

Chapter 3: Methods for Simulating Data

Statisticians (and other users of data) need to simulate data for many reasons.

For example, I simulate as a way to check whether a model is appropriate. If the observed data are similar to the data I generated, then this is one way to show my model may be a good one.

It is also sometimes useful to simulate data from a distribution when I need to estimate an expected value (approximate an integral).

- ch. 5

R can already generate data from many (named) distributions:

```
set.seed(400) #reproducibility
```

```
rnorm(10) # 10 observations of a N(0,1) r.v.
```

```
## [1] -1.0365488 0.6152833 1.4729326 -0.6826873 -0.6018386 -1.3526097  
## [7] 0.8607387 0.7203705 0.1078532 -0.5745512
```

```
rnorm(10, 0, 5) # 10 observations of a N(0,5^2) r.v.
```

```
## [1] -4.5092359 0.4464354 -7.9689786 -0.4342956 -5.8546081 2.7596877  
## [7] -3.2762745 -2.1184014 2.8218477 -5.0927654
```

```
rexp(10) # 10 observations from an Exp(1) r.v.
```

```
## [1] 0.67720831 0.04377997 5.38745038 0.48773005 1.18690322 0.92734297  
## [7] 0.33936255 0.99803323 0.27831305 0.94257810
```

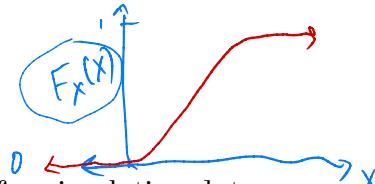
But what about when we don't have a function to do it?

↳ we need to write our own functions to simulate draws from distributions.

1 Inverse Transform Method

"PIT"

Theorem 1.1 (Probability Integral Transform) If X is a continuous r.v. with cdf F_X , then $U = F_X(X) \sim \text{Uniform}[0, 1]$.



This leads to the following method for simulating data.

$$U = F_x(X)$$

$$F_x^{-1}(U) = F_x^{-1}(F_x(X)) = X$$

First, generate u from $\text{Uniform}[0, 1]$. Then, $x = F_X^{-1}(u)$ is a realization from F_X .

Note: F may not be available in closed form. If that's the case, use something else.

1.1 Algorithm

1. Derive the inverse function F_X^{-1} . To do this, let $F(x) = u$. Then solve for x to find $x = F^{-1}(u)$.

2. Write a function to compute $x = F_X^{-1}(u)$.

\hookrightarrow in R

3. For each realization, \rightarrow simulated value.

a. generate a random value u from $\text{unif}(0, 1)$.

b. Compute $x = F^{-1}(u)$.

Typically repeat a-b many times.

Example 1.1 Simulate a random sample of size 1000 from the pdf $f_X(x) = 3x^2$, $0 \leq x \leq 1$.

1. Find the cdf F .

$$F(x) = \int_0^x 3y^2 dy = y^3 \Big|_0^x = x^3 \quad x \in [0, 1].$$

2. Find F^{-1}

$$u = F(x) = x^3 \Rightarrow u^{1/3} = x = F^{-1}(u) \quad 0 \leq u \leq 1$$

↑ range of
 $F(x)$.

3. # write code for inverse transform example

$f_X(x) = 3x^2$, $0 \leq x \leq 1$

① Write a function for F^{-1}

② Sample 1000 values from $\text{Unif}(0, 1)$

③ evaluate $x_i = F^{-1}(u_i)$ for $i=1, \dots, 1000$.

1.2 Discrete RVs

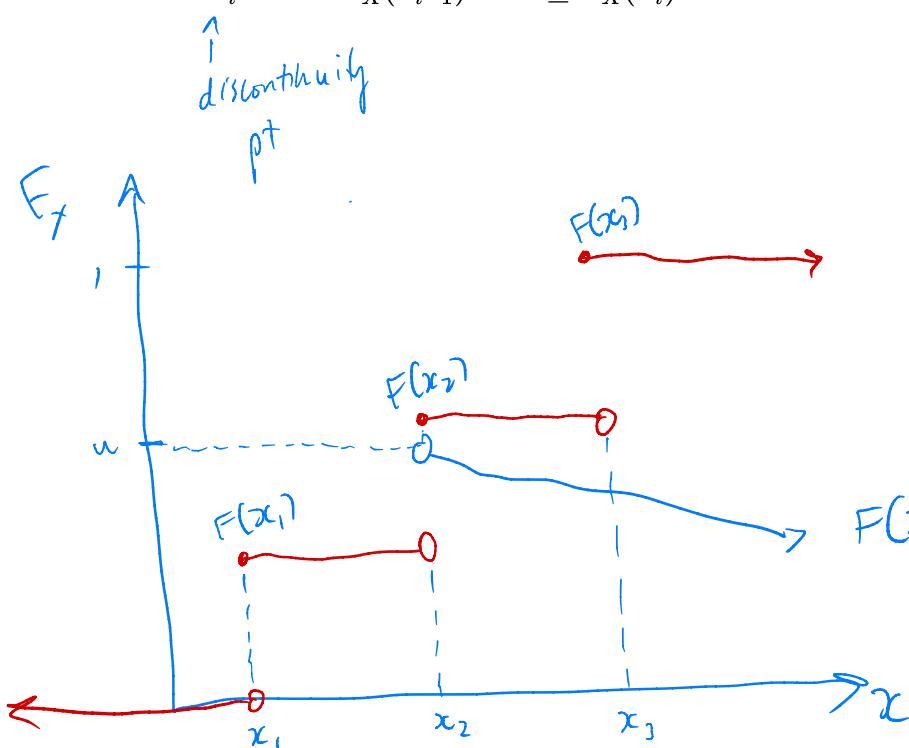
If X is a discrete random variable and $\dots < x_{i-1} < x_i < \dots$ are the points of discontinuity of $F_X(x)$, then the inverse transform is $F_X^{-1}(u) = x_i$ where $F_X(x_{i-1}) < u \leq F_X(x_i)$. This leads to the following algorithm:

1. Generate a r.v. U from $\text{Unif}(0, 1)$.

2. Select x_i where $F_X(x_{i-1}) < U \leq F_X(x_i)$.

} repeat many times.

