Intro to RMarkdown

David King

2/16/2021

R Markdown

Markdown is its own language. It uses syntax from Pandoc/John Gruber.

```r
# a vector is a collection of items that have all the same type
str_vector = c("a","b","c","d")
numeric_vector = c(1,4,5,8,3)
# make a second numeric vector here with at least three elements
numeric_vector_2 = c(...)
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```r
# like python, R has bracket indexing, but the first element is accessed at index 1, instead of 0
str_vector[1]
```

```
## [1] "a"
```

```r
numeric_vector[1:5]
```

```
## [1] 1 4 5 8 3
```

```r
numeric_vector_2[...] # get the last element of numeric_vector_2
```

```
## Error in eval(expr, envir, enclos): object 'numeric_vector_2' not found
```

```r
# Vector lengths:
# R has a function, length(), that acts the same as python's len() to get vector length
# run length() on str_vector, numeric_vector, and numeric_vector_2
length(...)
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```r
length(...)
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```r
length(...)
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```r
# Out-of-bounds indexes:
# Unlike python, accessing elements out-of-bounds is not an error, but NA
# Accessing:
str_vector[5]
```

```
## [1] NA
```

```r
is.na(str_vector[5]) # check for NA
```

```
## [1] TRUE
```

```
numeric_vector[3:10]
```

```
## [1]  5  8  3 NA NA NA NA NA
```

```
is.na(numeric_vector[3:10])
```

```
## [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
# Setting:
numeric_vector[6] = 30
# now check for NAs in the range 3 to 10.
# numeric_vector[...]

# fill in the rest of the vector up to 10. Check your work using "any(is.na(...))" below.
numeric_vector[7] = ...
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```
numeric_vector[...] = ...
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```
...
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```
# check for NAs
is.na(numeric_vector)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
any( is.na(numeric_vector) ) # if "any" are TRUE, return TRUE
```

```
## [1] FALSE
```

```
# combined vectors
numeric_vector_all = c(numeric_vector,numeric_vector_2)
```

```
## Error in eval(expr, envir, enclos): object 'numeric_vector_2' not found
```

Operations on vectors

Operators and Functions that take a single number often also work on vectors.

```
numeric_vector * 10
sqrt(5)
sqrt(numeric_vector)
sqrt(numeric_vector*numeric_vector)

# try filling in the dashes with the functions log, exp, sin, cos
___(numeric_vector)
___(numeric_vector_2)
___(numeric_vector_2[1])
___(numeric_vector_2[-1])
# question!! what does -1 do? It is different than in python
print("-1 as a vector index does: ... ")


# Functions that operate on a vector, but return a single value
# fill in with the function: mean,sum, sd, median, max
___(numeric_vector)
___(numeric_vector_2)
___(numeric_vector)
___(numeric_vector)
# fill-in with each numeric vector
summary(...)
summary(...)
```

