Basics of Sampling with R

Ryan McClarren

September 20, 2016

In this document I will use the common R functions to generate samples from distributions, and demonstrate how we can sample from copulas.

This next bit of code will load in some necessary libraries for plotting and multivariate normals.

```
library(ggplot2)
library(dplyr)
library(MASS)
library(gridExtra)
```

Uniform Random Variables

As we mentioned in class, the bedrock of many sampling strategies is sampling a uniform random variable between 0 and 1. We do this using the function `runif(N)`, which generates a list of `N` uniform random variables between 0 and 1.

```
N = 1e1
X = runif(N)
print(X)
```

```
##  [1] 0.5806918 0.3226436 0.1090324 0.5063774 0.9419624 0.7198082 0.2525950
##  [8] 0.0345124 0.0454551 0.6885542
```

As a result, we can readily generate lot's of samples. I will save this into a `data.frame`; a R container that is matrix of mixed data types. The `head` function prints the first 10 lines of a data frame.

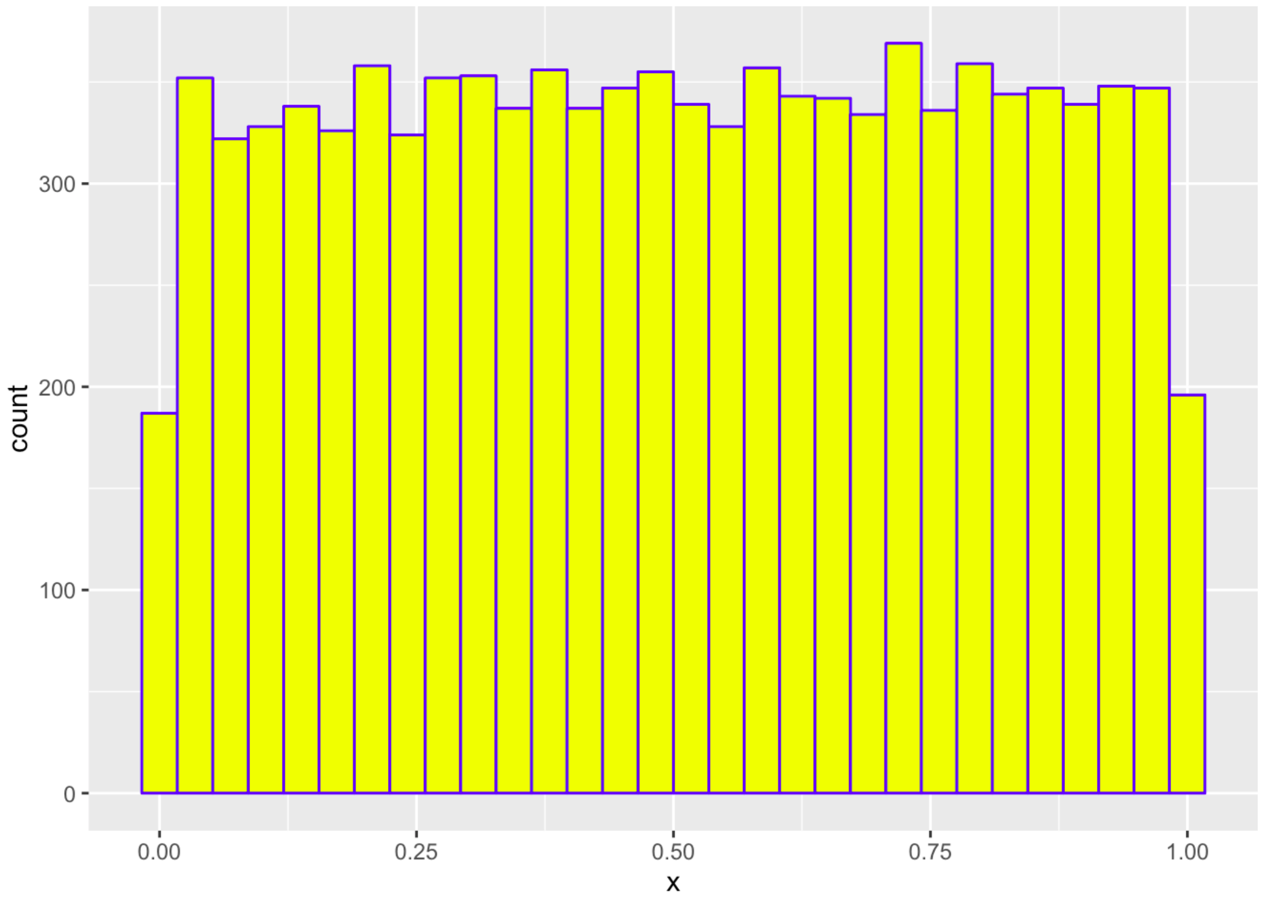
```
N = 1e4
df = data.frame(x = runif(N))
head(df)
```

```
##             x
## 1 0.97947502
## 2 0.99707695
## 3 0.55063576
## 4 0.45003879
## 5 0.07848921
## 6 0.21175098
```

We can plot the histogram of these samples using `ggplot` . The syntax of `ggplot` is a bit abstruse at first, but it makes some nice plots.

