Common Probability Distributions

R Function Conventions

R provides functions to do four different types of things with a probability distribution:

Task	Function name starts with...
Evaluate the **d**ensity function $f(x)$	d
Calculate the **p**robability $P(X \leq x)$	p
Find a **q**uantile	q
Generate a **r**andom number	r

I'll illustrate each of these with the `norm`al distribution:

1. Evaluate $f(x)$, the density function (pdf for continuous random variables or pmf for discrete random variables)

```
dnorm(0.5, mean = 0, sd = 1, log = FALSE)
```

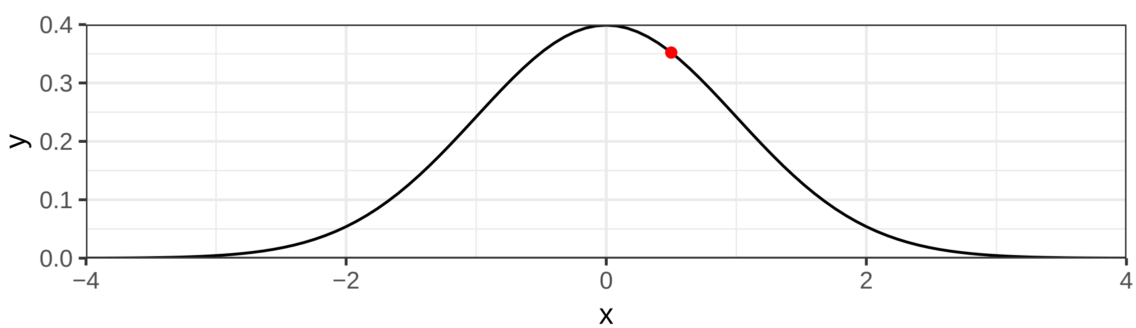
```
## [1] 0.3520653
```

This gives the vertical axis coordinate of the pdf (or pmf) evaluated at $x = 0.5$:

```
# evaluate the normal(0, 1) pdf at x = 0.5 and store in a data frame
evaluated_density_to_plot <- data.frame(
  x = 0.5,
  y = dnorm(0.5, mean = 0, sd = 1, log = FALSE)
)
evaluated_density_to_plot
```

```
##     x         y
## 1 0.5 0.3520653
```

```
# make a plot of the density function with a red point at the (x, y) pair found above
library(ggplot2)
ggplot(data = data.frame(x = c(-4, 4)), mapping = aes(x = x)) +
  stat_function(fun = dnorm) +
  geom_point(data = evaluated_density_to_plot, mapping = aes(x = x, y = y), color = "red") +
  coord_cartesian(xlim = c(-4, 4), ylim = c(0, 0.4), expand = FALSE) +
  theme_bw()
```



2. Calculate $P(X \leq q)$, the probability that a value drawn from this distribution is less than q

```r
pnorm(0.5, mean = 0, sd = 1, log = FALSE)
```