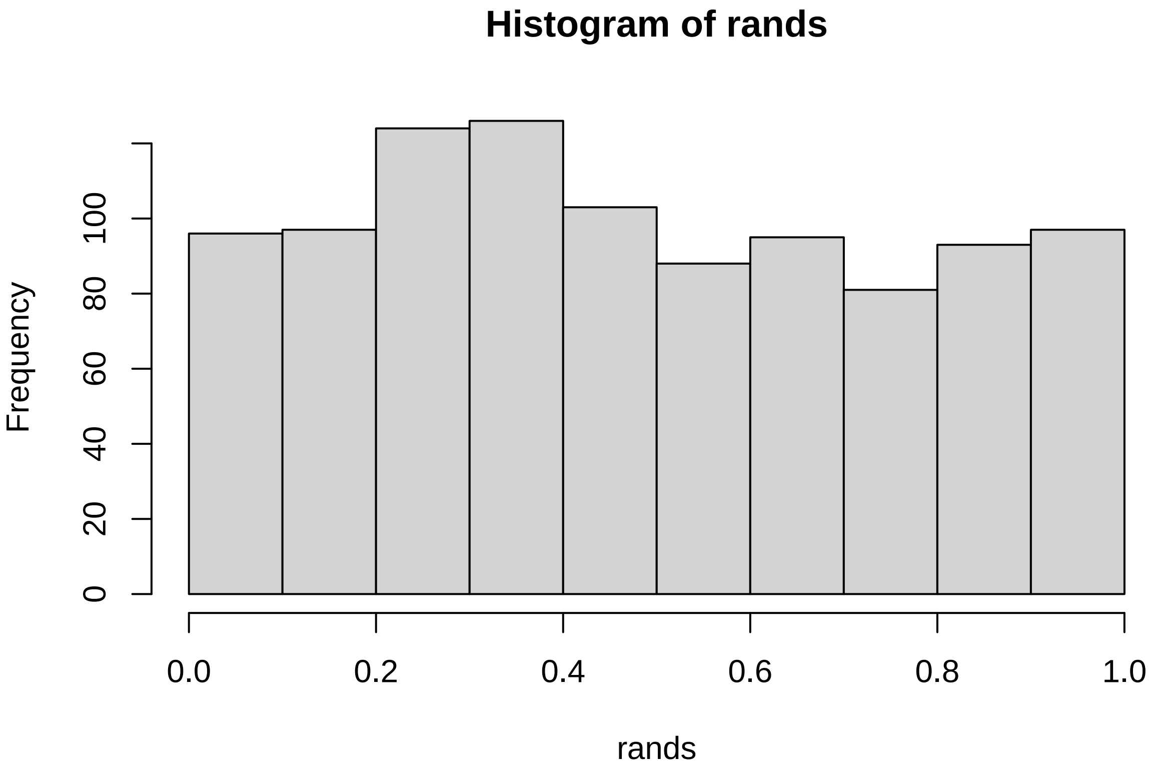
```
## Error: <text>:7:1: unexpected input
## 6: # try filling in the dashes with the functions log, exp, sin, cos
## 7: _
##    ^
```

It's time to start thinking about statistics

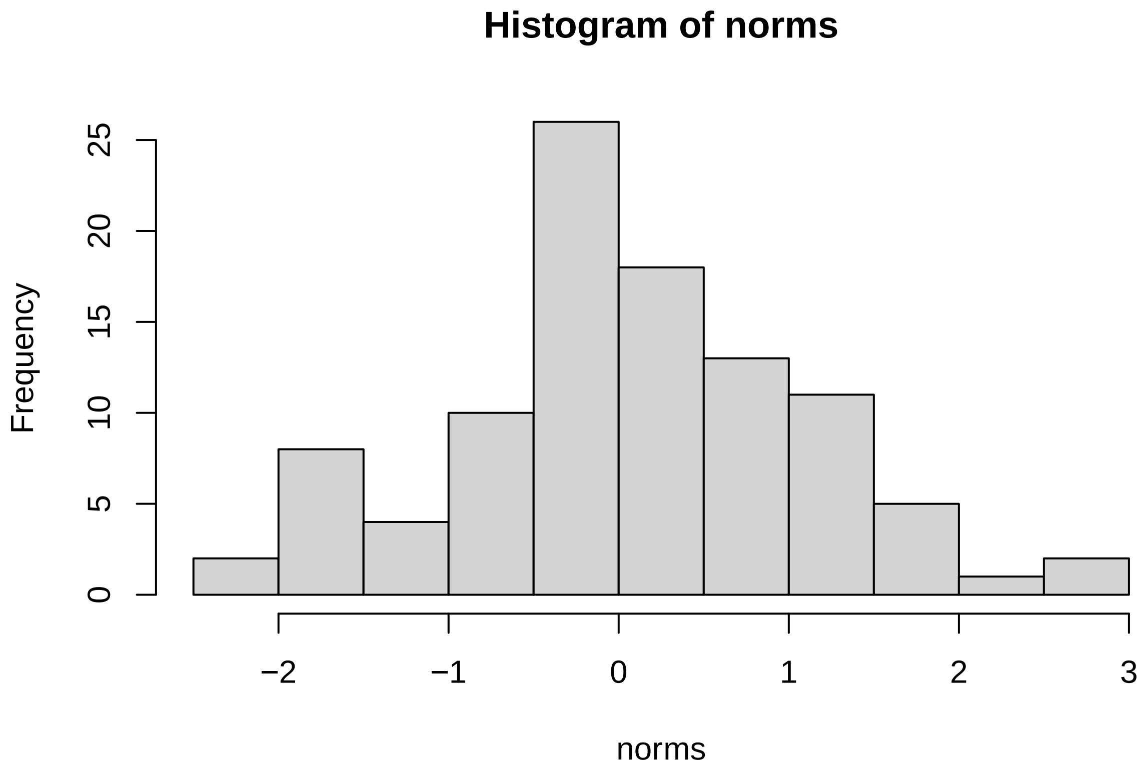
```
# get a large number of random numbers between 0 and 1
rands = runif(1000) # (r)andom (unif)orm distribution

# use summary() to check the mean and median... are they close to .5?
#summary(...)

# quick look: a histogram
hist(rands)
```



```
norms = rnorm(100)
hist(norms)
```



```r
# let's compare the two distributions
# Make a data frame with all the values in a vector
uniform_df = data.frame(values=rands, distribution="uniform") # it repeated "uniform" 1000 times
head(uniform_df)
```

```
##       values distribution
## 1 0.01784207      uniform
## 2 0.18222789      uniform
## 3 0.89885289      uniform
## 4 0.27714927      uniform
## 5 0.02534195      uniform
## 6 0.61110459      uniform
```

