/4 is 0.5 spacing between bins. Rounding to the nearest half doesn't have a specific command, so instead we can multiply by 2, round to the nearest single digit, and then divide the results by 2 to obtain the same result.

**Step 3.** Round data to obtain an appropriate number of bins.

```
IR=InflationRate*2
IRround<-signif(IR, digits = 1)/2
```

Then re-graph with the modified data.

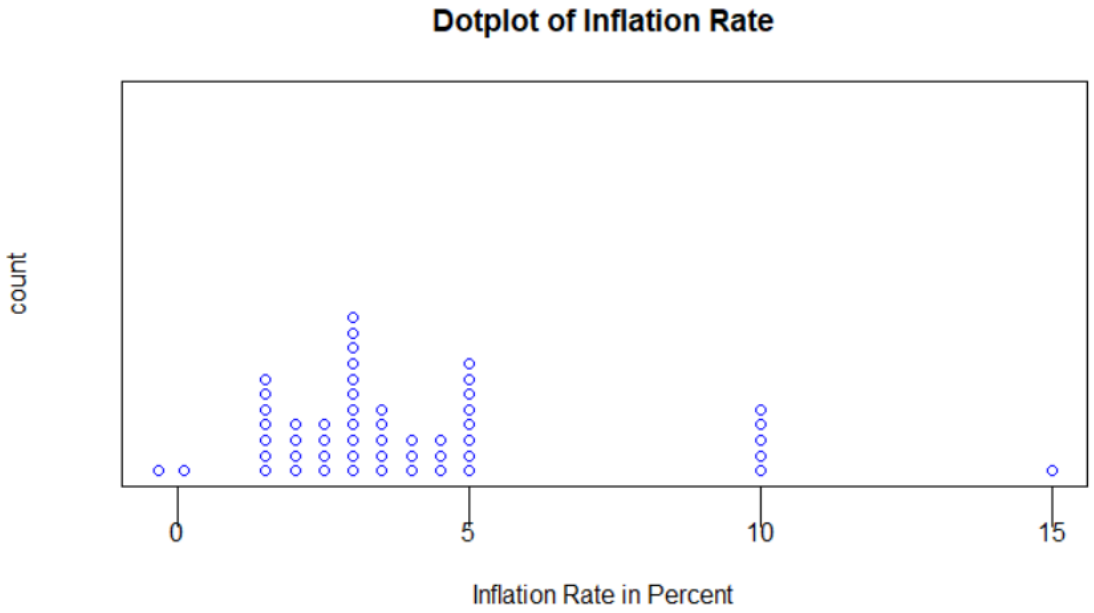
```
stripchart(IRround, method = "stack", offset = .275, at = 0, pch = 1, col="black", main = "
Dotplot of Inflation Rate", xlab = "Inflation Rate in Percent", ylab="count", tck=-0.1)
```




This looks better. Experiment with the spacing to obtain an attractive result.

