# My Data Science Notes

*Michael Foley*

*2020-01-02*

# Contents

# Intro

These notes are pulled from various classes, tutorials, books, etc. and are intended for my own consumption. If you are finding this on the internet, I hope it is useful to you, but you should know that I am just a student and there's a good chance whatever you're reading here is mistaken. In fact, that should probably be your null hypothesis… or your prior. Whatever.

# Chapter 1

# Probability

## 1.1 Principles

Here are three rules that come up all the time.

- $Pr(A \cup B) = Pr(A) + Pr(B) - Pr(AB)$. This rule generalizes to $Pr(A \cup B \cup C) = Pr(A) + Pr(B) + Pr(C) - Pr(AB) - Pr(AC) - Pr(BC) + Pr(ABC)$.
- $Pr(A|B) = \frac{P(AB)}{P(B)}$
- If A and B are independent, $Pr(A \cap B) = Pr(A)Pr(B)$, and $Pr(A|B) = Pr(A)$.

Uniform distributions on finite sample spaces often reduce to counting the elements of $A$ and the sample space $S$, a process called combinatorics. Here are three important combinatorial rules.

**Multiplication Rule**. $|S| = |S_1||S_k|$.

*How many outcomes are possible from a sequence of 4 coin flips and 2 rolls of a die?* $|S| = |S_1| \cdot |S_2| \ldots |S_6| = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 6 \cdot 6 = 288$.

*How many subsets are possible from a set of n=10 elements?* In each subset, each element is either included or not, so there are $2^n = 1024$ subsets.

*How many subsets are possible from a set of n=10 elements taken k at a time with replacement?* Each experiment has $n$ possible outcomes and is repeated $k$ times, so there are $n^k$ subsets.

**Permutations**. The number of *ordered* arrangements (permutations) of a set of $|S| = n$ items taken $k$ at a time *without* replacement has $n(n-1)\ldots(n-k+1)$

subsets because each draw is one of k experiments with decreasing number of possible outcomes.

$$ {}_nP_k = \frac{n!}{(n-k)!} $$

Notice that if $k = 0$ then there is 1 permutation; if $k = 1$ then there are $n$ permutations; if $k = n$ then there are $n!$ permutations.

*How many ways can you distribute 4 jackets among 4 people?* ${}_nP_k = \frac{4!}{(4-4)!} = 4! = 24$

*How many ways can you distribute 4 jackets among 2 people?* ${}_nP_k = \frac{4!}{(4-2)!} = 12$

**Subsets**. The number of *unordered* arrangements (combinations) of a set of $|S| = n$ items taken $k$ at a time *without* replacement has

$$ {}_nC_k = \binom{n}{k} = \frac{n!}{k!(n-k)!} $$

combinations and is called the binomial coefficient. The binomial coefficient is the number of different subsets. Notice that if k=0 then there is 1 subset; if k=1 then there are n subsets; if k=n then there is 1 subset. The connection with the permutation rule is that there are $n!/(n-k)!$ permutations and each permutation has $k!$ permutations.

*How many subsets of 7 people can be taken from a set of 12 persons?* ${}_{12}C_7 = \binom{12}{7} = \frac{12!}{7!(12-7)!} = 792$

*If you are dealt five cards, what is the probability of getting a "full-house" hand containing three kings and two aces (KKKAA)?*

$$ P(F) = \frac{\binom{4}{3}\binom{4}{2}}{\binom{52}{5}} $$

**Distinguishable permutations**. The number of *unordered* arrangements (distinguishable permutations) of a set of $|S| = n$ items in which $n_1$ are of one type, $n_2$ are of another type, etc., is

$$ \binom{n}{n_1, n_2, \ldots, n_k} = \frac{n!}{n_1! n_2! \ldots n_k!} $$

*How many ordered arrangements are there of the letters in the word PHILIPPINES?* There are n=11 objects. $|P| = n_1 = 3$; $|H| = n_2 = 1$; $|I| = n_3 = 3$; $|L| = n_4 = 1$; $|N| = n_5 = 1$; $|E| = n_6 = 1$; $|S| = n_7 = 1$.

$$ \binom{n}{n_1, n_2, \ldots, n_k} = \frac{11!}{3!1!3!1!1!1!1!} = 1,108,800 $$

*How many ways can a research pool of 15 subjects be divided into three equally sized test groups?*

$$\binom{n}{n_1, n_2, \ldots, n_k} = \frac{15!}{5!5!5!} = 756,756$$

## 1.2 Discrete Distributions

### 1.2.1 Binomial

If $X$ is the count of successful events in $n$ identical and independent Bernoulli trials of success probability $p$, then $X$ is a random variable with a binomial distribution $X \sim b(n, p)$ with mean $\mu = np$ and variance $\sigma^2 = np(1-p)$. The probability of $X = x$ successes in $n$ trials is

$$P(X = x) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}.$$

*What is the probability 2 out of 10 coin flips are heads if the probability of heads is 0.3?*

Function `dbinom()` calculates the binomial probability.

```
dbinom(x = 2, size = 10, prob = 0.3)
```