① If $u = 0.5$ for example,



$F(x_1) < u \leq F(x_2)$.

\Rightarrow ② select x_2 .

Example 1.2 Generate 1000 samples from the following discrete distribution.

```
x <- 1:3
p <- c(0.1, 0.2, 0.7)
```

x	1.0	2.0	3.0
f	0.1	0.2	0.7

```
# write code to sample from discrete dsn
n <- 1000
```

There is a simpler way using `sample()` function in R.

* Remember to allow replacement and specify the probability vector if using `sample()`

↓ Something we can do if we can't find F^{-1} in closed form

2 Acceptance-Reject Method

The goal is to generate realizations from a target density, f .

Most cdfs cannot be inverted in closed form.

\Rightarrow we can't use inverse transform method.

The Acceptance-Reject (or "Accept-Reject") samples from a distribution that is *similar* to f and then adjusts by only accepting a certain proportion of those samples.

↑
target

\Rightarrow we reject the rest.

The method is outlined below:

①

Let g denote another density from which we **know how to sample** and we can **easily calculate** $g(x)$.

②

Let $e(\cdot)$ denote an envelope, having the property $e(x) = cg(x) \geq f(x)$ for all

$x \in \mathcal{X} = \{x : f(x) > 0\}$ for a given constant $c \geq 1$.

The Accept-Reject method then follows by sampling $Y \sim g$ and $U \sim \text{Unif}(0, 1)$.

If $U < f(Y)/e(Y)$, accept Y . Set $X = Y$ and consider X to be an element of the target random sample.

Note: $1/c$ is the expected proportion of candidates that are accepted.

We can use this to evaluate the efficiency of the algorithm.

2.1 Algorithm

1. Find a suitable density g and envelope e . ↗ finding constant $c \geq 1$.

2. Sample $Y \sim g$.

3. Sample $U \sim \text{Unif}(0, 1)$.

4. If $U < f(Y)/e(Y)$, accept Y .

5. Repeat from Step 2 until you have generated your desired sample size.

what might be hard/slow?

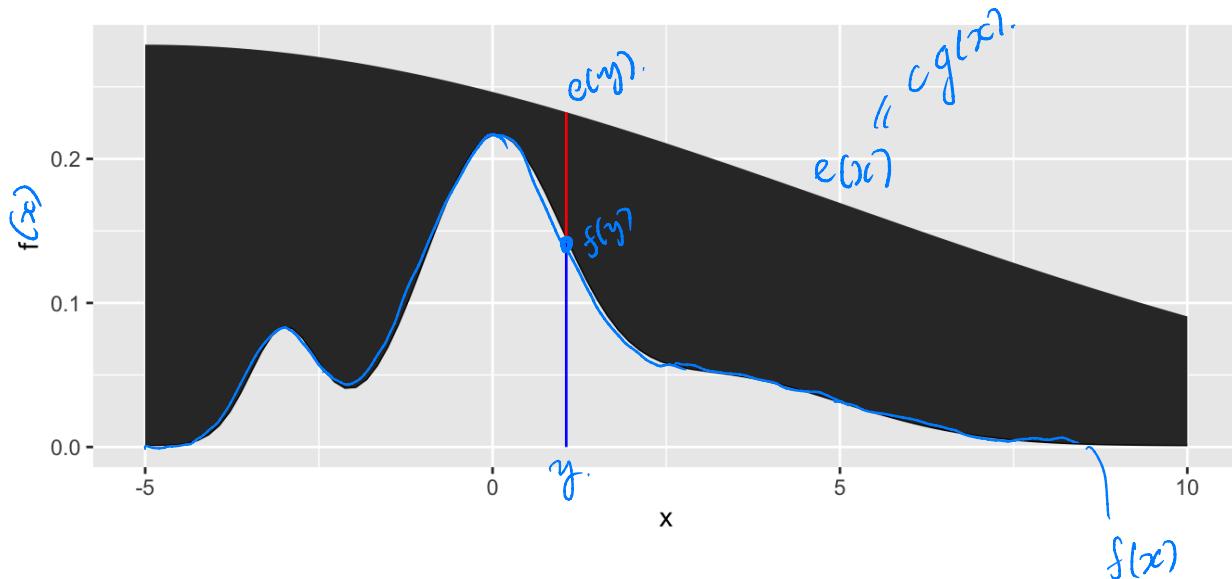
slow: we may throw away a lot of draws - depending on efficiency.

hard: finding $e(\cdot)$.

Requirement: The support of g MUST include the support of f . *

(BAD) Example: If $f \equiv N(0, 2)$ and $g \equiv \text{Unif}(-10, 10)$.

This would NOT be appropriate because the support of f $X_f = (-\infty, \infty)$
 $X_g = [-10, 10]$.



2.2 Envelopes

Good envelopes have the following properties:

- ① Envelope exceeds target everywhere \leftarrow Support of g must include the support of f .
- ② Easy to sample from g and easy to evaluate.
- ③ Generate few rejected draws (save time).