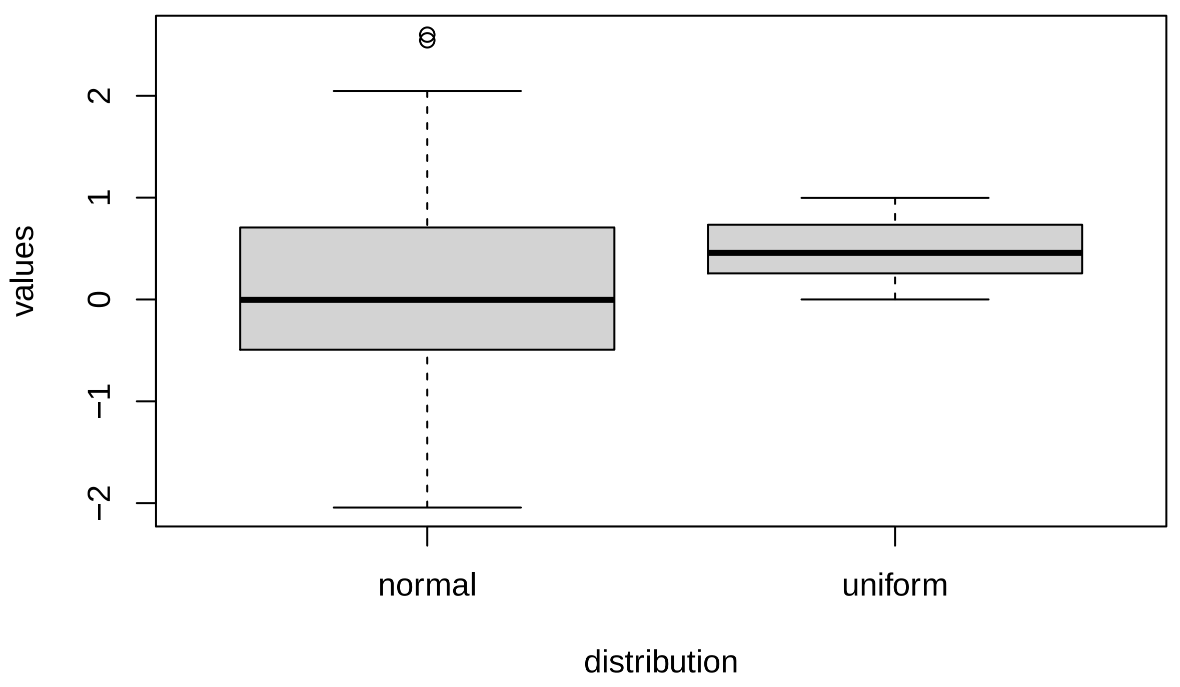
```r
normal_df = data.frame(values=norms, distribution="normal") # only 100 rows because there are 100 norms
head(normal_df)
```

```
##       values distribution
## 1 -0.1825075       normal
## 2 -0.3070500       normal
## 3  1.1521264       normal
## 4  0.5941377       normal
## 5 -0.4782100       normal
## 6 -0.9944037       normal
```

```r
comparison = rbind(uniform_df, normal_df)
head(comparison)
```

```
##       values distribution
## 1 0.01784207      uniform
## 2 0.18222789      uniform
## 3 0.89885289      uniform
## 4 0.27714927      uniform
## 5 0.02534195      uniform
## 6 0.61110459      uniform
```

```
boxplot(values ~ distribution, data=comparison) # left_side ~ right_side
```



```
                                                # left_side: value to analyze
                                                # right_side: grouping
# Question: where was the normal distribution centered? higher or lower to the uniform?

# Question: how does the boxplot diagram relate to the "summary()" function used above?
```