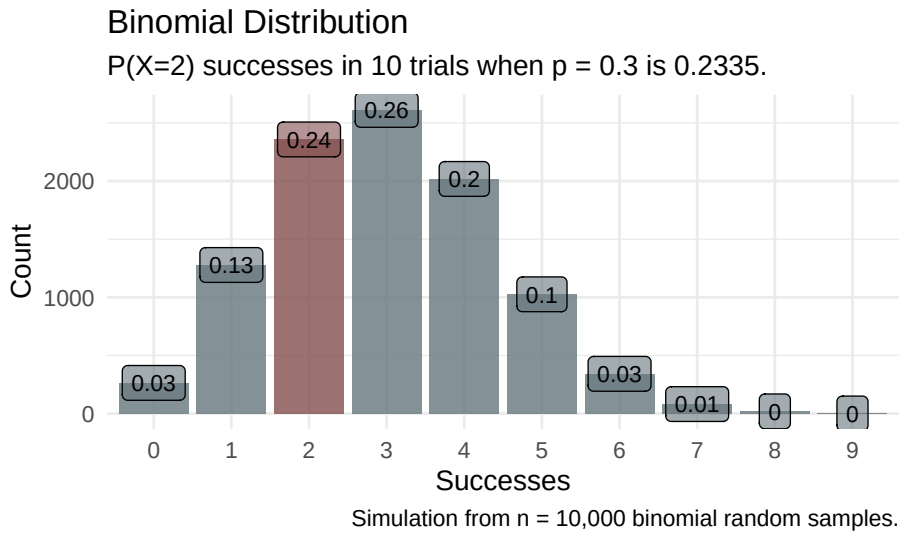
```
## [1] 0.2334744
```

A simulation of n = 10,000 random samples of size 10 gives a similar result. `rbinom()` generates a random sample of numbers from the binomial distribution.

```
library(tidyverse)

data.frame(cnt = rbinom(n = 10000, size = 10, prob = 0.3)) %>%
  count(cnt) %>%
  ungroup() %>%
  mutate(pct = n / sum(n),
         X_eq_x = cnt == 2) %>%
  ggplot(aes(x = as.factor(cnt), y = n, fill = X_eq_x, label = pct)) +
  geom_col(alpha = 0.8) +
  scale_fill_manual(values = c(my_colors$grey, my_colors$red)) +
  geom_label(aes(label = round(pct, 2)), size = 3, alpha = .6) +
  theme_minimal() +
  theme(legend.position = "none") +
```

```
  labs(title = "Binomial Distribution",
       subtitle = paste0("P(X=2) successes in 10 trials when p = 0.3 is ", round(dbinom
       x = "Successes",
       y = "Count",
       caption = "Simulation from n = 10,000 binomial random samples.")
```




*What is the probability of <=2 heads in 10 coin flips where probability of heads is 0.3?*

The cumulative probability is the sum of the first three bars in the simulation above. Function `pbinom()` calculates the *cumulative* binomial probability.

```
pbinom(q = 2, size = 10, prob = 0.3, lower.tail = TRUE)
```

```
## [1] 0.3827828
```

*What is the expected number of heads in 25 coin flips if the probability of heads is 0.3?*

The expected value, $\mu = np$, is 7.5. Here's an empirical test from 10,000 samples.

```
mean(rbinom(n = 10000, size = 25, prob = .3))
```

```
## [1] 7.5288
```

The variance, $\sigma^2 = np(1-p)$, is 5.25. Here's an empirical test.

```
var(rbinom(n = 10000, size = 25, prob = .3))
```

```
## [1] 5.199654
```

*Suppose $X$ and $Y$ are independen random variables distributed $X \sim b(10, .6)$ and $Y \sim b(10, .7)$. What is the probability that either variable is <=4?*

Let $P(A) = P(X <= 4)$ and $P(B) = P(Y <= 4)$. Then $P(A|B) = P(A) + P(B) - P(AB)$, and because the events are independent, $P(AB) = P(A)P(B)$.

```
p_a <- pbinom(q = 4, size = 10, prob = 0.6, lower.tail = TRUE)
p_b <- pbinom(q = 4, size = 10, prob = 0.7, lower.tail = TRUE)
p_a + p_b - (p_a * p_b)
```

```
## [1] 0.2057164
```

Here's an empirical test.

```
df <- data.frame(
  x = rbinom(10000, 10, 0.6),
  y = rbinom(10000, 10, 0.7)
  )
mean(if_else(df$x <= 4 | df$y <= 4, 1, 0))
```

```
## [1] 0.2045
```

### 1.2.2 Negative-Binomial

If $X$ is the count of trials required to reach a target number $r$ of successful events in identical and independent Bernoulli trials of success probability $p$, then $X$ is a random variable with a negative-binomial distribution $X \sim nb(r, p)$ with mean $\mu = r/p$ and variance $\sigma^2 = r(1-p)/p^2$. The probability of $X = x$ trials prior to $r$ successes is

$$P(X = x) = \binom{x-1}{r-1} p^r (1-p)^{x-r}.$$

*An oil company has a $p = 0.20$ chance of striking oil when drilling a well. What is the probability the company drills $x = 7$ wells to strike oil $r = 3$ times?*

$$P(X = 7) = \binom{7-1}{3-1} (0.2)^3 (1-0.2)^{(7-3)} = 0.049.$$

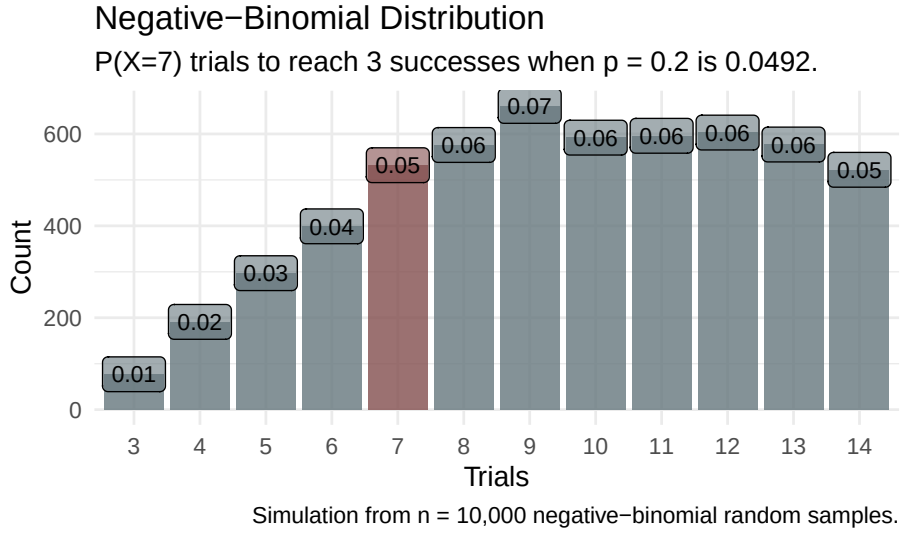
Function `dnbinom()` calculates the negative-binomial probability. Parameter `x` equals the number of failures, $x - r$.

```
dnbinom(x = 4, size = 3, prob = 0.2)
```

```
## [1] 0.049152
```