A simple approach to finding the envelope: *In some cases*

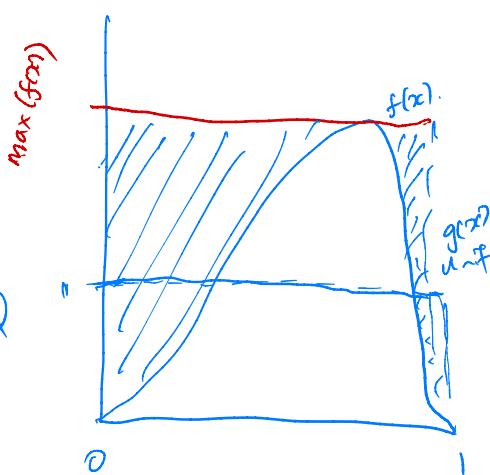
Say the support of $X_f = [0, 1]$.
Let $g(x) = \text{Unif}(0, 1) = \begin{cases} 1 & \text{if } x \in [0, 1] \\ 0 & \text{o.w.} \end{cases}$

Find $\max(f(x))$ and let $c = \max(f(x))$.

This is ONLY USEFUL if support is $[0, 1]$.

Plotting is your friend here!

This may not be efficient. If you know more about the shape, use it!



Example 2.1 We want to generate a random variable with pdf $f(x) = 60x^3(1-x)^2$, $0 \leq x \leq 1$. This is a Beta(4, 3) distribution.

Can we invert $F(x)$ analytically?

NO,

If not, find the maximum of $f(x)$.

$$\begin{aligned} f'(x) &= 60 \cdot 3x^2(1-x)^2 + 60x^3 \cdot 2(1-x) \cdot -1 \\ &= 60x^2(1-x)(3(1-x)-2x) \\ &= 60x^2(1-x)(3-5x) = 0. \end{aligned}$$

could be
 $x=0, x=1$ $x = \frac{3}{5}$

$$f(0) = f(1) = 0.$$

$$c = f\left(\frac{3}{5}\right) = 2.0736$$

pdf function, could use dbeta() instead
 $f \leftarrow \text{function}(x) \{$ in base R.
 $\quad 60 * x^3 * (1-x)^2$
 $\}$

plot pdf
 $x \leftarrow \text{seq}(0, 1, \text{length.out} = 100)$

$\text{ggplot}() +$

$\text{geom_line}(\text{aes}(x, f(x)))$

draw ...

evaluate target pdf at x values.

↑ evaluate target pdf at x values.

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

target pdf

make x values to evaluate f at.

```

envelope <- function(x) {
  ## create the envelope function
}

# Accept reject algorithm
n <- 1000 # number of samples wanted
accepted <- 0 # number of accepted samples
samples <- rep(NA, n) # store the samples here empty vector of length n.

while(accepted < n) {
  # sample y from g <- unif(0,1).
  y <- runif(0,1)
  # sample u from uniform(0,1)
  u <- runif(1)

  calculating the proportion.
  if(u < f(y)/envelope(y)) {
    # accept
    accepted <- accepted + 1
    samples[accepted] <- y
  }
}

loop accepted so my loop ends eventually.

```

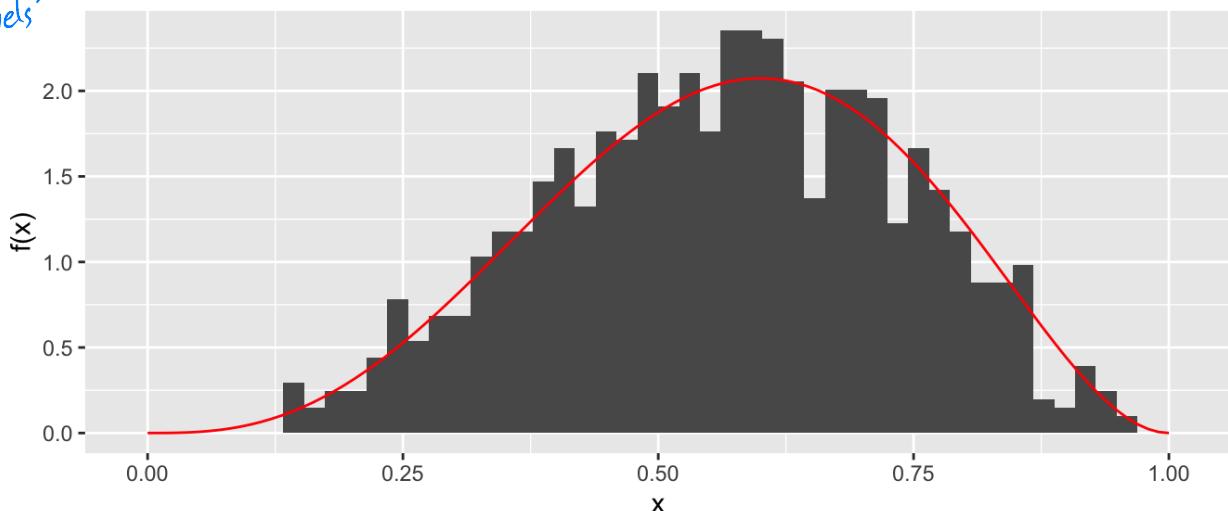
plot histogram of samples w/ theoretical pdf on top.

```

ggplot() +
  geom_histogram(aes(sample, y = ..density..), bins = 50, ) +
  geom_line(aes(x, f(x)), colour = "red") +
  xlab("x") + ylab("f(x)")

```

samples obtained from accept-reject. necessary so that histogram is on same scale as the density instead of raw counts.



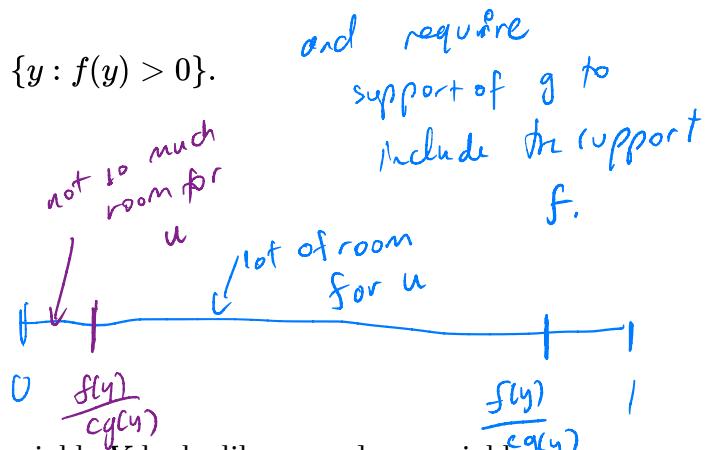
2.3 Why does this work?