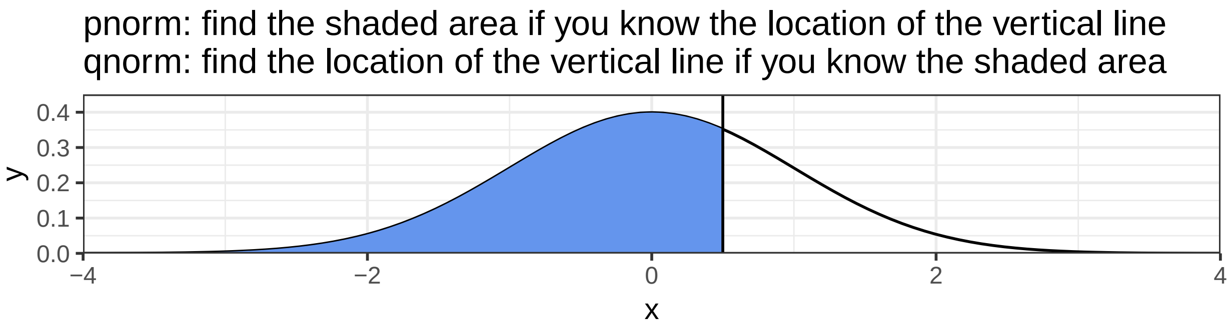
```
## [1] 0.6914625
```

This calculates $P(X \leq 0.5)$ if $X \sim \text{Normal}(0, 1)$, the shaded area below:

```r
x_grid <- seq(from = -4, to = 0.5, length = 101)

region_to_shade <- data.frame(
  x = c(-4, x_grid, 0.5),
  y = c(0, dnorm(x_grid, mean = 0, sd = 1, log = FALSE), 0)
)

ggplot(data = data.frame(x = c(-4, 4)), mapping = aes(x = x)) +
  stat_function(fun = dnorm) +
  geom_polygon(
    mapping = aes(x = x, y = y),
    fill = "cornflowerblue",
    data = region_to_shade) +
  geom_vline(xintercept = 0.5) +
  coord_cartesian(xlim = c(-4, 4), ylim = c(0, 0.45), expand = FALSE) +
  theme_bw() +
  ggtitle("pnorm: find the shaded area if you know the location of the vertical line\nqnorm: find the locatic
```



3. Find quantiles: For a given number p, find the value q such that $P(X \leq q) = p$

```r
qnorm(0.6914625, mean = 0, sd = 1, log = FALSE)
```

```
## [1] 0.5000001
```

This calculates the location of the vertical line in the plot above, given that the area to the left of that line is 0.6914625: $P(X \leq ?) = 0.6914625$

4. Generate random numbers

```r
rnorm(n = 5, mean = 0, sd = 1)
```

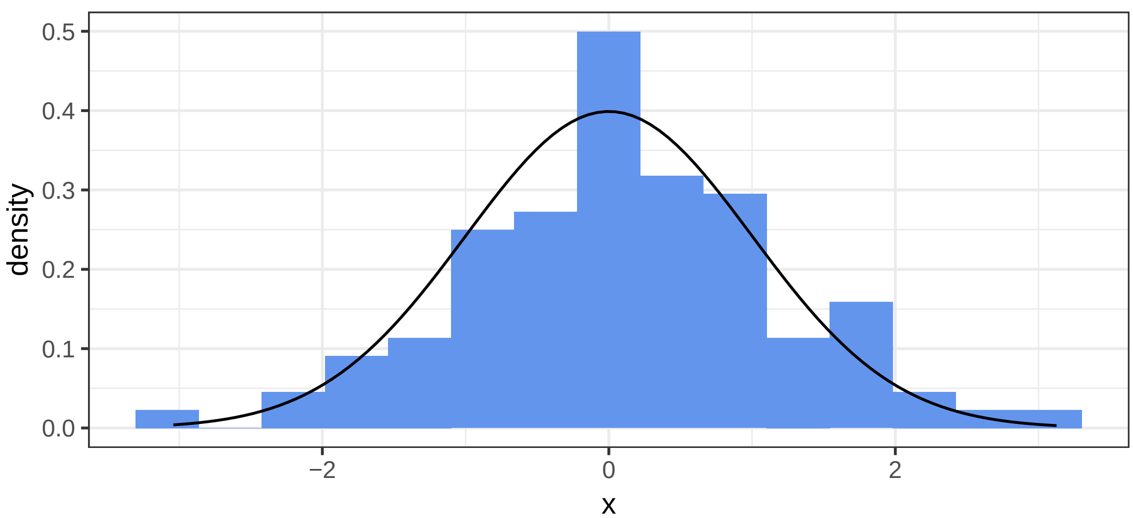
```
## [1] -1.6494928 -2.3563596  0.6839980 -0.5959463  0.8583114
```

If we want to plot these values using ggplot2, it's convenient to store them as a variable in a data frame:

```r
# set a seed for random number generation so that I get reproducible results
set.seed(59749)

# create a new data frame called data_to_plot, with one variable in it: x
data_to_plot <- data.frame(
  x = rnorm(n = 100, mean = 0, sd = 1)
)

# make a plot
ggplot(data = data_to_plot, mapping = aes(x = x)) +
  geom_histogram(mapping = aes(y = ..density..),
    fill = "cornflowerblue",
    bins = 15) +
  stat_function(fun = dnorm) +
  theme_bw()
```



