# Statistical Software Camp: Introduction to R

**Day 3**
**Probability and Statistical Inference via Simulation**

January 28, 2009

## 1 Calculating Probability through Simulation

### 1.1 Basic Concepts

- Recall that the probability can be thought of as the "limit" of repeated identical experiments.
- Use loop to repeat an experiment and calculate an approximate probability of certain events.

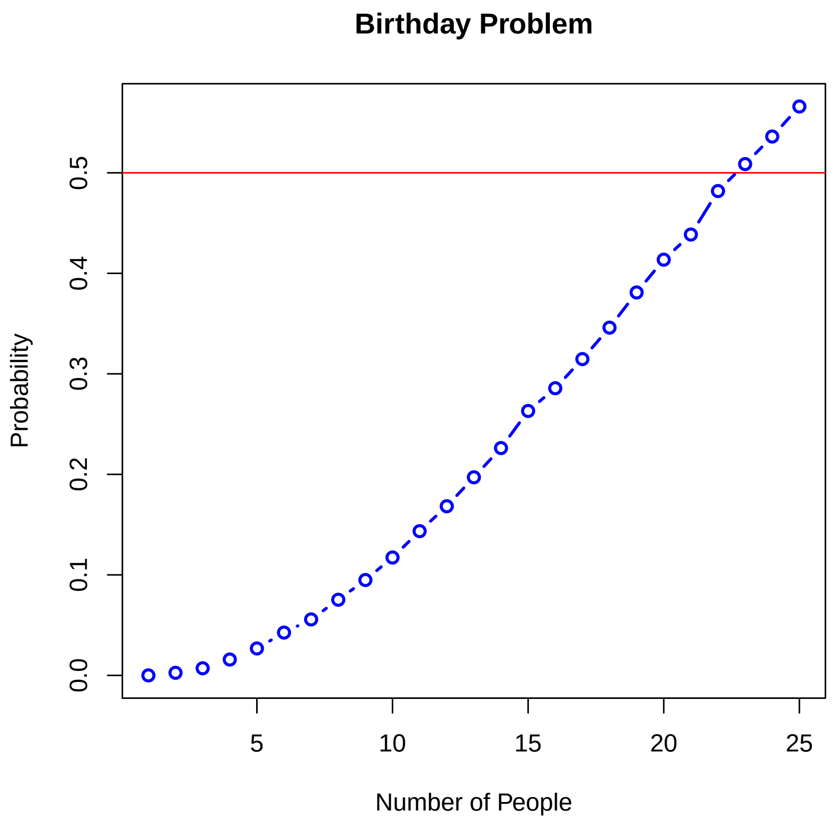
### 1.2 Example 1: Birthday Problem

- How many people do you need in order for the probability that at least two people have the same birthday exceeds 0.5?

```
> sims <- 10000 ## number of simulations
> bday <- 1:365 ## possible birthdays
> answer <- rep(NA, 25) ## holder for our answers
> for (k in 1:25) {
+   count <- 0  ## counter
+   for (i in 1:sims) {
+     class <- sample(bday, k, replace = TRUE) # sampling with replacement
+     if (length(unique(class)) < length(class)) {
+       count <- count + 1
+     }
+   }
+   ## printing the estimate
+   cat("The estimated probability for", k,"people is:", count/sims, "\n")
+   answer[k] <- count/sims # store the answers
+ }
```

```
The estimated probability for 1 people is: 0
The estimated probability for 2 people is: 0.0025
The estimated probability for 3 people is: 0.0071
The estimated probability for 4 people is: 0.0159
The estimated probability for 5 people is: 0.0294
The estimated probability for 6 people is: 0.0414
The estimated probability for 7 people is: 0.0543
The estimated probability for 8 people is: 0.0719
The estimated probability for 9 people is: 0.0928
The estimated probability for 10 people is: 0.1156
The estimated probability for 11 people is: 0.1435
The estimated probability for 12 people is: 0.1641
The estimated probability for 13 people is: 0.1938
The estimated probability for 14 people is: 0.2256
The estimated probability for 15 people is: 0.2571
The estimated probability for 16 people is: 0.2859
The estimated probability for 17 people is: 0.32
The estimated probability for 18 people is: 0.3425
The estimated probability for 19 people is: 0.3805
The estimated probability for 20 people is: 0.4106
The estimated probability for 21 people is: 0.4467
The estimated probability for 22 people is: 0.477
The estimated probability for 23 people is: 0.5239
The estimated probability for 24 people is: 0.5394
The estimated probability for 25 people is: 0.5706
```

```
> ## plotting probability that was saved during the loop
> plot(1:25, answer, type = "b", xlab = "Number of People",
+      ylab = "Probability", main = "Birthday Problem", col="blue", lwd = 2)
> abline(h = 0.5, col = "red")
```



## 1.3 Example 2: Monty Hall Problem

- You must choose one of three doors where one conceals a new car and two conceal old goats. Assume that the probability that each door has a new car is equal to 1/3. Also, assume that your initial choice is random. After you randomly choose one door, the host of the game show, Monty, opens another door which does not conceal a new car. Then, Monty asks you if you would like to switch to the (unopened) third door. Assume that Monty mentally flips a coin to decide which door to open if you initially pick the door with a car. Should you switch?

```
> sims <- 10000
> WinNoSwitch <- 0 # Counter: +1 if you win when not switch
> WinSwitch <- 0 # Counter: +1 if you win when switch
> doors <- 1:3
> for (i in 1:sims){
+   WinDoor <- sample(doors, 1)
+   choice <- sample(doors, 1)
+   if (WinDoor == choice) {
+     WinNoSwitch <- WinNoSwitch + 1
+   }
+     WinSwitch <- WinSwitch + 1
+ }
> cat("Prob(Car | no swtich)=", WinNoSwitch/sims, "\n")

Prob(Car | no swtich)= 0.3405

> cat("Prob(Car | swtich)=", WinSwitch/sims, "\n")

Prob(Car | swtich)= 1
```

# 2 Probability Distributions in R

## 2.1 Random Draws

- The function `rnorm(n, mean, sd)` will create a vector of length `n` containing independent, random draws from a **normal** distribution with the specified `mean` and `sd` (standard deviation).