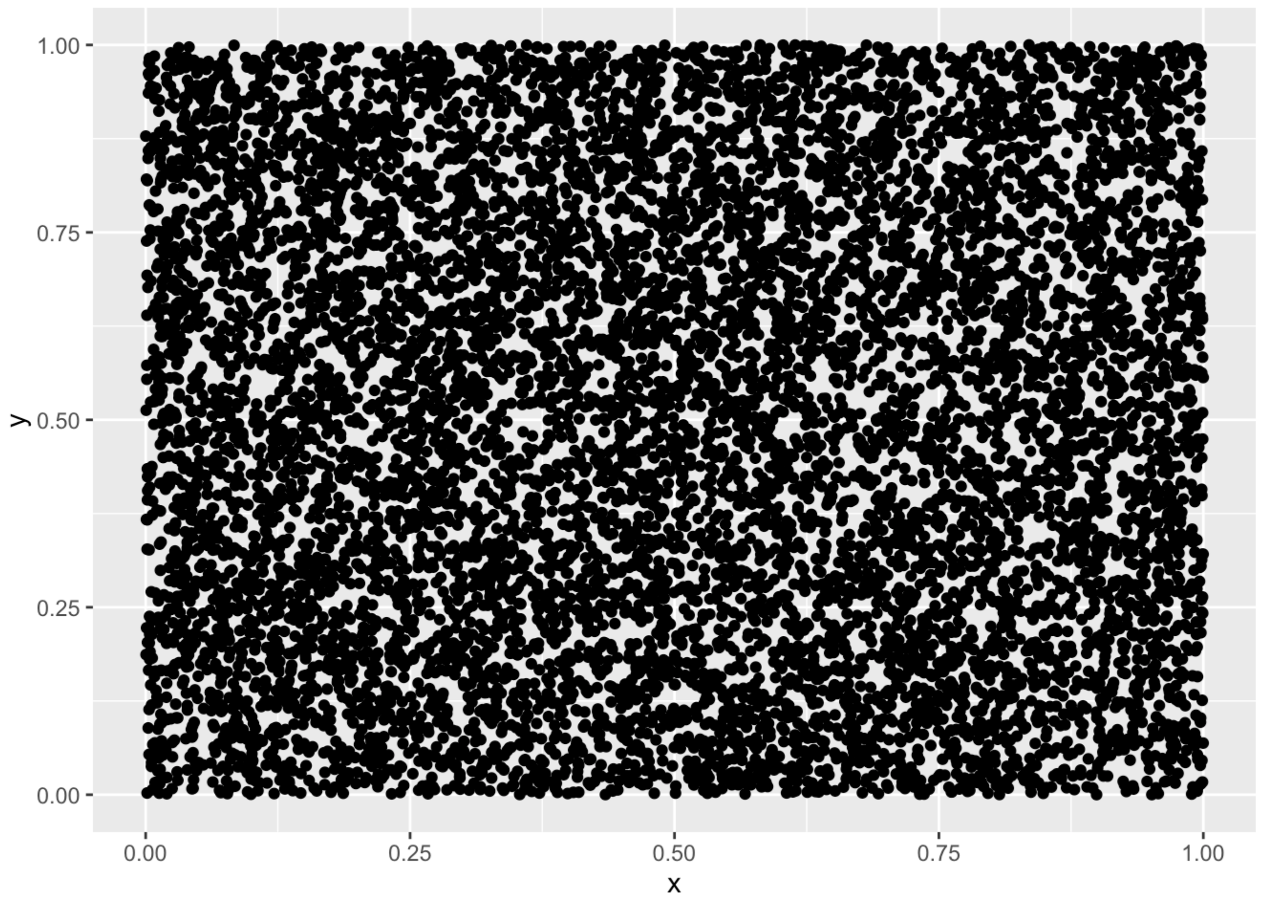
```
ggplot(df, aes(x = x)) + geom_histogram(fill="yellow", color="blue")
```

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



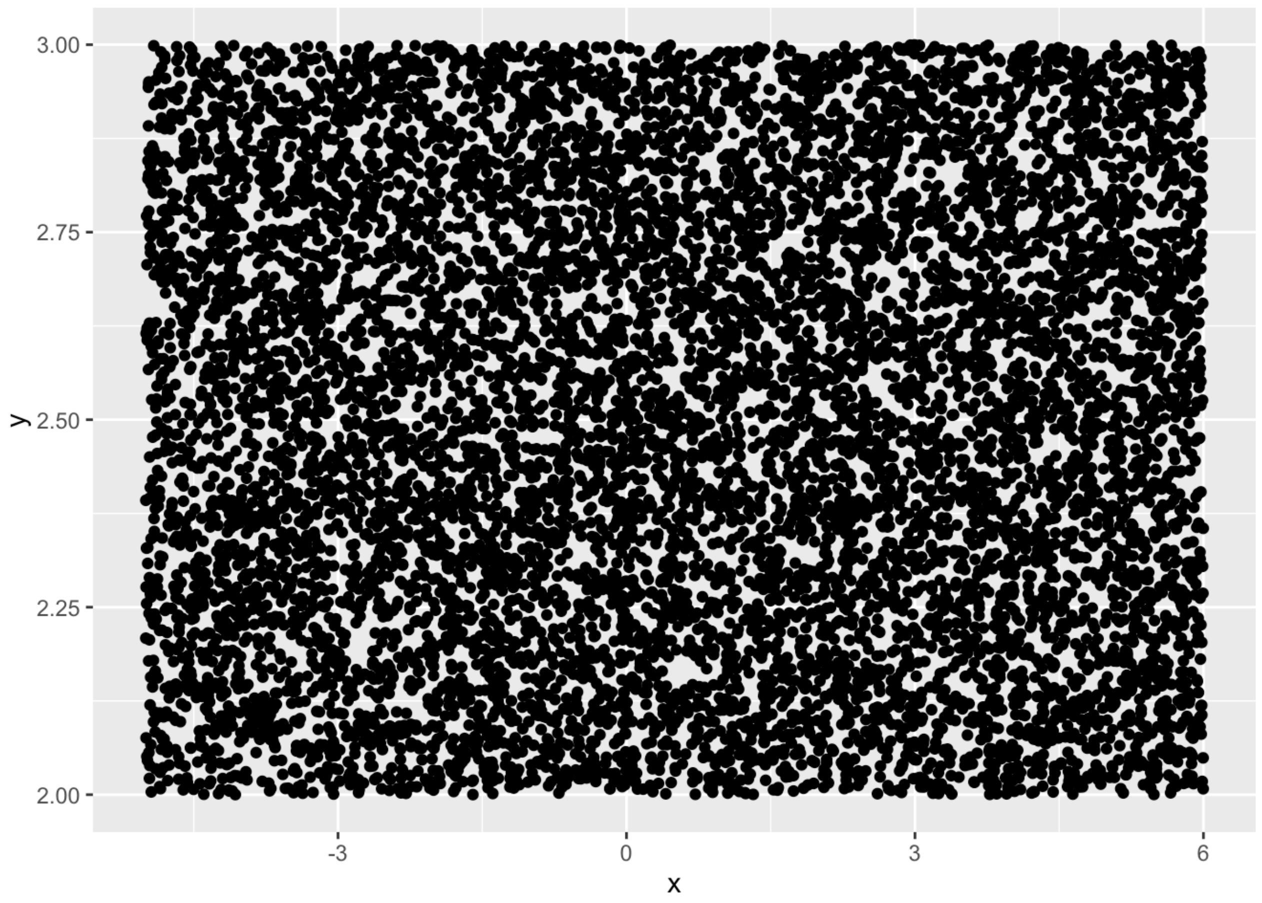
One of the benefits of the data frame is that we can have multiple variables.

```
df = data.frame(x = runif(N), y = runif(N))
ggplot(df, aes(x = x, y = y)) + geom_point()
```