Add a new series of observations to the data frame `comparison`. This will result in a 3rd boxplot in the figure.

```
series3 = ___(n=...) # generate a new series with `n` elements (you choose how many). Random variates c

# preview some of the properties of series3
hist(series3)
summary(series3)

series3_df = data.frame(values=series3, distribution="...") # for distribution="...", replace the ... w

# append series3_df to `comparison`
```

```
comparison = rbind(comparison, series3_df)
boxplot(values ~ distribution, data=comparison) # will include the new data

# Lastly, do statistical tests on series3 versus rands. You may use t.test again, and use wilcox.test.

t.test(...,...)
wilcox.test(...,...) # you must answer a question below about wilcox.test


# For example, I centered a normal distribution on mean=.5. The p-value btw my new, .5-centered normal
```

```
## Error: <text>:2:11: unexpected input
## 1:
## 2: series3 = _
##              ^
```

How does the Wilcoxon test differ from the t-test?

You have to look it up.

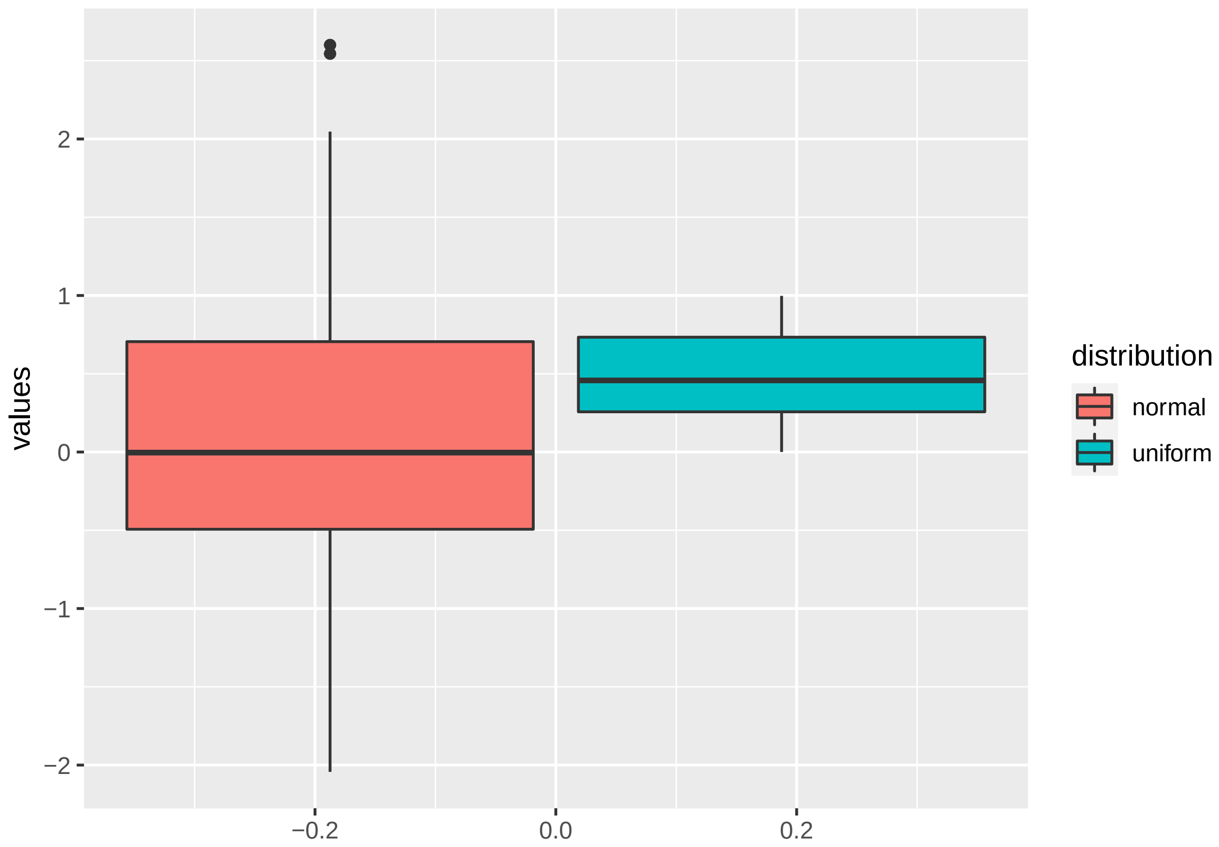
> Write your answer here. In this block quote syntax. All lines start with > for blockquotes.

