Data Transformation and Exploration

Lecture 3.4: Data Simulation

Lauren Sullivan

Module: Data Management, Visualization & Reproducibility

Data Simulation

Sometimes you need to simulate data to determine:

- if your models are working like you think they are
- to determine appropriate sample size (e.g. power analysis)
- to understand your data better

the 4 types of simulator functions in R

R has 4 built in functions for most distributions that you will find useful when simulating data.

- `r` - for generating random numbers from a distribution
- `d` - for density which evaluates a distribution at a certain point
- `p` - for evaulating the cumulative distribution function for a given distribution
- `q` - for the quantile function

rnorm()

```r
rnorm(10, mean = 1, sd = 0.5)
```

```
##  [1]  0.8861865  1.2444820  1.1387855 -0.1856769  0.7538
##  [7]  1.3850880  1.3434737  1.6532880  0.5750528
```

rnorm()

```r
rnorm(10, mean = 1, sd = 0.5)
```

```
##  [1] 1.6245152 1.5677174 1.4444282 1.4302982 0.1513304 1
##  [8] 1.1247265 1.6701207 0.1788031
```

```r
rnorm(10, mean = 1, sd = 0.5)
```

```
##  [1]  0.69248895  0.50261309  0.54147589  1.24738622  0.
##  [7]  1.06004819  0.27793104  0.49887083  1.08682311
```

set.seed()

```r
set.seed(10)
rnorm(10, mean = 1, sd = 0.5)
```

```
##  [1] 1.0093731 0.9078737 0.3143347 0.7004161 1.1472726 1
##  [8] 0.8181620 0.1866637 0.8717608
```

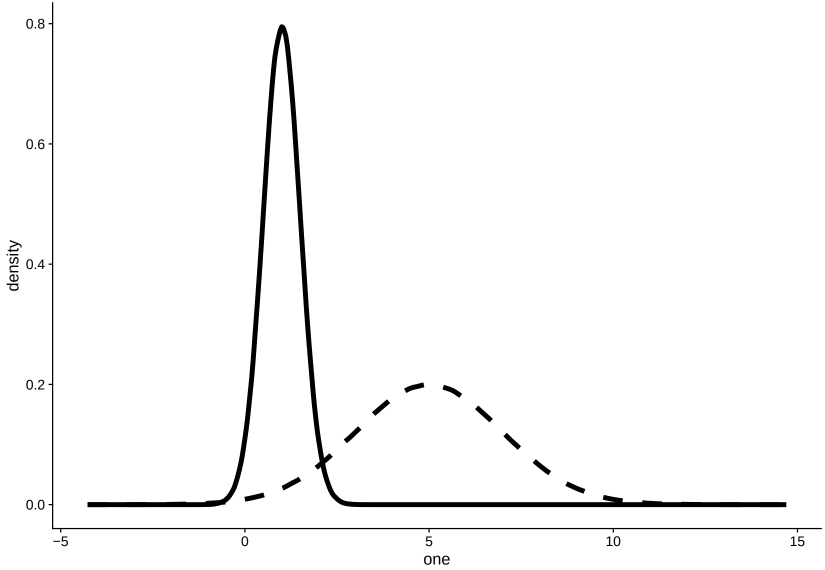
```r
set.seed(10)
rnorm(10, mean = 1, sd = 0.5)
```

```
##  [1] 1.0093731 0.9078737 0.3143347 0.7004161 1.1472726 1
##  [8] 0.8181620 0.1866637 0.8717608
```

Plot the distribution

```
one <- rnorm(1000000, mean = 1, sd = 0.5)
dat <- tibble(one)

ten <- rnorm(1000000, mean = 5, sd = 2)
dat2 <- tibble(ten)
```