The `runif` function can take in two other arguments that are the range of the uniform random variable. The syntax `runif(n, min, max)` produces `n` samples uniformly between `min` and `max`.

```
df = data.frame(x = runif(n = N, min = -5, max = 6), y = runif(n = N, min = 2
, max = 3))
ggplot(df, aes(x = x, y = y)) + geom_point()
```

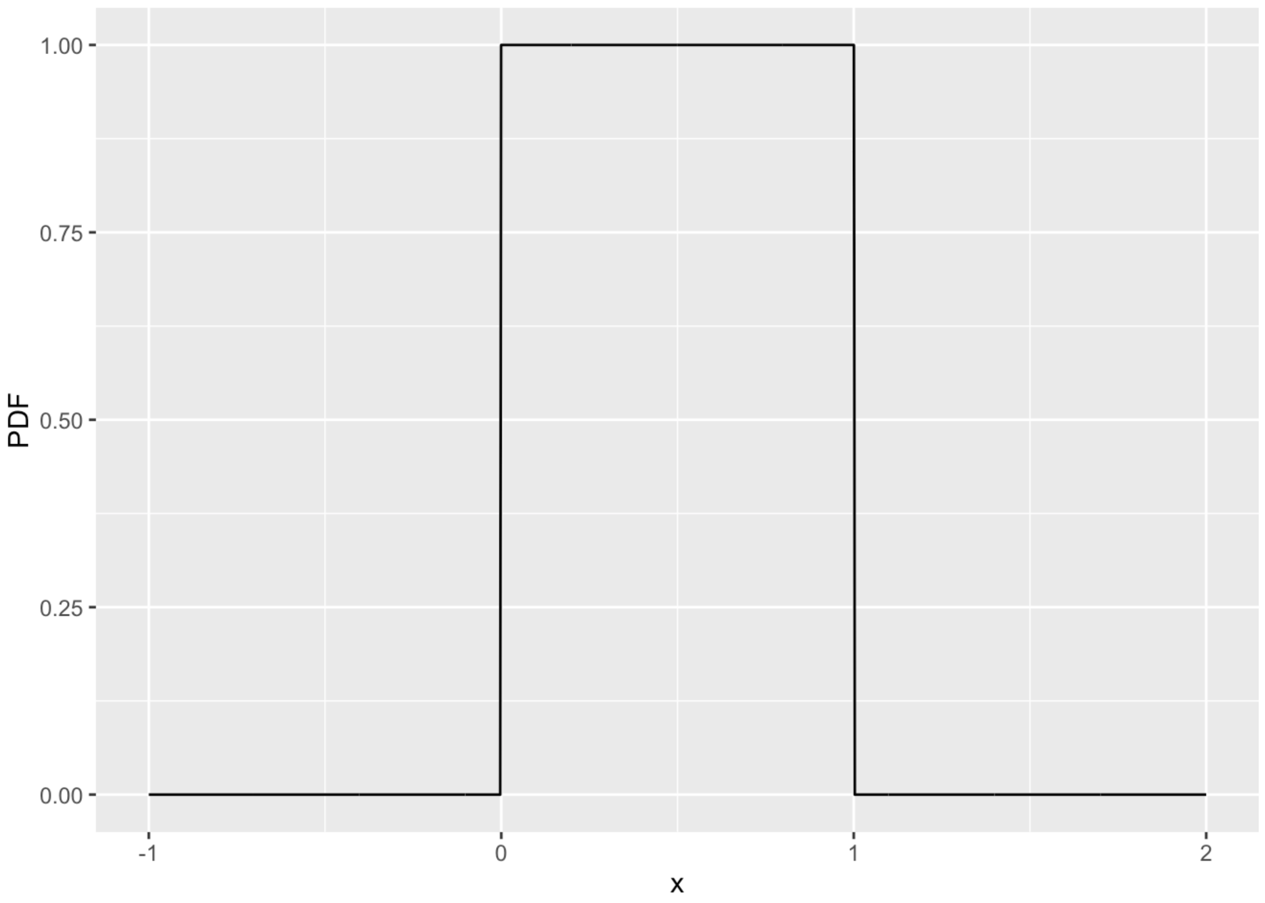


Additionally, R provides functions for the PDF, CDF, and inverse CDF for uniform random variables.