ggplot

The library ggplot2 has more complex construction, but makes better plots. To understand the `aes()` argument of ggplot, do the interactive course on Datacamp: **Introduction to Data Visualization with ggplot2.**

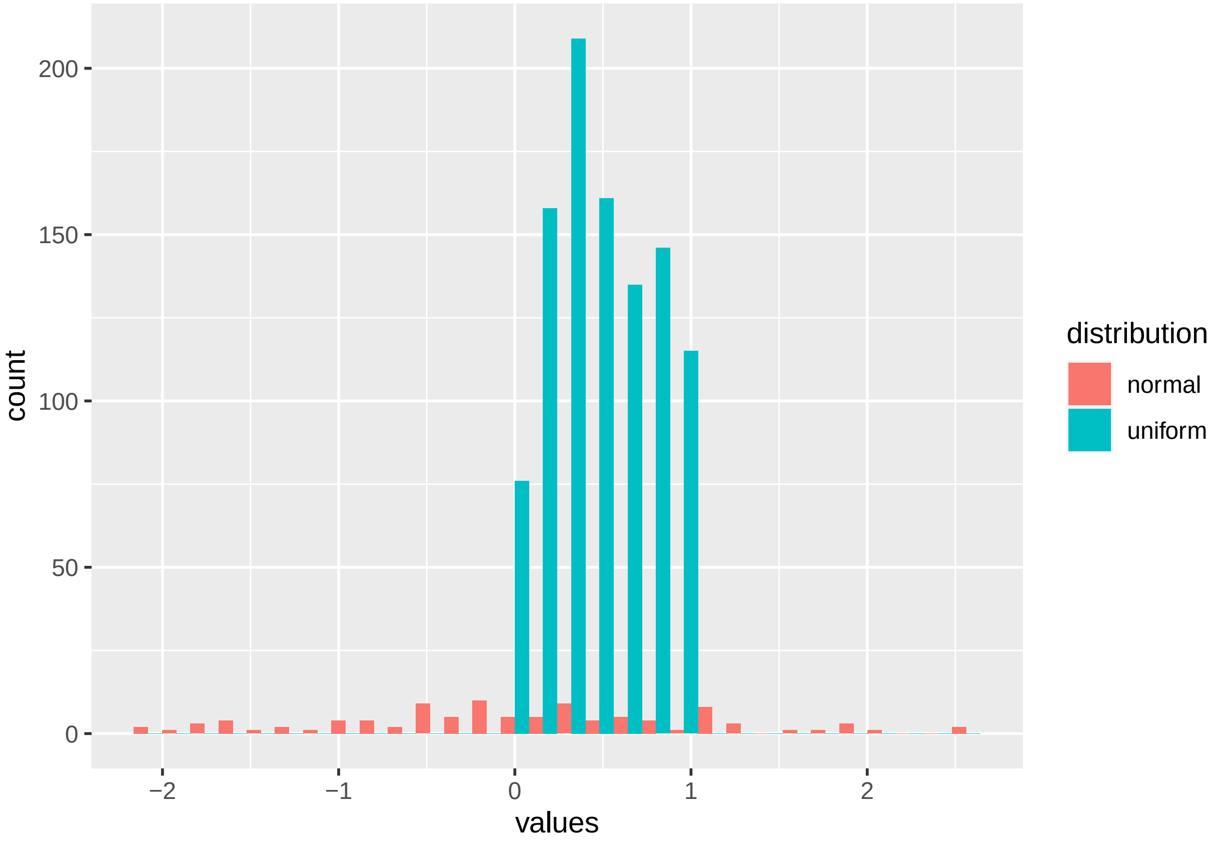
```
library(ggplot2)

# THE boxplot
ggplot(data=comparison,
       aes(x=values, fill=distribution)) +
       geom_boxplot() +
       coord_flip() # I'm used to seeing vertical boxplots. Delete/comment this line and the plus sign
```