Discrete Distributions

Bernoulli(p)

X = the result of a single experiment with one of two outcomes ("success", coded as 1, or "failure", coded as 0), where the probability of success is p.

parameters	p = probability of success
p.f.	$f(x\|p) = p^x(1-p)^{(1-x)}$
Mean	p
Variance	$p(1-p)$
R functions	`dbinom(..., size = 1, prob = p), pbinom, qbinom, rbinom`

Binomial(n, p)

X = the total number of successes in n independent and identically distributed Bernoulli trials, each with probability of success p.

parameters	n = number of trials, p = probability of success
p.f.	$f(x\|n,p) = \binom{n}{x}p^x(1-p)^{(n-x)}$
Mean	np
Variance	$np(1-p)$
R functions	`dbinom(..., size = n, prob = p), pbinom, qbinom, rbinom`

Uniform(a, b)

X = an integer between a and b (inclusive), where each integer from a to b is equally likely.

parameters	a: lower endpoint, b: upper endpoint
p.f.	$f(x\|a,b) = \frac{1}{b-a+1}$
Mean	$\frac{a+b}{2}$
Variance	$\frac{(b-a)(b-a+2)}{12}$
R functions	No specific `d`, `p`, or `q` functions. For random number generation, you could use `sample(seq(from = a, to = b, by = 1), size = n, replace = TRUE)`

Geometric(p)

X = the number of failures that occur before the first success in a sequence of independent and identically distributed Bernoulli trials.

parameters	$p \in (0,1)$: probability of success on each trial
p.f.	$f(x\|p) = p(1-p)^x$
Mean	$\frac{1-p}{p}$
Variance	$\frac{1-p}{p^2}$
R functions	`dgeom(..., prob = p, ...), pgeom, qgeom, rgeom`

Be careful – there are other parameterizations used in other sources.

Negative Binomial(r, p)

X = the number of failures which occur in a sequence of independent and identically distributed Bernoulli trials before r successes occur.

parameters	$r > 0$: target number of successes, $p > 0$ = probability of success on each trial
p.f.	$f(x\|r, p) = \binom{r+x-1}{x} p^r (1-p)^x$
Mean	$\frac{r(1-p)}{p}$
Variance	$\frac{r(1-p)}{p^2}$
R functions	`dnbinom(..., size = r, prob = p, ...), pnbinom, qnbinom, rnbinom`

Be careful – there are multiple other parameterizations used in other sources.

Hypergeometric(A, B, n)

X = the number of successes in n draws (without replacement) from a finite population that contains exactly A successes and B failures.

parameters	A: number of successes in the population, B: number of failures in the population, n: sample size
p.f.	$f(x\|A, B, n) = \frac{\binom{A}{x}\binom{B}{n-x}}{\binom{A+B}{n}}$
Mean	$\frac{nA}{A+B}$
Variance	$\frac{nAB}{(A+B)^2} \frac{A+B-n}{A+B-1}$
R functions	`dhyper(..., m = A, n = B, k = n, ...), phyper, qhyper, rhyper`

Poisson(λ)

X = the number of events occurring in a fixed interval of time or space if these events occur with a known constant rate and independently of the time since the last event.

parameters	λ: rate parameter
p.f.	$f(x\|\lambda) = e^{-\lambda} \frac{\lambda^x}{x!}$
Mean	λ
Variance	λ
R functions	`dpois(..., lambda = lambda), ppois, qpois, rpois`

Multinomial(n, p), where $p = (p_1, p_2, \dots, p_k)$

$X = (X_1, X_2, \dots, X_k)$ = the vector of counts for how many observations fell into each of k categories in a sample of n independent trials where the item sampled in each trial falls into category j with probability p_j. (Roll a weighted die with k sides n times. How many times did each face of the die come up?)