Recall that we require

$$\frac{e(y)}{cg(y)} \geq f(y) \quad \forall y \in \{y : f(y) > 0\}.$$

Thus,

$$0 \leq \frac{f(y)}{cg(y)} \leq 1$$



The larger the ratio $\frac{f(y)}{cg(y)}$, the more the random variable Y looks like a random variable distributed with pdf f and the more likely Y is to be accepted.

2.4 Additional Resources

See p.g. 69-70 of Rizzo for a proof of the validity of the method.

\uparrow
 Come read in OHT or in library on reserve.

3 Transformation Methods

We have already used one transformation method – **Inverse transform method** – but there are many other transformations we can apply to random variables.

1. If $Z \sim N(0, 1)$, then $V = Z^2 \sim \chi^2_1$
2. If $U \sim \chi^2_m$ and $V \sim \chi^2_n$ are independent, then $F = \frac{U/m}{V/n} \sim F_{m,n}$
3. If $Z \sim N(0, 1)$ and $V \sim \chi^2_n$ are independent, then $T = \frac{Z}{\sqrt{V/n}} \sim t_n$
4. If $U \sim \text{Gamma}(r, \lambda)$ and $V \sim \text{Gamma}(s, \lambda)$ are independent, then $X = \frac{U}{U+V} \sim \text{Beta}(r, s)$

$$X \rightarrow g(X).$$

Definition 3.1 A *transformation* is any function of one or more random variables.

Sometimes we want to transform random variables if observed data don't fit a model that might otherwise be appropriate. Sometimes we want to perform inference about a new statistic.

* *iid. ↗ identically distributed, i.e. "random sample."*

Example 3.1 If $X_1, \dots, X_n \stackrel{iid}{\sim} \text{Bernoulli}(p)$. What is the distribution of $\sum_{i=1}^n X_i$?

Can derive $\sum X_i \sim \text{Binomial}(n, p)$.

Example 3.2 If $X \sim N(0, 1)$, what is the distribution of $X + 5$?

Can derive $X+5 \sim N(5, 1)$.

Example 3.3 For X_1, \dots, X_n iid random variables, what is the distribution of the median of X_1, \dots, X_n ? What is the distribution of the order statistics? $X_{[i]}$?

This is complex...

There are many approaches to deriving the pdf of a transformed variable.

– change of variable

If g monotone, then for cts X

and $Y = g(X)$,

$$f_Y(y) = \begin{cases} f_X(g^{-1}(y)) \left| \frac{d}{dy} g^{-1}(y) \right|, & y \in Y \\ 0 & \text{o.w.} \end{cases}$$

– moment generating function

$$M_X(t) = E(e^{tX})$$

– Convolution Theorem

$$Z = X+Y$$

10

etc.

3.1 Algorithm

But the theory isn't always available. What can we do?

Statistical
V 11

Use computational methods
to simulate from the transformed
distribution.

3.1 Algorithm

Let X_1, \dots, X_p be a set of independent random variables with pdfs f_{X_1}, \dots, f_{X_p} , respectively, and let $g(X_1, \dots, X_p)$ be some transformation we are interested in simulating from.

1. Simulate $X_1 \sim f_{X_1}, \dots, X_p \sim f_{X_p}$ either be straight forward (named)
could use inverse method, accept-reject.
2. Compute $G = g(X_1, \dots, X_p)$. This is one draw from $g(X_1, \dots, X_p)$.
3. Repeat Steps 1-2 many times to simulate from the target distribution.

Example 3.4 It is possible to show for $X_1, \dots, X_p \stackrel{iid}{\sim} N(0, 1)$, $Z = \sum_{i=1}^p X_i^2 \sim \chi_p^2$. Imagine that we cannot use the `rchisq` function. How would you simulate Z ?

1. Sample p draws from the $N(0, 1)$.
2. square all X 's, sum them up. $\sum X_i^2$
3. Repeat 1-2.

`library(tidyverse)`

```
# function for squared r.v.s
squares <- function(x) x^2
```

```
sample_z <- function(n, p) {
  # store the samples
  samples <- data.frame(matrix(rnorm(n*p), nrow = n))
```

```
samples %>%
  mutate_all("squares") %>% # square the rvs
  rowSums() # sum over rows
```

} ↑ this will add up the p squared X 's, n times.

```
# get samples
n <- 1000 # number of samples
```

```
# apply our function over different degrees of freedom
samples <- data.frame(chisq_2 = sample_z(n, 2),
                      chisq_5 = sample_z(n, 5),
                      chisq_10 = sample_z(n, 10),
```