- The function `runif(n, min, max)` will create a vector of length `n` containing independent, random draws from a **uniform** distribution with lower bound `min` and upper bound `max`.

- The function `rbinom(n, s, p)` will create a vector of length `n` containing independent, random draws from a **binomial** distribution with the size of `s` and the probability of success `p`.

- The function `rt(n, df)` will create a vector of length `n` containing independent, random draws from a *t* distribution with the specified degrees of freedom, `df`.

```
> rnorm(5)  # Five random draws from N(0,1)

[1]  1.6173432  0.2188151 -2.1953822 -1.3520070 -0.8164277

> rnorm(10, -1, 0.2)  # Ten random draws from N(-1,0.2)
```

```
 [1] -1.2700309 -1.0150373 -0.8996691 -1.0163568 -1.3214218 -1.1676961
 [7] -1.1734292 -0.7921740 -0.8313681 -0.9106744
```

```
> runif(2)  # Two random draws from U[0,1]
```

```
[1] 0.7643113 0.9610185
```

```
> runif(6, 70, 90) # Six random draws from U[70,90]
```

```
[1] 89.58772 80.68527 87.33140 70.72850 76.26640 73.88597
```

```
> rbinom(20, 7, 0.55) # Twenty draws from B(7,0.55)
```

```
 [1] 3 5 5 4 3 5 4 4 2 4 3 5 4 4 3 2 3 7 4 4
```

```
> rt(5, 20) # Five draws from t(df=20)
```

```
[1] -1.05958625 -0.01117746 -1.27132233 -1.97743043  0.23589581
```

## 2.2 Distribution Functions

- The **(cumulative) distribution function (CDF)** of the random variable $X$, denoted by $F(x)$, is the function that is equal to the probability of $X$ taking a value less than or equal to $x$.
- The function `pnorm(q, mean, sd, lower.tail=TRUE)` will take in a vector of values, `q`, and report the proportion of all observations we would expect to witness having values `q` or lower, given that the distribution is normal with the specified `mean` and `sd`.
- The function `punif(q, min, max, lower.tail=TRUE)` will take in a vector of values, `q`, and report the proportion of all observations we would expect to witness having values `q` or lower, given that the distribution is uniform with lower bound `min` and upper bound `max`.
- The function `pbinom(q, s, p, lower.tail=TRUE)` will take in a vector of values, `q`, and report the proportion of times we would expect to see `q` or fewer successes when we have sample size `s` and probability of success `p`.
- The function `pt(q, df, lower.tail=TRUE)` will take in a vector of values, `q`, and report the proportion of all observations we would expect to witness having values `q` or lower, given that the distribution is $t$ with the specified `df`.
- For all of these functions, choosing `lower.tail=FALSE` will produce the probility of values *greater* than `q`.

```
> # Normal cumulative probabilities at various sd units
> x <- c(-6, -4, -2, 0, 2, 4, 6)
> pnorm(x, 0, 2)
```

```
[1] 0.001349898 0.022750132 0.158655254 0.500000000 0.841344746 0.977249868
[7] 0.998650102
```

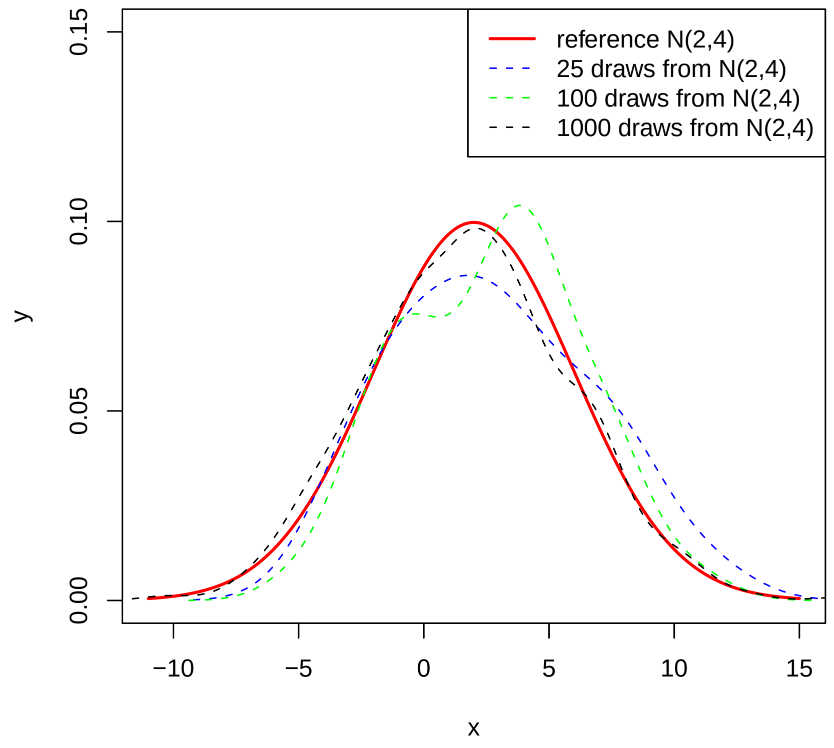
```
> # Uniform cumulative probabilities:
> x <- c(1.2, 1.5, 1.7)
> punif(x, 1, 2) # x or less on a uniform distribution [1,2]

[1] 0.2 0.5 0.7

> punif(x, 1, 2, lower.tail=FALSE) # greater than x on U[1,2]

[1] 0.8 0.5 0.3

> # Binomial cumulative probability of x or fewer successes
> x <- 0:10
> pbinom(x, 10, .88)

 [1] 6.191736e-10 4.602524e-08 1.544425e-06 3.084647e-05 4.068894e-04
 [6] 3.716067e-03 2.393882e-02 1.086818e-01 3.417250e-01 7.214990e-01
[11] 1.000000e+00
```