parameters	n: number of trials, $p = (p_1, p_2, \dots, p_k)$: vector of probabilities for each category
p.f.	$f(x\|p) = \frac{n!}{x_1! x_2! \cdots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}$
Mean	$E(X_i) = np_i$
Variance	$Var(X_i) = np_i(1-p_i)$
R functions	`dmultinom(..., size = n, prob = p), rmultinom`

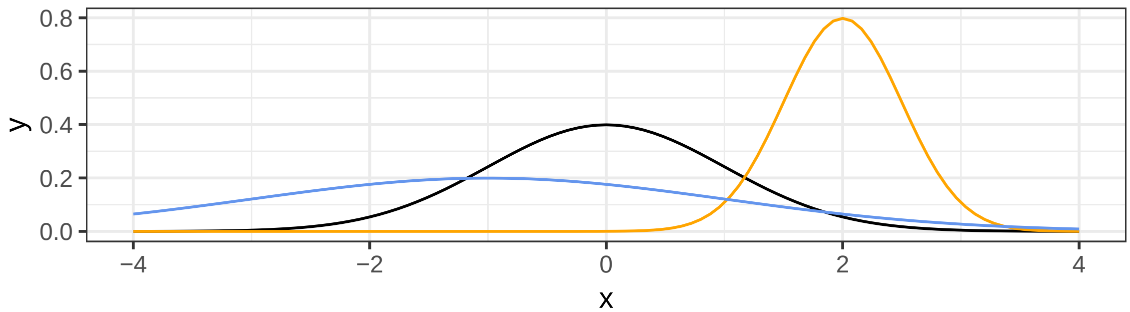
Continous Distributions

Normal(μ, σ^2)

A real number.

parameters	μ: mean, $\sigma^2 > 0$: variance
p.f.	$f(x\|\mu, \sigma^2) = (2\pi\sigma^2)^{-\frac{1}{2}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$
Mean	μ
Variance	σ^2
R functions	`dnorm(..., mean =` μ`, sd =` σ`), pnorm, qnorm, rnorm`

```r
ggplot(data = data.frame(x = c(-4, 4)), mapping = aes(x = x)) +
  stat_function(fun = dnorm, args = list(mean = 0, sd = 1), color = "black") +
  stat_function(fun = dnorm, args = list(mean = 2, sd = 0.5), color = "orange") +
  stat_function(fun = dnorm, args = list(mean = -1, sd = 2), color = "cornflowerblue") +
  theme_bw()
```

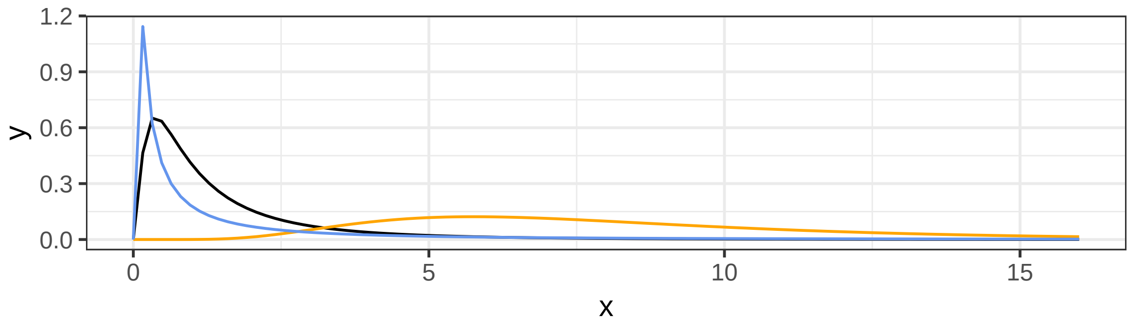


Lognormal(μ, σ^2)

A positive number. If $X \sim$ Lognormal(μ, σ^2) and $Y = \log(X)$, then $Y \sim$ Normal(μ, σ^2).