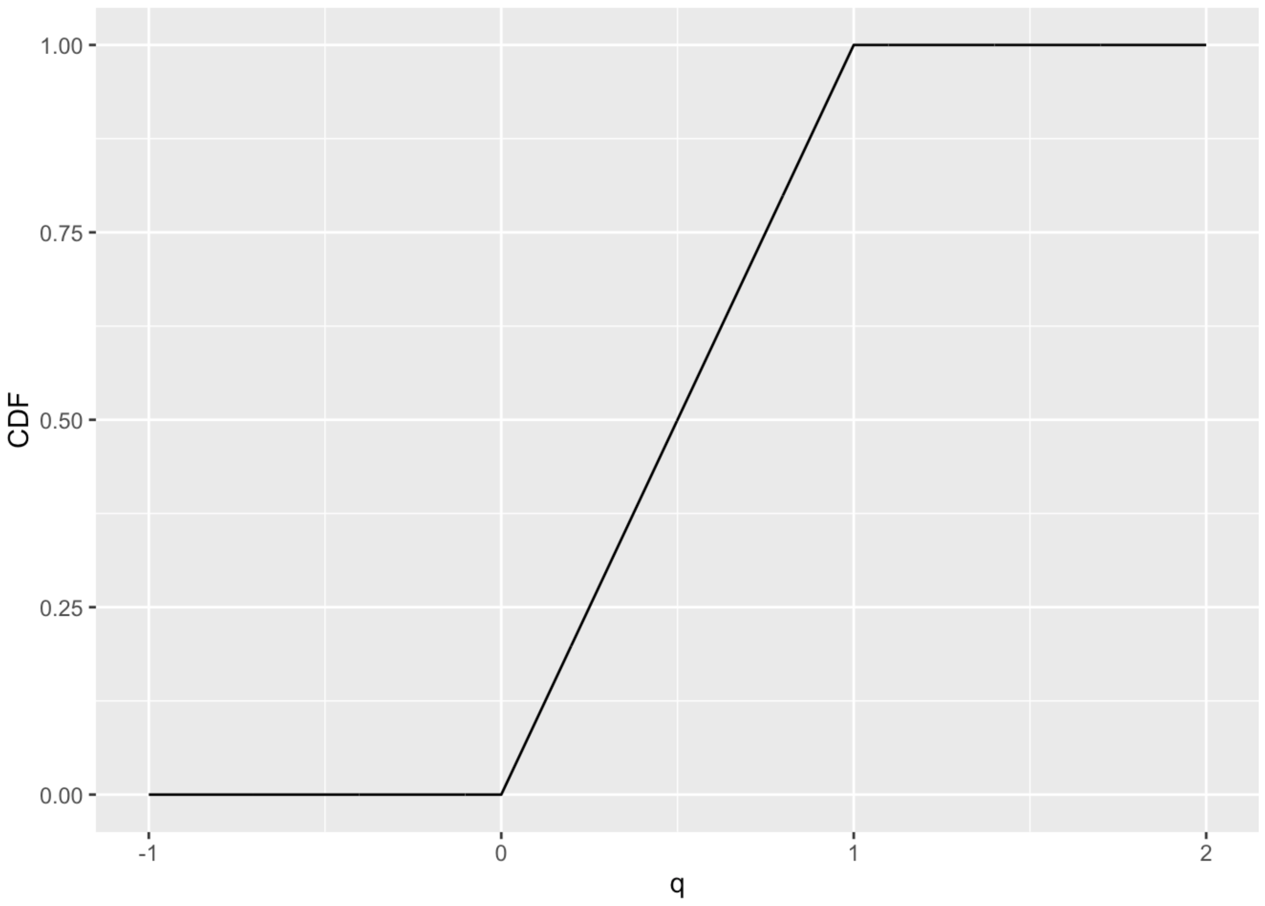
- `dunif(x, min, max)` returns the probability density at `x` for a uniform random variable between `min` and `max`.
- `punif(q, min, max)` returns the value of the CDF at `q` for a uniform random variable between `min` and `max`.
- `quinf(p, min, max)` returns the inverse of the CDF for probability `p` for a uniform random variable between `min` and `max`.

Here we demonstrate the use of these functions next.

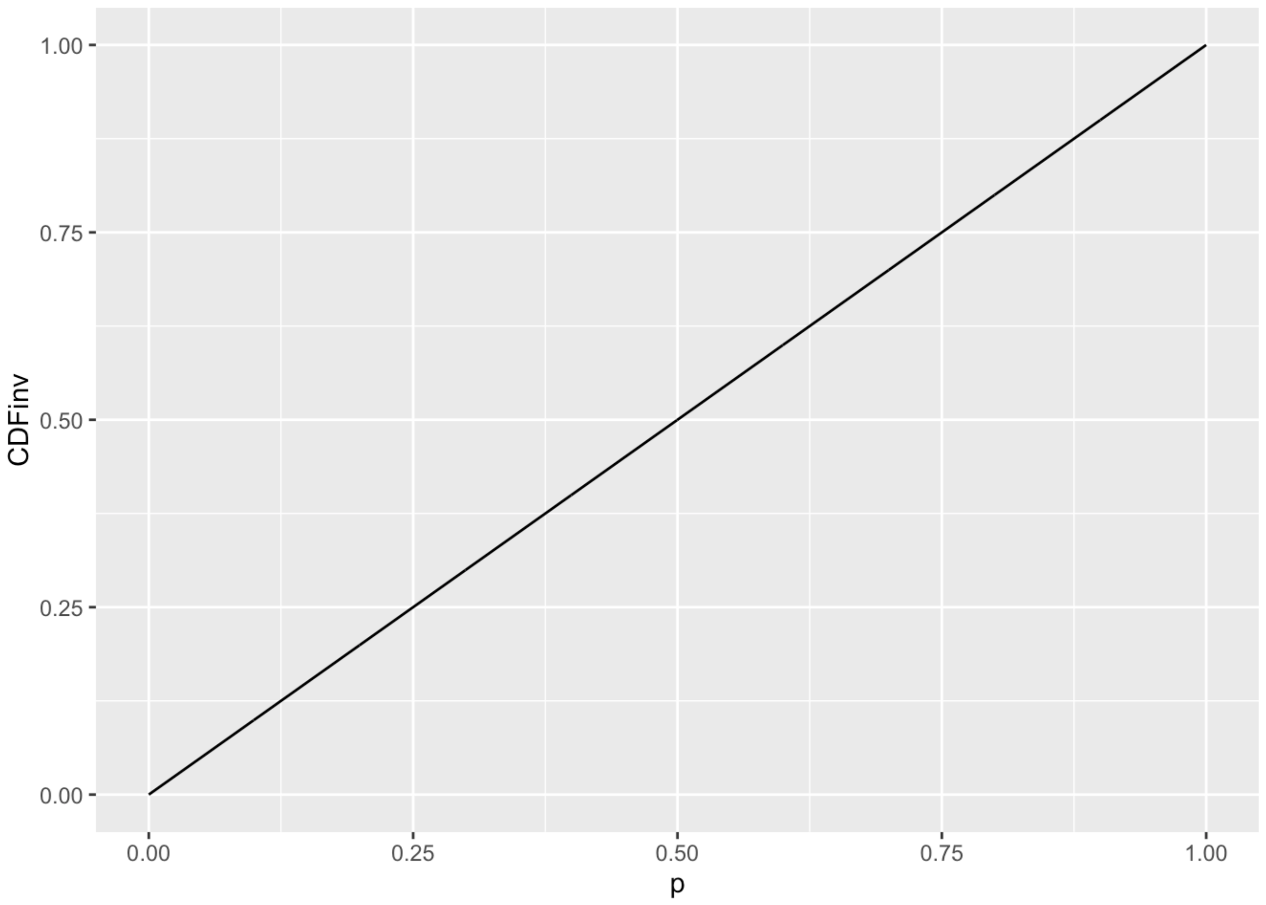
```
N = 1e3
df = data.frame(x = seq(from = -1, to = 2, length.out = N))
df$PDF = dunif(df$x, min = 0, max = 1)
ggplot(df, aes(x = x, y = PDF)) + geom_line()
```



```
N = 1e3
df = data.frame(q = seq(from = -1, to = 2, length.out = N))
df$CDF = punif(df$q, min = 0, max = 1)
ggplot(df, aes(x = q, y = CDF)) + geom_line()
```



```
N = 1e3
df = data.frame(p = seq(from = 0, to = 1, length.out = N))
df$CDFinv = qunif(df$p, min = 0, max = 1)
ggplot(df, aes(x = p, y = CDFinv)) + geom_line()
```