## 2.3 Probability Density Functions

- The **probability density function (PDF)**, denoted by $f(x)$, charts the height of the density function at point $x$.
- The function `dnorm(x, mean, sd)` will take in a vector of values, `x`, and report the height of the density function at point `x` for the normal distribution with a specific `mean` and `sd`.
- The function `dunif(x, min, max)` will take in a vector of values, `x`, and report the height of the density function at point `x`.
- The function `dbinom(x, s, p)` will take in a vector of values, `x`, and report the probability of seeing exactly `x` successes when we have sample size `s` and probability of success `p`.
- The function `dt(x, df)` will take in a vector of values, `x`, and report the height of the density function at point `x` for the t distribution with a specific `df`.

```
> x <- -1:4
> dnorm(x, 1, 2) # understand why the first=fifth; second=fourth

[1] 0.1209854 0.1760327 0.1994711 0.1760327 0.1209854 0.0647588

> dunif(x, 1, 5) # -1 never occurs, others have equal likelihood

[1] 0.00 0.00 0.25 0.25 0.25 0.25

> dbinom(x,5, 0.4) # -1 never occurs, hence the zero probability

[1] 0.00000 0.07776 0.25920 0.34560 0.23040 0.07680
```

```
> # Compare Random Draws with PDFs
> a1 <- rnorm(25, mean=2, sd=4)
> a2 <- rnorm(100, mean=2, sd=4)
> a3 <- rnorm(1000, mean=2, sd=4)
> x <- seq(from=-11, to=15, length.out=1000)
> y <- dnorm(x, 2, 4)
> plot(x,y,type="l", col="red", lwd=2, ylim=c(0, 0.15))
> lines(density(a1), col="blue", lty=2)
> lines(density(a2), col="green", lty=2)
> lines(density(a3), col="black", lty=2)
> legend("topright", c("reference N(2,4)", "25 draws from N(2,4)",
+     "100 draws from N(2,4)", "1000 draws from N(2,4)"),
+     lty=c(1,2,2,2), lwd=c(2,1,1,1),
+     col=c("red", "blue", "green", "black"))
```



## 2.4 Quantile Functions

- The `qnorm(p, mean, sd, lower.tail=TRUE)` function returns the 100p-th percentile of the normal distribution, with the mean and standard deviation equal to `mean` and `sd`, respectively.

- The `qunif(p, min, max, lower.tail=TRUE)` function returns the 100p-th percentile of the uniform distribution with lower bound `min` and upper bound `max`.

- The `qbinom(p, s, prob, lower.tail=TRUE)` function returns the 100p-th percentile of the binomial distribution with total number of trials `n` and probability of success `prob`.

- The `qt(p, df, lower.tail=TRUE)` function returns the 100`p`-th percentile of the $t$ distribution with degrees of freedom `df`.
- For all of these functions, choosing `lower.tail=FALSE` will return the $100(1-$`p`$)$-th percentile instead.

```
> qnorm(0.975) # 97.5 percentile of N(0,1)

[1] 1.959964

> qt(0.975, 2) # 97.5 percentile of t(df=2)

[1] 4.302653
```

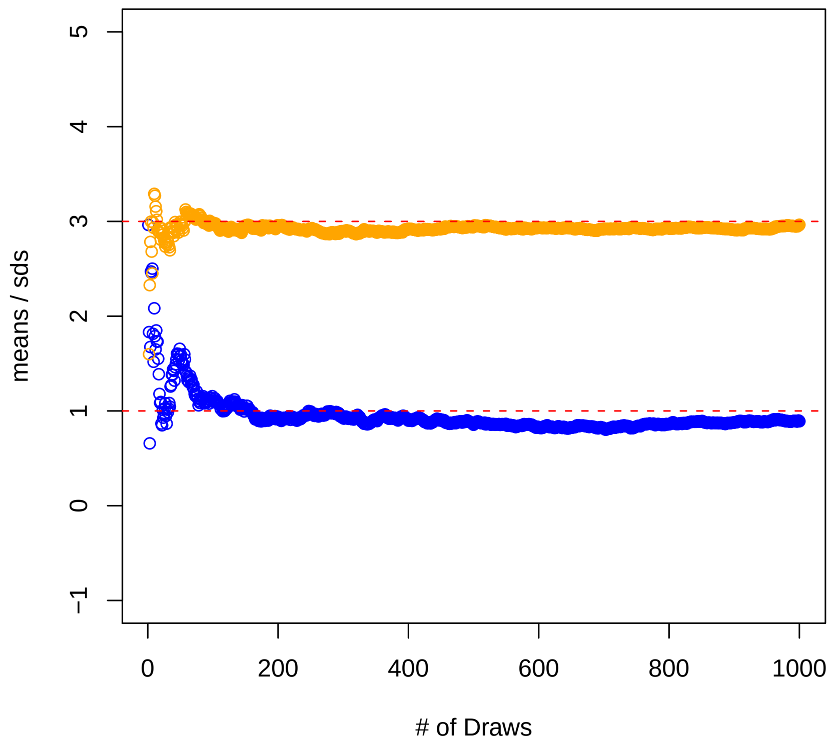
## 3 Statistical Inference via Simulation

### 3.1 Law of Large Numbers

- The **Law of Large Numbers** states that the sample mean approaches the population mean as we increase the sample size.

```
> x5 <- rnorm(5, 1, 3) # Sample size = 5
> mean(x5)

[1] 1.111967

> sd(x5)

[1] 3.640181

> x50 <- rnorm(50, 1, 3) # Sample size = 50
> mean(x50)

[1] 0.3646407

> sd(x50)

[1] 3.050406

> x500 <- rnorm(500, 1, 3) # Sample size = 500
> mean(x500)

[1] 0.8999008

> sd(x500)

[1] 3.223228
```