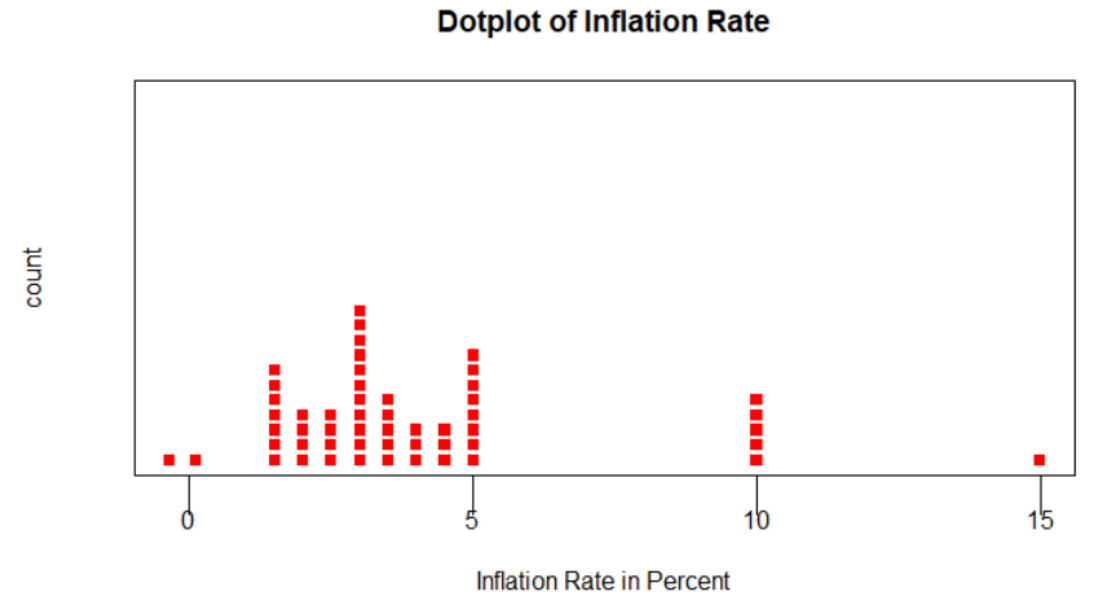
```
stripchart(IRround, method = "stack", offset = .5, at = 0, pch = 1, col="blue", main = "Dot
plot of Inflation Rate", xlab = "Inflation Rate in Percent", ylab="count", tck=-0.1)
```

We can increase the offset when using less data, for instance, and decrease it when there is more.




Or,

```
stripchart(IRround, method = "stack", offset = .5, at = 0, pch = 15, col="red", main = "Dot plot of Inflation Rate", xlab = "Inflation Rate in Percent", ylab="count", tck=-0.1)
```




**Step 4.** Add additional tickmarks as needed to make the graph easier to read. Or other optional features.

If you have not already done so, you'll need to install some packages. "minor.tick" adds additional tickmarks to the graph. "nx" is the number of such ticks, and "tick.ratio" is the length. You want fewer longer ticks, and more shorter ticks.

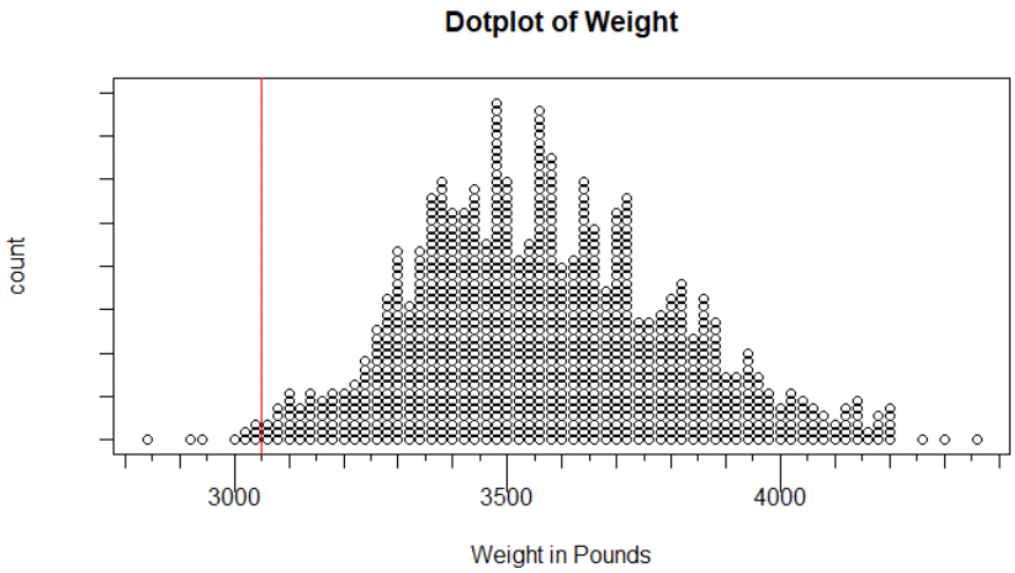
```
install.packages("ggplot2")
library(Hmisc)
minor.tick(nx=5, tick.ratio=1)
minor.tick(nx=10, tick.ratio=0.5)
```

You can also add a straight line to graph to make it easier to read for a specific value, such as

```
abline(v = 3050, col="red", lwd=1, lty=1)
```

Here, "v" is the location of the vertical line you want to add. "lwd" and "lty" change the weight and type of line (i.e. its thickness and whether it's solid, dotted, etc.)