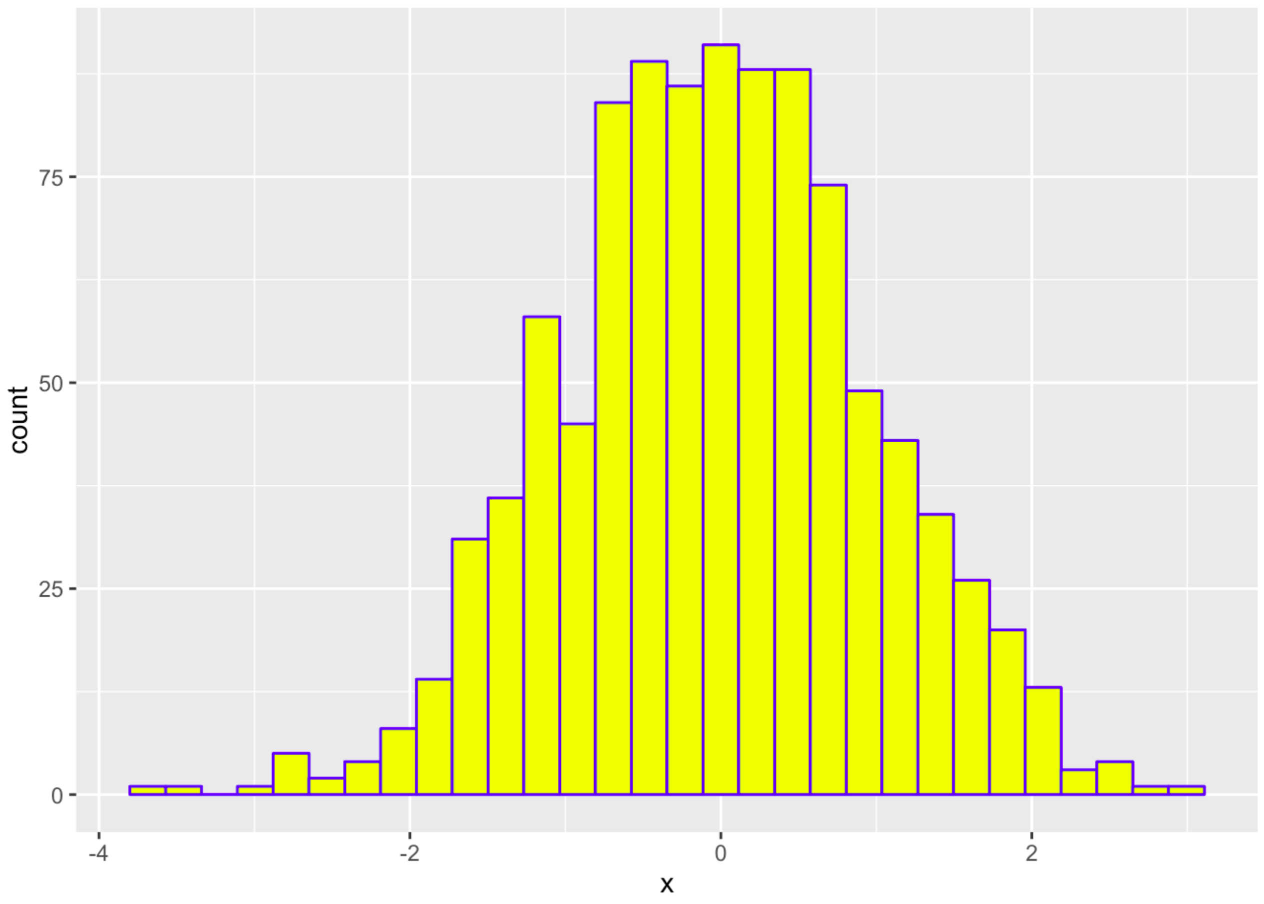


Normal (Gaussian) Random Variables

R also has built in functions for sampling from a normal random variable. The R function is `rnorm(n, mean = 0, sd = 1)`. This gives `n` samples from a normal distribution with mean = `mean` and standard deviation = `sd`. If the mean and standard deviation are not given, the function assumes a standard normal, i.e., `mean` = 0 and `sd` = 1.

```
N = 1e3
df = data.frame(x = rnorm(n = N))
ggplot(df, aes(x = x)) +  geom_histogram(fill="yellow", color="blue")
```