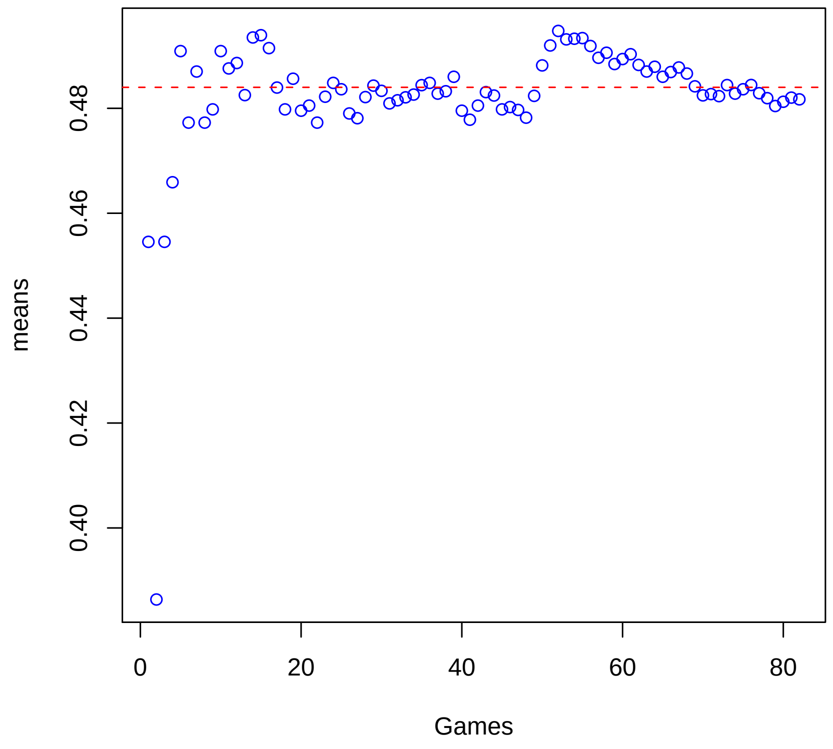
```
> # Verifying the LLN via Simulation:
> n.draws <- 1000
> means <- rep(NA, n.draws); sds <- rep(NA, n.draws)
> x <- NULL
> for (i in 1:n.draws){
+   # Add one draw from N(1,3) to previous draws
+   x <- c(x, rnorm(1,mean=1, sd=3))
+   means[i] <- mean(x)
+   sds[i] <- sd(x)
+ }
> plot(means, col="blue", ylim=c(-1,5), ylab="means / sds", xlab = "# of Draws")
> abline(h=1, col="red", lty=2)
> points(sds, col="orange")
> abline(h=3, col="red", lty=2)
```



- Example: Lebron James' field goal percentage
- Last season James averaged 22 field goal attempts per game and made 48.4% of his shots
- We will take this percentage as his 'true' ability.

```
> # Two games
> lebron.goals.2 <- rbinom(2, 22, 0.484) #22 shots taken, 48.4% success rate
> lebron.ave.2 <- lebron.goals.2 / 22
> mean(lebron.ave.2)
```

```
[1] 0.2954545

> # Season performance
> n.games <- 82 #82 games in the season
> means <- rep(NA, n.games)
> lebron.ave <- NULL
> for (i in 1:n.games){
+   # Add game performance to season performance
+   goals <- rbinom(1, size=22, prob=0.484)
+   lebron.ave <- c(lebron.ave, goals/22)
+   means[i] <- mean(lebron.ave)
+ }
> plot(means, col="blue", xlab="Games")
> abline(h=0.484, col="red", lty=2)
```

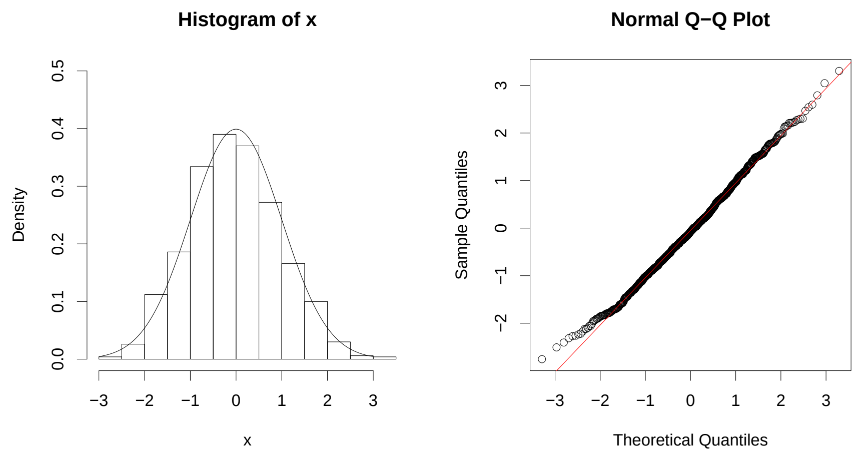


## 3.2 Quantile-Quantile (QQ) Plots

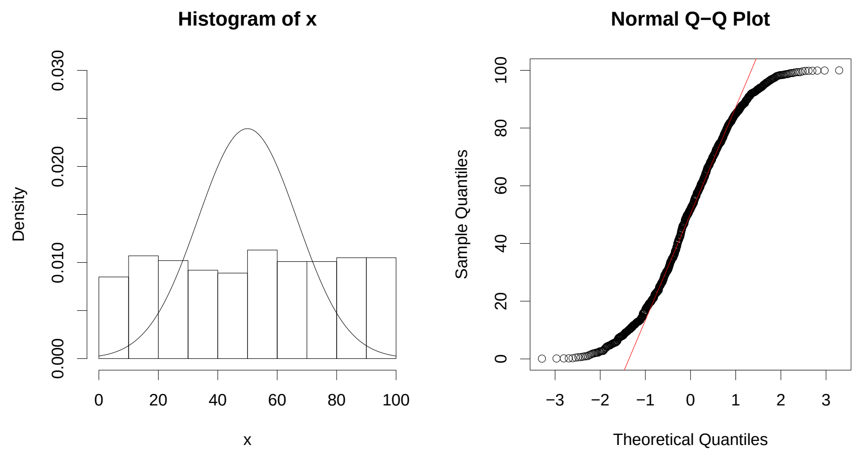
- A quantile-quantile plot (**QQ plot**) allows us to diagnose differences between two probability distributions.
- Typically, we use a QQ plot to compare the empirical distribution of our sample data with a reference distribution (e.g. standard normal).
- If the two distributions are exactly the same, the data points will be on the straight diagonal line.