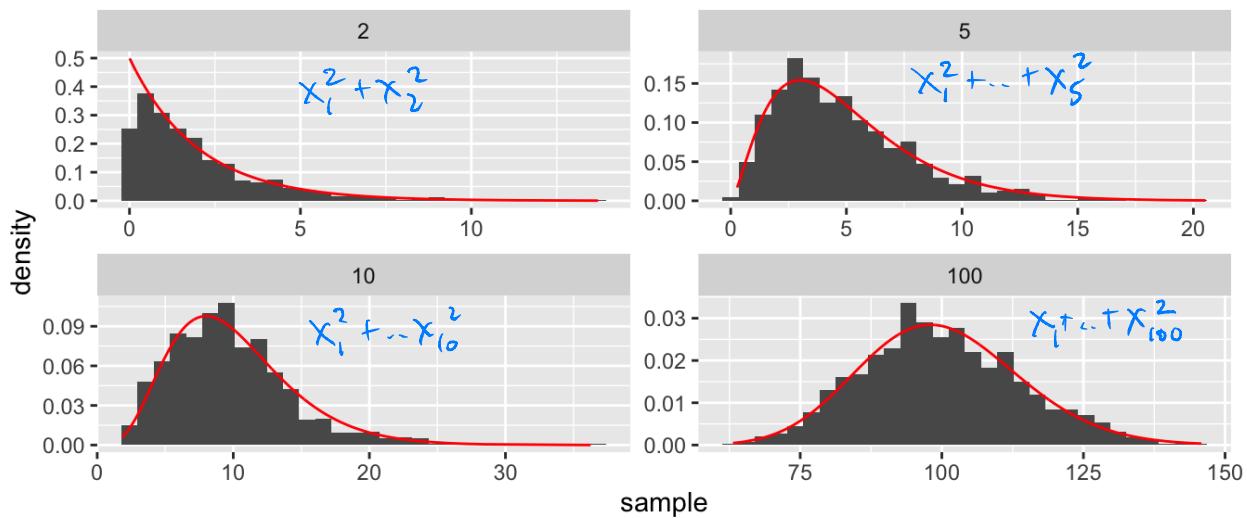
```

chisq_100 = sample_z(n, 100)

# plot results
samples %>%
  gather(distribution, sample, everything()) %>% # make easier to
  plot w/ facets
  separate(distribution, into = c("dsn_name", "df")) %>% # get the df
  mutate(df = as.numeric(df)) %>% # make numeric
  mutate(pdf = dchisq(sample, df)) %>% # add density function values
  ggplot() + # plot
  geom_histogram(aes(sample, y = ..density..)) + # samples
  geom_line(aes(sample, pdf), colour = "red") + # true pdf in red,
  facet_wrap(~df, scales = "free")

```

different scales for each df scales = "free-y" scales = "free-x"



4 Mixture Distributions

A special transformation.

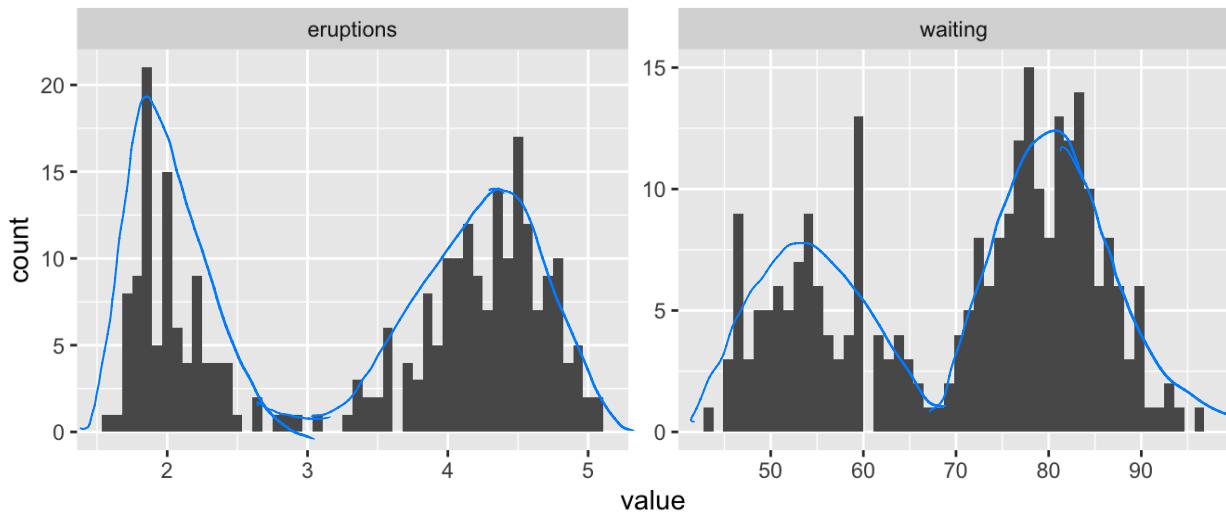
The `faithful` dataset in R contains data on eruptions of Old Faithful (Geyser in Yellowstone National Park).

```
head(faithful)
##   eruptions waiting
## 1      3.600     79
## 2      1.800     54
## 3      3.333     74
## 4      2.283     62
## 5      4.533     85
## 6      2.883     55
```

length of eruptions
waiting time until next eruption

```
faithful %>%
  gather(variable, value) %>%
  ggplot() +
  geom_histogram(aes(value), bins = 50) +
  facet_wrap(~variable, scales = "free")
```

bimodal.



What is the shape of these distributions?

Bimodal
i.e. two modes.

Definition 4.1 A random variable Y is a discrete mixture if the distribution of Y is a weighted sum $F_Y(y) = \sum \theta_i F_{X_i}(y)$ for some sequence of random variables X_1, X_2, \dots and $\theta_i > 0$ such that $\sum \theta_i = 1$.

weights ↗
For 2 r.v.s,

↓
| summing
countable #
of distributions.

$$f(x) = \theta f_{X_1}(x) + (1-\theta) f_{X_2}(x).$$

↖ ↘
two different
distributions.

How do we simulate from this distribution?