Plot the distribution with `rnorm()`

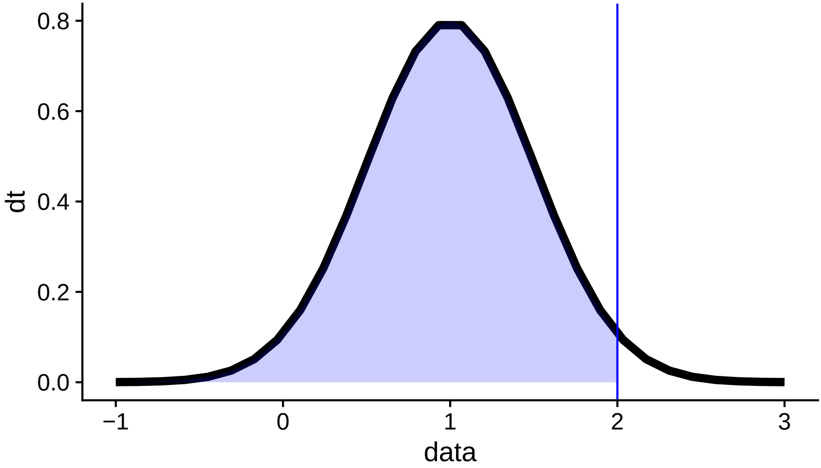


pnorm()

Calculate the probability of a certain point on a distribution

```r
pnorm(2, mean = 1, sd = 0.5)
```

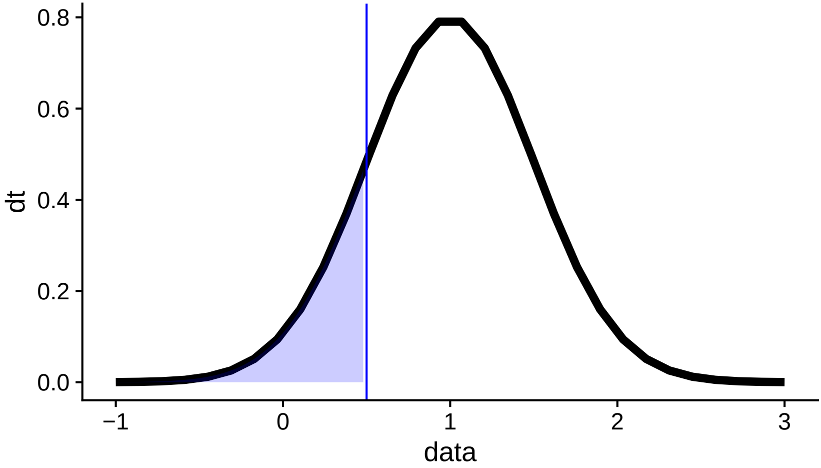
```
## [1] 0.9772499
```



pnorm()

```
pnorm(0.5, mean = 1, sd = 0.5)
```

```
## [1] 0.1586553
```

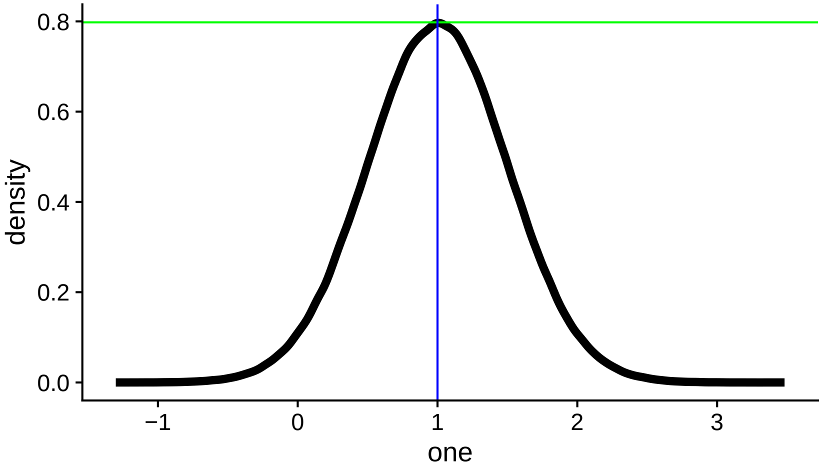


dnorm()

Find what the probability is at any point with the `dnorm()` function.

```
dnorm(1, mean = 1, sd = 0.5)
```