- The function `qqnorm(x)` creates a normal quantile-quantile (QQ) plot of the values in $x$. We order our data and plot its sample quantiles against the stanard normal quantiles.
- The function `qqline(x)` adds a line to a normal QQ plot which passes through the first and third quartiles. Systematic departure of points from this QQ line would indicate some type of departure from normality (e.g., long tails or skewed distribution) for the sample points.

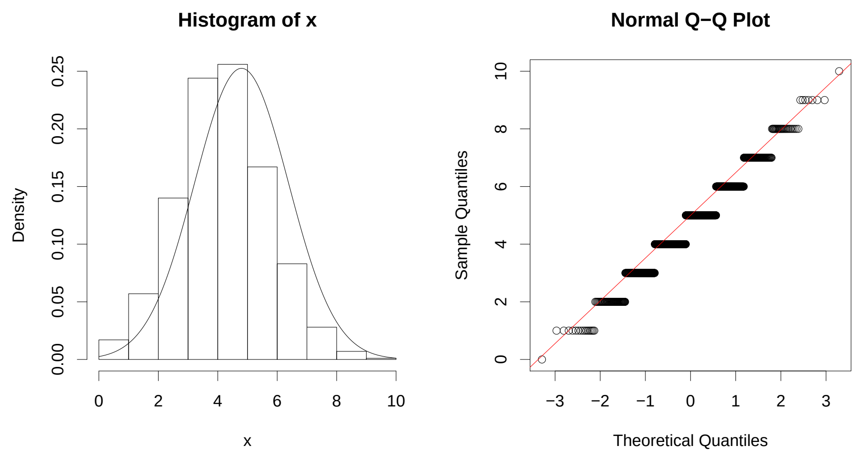
```
> # Normally Distributed Data
> par(mfrow=c(1,2), cex=2)
> x <- rnorm(1000)
> hist(x, freq=F, asp=1, ylim = c(0, 0.5)) # See lecture notes; asp part of plot
> lines(seq(-3, 3, by = 0.01), dnorm(seq(-3, 3, by = 0.01)))
> qqnorm(x); qqline(x, col="red")
```



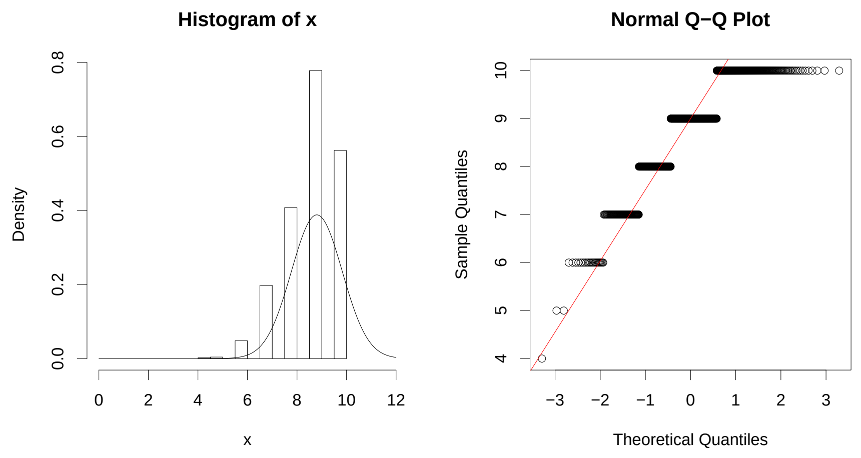
```
> # Uniformly Distributed Data
> par(mfrow=c(1,2), cex=2)
> x <- runif(1000, 0, 100)
> hist(x, freq=F, ylim=c(0,0.03))
> lines(seq(0, 100, by = 0.1), dnorm(seq(0, 100, by = 0.1), 50, 50/3))
> qqnorm(x); qqline(x, col="red") # Notice the divergence from the line at the edges
```



```
> # Binomial Distribution with 50% Success Rate
> par(mfrow=c(1,2), cex=2)
> x <- rbinom(1000, 10, 0.48)
> hist(x, freq=F, ylim=c(0,.25))
> lines(seq(0, 10, by = 0.01),
+       dnorm(seq(0, 10, by = 0.01), 10*0.48, sqrt(10*0.48*0.52)))
> qqnorm(x); qqline(x, col="red") # Reasonably approximates normal distribution
```



```
> # Binomial Distribution with 88% Success Rate
> par(mfrow=c(1,2), cex=2)
> x <- rbinom(1000, 10, 0.88)
> hist(x, freq=F, xlim = c(0, 12))
> lines(seq(0, 12, by = 0.01),
+       dnorm(seq(0, 12, by = 0.01), 10*0.88, sqrt(10*0.88*0.12)))
> qqnorm(x); qqline(x, col="red") # Does not approximate normal distribution
```

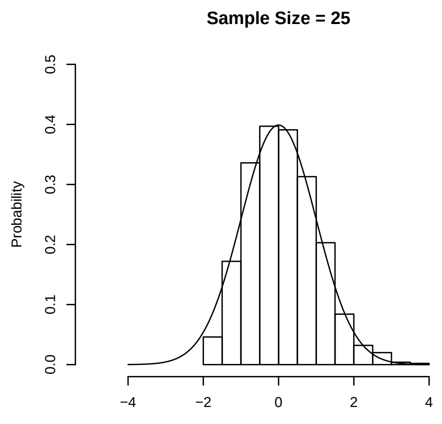


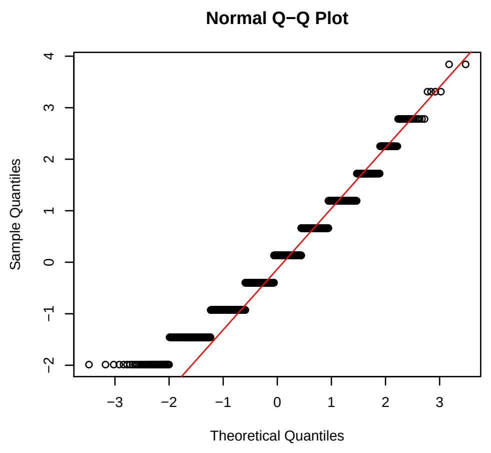
## 3.3 Central Limit Theorem

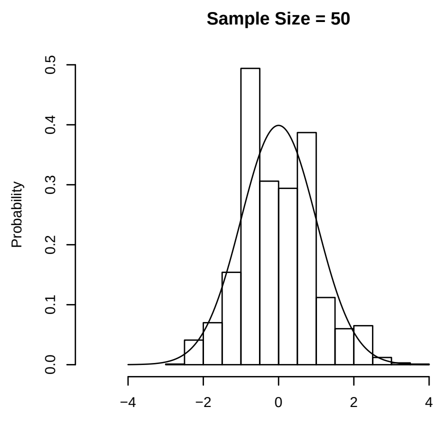
- The **Central Limit Theorem** (CLT) states that the sample mean of identically distributed independent random variables is approximately normally distributed if the sample size is large.