There are two sources of variability.

$$Y \sim \text{Bernoulli}(\theta). \text{ Then } \text{if } \begin{cases} y=1 & \text{with probability } \theta. \\ y=0 & \text{with probability } 1-\theta. \end{cases} \quad \begin{array}{ll} X \sim f_{X_1}, & \\ X \sim f_{X_2}. & \end{array}$$

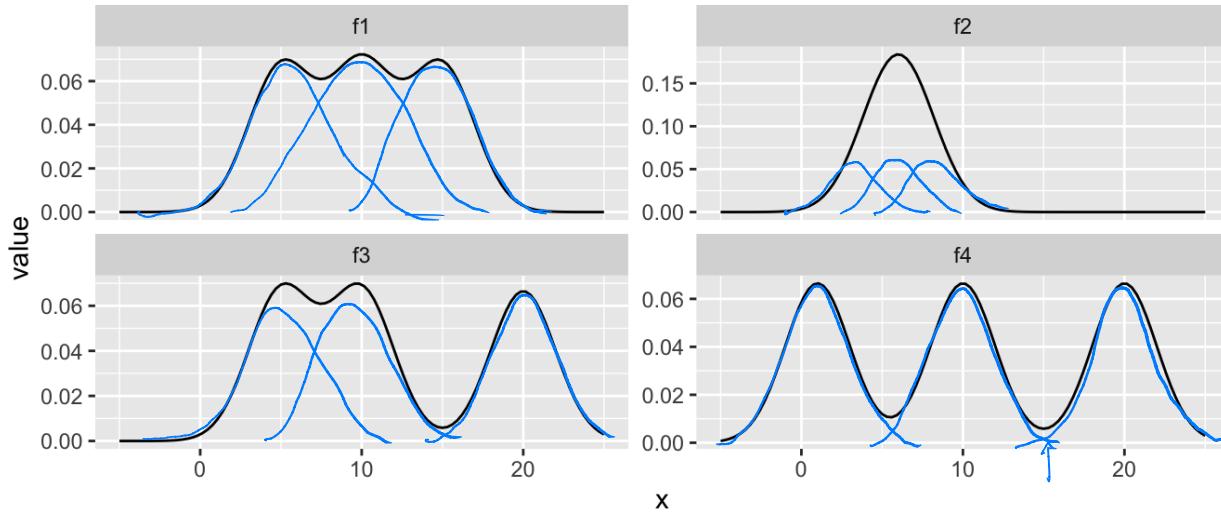
Example 4.1

```

x <- seq(-5, 25, length.out = 100)
// vector of means
mixture <- function(x, means, sd) {
  # x is the vector of points to evaluate the function at
  # means is a vector, sd is a single number
  f <- rep(0, length(x)) ← storage container to store pdf values.
  for(mean in means) {
    f <- f + dnorm(x, mean, sd)/length(means) # why do I divide?
  }
  f
}

# look at mixtures of N(mu, 4) for different values of mu need  $\sum \theta_i = 1$ .
data.frame(x,
  f1 = mixture(x, c(5, 10, 15), 2),
  f2 = mixture(x, c(5, 6, 7), 2),
  f3 = mixture(x, c(5, 10, 20), 2),
  f4 = mixture(x, c(1, 10, 20), 2)) %>%
gather(mixture, value, -x) %>%
ggplot() +
  geom_line(aes(x, value)) +
  facet_wrap(~mixture, scales = "free_y")
  
```

*I am equally weightily each dsn.
[We don't have to equally weight, we just*



4.1 Mixtures vs. Sums

Note that mixture distributions are *not* the same as the distribution of a sum of r.v.s.

Mixtures are weighted sums of distributions.

NOT distributions of weighted sums!

Example 4.2 Let $X_1 \sim N(0, 1)$ and $X_2 \sim N(4, 1)$, independent.

$$S = \frac{1}{2}(X_1 + X_2)$$

also called a convolution.

$$\begin{aligned} E(S) &= E\left(\frac{1}{2}(X_1 + X_2)\right) \\ &= \frac{1}{2}(E(X_1) + E(X_2)) = \frac{1}{2}(0+4) = 2. \end{aligned}$$

$$\text{Var}(S) = \text{Var}\left(\frac{1}{2}(X_1 + X_2)\right) \stackrel{\text{indep}}{=} \frac{1}{4}(\text{Var}X_1 + \text{Var}X_2) = \frac{1}{4}(1+1) = \frac{1}{2}$$

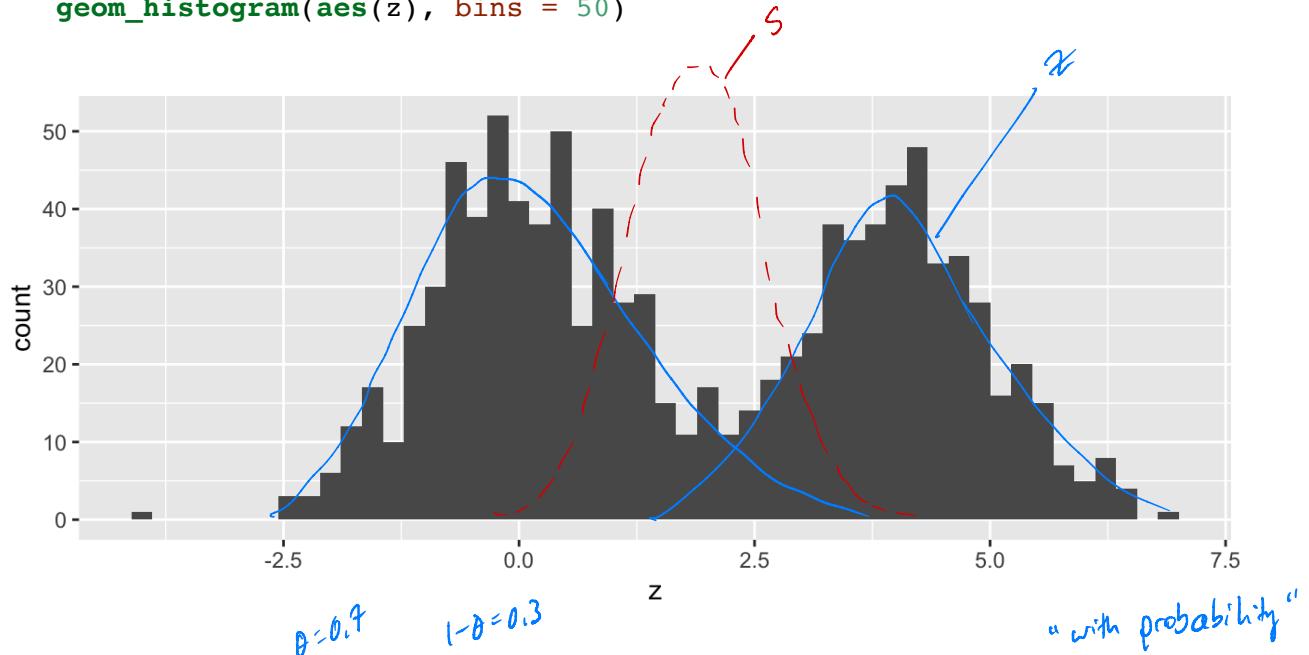
Can show in fact that $S = \frac{1}{2}(X_1 + X_2) \sim N(2, \frac{1}{2})$