parameters	μ: mean of $\log(X)$, $\sigma^2 > 0$: variance of $\log(X)$
p.f.	$f(x\|\mu, \sigma^2) = x^{-1}(2\pi\sigma^2)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\frac{\{\log(x)-\mu\}^2}{\sigma^2}\right]$
Mean	$\exp\left(\mu + \frac{\sigma^2}{2}\right)$
Variance	$\{\exp(\sigma^2) + 1\} \exp(2\mu + \sigma^2)$
R functions	`dlnorm(..., meanlog =` μ`, sdlog =` σ`), plnorm, qlnorm, rlnorm`

```r
ggplot(data = data.frame(x = c(0, 16)), mapping = aes(x = x)) +
  stat_function(fun = dlnorm, args = list(meanlog = 0, sdlog = 1), color = "black") +
  stat_function(fun = dlnorm, args = list(meanlog = 2, sdlog = 0.5), color = "orange") +
  stat_function(fun = dlnorm, args = list(meanlog = -1, sdlog = 2), color = "cornflowerblue") +
  theme_bw()
```



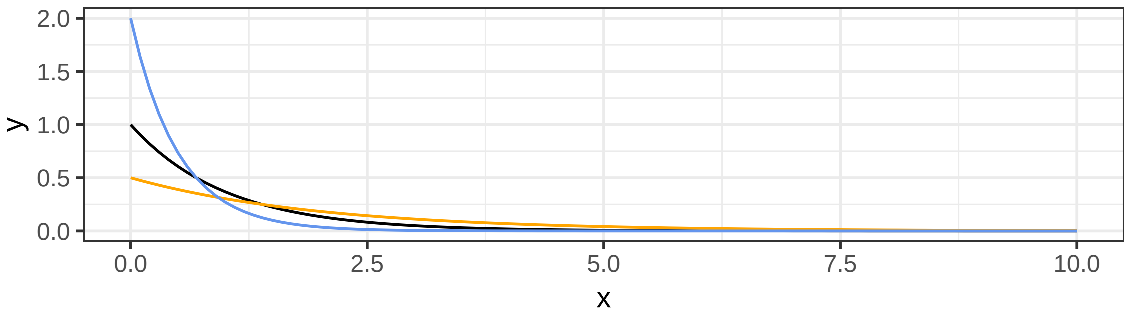
Exponential(λ)

A non-negative real number. Often used as a model for waiting times – but need to check whether this is a good model for a given data set.

parameters	$\lambda > 0$: rate parameter
p.f.	$f(x\|\lambda) = \lambda e^{-\lambda x}$
Mean	$\frac{1}{\lambda}$
Variance	$\frac{1}{\lambda^2}$
R functions	`dexp(..., rate =` λ`), pexp, qexp, rexp`

Be careful – in some sources, the exponential distribution is parameterized in terms of a scale parameter $\beta = \frac{1}{\lambda}$

```
ggplot(data = data.frame(x = c(0, 10)), mapping = aes(x = x)) +
  stat_function(fun = dexp, args = list(rate = 1), color = "black") +
  stat_function(fun = dexp, args = list(rate = 0.5), color = "orange") +
  stat_function(fun = dexp, args = list(rate = 2), color = "cornflowerblue") +
  theme_bw()
```



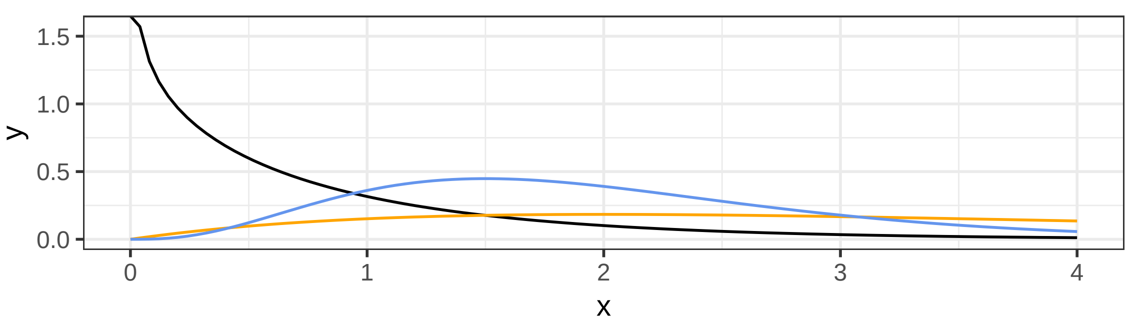
Gamma(α, λ)