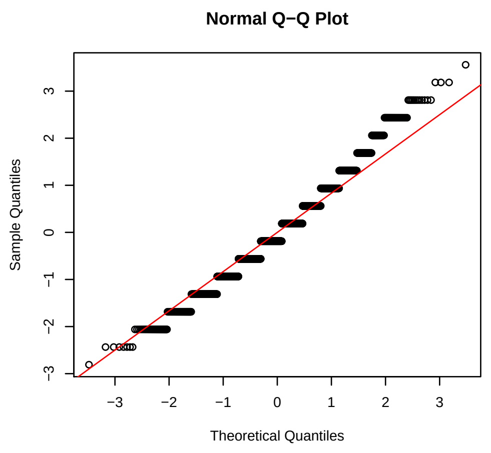
- This is true for virtually any population distribution!

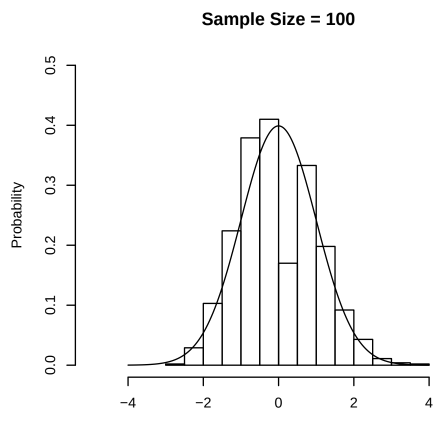
```
> # Example: Binomial population
> p <- 0.05 #success rate -- notice the skew
> n <- 3 #number of trials
> r <- 2000 #number of simulations
> m <- c(25, 50, 100, 500) #sample sizes
> xbar <- NULL
> Z <- matrix(NA, ncol = r, nrow = length(m)) #container matrix
> for (i in 1:length(m)) {
+  for (j in 1:r) {
+    xbar <- mean(rbinom(m[i], size = n, prob = p)) #sample mean
+    Z[i,j] <- (xbar - n*p) / sqrt(n*p*(1-p)/m[i]) #Z score for sample mean
+  }
+ }
> # Displaying the distribution of means
> par(mfrow=c(4,2))
> for (i in 1:length(m)) {
+   hist(Z[i,], xlim = c(-5, 5), freq = FALSE, ylim = c(0, 0.5),
+        ylab = "Probability", xlab = "", main = paste("Sample Size =", m[i]))
+   lines(seq(-4, 4, by = 0.01), dnorm(seq(-4, 4, by = 0.01)))
+   qqnorm(Z[i,]); qqline(Z[i,], col="red") #compare to normal distribution
+ }
```

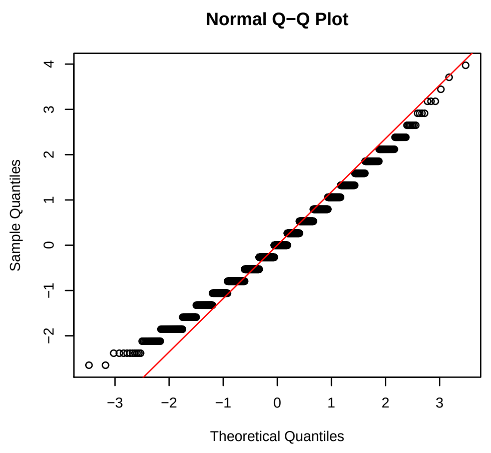

Sample Size = 25
Probability
0.0
0.1
0.2
0.3
0.4
0.5
−4
−2
0
2
4

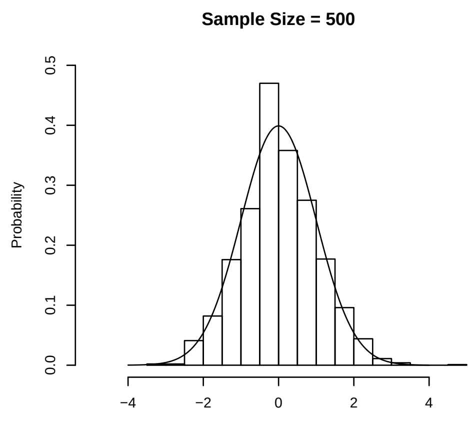

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−2
−1
0
1
2
3
4
−3
−2
−1
0
1
2
3

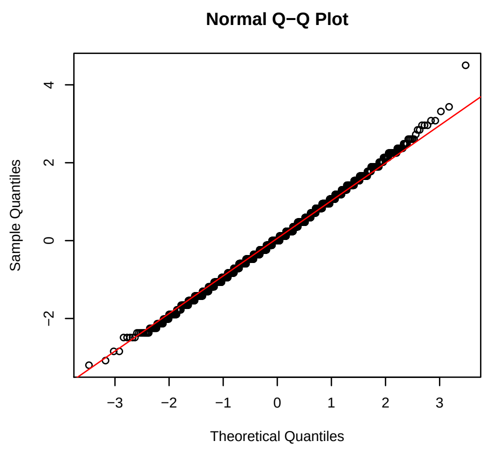

Sample Size = 50
Probability
0.0
0.1
0.2
0.3
0.4
0.5
−4
−2
0
2
4


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−3
−2
−1
0
1
2
3
−3
−2
−1
0
1
2
3