Z such that $f_Z(z) = 0.5f_{X_1}(z) + 0.5f_{X_2}(z)$.

$n \leftarrow 1000$ θ " θ
 $u \leftarrow \text{rbinom}(n, 1, 0.5)$ ← choose which dsn we sample from w/ $\theta = 0.5$,

$z \leftarrow u * \text{rnorm}(n) + (1 - u) * \text{rnorm}(n, 4, 1)$
 $N(0, 1)$ $N(u, 1)$

`ggplot() +
 geom_histogram(aes(z), bins = 50)`



What about $f_Z(z) = 0.7f_{X_1}(z) + 0.3f_{X_2}(z)$?

change $u \leftarrow \text{rbinom}(n, 1, 0.7)$ to choose f_{X_1} w.p. 0.7.

4.2 Models for Count Data (refresher)

Recall that the Poisson(λ) distribution is useful for modeling count data.

$$f(x) = \frac{\lambda^x \exp\{-\lambda\}}{x!}, \quad x = 0, 1, 2, \dots$$

Where X = number of events occurring in a fixed period of time or space.

When the mean λ is low, then the data consists of mostly low values (i.e. 0, 1, 2, etc.) and less frequently higher values.

As the mean count increases, the skewness goes away and the distribution becomes approximately normal.

With the Poisson distribution,

$$E[X] = \text{Var } X = \lambda.$$

Example 4.3

- # of meows in a 2 minute cat video on youtube.
- # of baskets made in a minute.
- # of cars that drive by during class.

restricts the shape of the dsn!

High λ

Example 4.4 The Colorado division of Parks and Wildlife has hired you to analyze their data on the number of fish caught in Horsetooth reservoir by visitors. Each visitor was asked - How long did you stay? - How many fish did you catch? - Other questions: How many people in your group, were children in your group, etc.

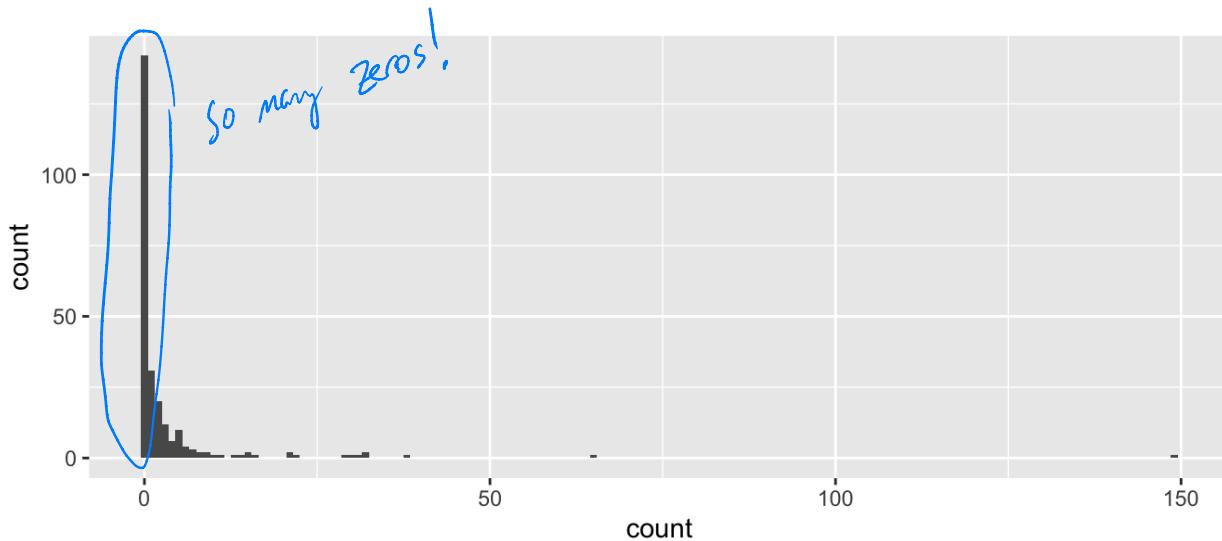
Some visitors do not fish, but there is not data on if a visitor fished or not. Some visitors who did fish did not catch any fish.

Note, this is modified from <https://stats.idre.ucla.edu/r/dae/zip/>.