Here is a simulation of n = 10,000 random samples. `rnbinom()` generates a random sample of numbers from the negative-binomial distribution.

```
data.frame(cnt = rnbinom(n = 10000, size = 3, prob = 0.2)) %>%
  count(cnt) %>%
  ungroup() %>%
  mutate(pct = n / sum(n),
         X_eq_x = cnt == 7-3,
         cnt = cnt + 3) %>%
  filter(cnt < 15) %>%
  ggplot(aes(x = as.factor(cnt), y = n, fill = X_eq_x, label = pct)) +
  geom_col(alpha = 0.8) +
  scale_fill_manual(values = c(my_colors$grey, my_colors$red)) +
  geom_label(aes(label = round(pct, 2)), size = 3, alpha = .6, check_overlap = TRUE) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(title = "Negative-Binomial Distribution",
       subtitle = paste0("P(X=7) trials to reach 3 successes when p = 0.2 is ", round(
       x = "Trials",
       y = "Count",
       caption = "Simulation from n = 10,000 negative-binomial random samples.")
```



### 1.2.3 Geometric

If $X$ is the count of independent Bernoulli trials of success probability $p$ required to achieve the first successful trial, then $X$ is a random variable with a geometric distribution $X \sim G(p)$ with mean $\mu = \frac{n}{p}$ and variance $\sigma^2 = \frac{(1-p)}{p^2}$ . The probability of $X = n$ trials is

$$f(X = n) = p(1-p)^{n-1}.$$

The probability of $X <= n$ trials is

$$F(X = n) = 1 - (1-p)^n.$$

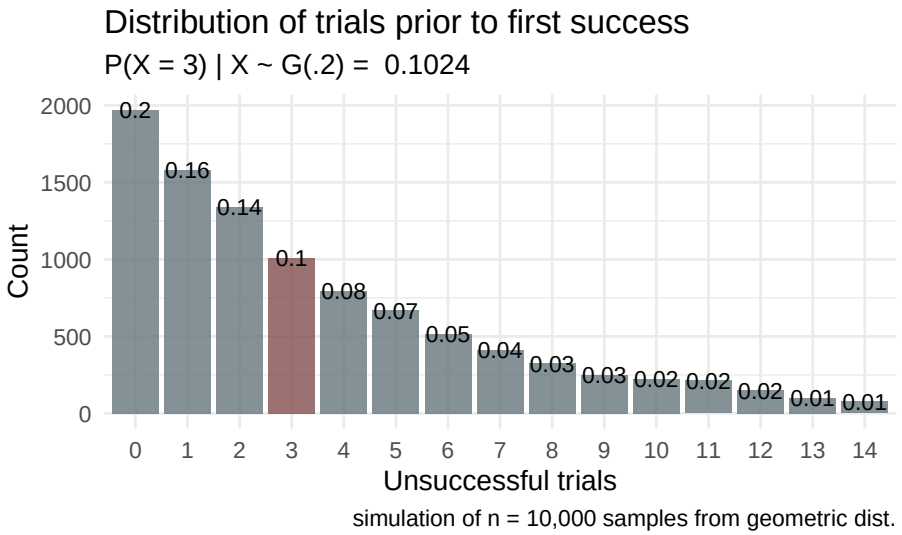
*Example. A sports marketer randomly selects persons on the street until he encounters someone who attended a game last season. What is the probability the marketer encounters $x = 3$ people who did not attend a game before the first success if $p = 0.20$ of the population attended a game?*

Function `pgeom()` calculates the geometric distribution probability.

```
dgeom(x = 3, prob = 0.20)
```

```
## [1] 0.1024
```

```
data.frame(cnt = rgeom(n = 10000, prob = 0.20)) %>%
  count(cnt) %>%
  top_n(n = 15, wt = n) %>%
  ungroup() %>%
  mutate(pct = round(n / sum(n), 2),
         X_eq_x = cnt == 3) %>%
  ggplot(aes(x = as.factor(cnt), y = n, fill = X_eq_x, label = pct)) +
  geom_col(alpha = 0.8) +
  scale_fill_manual(values = c(my_colors$grey, my_colors$red)) +
  geom_text(size = 3) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(title = "Distribution of trials prior to first success",
       subtitle = paste("P(X = 3) | X ~ G(.2) = ", round(dgeom(3, .2), 4)),
       x = "Unsuccessful trials",
       y = "Count",
       caption = "simulation of n = 10,000 samples from geometric dist.")
```




## 1.3 Continuous Distributions

### 1.3.1 Normal

Random variable $X$ is distributed $X \sim N(\mu, \sigma^2)$ if

$$f(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{-.5(\frac{x-\mu}{\sigma})^2}$$

.

#### Example

*IQ scores are distributed $X \sim N(100, 16^2)$. What is the probability a randomly selected person's IQ is <90?*

```
my_mean = 100
my_sd = 16
my_x = 90
# exact
pnorm(q = my_x, mean = my_mean, sd = my_sd, lower.tail = TRUE)
```

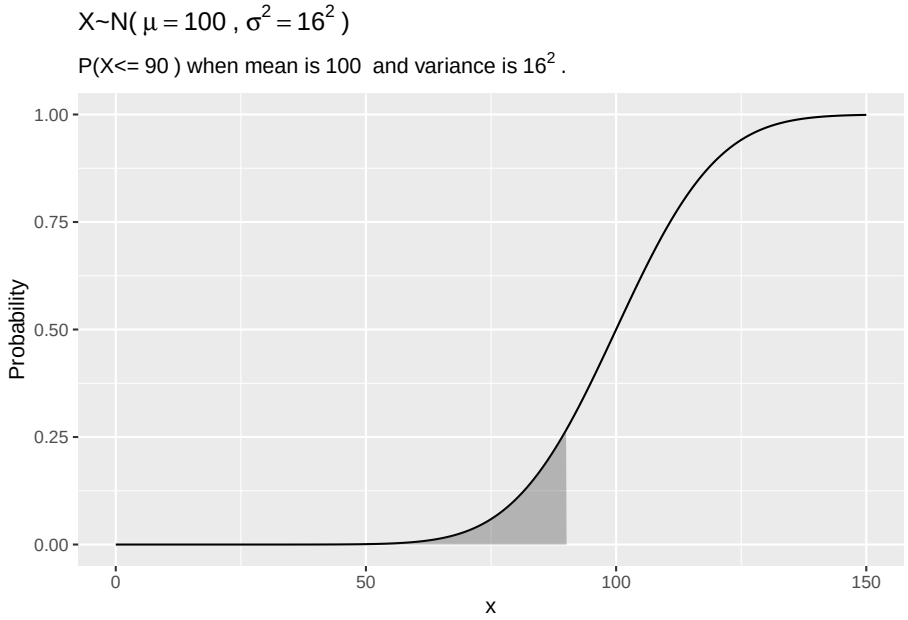
```
## [1] 0.2659855
```

```r
# simulated
mean(rnorm(n = 10000, mean = my_mean, sd = my_sd) <= my_x)
```

```
## [1] 0.264
```

```r
library(dplyr)
library(ggplot2)

data.frame(x = 0:1500 / 10,
           prob = pnorm(q = 0:1500 / 10,
                        mean = my_mean,
                        sd = my_sd,
                        lower.tail = TRUE)) %>%
  mutate(cdf = ifelse(x > 0 & x <= my_x, prob, 0)) %>%
ggplot() +
  geom_line(aes(x = x, y = prob)) +
  geom_area(aes(x = x, y = cdf), alpha = 0.3) +
  labs(title = bquote('X~N('~mu==.(my_mean)~','~sigma^{2}==.(my_sd)^{2}~')'),
       subtitle = bquote('P(X<='~.(my_x)~') when mean is'~.(my_mean)~' and variance is'~.(my_sd)^
       x = "x",
       y = "Probability")
```