Sample Size = 100
Probability
0.0
0.1
0.2
0.3
0.4
0.5
−4
−2
0
2
4

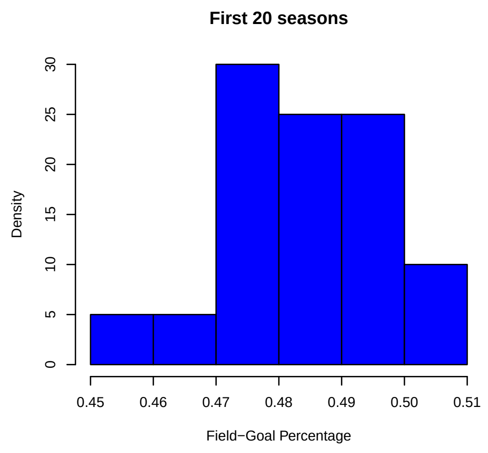

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−2
−1
0
1
2
3
4
−3
−2
−1
0
1
2
3

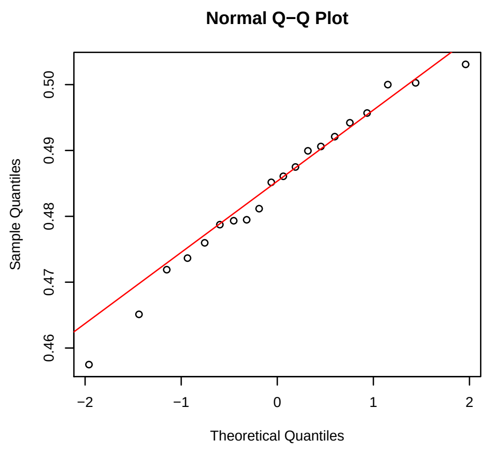

Sample Size = 500
Probability
0.0
0.1
0.2
0.3
0.4
0.5
−4
−2
0
2
4

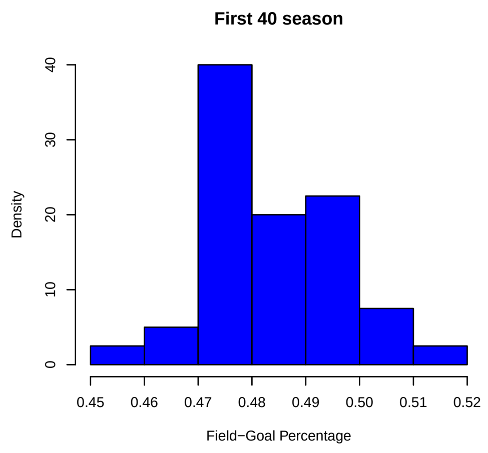

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−2
0
2
4
−3
−2
−1
0
1
2
3

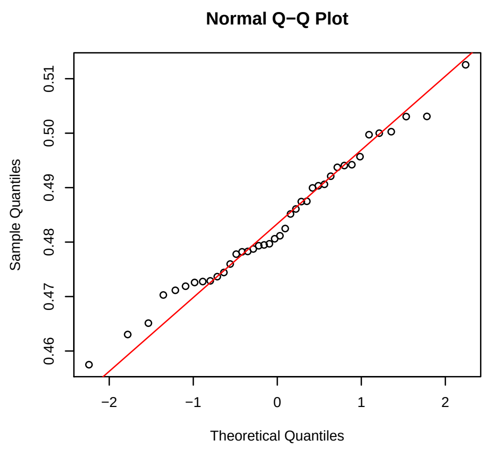
- Example: Lebron James' revisited

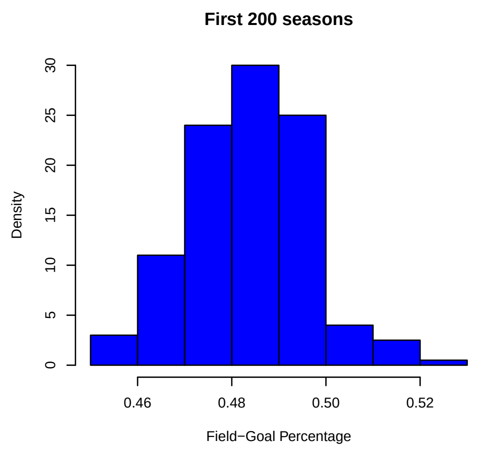
```
> # Simulate 1000 seasons
> n.games <- 82 #number of games in the season
> true.accuracy <- 0.484 #true success rate
> n.sims <- 1000 #number of seasons simulated
> percentages <- rep(NA, n.sims) #stores season-ending field goal percentage
> for (i in 1:n.sims){
+   shots.taken.season <- NULL
+   shots.made.season <- NULL
+   for (j in 1:n.games){
+     shots.taken.game <- round(rnorm(1, 22, 5))
+     shots.taken.game <- ifelse(shots.taken.game<0, 0, shots.taken.game)
+     shots.made.game <- rbinom(1, shots.taken.game, true.accuracy)
+     shots.taken.season <- c(shots.taken.season, shots.taken.game)
+     shots.made.season <- c(shots.made.season, shots.made.game)
+   }
+   percentages[i] <- sum(shots.made.season) / sum(shots.taken.season)
+ }
> # See how season-ending percentages are distributed
> par(mfrow=c(4,2))
> ##First 20 simulated seasons
> hist(percentages[1:20], freq = F, col="blue",
+      main="First 20 seasons", xlab = "Field-Goal Percentage")
> qqnorm(percentages[1:20]); qqline(percentages[1:20], col="red")
> #First 40 simulated seasons
> hist(percentages[1:40], freq = F, col="blue",
+      main="First 40 season", xlab = "Field-Goal Percentage")
> qqnorm(percentages[1:40]); qqline(percentages[1:40], col="red")
> #First 200 simulated seasons
> hist(percentages[1:200], freq = F, col="blue",
+      main="First 200 seasons", xlab = "Field-Goal Percentage")
> qqnorm(percentages[1:200]); qqline(percentages[1:200], col="red")
> #All 1000 simulated seasons
> hist(percentages, freq = F, col="blue",
+      main="All 1000 seasons", xlab = "Field-Goal Percentage")
> qqnorm(percentages); qqline(percentages, col="red")
```