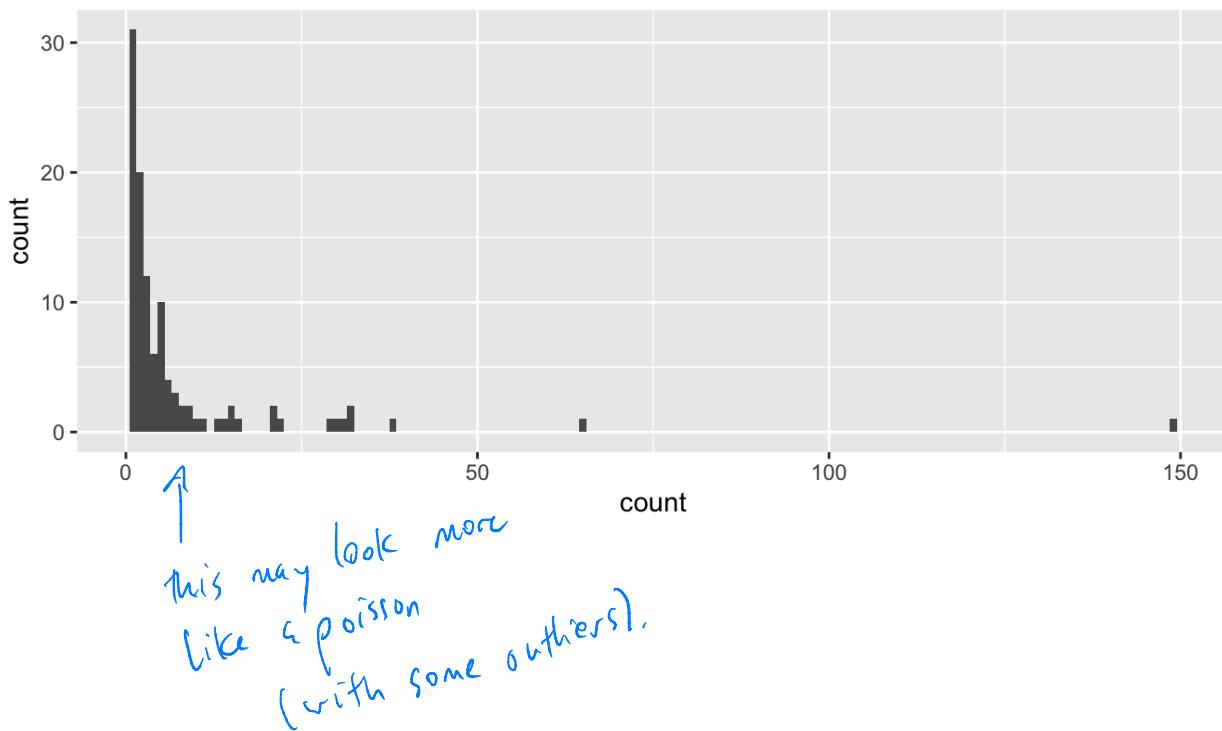
```
fish <- read_csv("https://stats.idre.ucla.edu/stat/data/fish.csv")
```

of fish caught by each party.
 # with zeroes

```
ggplot(fish) + geom_histogram(aes(count), binwidth = 1)
```



```
# without zeroes
fish %>%
  filter(count > 0) %>%
  ggplot() +
  geom_histogram(aes(count), binwidth = 1)
```



A zero-inflated model assumes that the zero observations have two different origins – structural and sampling zeroes. → a zero is possible and occurs by random chance.
 Example 4.5 ↗ a non-zero value is impossible.

A zero-inflated model is a **mixture model** because the distribution is a weighted average of the sampling model (i.e. Poisson) and a point-mass at 0.

For $Y \sim ZIP(\lambda)$,

$$Y \sim \begin{cases} 0 & \text{with probability } \pi \\ \text{Poisson}(\lambda) & \text{with probability } 1 - \pi \end{cases}$$

So that,

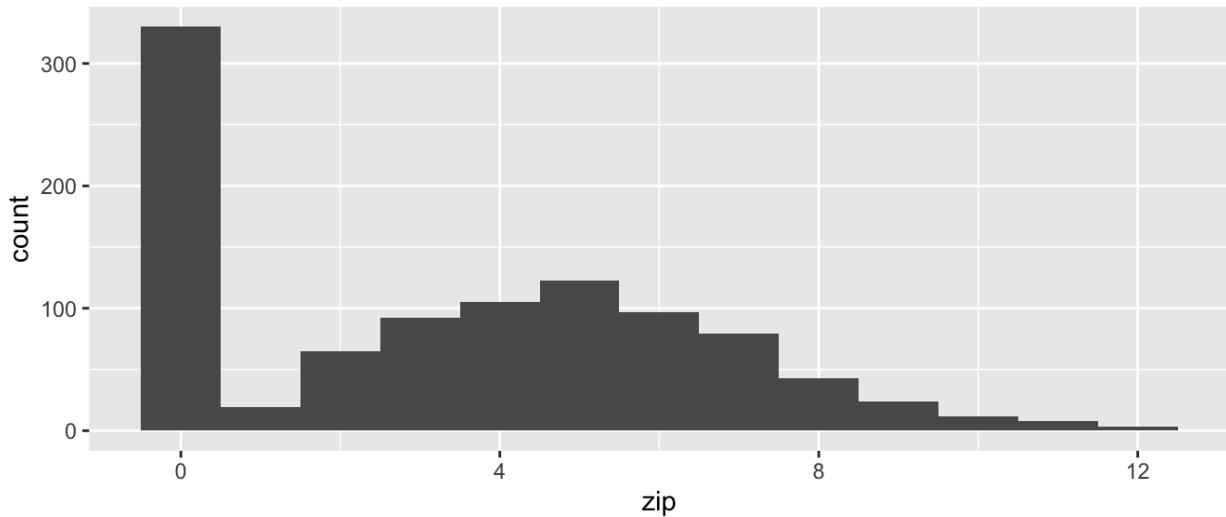
$$Y =$$

To simulate from this distribution,

```
n <- 1000
lambda <- 5
pi <- 0.3

u <- rbinom(n, 1, pi)
zip <- u*0 + (1-u)*rpois(n, lambda)
```

```
# zero inflated model  
ggplot() + geom_histogram(aes(zip), binwidth = 1)
```



```
# Poisson(5)  
ggplot() + geom_histogram(aes(rpois(n, lambda)), binwidth = 1)
```