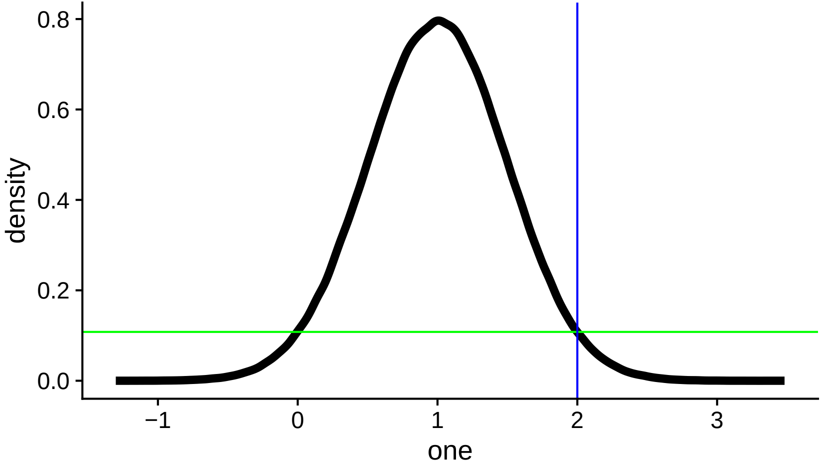
```
## [1] 0.7978846
```



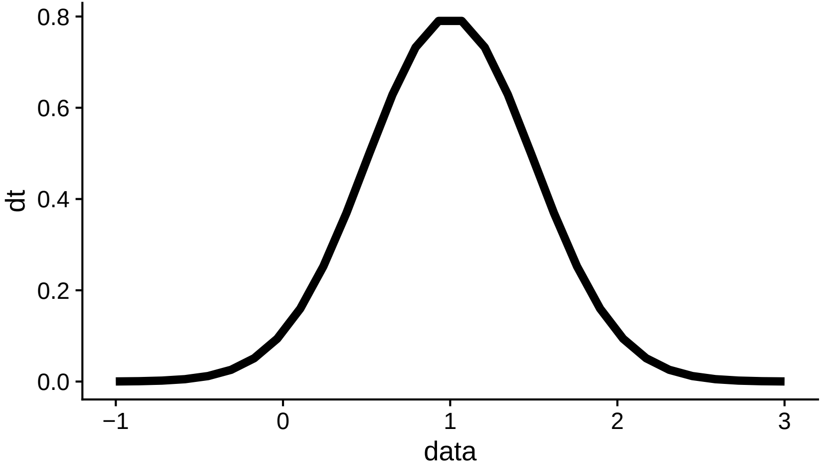
dnorm()

```
dnorm(2, mean = 1, sd = 0.5)
```

```
## [1] 0.1079819
```



Plotting the distribution with `dnorm()`

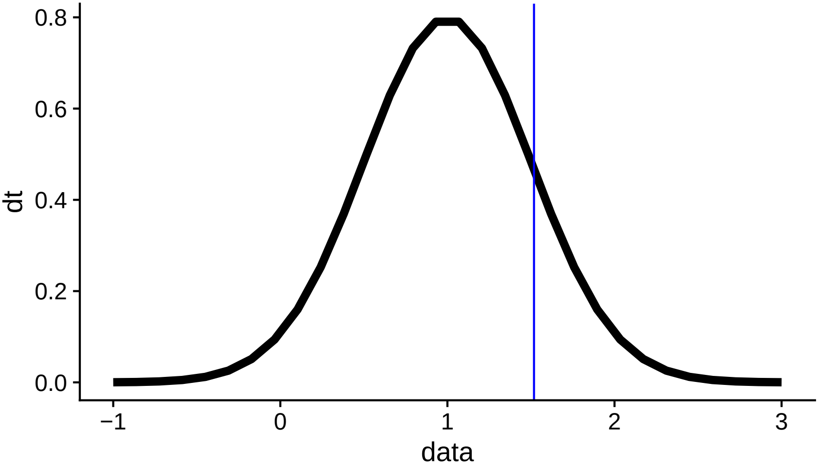


qnorm()

Used to find the probabilities of certain quantiles

```
qnorm(0.85, mean = 1, sd = 0.5)
```

```
## [1] 1.518217
```



Other distributions

```r
rpois(10, lambda = 1)
```

```
##  [1] 1 3 1 0 2 0 0 3 0 0
```

```r
rbinom(10, size=12, prob=0.2)
```

```
##  [1] 2 2 0 1 2 5 1 4 2 1
```

```r
rgamma(10, shape = 2, rate = 0.5)
```

```
##  [1]  1.5469101  8.8439521  0.8100006  2.8831475  1.2912295 10.8960108
##  [7]  5.9532477  4.5479624  1.5464896  3.0237713
```