A non-negative real number. Often used as a model for waiting times – but need to check whether this is a good model for a given data set.

parameters	$\alpha \geq 0$: shape parameter, $\lambda > 0$: rate parameter
p.f.	$f(x\|\alpha, \lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x}$
Mean	$\frac{\alpha}{\lambda}$
Variance	$\alpha\lambda^2$
R functions	`dgamma(..., shape =` α`, rate =` λ`), pgamma, qgamma, rgamma`

Be careful – there are multiple other parameterizations used in other sources.

```
ggplot(data = data.frame(x = c(0, 4)), mapping = aes(x = x)) +
  stat_function(fun = dgamma, args = list(shape = 0.8, rate = 1), color = "black") +
  stat_function(fun = dgamma, args = list(shape = 2, rate = 0.5), color = "orange") +
  stat_function(fun = dgamma, args = list(shape = 4, rate = 2), color = "cornflowerblue") +
  theme_bw()
```

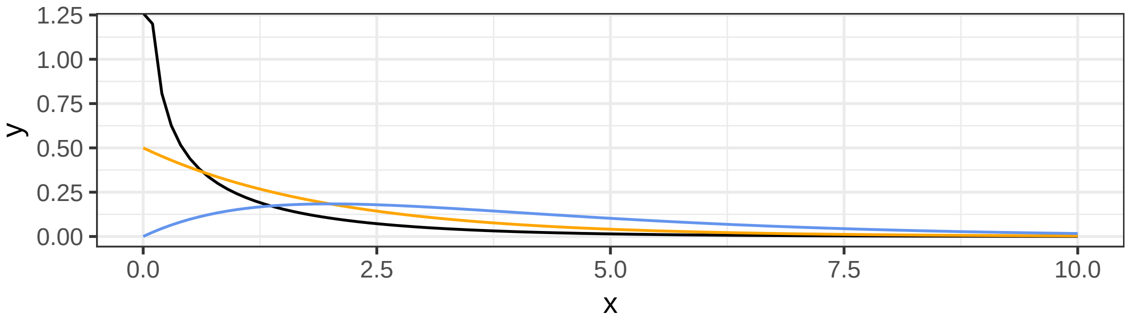


Chi-Squared(n)

A non-negative real number.

parameters	n: degrees of freedom
p.f.	$f(x\|n) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{\frac{n}{2}-1} e^{-\frac{x}{2}}$
Mean	n
Variance	$2n$
R functions	`dchisq(..., df = n)`, `pchisq`, `qchisq`, `rchisq`

```r
ggplot(data = data.frame(x = c(0, 10)), mapping = aes(x = x)) +
  stat_function(fun = dchisq, args = list(df = 1), color = "black") +
  stat_function(fun = dchisq, args = list(df = 2), color = "orange") +
  stat_function(fun = dchisq, args = list(df = 4), color = "cornflowerblue") +
  theme_bw()
```

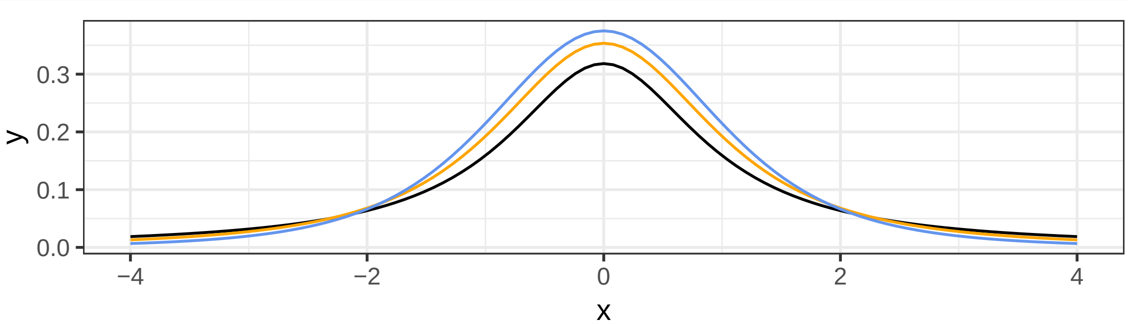


t_ν

A real number.