### 1.3.2 Example

*IQ scores are distributed* $X \sim N(100, 16^2)$*. What is the probability a randomly selected person's IQ is >140?*

```
my_mean = 100
my_sd = 16
my_x = 140
# exact
pnorm(q = my_x, mean = my_mean, sd = my_sd, lower.tail = FALSE)
```

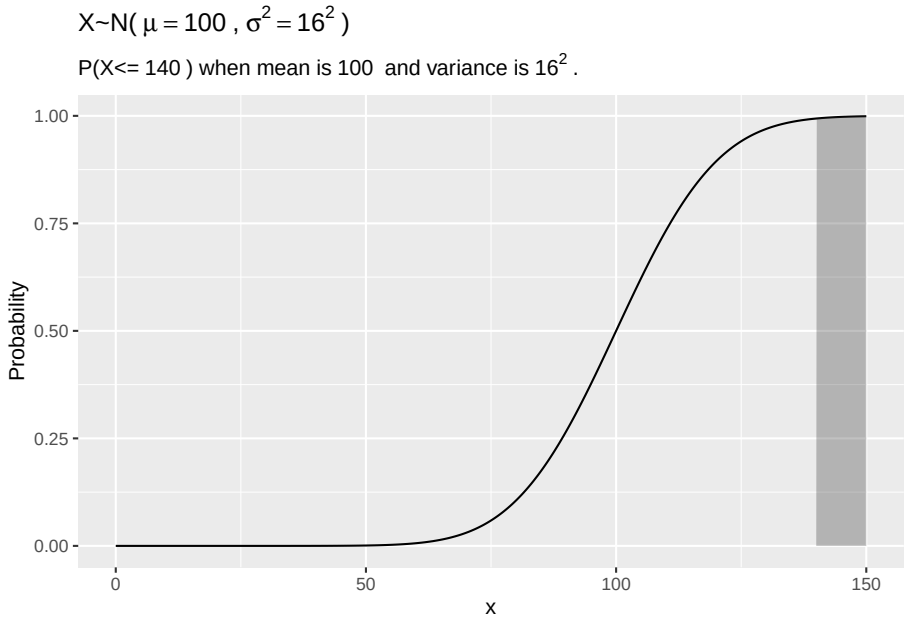
```
## [1] 0.006209665
```

```
# simulated
mean(rnorm(n = 10000, mean = my_mean, sd = my_sd) > my_x)
```

```
## [1] 0.008
```

```
library(dplyr)
library(ggplot2)

data.frame(x = 0:1500 / 10,
           prob = pnorm(q = 0:1500 / 10,
                        mean = my_mean,
                        sd = my_sd,
                        lower.tail = TRUE)) %>%
  mutate(cdf = ifelse(x > my_x & x < 1000, prob, 0)) %>%
ggplot() +
  geom_line(aes(x = x, y = prob)) +
  geom_area(aes(x = x, y = cdf), alpha = 0.3) +
  labs(title = bquote('X~N('~mu==.(my_mean)~','~sigma^{2}==.(my_sd)^{2}~')'),
       subtitle = bquote('P(X<='~.(my_x)~') when mean is'~.(my_mean)~' and variance is
       x = "x",
       y = "Probability")
```




### 1.3.3 Example

*IQ scores are distributed* $X \sim N(100, 16^2)$. *What is the probability a randomly selected person's IQ is between 92 and 114?*

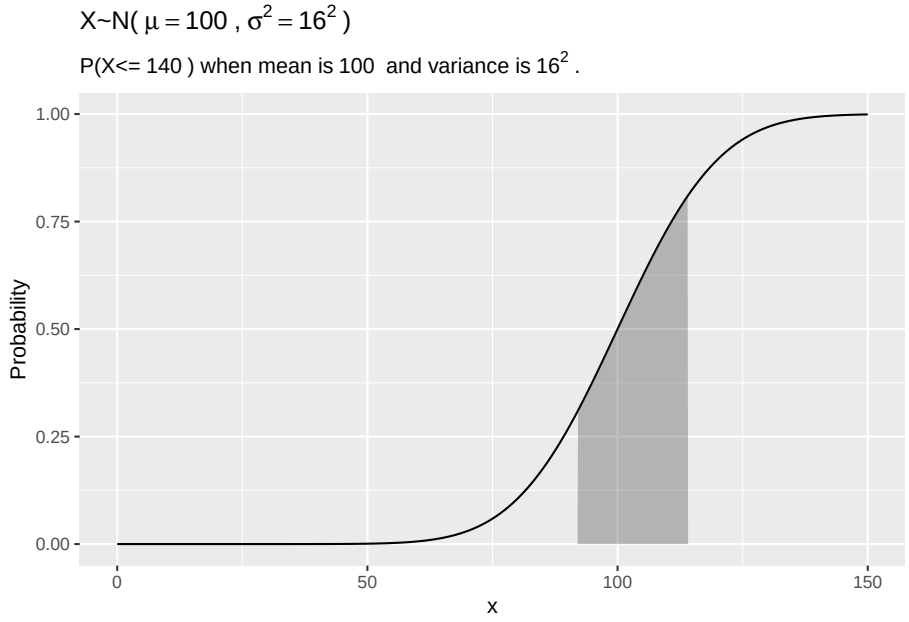
```
my_mean = 100
my_sd = 16
my_x_l = 92
my_x_h = 114
# exact
pnorm(q = my_x_h, mean = my_mean, sd = my_sd, lower.tail = TRUE) -
  pnorm(q = my_x_l, mean = my_mean, sd = my_sd, lower.tail = TRUE)
```

```
## [1] 0.5006755
```

```
library(dplyr)
library(ggplot2)

data.frame(x = 0:1500 / 10,
           prob = pnorm(q = 0:1500 / 10,
                        mean = my_mean,
                        sd = my_sd,
                        lower.tail = TRUE)) %>%
```

```
  mutate(cdf = ifelse(x > my_x_l & x <= my_x_h, prob, 0)) %>%
ggplot() +
  geom_line(aes(x = x, y = prob)) +
  geom_area(aes(x = x, y = cdf), alpha = 0.3) +
  labs(title = bquote('X~N('~mu==.(my_mean)~','~sigma^{2}==.(my_sd)^{2}~')'),
       subtitle = bquote('P(X<='~.(my_x)~') when mean is'~.(my_mean)~' and variance is
       x = "x",
       y = "Probability")
```