Simulate Data Based on Small Samples

```
head(dat)
```

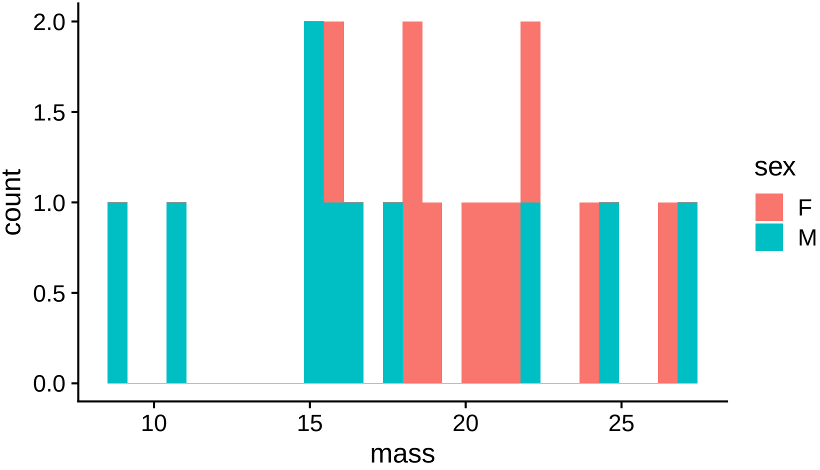
```
## # A tibble: 6 x 3
##      ID  mass sex
##   <dbl> <dbl> <chr>
## 1     1  16.0 F
## 2     2  21.4 F
## 3     3  19.1 F
## 4     4  20.5 F
## 5     5  18.1 F
## 6     6  23.8 F
```

```
dat %>%
  group_by(sex) %>%
  summarize(mean_mass = mean(mass),
            sd_mass = sd(mass))
```

```
## # A tibble: 2 x 3
##   sex   mean_mass sd_mass
##   <chr>     <dbl>   <dbl>
## 1 F          20.6    3.03
## 2 M          17.4    5.75
```

Simulate Data Based on Small Samples

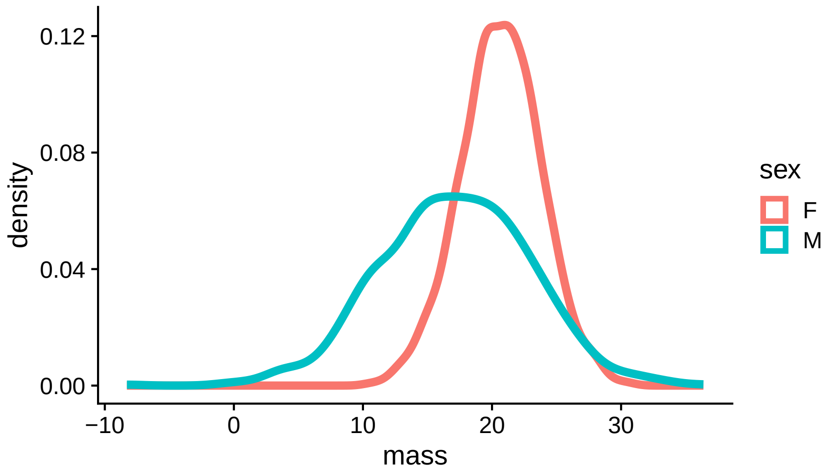
```
ggplot(dat)+
  geom_histogram(aes(x = mass, fill = sex))
```



Simulate Data Based on Small Samples

```
set.seed(100)
dat_females <- rnorm(1000, mean = 20.61769, sd = 3.028194)
set.seed(24)
dat_males <- rnorm(1000, mean = 17.40361, sd = 5.748625)

dat_all <- tibble(c(dat_females, dat_males))
dat_all$sex <- c(rep("F", 1000), rep("M", 1000))
names(dat_all)[1] <- 'mass'
```