parameters	ν: degrees of freedom
p.f.	$f(x\|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}$
Mean	0 for $\nu > 1$, otherwise undefined
Variance	$\frac{\nu}{\nu-2}$ for $\nu > 2$
R functions	`dt(..., df = ν)`, `pt`, `qt`, `rt`

```r
ggplot(data = data.frame(x = c(-4, 4)), mapping = aes(x = x)) +
  stat_function(fun = dt, args = list(df = 1), color = "black") +
  stat_function(fun = dt, args = list(df = 2), color = "orange") +
  stat_function(fun = dt, args = list(df = 4), color = "cornflowerblue") +
  theme_bw()
```

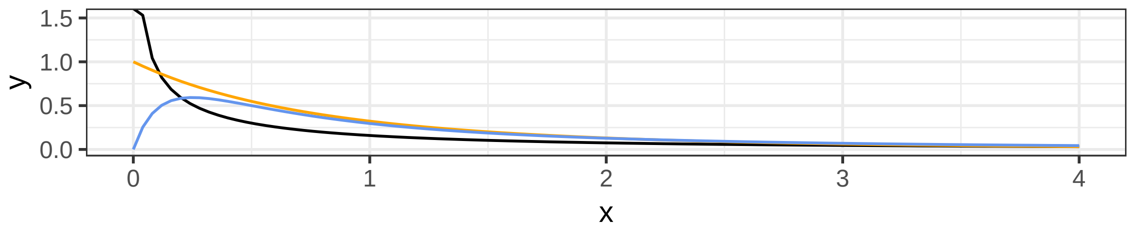


$\mathbf{F}(\nu_1, \nu_2)$

A non-negative real number.

parameters	ν_1: degrees of freedom, ν_2: degrees of freedom
R functions	df(..., df1 = ν_1, df2 = ν_2), pf, qf, rf

```
ggplot(data = data.frame(x = c(0, 4)), mapping = aes(x = x)) +
  stat_function(fun = df, args = list(df1 = 1, df2 = 1), color = "black") +
  stat_function(fun = df, args = list(df1 = 2, df2 = 7), color = "orange") +
  stat_function(fun = df, args = list(df1 = 4, df2 = 2), color = "cornflowerblue") +
  theme_bw()
```



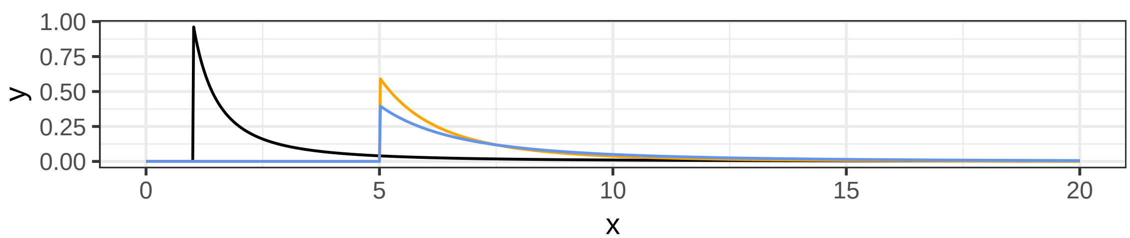
$\mathbf{Pareto}(x_0, \alpha)$

A real number that is greater than or equal to x_0.