### 1.3.4 Example

*Class scores are distributed* $X \sim N(70, 10^2)$*. If the instructor wants to give A's to >=85th percentile and B's to 75th-85th percentile, what are the cutoffs?*

```
my_mean = 70
my_sd = 10
my_pct_l = .75
my_pct_h = .85

qnorm(p = my_pct_l, mean = my_mean, sd = my_sd, lower.tail = TRUE)
```

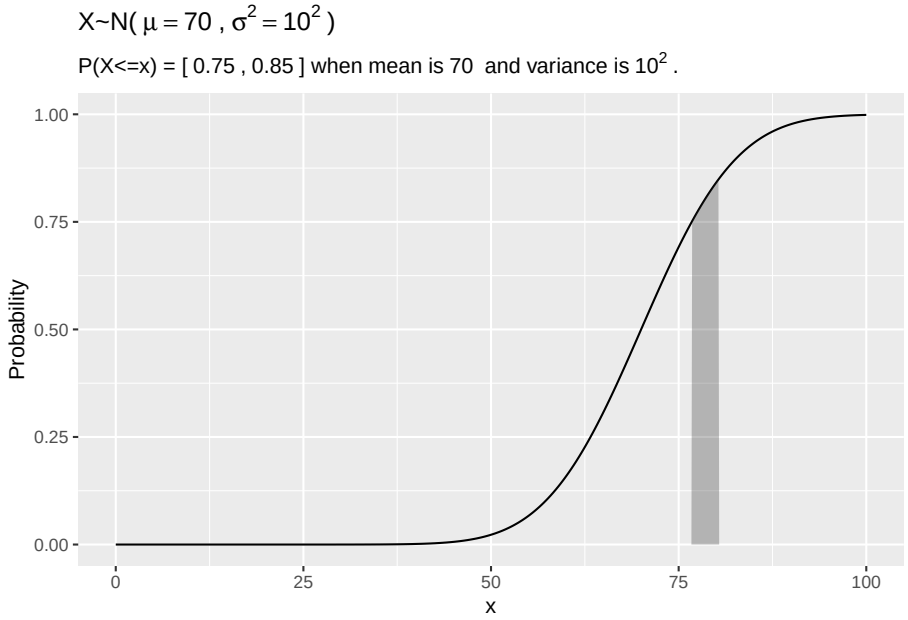
```
## [1] 76.7449
```

```
qnorm(p = my_pct_h, mean = my_mean, sd = my_sd, lower.tail = TRUE)
```

```
## [1] 80.36433
```

```
library(dplyr)
library(ggplot2)

data.frame(x = 0:1000 / 10,
           prob = pnorm(q = 0:1000 / 10,
                        mean = my_mean,
                        sd = my_sd,
                        lower.tail = TRUE)) %>%
  mutate(cdf = ifelse(prob > my_pct_l & prob <= my_pct_h, prob, 0)) %>%
ggplot() +
  geom_line(aes(x = x, y = prob)) +
  geom_area(aes(x = x, y = cdf), alpha = 0.3) +
  labs(title = bquote('X~N('~mu==.(my_mean)~','~sigma^{2}==.(my_sd)^{2}~')'),
       subtitle = bquote('P(X<=x) = ['~.(my_pct_l)~','~.(my_pct_h)~'] when mean is'~.(my_mean)~'
       x = "x",
       y = "Probability")
```



### 1.3.5 Normal Approximation to Binomial

The CLT implies that certain distributions can be approximated by the normal distribution.

The binomial distribution $X \sim B(n, p)$ is approximately normal with mean $\mu = np$ and variance $\sigma^2 = np(1-p)$. The approximation is useful when the expected number of successes and failures is at least 5: $np >= 5$ and $n(1-p) >= 5$.

### 1.3.6 Example

*A measure requires p>=50% popular to pass. A sample of n=1,000 yields x=460 approvals. What is the probability that the overall population approves, P(X)>0.5?*

```
my_x = 460
my_p = 0.50
my_n = 1000

my_mean = my_p * my_n
my_sd = round(sqrt(my_n * my_p * (1 - my_p)), 1)

# Exact binomial
pbinom(q = my_x, size = my_n, prob = my_p, lower.tail = TRUE)
```

```
## [1] 0.006222073
```