Simulate Data Based on Small Samples



Simulate a Linear Model

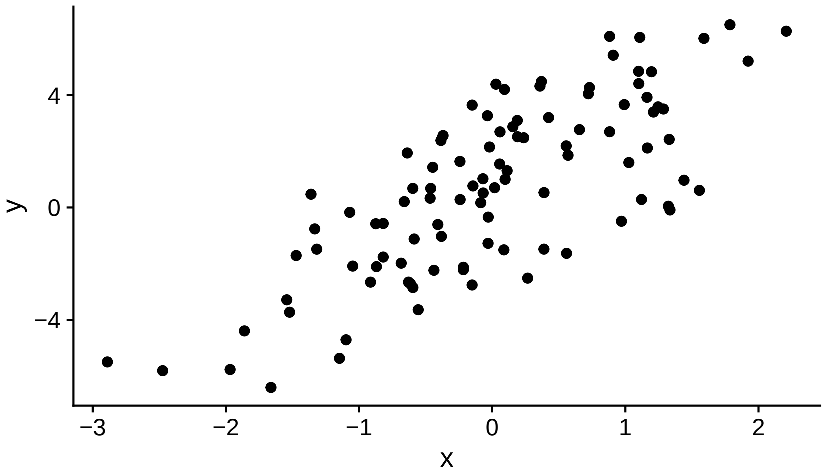
You may want to simulate from a specific model, instead of from a distribution of random numbers. So say you want to simulate from this equation:

$y = \beta_0 + \beta_1 x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 2), \beta_0 = 0.5$, and $\beta_1 = 2$

```
set.seed(20)

#simulate predictor (x) variables
x <- rnorm(100)

#simulate the error
e <- rnorm(100, 0, 2)

# calculate the response variables (y) from the model
y <- 0.5 + 2*x + e

summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -6.4084 -1.5402  0.6789  0.6893  2.9303  6.5052
```

Simulate a Linear Model



Simulate a Linear Model with a Binomial x

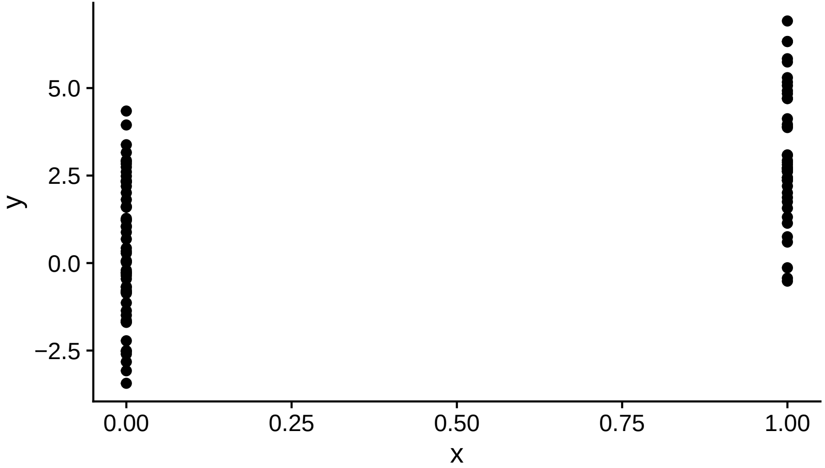
You could also use the same model, but instead have your x's distributed as a binomial variable (1's or 0's)

```
set.seed(20)

#simulate predictor (x) variables
x <- rbinom(100, 1, 0.5)
str(x)
```

```
##  int [1:100] 1 1 0 1 1 1 0 0 0 0 ...
```

```
#simulate the error
e <- rnorm(100, 0, 2)

# calculate the response variables (y) from the model
y <- 0.5 + 2*x + e
```

Simulate a Linear Model with a Binomial x



Simulate a Generalized Linear Model with Poisson

You may also want to simulate a model from something other than a normal distribution. For example, say you want to use a Poisson log-linear model (often good for count data):

$Y \sim Poisson(\mu)$

$log(\mu) = \beta_0 + \beta_1 x$, where $\beta_0 = 0.5$, and $\beta_1 = 0.2$

```r
set.seed(20)

#simulate the predictor variable as before
x <- rnorm(100)

#calculate the log mean of the model
log.mu <- 0.5 + 0.2*x

# calculate the response variables (y) from the possion distn.
y <- rpois(100, exp(log.mu))
```

Simulate a Generalized Linear Model with Poisson