Below is a graph I made with more data using some of these optional features.




**Practice.**
To get some random data to practice with, consider the following commands:

1. `test1=c(runif(100, min=80, max=100))`

runif produces a list of uniformly distributed random numbers between the specified min and max values. In this case, 100 such numbers are generated.

2. test2=c(rnorm(150, mean=64, sd=3.1))

rnorm generates a list of normally distributed numbers with the specified mean and standard deviation. In this case 150 such numbers are generated.

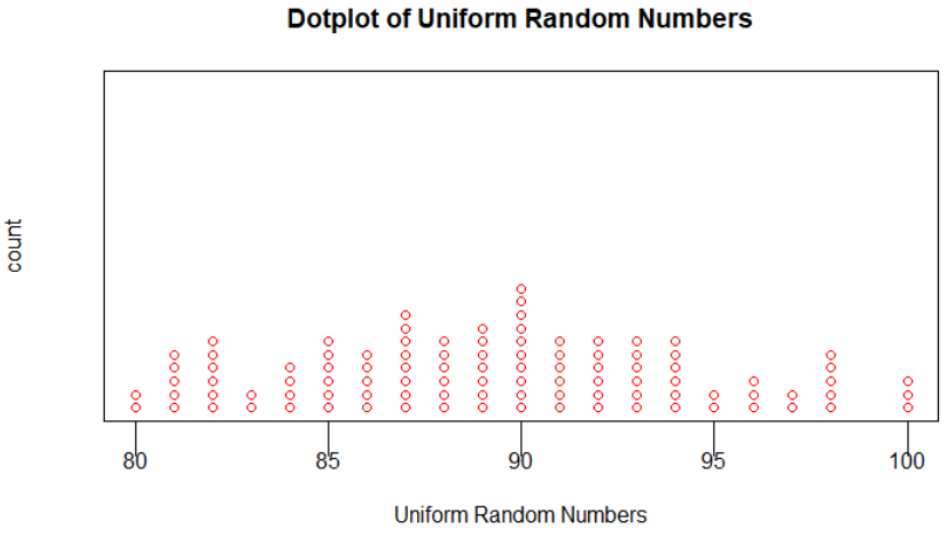
As you experiment, adjust the number of values included, and the range of such values to develop a feel for how to round/bin values, and stack them, given the number of values and how they are spread out.

**Solutions.**

Some example "solutions" to the practice are shown below. Your graphs will be different even with the same settings since the values are random.

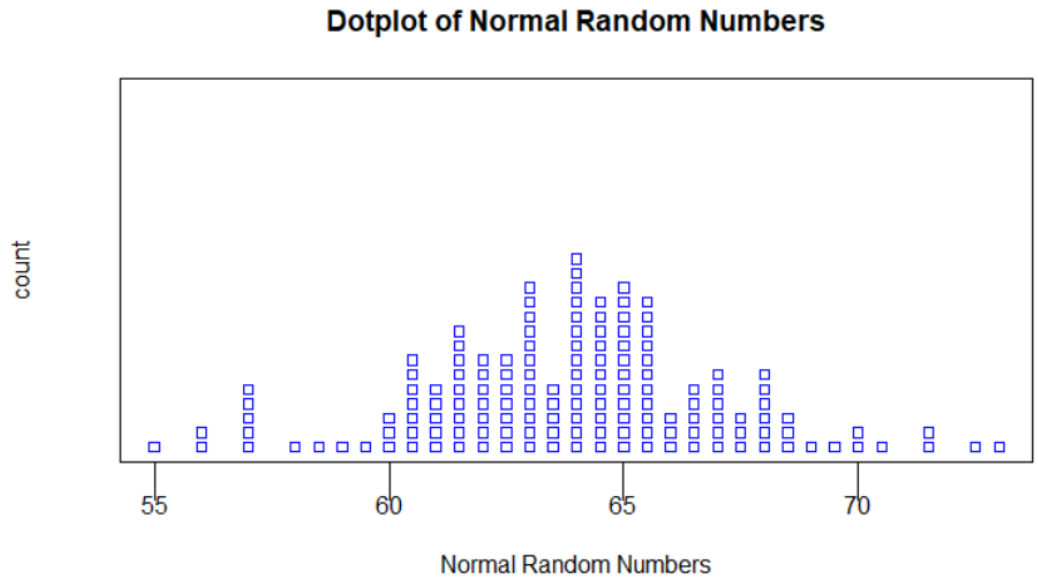
1.

```
test1round<-signif(test1, digits = 2)

stripchart(test1round, method = "stack", offset = .5, at = 0, pch = 21, col="red", main = "
Dotplot of Uniform Random Numbers", xlab = "Uniform Random Numbers", ylab="count
", tck=-0.1)
```