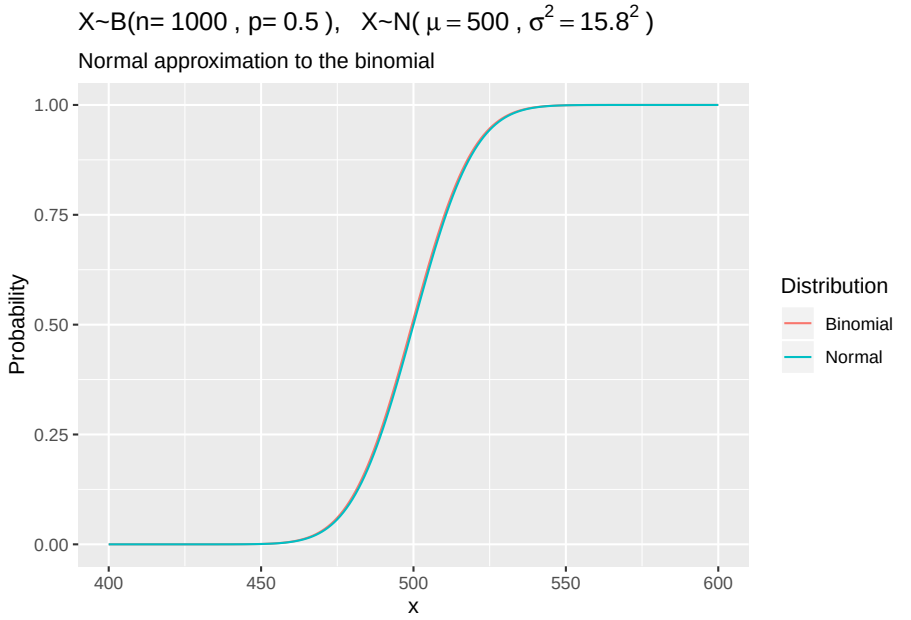
```
# Normal approximation
pnorm(q = my_x, mean = my_p * my_n, sd = sqrt(my_n * my_p * (1 - my_p)), lower.tail = T
```

```
## [1] 0.005706018
```

```
library(dplyr)
library(ggplot2)
library(tidyr)

data.frame(x = 400:600,
           Normal = pnorm(q = 400:600,
                          mean = my_p * my_n,
                          sd = sqrt(my_n * my_p * (1 - my_p)),
                          lower.tail = TRUE),
           Binomial = pbinom(q = 400:600,
                          size = my_n,
                          prob = my_p,
```

```
                        lower.tail = TRUE)) %>%
  gather(key = "Distribution", value = "cdf", c(-x)) %>%
  ggplot(aes(x = x, y = cdf, color = Distribution)) +
  geom_line() +
  labs(title = bquote('X~B(n='~.(my_n)~', p='~.(my_p)~'),  '~'X~N('~mu==.(my_mean)~','~sigma^{2}=
       subtitle = "Normal approximation to the binomial",
       x = "x",
       y = "Probability")
```




The Poisson distribution $x$ $P(\lambda)$ is approximately normal with mean $\mu = \lambda$ and variance $\sigma^2 = \lambda$, for large values of $\lambda$.

### 1.3.7 Example

*The annual number of earthquakes registering at least 2.5 on the Richter Scale and having an epicenter within 40 miles of downtown Memphis follows a Poisson distribution with mean* $\lambda = 6.5$. *What is the probability that at least* $x >= 9$* such earthquakes will strike next year?*

```
my_x = 9
my_lambda = 6.5
my_sd = round(sqrt(my_lambda), 2)
```

```
# Exact Poisson
ppois(q = my_x - 1, lambda = my_lambda, lower.tail = FALSE)
```

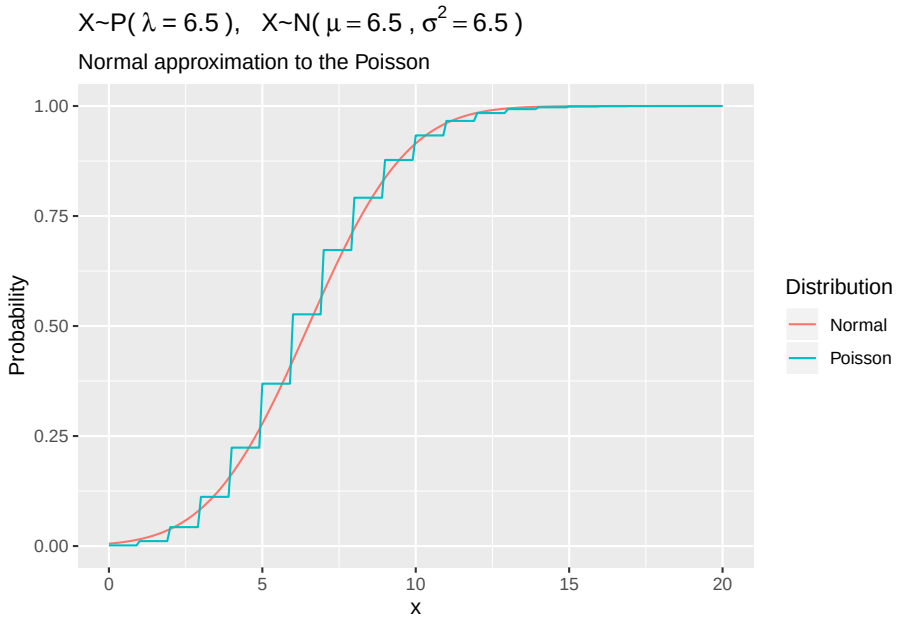
```
## [1] 0.208427
```

```
# Normal approximation
pnorm(q = my_x - 0.5, mean = my_lambda, sd = my_sd, lower.tail = FALSE)
```

```
## [1] 0.216428
```

```
library(dplyr)
library(ggplot2)
library(tidyr)

data.frame(x = 0:200 / 10,
           Normal = pnorm(q = 0:200 / 10,
                          mean = my_lambda,
                          sd = my_sd,
                          lower.tail = TRUE),
           Poisson = ppois(q = 0:200 / 10,
                           lambda = my_lambda,
                           lower.tail = TRUE)) %>%
  gather(key = "Distribution", value = "cdf", c(-x)) %>%
  ggplot(aes(x = x, y = cdf, color = Distribution)) +
  geom_line() +
  labs(title = bquote('X~P('~lambda~'='~.(my_lambda)~'),  '~'X~N('~mu==.(my_lambda)~',
       subtitle = "Normal approximation to the Poisson",
       x = "x",
       y = "Probability")
```