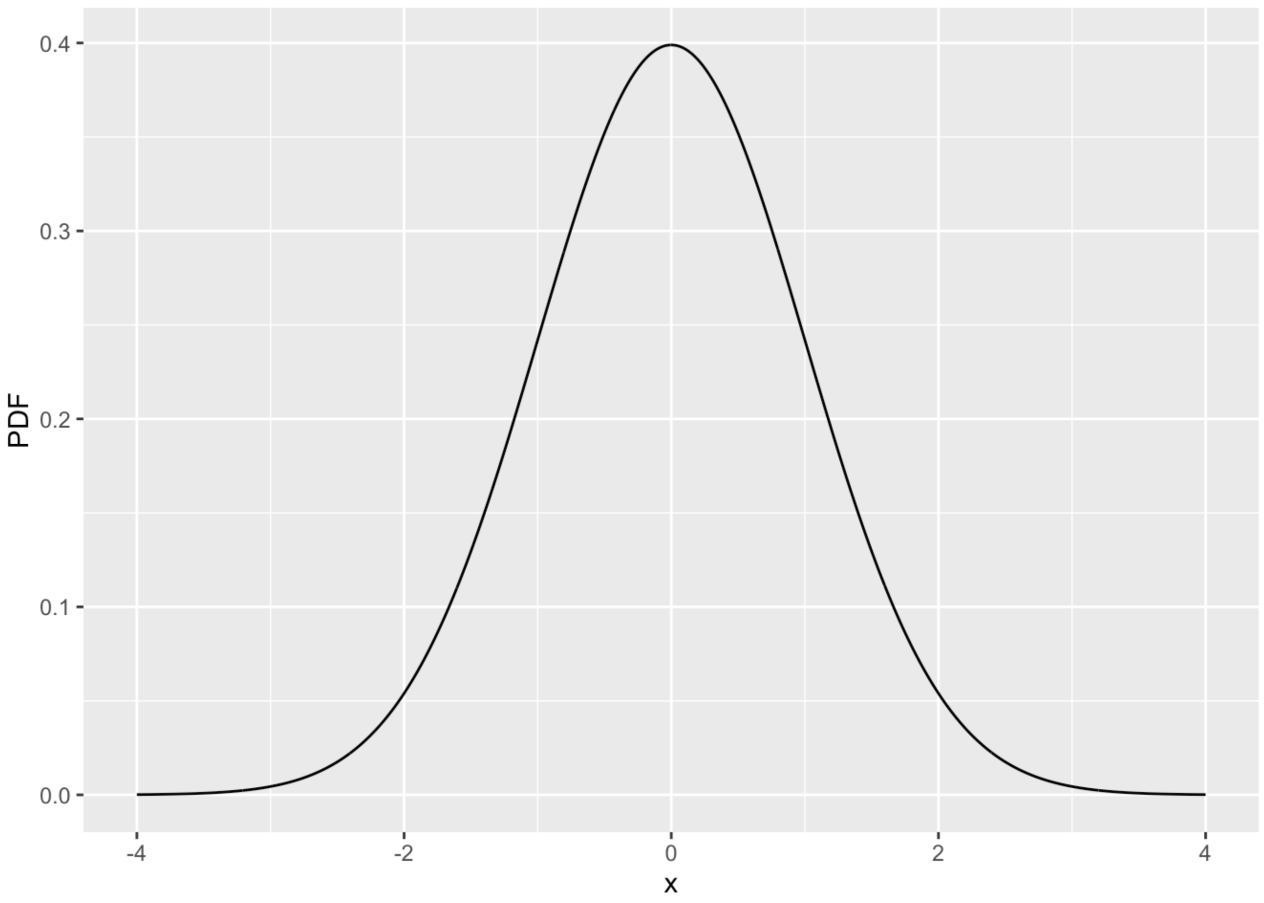
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



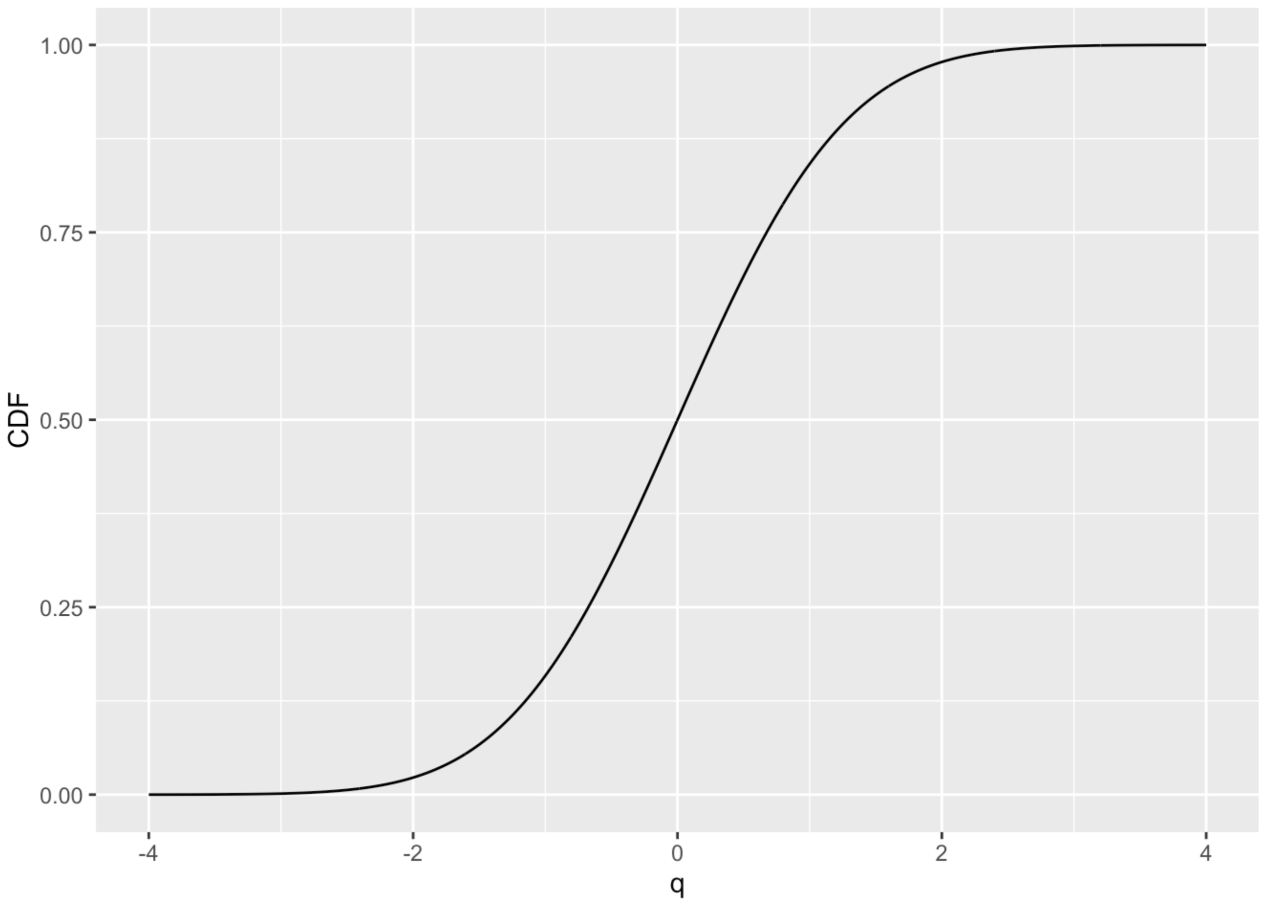
Furthermore, R also provides the PDF, CDF, and inverse CDF functions for normal random variables.