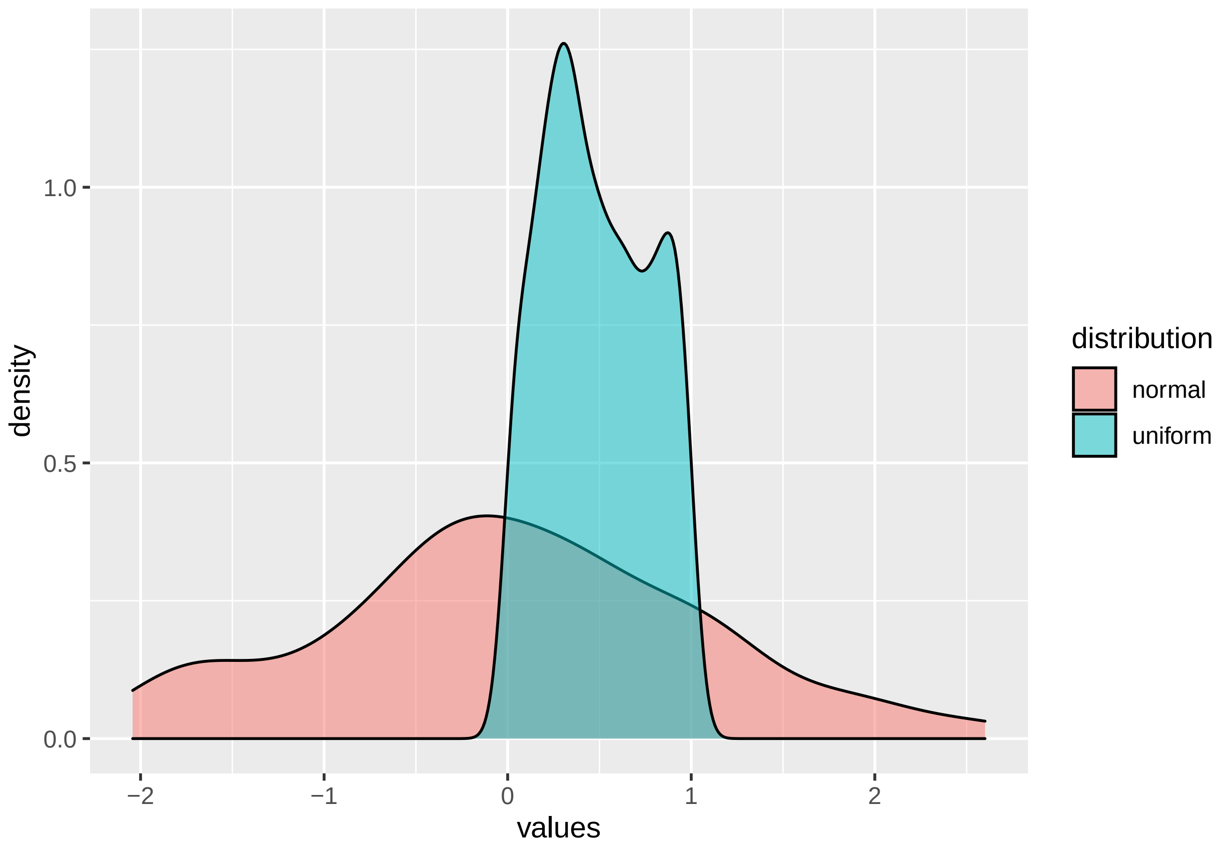


```
# THE histogram
ggplot(data=comparison,
       aes(x=values, fill=distribution)) +
       geom_histogram(position="dodge")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
# A better vis than a histogram.
# geom_density smooths the histogram and sets all counts relative to 1
ggplot(data=comparison,
       aes(x=values, fill=distribution)) +
       geom_density(alpha=.5) # alpha is transparency, so the curves can overlap
```


density
1.0
0.5
0.0
−2
−1
0
1
2
values
distribution
normal
uniform