parameters	x_0: lower bound of support, α: shape parameter
p.f.	$f(x\|x_0, \alpha) = \frac{\alpha x_0^\alpha}{x^{\alpha+1}} \mathbb{I}_{[x_0,\infty)}(x) = \begin{cases} \frac{\alpha x_0^\alpha}{x^{\alpha+1}} \text{ if } x \geq x_0 \\ 0 \text{ otherwise} \end{cases}$
R functions	Not provided as part of base R

```
#' Calculate pdf of a Pareto distribution
#'
#' @param x vector of real numbers at which to evaluate the Pareto density
#' @param x_0 location parameter for the Pareto distribution
#' @param alpha scale parameter for the Pareto distribution
#'
#' @return vector of the same length as x of values of the Pareto density function.
dpareto <- function(x, x_0, alpha) {
    result <- rep(-Inf, length(x))
    inds_x_greater_x_0 <- (x > x_0)
    result[inds_x_greater_x_0] <- log(alpha) + alpha * log(x_0) - (alpha + 1) * log(x[inds_x_greater_x_0])
    return(exp(result))
}

ggplot(data = data.frame(x = c(0, 20)), mapping = aes(x = x)) +
    stat_function(fun = dpareto, args = c(x_0 = 1, alpha = 1), n = 1001) +
    stat_function(fun = dpareto, args = c(x_0 = 5, alpha = 3), n = 1001, color = "orange") +
    stat_function(fun = dpareto, args = c(x_0 = 5, alpha = 2), n = 1001, color = "cornflowerblue") +
  theme_bw()
```

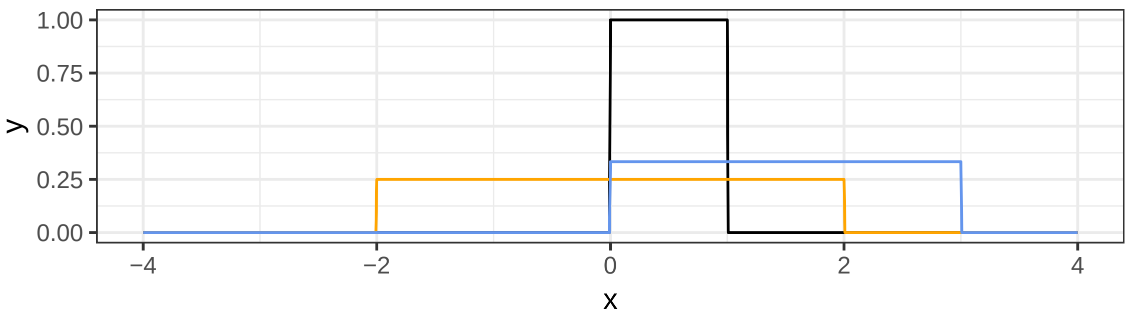


Uniform(a, b)

A real number between a and b (inclusive).

parameters	a: lower endpoint, b: upper endpoint
p.f.	$f(x\|a,b) = \begin{cases} \frac{1}{b-a} \text{ if } x \in [a,b] \\ 0 \text{ otherwise} \end{cases}$
Mean	$\frac{1}{2}(a+b)$
Variance	$\frac{1}{12}(b-a)^2$
R functions	`dunif(..., min = ` a`, max = `b`), punif, qunif, runif`

```
ggplot(data = data.frame(x = c(-4, 4)), mapping = aes(x = x)) +
  stat_function(fun = dunif, args = list(min = 0, max = 1), color = "black", n = 1001) +
  stat_function(fun = dunif, args = list(min = -2, max = 2), color = "orange", n = 1001) +
  stat_function(fun = dunif, args = list(min = 0, max = 3), color = "cornflowerblue", n = 1001) +
  theme_bw()
```



Beta(a, b)

A real number between 0 and 1.

parameters	a: lower endpoint, b: upper endpoint
p.f.	$f(x\|a,b) = \begin{cases} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{(a-1)}(1-x)^{(b-1)}, \text{ if } 0 \leq x \leq 1 \\ 0 \text{ otherwise} \end{cases}$
Mean	$\frac{a}{a+b}$
Variance	$\frac{ab}{(a+b)^2(a+b+1)}$
R functions	`dbeta(..., min = ` a`, max = `b`), punif, qunif, runif`

```
ggplot(data = data.frame(x = c(0, 1)), mapping = aes(x = x)) +
  stat_function(fun = dbeta, args = list(shape1 = 1, shape2 = 1), color = "black") +
  stat_function(fun = dbeta, args = list(shape1 = 0.5, shape2 = 2), color = "orange") +
  stat_function(fun = dbeta, args = list(shape1 = 10, shape2 = 3), color = "cornflowerblue") +
  theme_bw()
```