- `dnorm(x, mean = 0, sd = 1)` returns the probability density at `x` for a normal random variable with mean = `mean`, and standard deviation = `sd`.
- `pnorm(q, mean = 0, sd = 1)` returns the value of the CDF at `q` for a normal random variable with mean = `mean`, and standard deviation = `sd`.
- `qnorm(p, mean = 0, sd = 1)` returns the inverse of the CDF for probability `p` for a normal random variable with mean = `mean`, and standard deviation = `sd`.

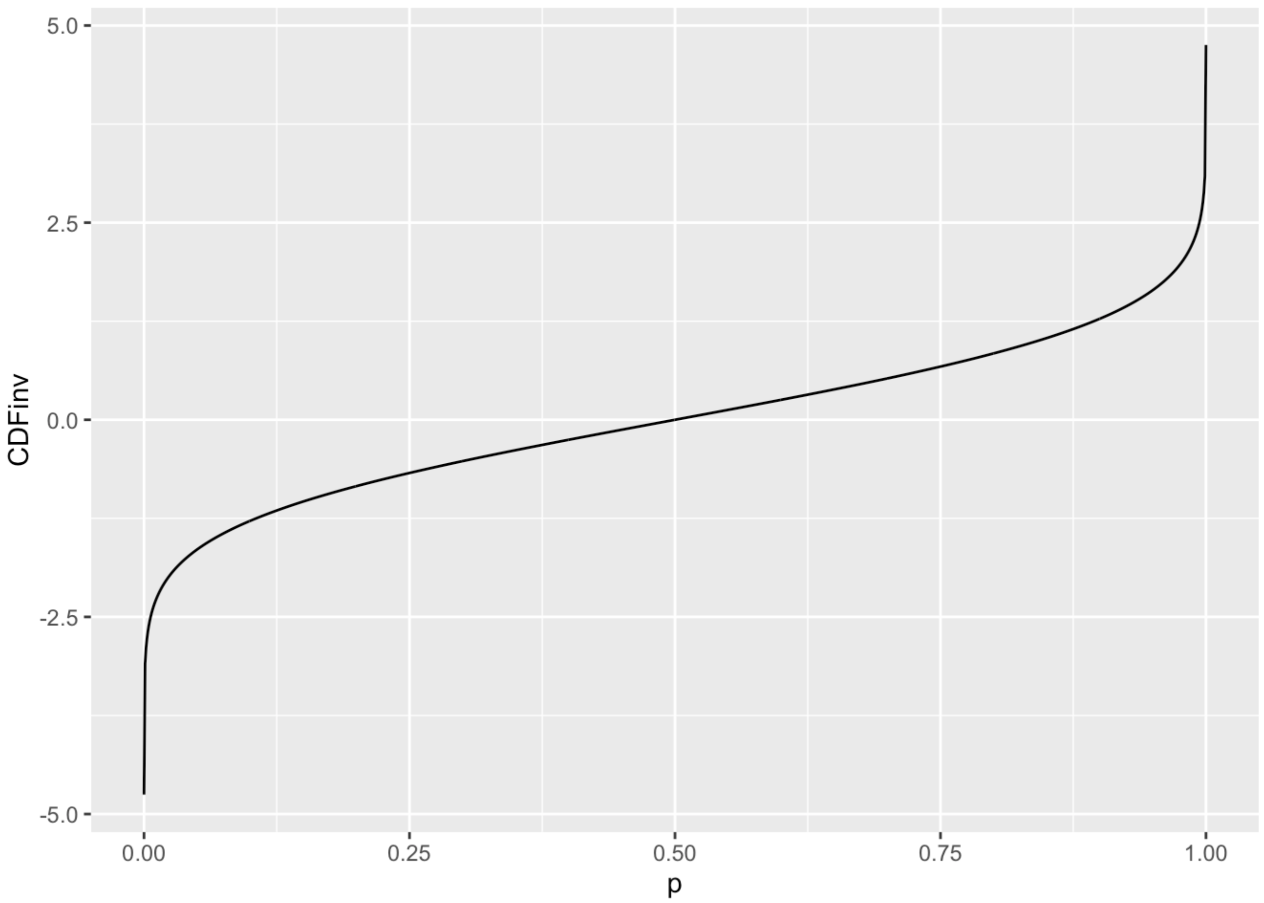
```
N = 1e3
df = data.frame(x = seq(from = -4, to = 4, length.out = N))
df$PDF = dnorm(df$x)
ggplot(df, aes(x = x, y = PDF)) + geom_line()
```



```
N = 1e3
df = data.frame(q = seq(from = -4, to = 4, length.out = N))
df$CDF = pnorm(df$q)
ggplot(df, aes(x = q, y = CDF)) + geom_line()
```



```
N = 1e3
df = data.frame(p = seq(from = 1e-6, to = 1-1e-6, length.out = N))
df$CDFinv = qnorm(df$p)
ggplot(df, aes(x = p, y = CDFinv)) + geom_line()
```



Other distributions

R also has several other built in distributions that you can use. They have the same interface as the normal and uniform distributions in terms of the prefixes `r`, `d`, `p`, and `q`. The functions for each for generating random variables from certain distributions are

- Beta: `rbeta`
- Binomial: `rbinom`
- Cauchy: `rcauchy`
- Exponential: `rexp`
- Gamma: `rgamma`
- Possion: `rpoisson`