### 1.3.8 From Sample to Population

*Suppose a person's blood pressure typically measures 160?20 mm. If one takes n=5 blood pressure readings, what is the probability the average will be <=150?*

```
my_mu = 160
my_sigma = 20
my_n = 5
my_x = 150

my_se = round(my_sigma / sqrt(my_n), 1)

pnorm(q = my_x, mean = my_mu, sd = my_sigma / sqrt(my_n), lower.tail = TRUE)
```

```
## [1] 0.1317762
```

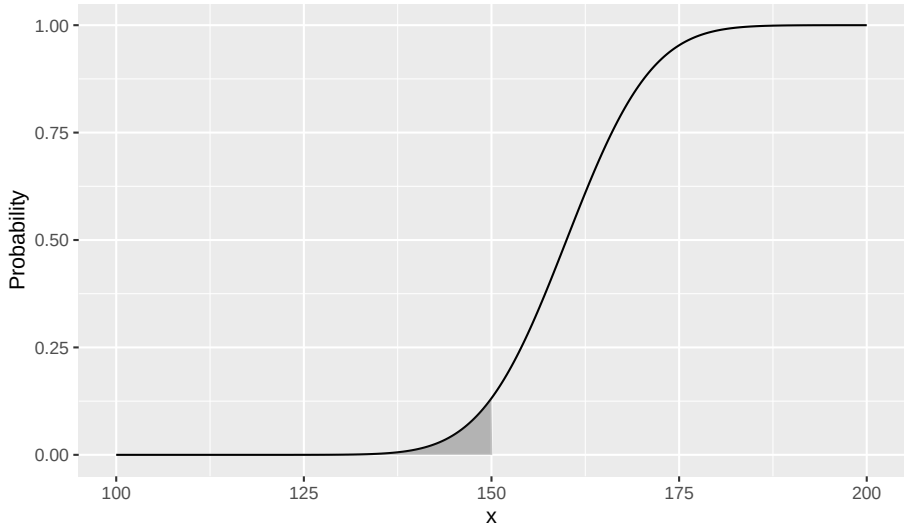
```
library(dplyr)
library(ggplot2)

data.frame(x = 1000:2000 / 10,
           prob = pnorm(q = 1000:2000 / 10,
                        mean = my_mu,
                        sd = my_sigma / sqrt(my_n),
```

```
                          lower.tail = TRUE)) %>%
  mutate(cdf = ifelse(x > 0 & x <= my_x, prob, 0)) %>%
ggplot() +
  geom_line(aes(x = x, y = prob)) +
  geom_area(aes(x = x, y = cdf), alpha = 0.3) +
  labs(title = bquote('X~N('~mu==.(my_mu)~','~sigma^{2}==.(my_se)^{2}~')'),
       subtitle = bquote('P(X<='~.(my_x)~') when mean is'~.(my_mu)~' and variance is'~s
       x = "x",
       y = "Probability")
```

$X \sim N(\mu = 160, \sigma^2 = 8.9^2)$

P(X<= 150 ) when mean is 160 and variance is σ /sqrt(n) $8.9^2$ .




```
knitr::include_app("https://mpfoley73.shinyapps.io/shiny_dist/",
  height = "600px")
```

# Chapter 2

# Inference

# Chapter 3

# Experiments

Some *significant* applications are demonstrated in this chapter.

## 3.1 Example one

## 3.2 Example two

# Chapter 4

# Regression

# Chapter 5

# Classification

# Chapter 6

# Regularization

# Chapter 7

# Non-linear Models

Linear methods can model nonlinear relationships by including polynomial terms, interaction effects, and variable transformations. However, it is often difficult to identify how to formulate the model. Nonlinear models may be preferable because you do not need to know the the exact form of the nonlinearity prior to model training.

## 7.1 Splines

A regression spline fits a piecewise polynomial to the range of $X$ partitioned by *knots* ($K$ knots produce $K + 1$ piecewise polynomials) **James et al** (et al, 2013). The polynomials can be of any degree $d$, but are usually in the range [0, 3], most commonly 3 (a cubic spline). To avoid discontinuities in the fit, a degree-$d$ spline is constrained to have continuity in derivatives up to degree $d-1$ at each knot.

A cubic spline fit to a data set with $K$ knots, performs least squares regression with an intercept and $3 + K$ predictors, of the form

$$y_i = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 h(X, \xi_1) + \beta_5 h(X, \xi_2) + \cdots + \beta_{K+3} h(X, \xi_K)$$

where $\xi_1, \ldots, \xi_K$ are the knots are truncated power basis functions $h(X, \xi) = (X - \xi)^3$ if $X > \xi$, else 0.

Splines can have high variance at the outer range of the predictors. A **natural spline** is a regression spline additionally constrained to be linear at the boundaries.

How many knots should there be, and Where should the knots be placed? It is common to place knots in a uniform fashion, with equal numbers of points

between each knot. The number of knots is typically chosen by trial and error using cross-validation to minimize the RSS. The number of knots is usually expressed in terms of degrees of freedom. A cubic spline will have $K + 3 + 1$ degrees of freedom. A natural spline has $K + 3 + 1$ - 5 degrees of freedom due to the constraints at the endpoints.

A further constraint can be added to reduce overfitting by enforcing smoothness in the spline. Instead of minimizing the loss function $\sum (y - g(x))^2$ where $g(x)$ is a natural spline, minimize a loss function with an additional penalty for variability:

$$L = \sum (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt.$$

The function $g(x)$ that minimizes the loss function is a *natural cubic spline* with knots at each $x_1, \ldots, x_n$. This is called a **smoothing spline**. The larger g is, the greater the penalty on variation in the spline. In a smoothing spline, you do not optimize the number or location of the knots – there is a knot at each training observation. Instead, you optimize $\lambda$. One way to optimze $\lambda$ is cross-validation to minimize RSS. Leave-one-out cross-validation (LOOCV) can be computed efficiently for smoothing splines.

## 7.2 MARS

Multivariate adaptive regression splines (MARS) is a non-parametric algorithm that creates a piecewise linear model to capture nonlinearities and interactions effects. The resulting model is a weighted sum of *basis* functions $B_i(X)$:

$$\hat{y} = \sum_{i=1}^{k} w_i B_i(x)$$

The basis functions are either a constant (for the intercept), a *hinge* function of the form $\max(0, x - x_0)$ or $\max(0, x_0 - x)$ (a more concise representation is $[\pm(x - x_0)]_+$), or products of two or more hinge functions (for interactions). MARS automatically selects which predictors to use and what predictor values to serve as the *knots* of the hinge functions.