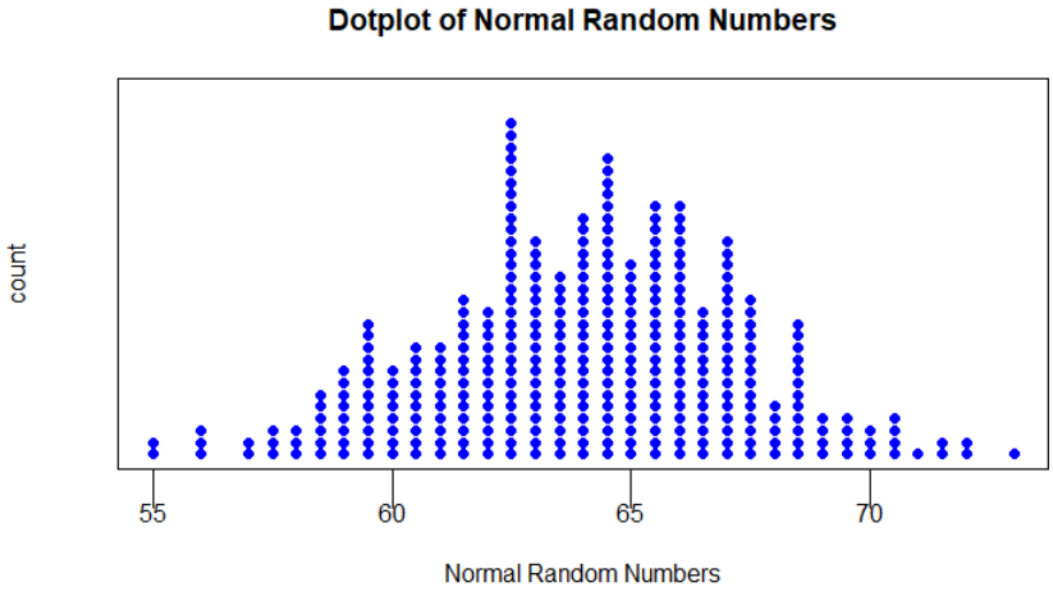

2.

```
test2round<-signif(test2*2, digits = 3)/2

stripchart(test2round, method = "stack", offset = .5, at = 0, pch = 22, col="blue", main =
"Dotplot of Normal Random Numbers", xlab = "Normal Random Numbers", ylab="count
", tck=-0.1)
```




If you redo the last example with 350 samples instead of just 150, you can see how the stacking height needs to be adjusted.

```
test3=c(rnorm(350, mean=64, sd=3.1))
test3round<-signif(test3*2, digits = 3)/2
stripchart(test3round, method = "stack", offset = .4, at = 0, pch = 19, col="blue", main = "Dotplot of Normal Random Numbers", xlab = "Normal Random Numbers", ylab="count ", tck=-0.1)
```




Or change the number of bins.

```
test3round<-signif(test3*3, digits = 3)/3

stripchart(test3round, method = "stack", offset = .5, at = 0, pch = 19, col="blue", main = "Dotplot of Normal Random Numbers", xlab = "Normal Random Numbers", ylab="count ", tck=-0.1)
```