Multivariate Normal

To sample from a multivariate normal we use the `mvrnorm` function from the `MASS` library. This function has the inputs given by `mvrnorm(n = 1, mu, Sigma)` where `mu` is a vector of length `n` with mean of each variable, and `Sigma` is the symmetric, positive-definite covariance matrix. The code below produces samples from a 2-D multivariate normal.

```
N = 1e4
covar = matrix(c(1,0.35,0.35,0.5), nrow = 2, ncol = 2)
bvn = mvrnorm(N,mu = c(0,0), Sigma= covar)
bvn = data.frame(V1 = bvn[,1], V2 = bvn[,2])
```

Now a fancy plot

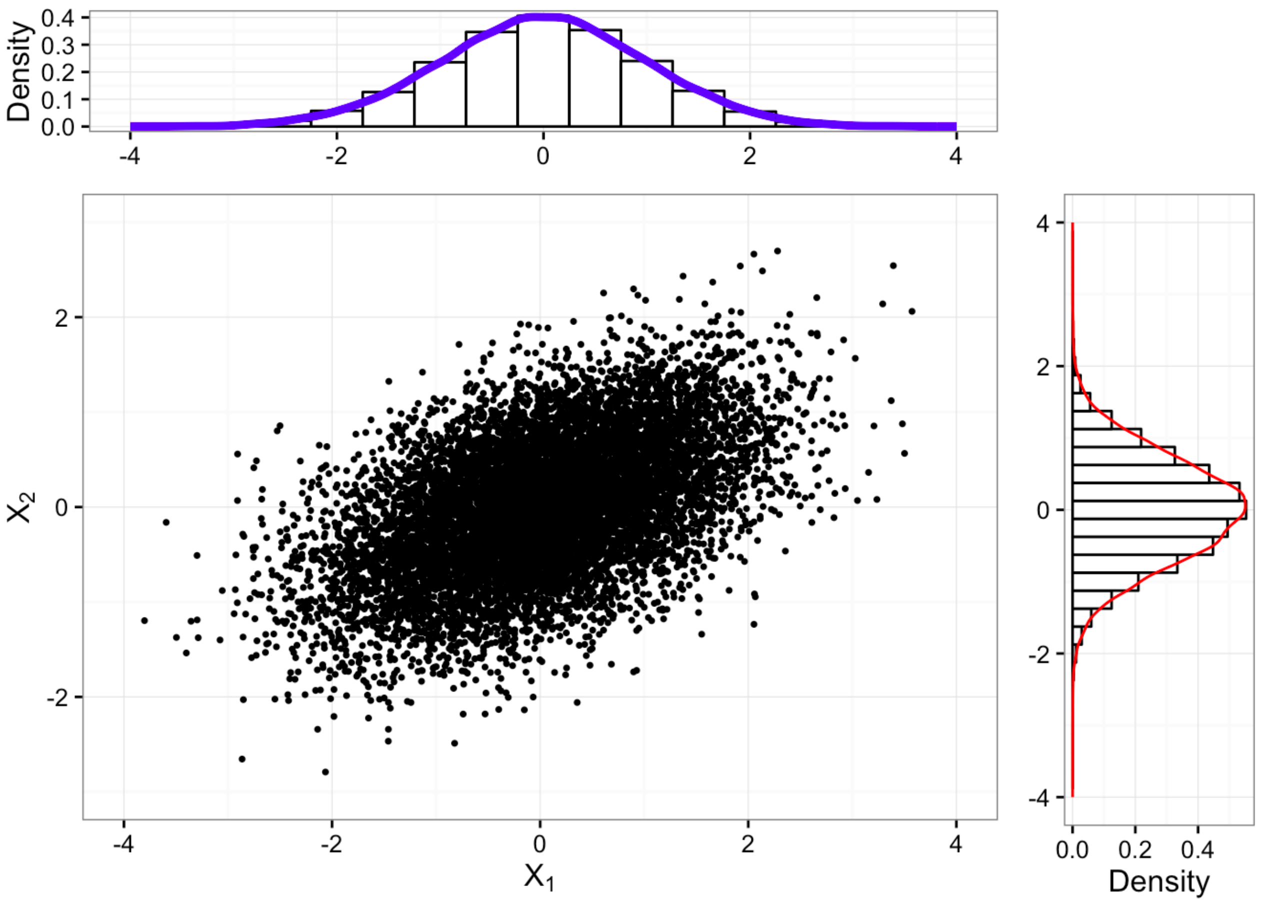
```
htop <- ggplot(data=bvn, aes(x=V1)) +
  geom_histogram(aes(y=..density..), fill = "white", color = "black", binwidt
h = 0.5) +
  stat_density(colour = "blue", geom="line", size = 1.5, position="identity",
 show_guide=FALSE) +
  scale_x_continuous("X1", limits = c(-4,4), breaks = c(-4,-2,0,2,4)) +
  scale_y_continuous("Density")+  theme_bw() + theme(axis.title.x = element_b
lank())

blank <- ggplot() + geom_point(aes(1,1), colour="white") +
  theme(axis.ticks=element_blank(), panel.background=element_blank(), panel.g
rid=element_blank(),
        axis.text.x=element_blank(), axis.text.y=element_blank(), axis.title.
x=element_blank(), axis.title.y=element_blank())

scatter <- ggplot(data=bvn, aes(x=V1, y=V2)) +
  geom_point(size = 0.6) +
  scale_x_continuous(expression(X[1]), limits = c(-4,4), breaks = c(-4,-2,0,2
,4)) +
  scale_y_continuous(expression(X[2]), limits = c(-3,3), breaks = c(-4,-2,0,2
,4)) +
  theme_bw()

hright <- ggplot(data=bvn, aes(x=V2)) +
  geom_histogram(aes(y=..density..), fill = "white", color = "black", binwidt
h = 0.25) +
  stat_density(colour = "red", geom="line", size = 0.5, position="identity",
show_guide=FALSE) +
  scale_x_continuous("X2", limits = c(-4,4), breaks = c(-4,-2,0,2,4)) +
  scale_y_continuous("Density")+  coord_flip() + theme_bw() + theme(axis.titl
e.y = element_blank())

grid.arrange(htop, blank, scatter, hright, ncol=2, nrow=2, widths=c(4, 1), he
ights=c(1, 4))
```


Density
0.4
0.3
0.2
0.1
0.0
-4
-2
0
2
4
X_2
X_1
Density
0.0
0.2
0.4