MARS builds a model in two phases: the forward pass and the backward pass, similar to growing and pruning of tree models. MARS starts with a model consisting of just the intercept term equaling the mean of the response values. It then asseses every predictor to find a basis function pair consisting of opposing sides of a mirrored hinge function which produces the maximum improvement in the model error. MARS repeats the process until either it reaches a predefined limit of terms or the error improvement reaches a predefined limit.

MARS generalizes the model by removing terms according to the generalized cross validation (GCV) criterion. GCV is a form of regularization: it trades off goodness-of-fit against model complexity.

The `earth::earth()` function (documentation) performs the MARS algorithm (*the term "MARS" is trademarked, so open-source implementations use "Earth" instead*). The caret implementation tunes two parameters: `nprune` and `degree`. `nprune` is the maximum number of terms in the pruned model. `degree` is the maximum degree of interaction (default is 1 (no interactions)). However, there are other hyperparameters in the model that may improve performance, including `minspan` which regulates the number of knots in the predictors.

Here is an example using the Ames housing data set (following this tutorial.

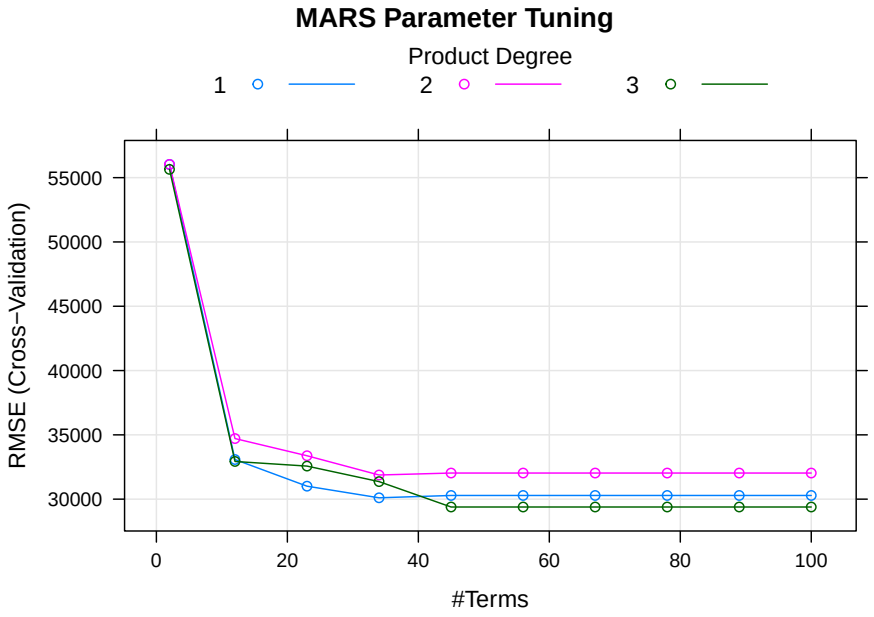
```
library(tidyverse)
library(earth)
library(caret)

# set up
ames <- AmesHousing::make_ames()
set.seed(12345)
idx <- createDataPartition(ames$Sale_Price, p = 0.80, list = FALSE)
ames_train <- ames[idx, ] %>% as.data.frame()
ames_test  <- ames[-idx, ]

m <- train(
  x = subset(ames_train, select = -Sale_Price),
  y = ames_train$Sale_Price,
  method = "earth",
  metric = "RMSE",
  minspan = -15,
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(
    degree = 1:3,
    nprune = seq(2, 100, length.out = 10) %>% floor()
  )
)
```

The model plot shows the best tuning parameter combination.

```
plot(m, main = "MARS Parameter Tuning")
```




```
m$bestTune
```

```
##    nprune degree
## 25     45      3
```

How does this model perform against the holdout data?

```
caret::postResample(
  pred = log(predict(m, newdata = ames_test)),
  obs = log(ames_test$Sale_Price)
)
```

```
##       RMSE   Rsquared        MAE
## 0.16515620 0.85470300 0.09319503
```

## 7.3 GAM

Generalized additive models (GAM) allow for non-linear relationships between each feature and the response by replacing each linear component $\beta_j x_{ij}$ with a nonlinear function $f_j(x_{ij})$. The GAM model is of the form

$$y_i = \beta_0 + \sum f_j(x_{ij}) + \epsilon_i.$$

It is called an additive model because we calculate a separate $f_j$ for each $X_j$, and then add together all of their contributions.

The advantage of GAMs is that they automatically model non-linear relationships so you do not need to manually try out many different transformations on each variable individually. And because the model is additive, you can still examine the effect of each $X_j$ on $Y$ individually while holding all of the other variables fixed. The main limitation of GAMs is that the model is restricted to be additive, so important interactions can be missed unless you explicitly add them.

# Chapter 8

# Classification and Regression Trees

# Chapter 9

# Support Vector Machines

# Chapter 10

# Principal Components Analysis

# Chapter 11

# Clustering

# Chapter 12

# Text Mining

# Appendix

Here are miscellaneous skills, knowledge, and technologies I should know.

# Publishing to BookDown

The **bookdown** package, written by Yihui Xie, is built on top of R Markdown and the **knitr** package. Use it to publish a book or long manuscript where each chapter is a separate file. There are instructions for how to author a book in his bookdown book (Xie, 2019). The main advantage of **bookdown** over R Markdown is that you can produce multi-page HTML output with numbered headers, equations, figures, etc., just like in a book. I'm using **bookdown** to create a compendium of all my data science notes.

The first step to using **bookdown** is installing the **bookdown* package with `install.packages("bookdown")`.

Next, create an account at bookdown.org, and connect the account to RStudio. Follow the instructions at https://bookdown.org/home/about/.

Finally, create a project in R Studio by creating a new project of type *Book Project using Bookdown*.

After creating all of your Markdown pages, knit the book or click the **Build Book** button in the Build panel.

# Bibliography

et al, G. J. (2013). *An introduction to statistical learning : with applications in R*. Springer, New York, NY, 1st edition. ISBN 978-1461471370.

Xie, Y. (2019). *bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman and Hall/CRC, Boca Raton, Florida, 1st edition. ISBN 978113870010.