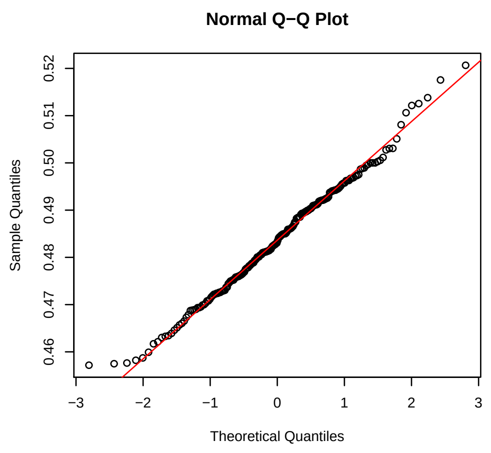

First 20 seasons
Density
Field−Goal Percentage

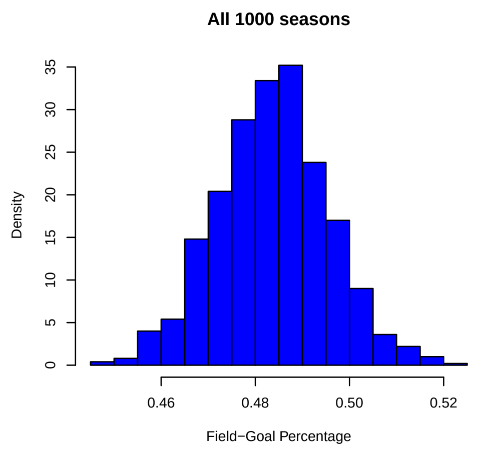

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles

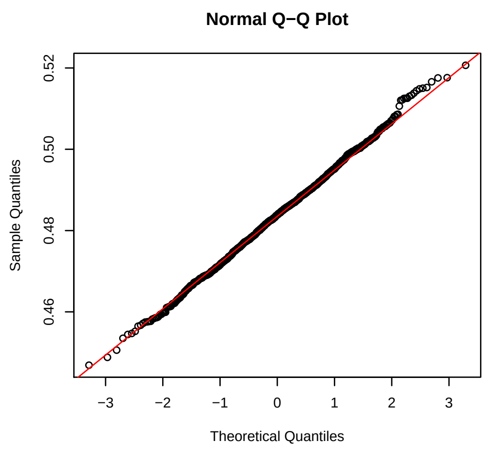

First 40 season
Density
Field−Goal Percentage


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles


First 200 seasons
Density
Field−Goal Percentage


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles

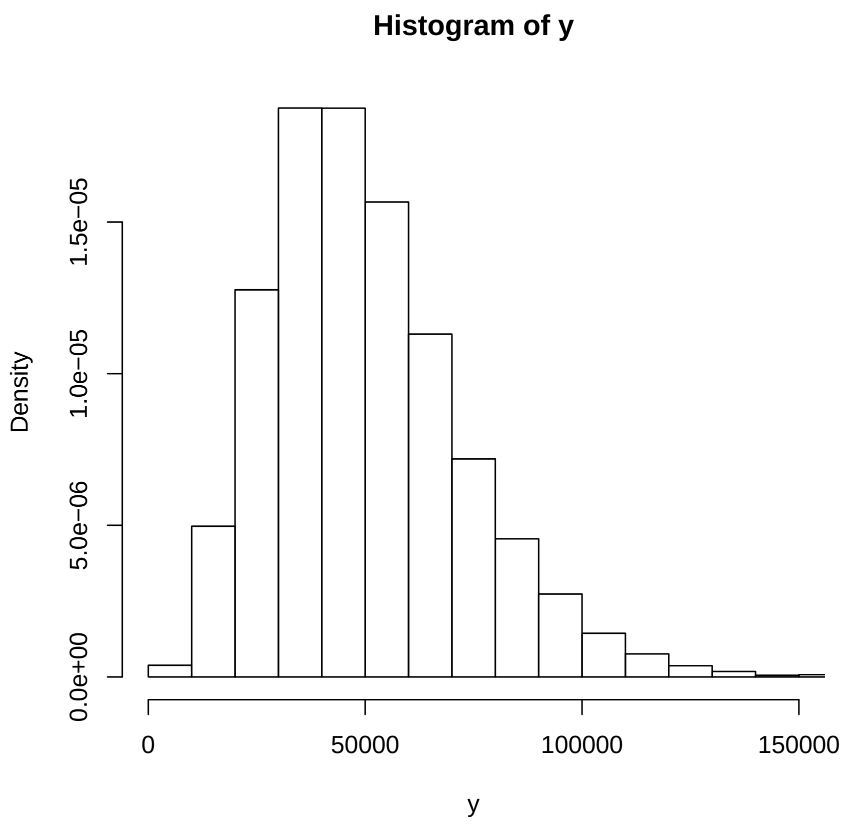

All 1000 seasons
Density
Field−Goal Percentage


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles

## 3.4 Unbiasedness

- A good estimator of a parameter has a sampling distribution that (1) is centered around the true value of the parameter (unbiasedness) and (2) has a small standard error (efficiency).
- An estimator is said to be **unbiased** if its expected value is equal to the parameter. In contrast, a **biased** estimator tests to underestimate the parameter, on average, or it tends to overestimate the parameter.

```
> # Generating income for the population of size 20,000 from the Gamma distribution
> y <- rgamma(20000, 5)*10000
> under100000 <- ifelse(y<100000, 1, 0) # Indicator variable
> hist(y, xlim=c(0,150000), freq=F) # Look at the skewed distribution
> # Generate 1000 samples from this distribution
> # Each time survey 200 people; save just the mean
>
> X1 <- rep(NA, 1000); X2 <- rep(NA, 1000)
> for (i in 1:1000){
+   # All people have equal chance of being surveyed
+   survey <- sample(y, 200)
+   X1[i] <- mean(survey)
+
+   # People with incomes over $100,000 aren't sampled
+   survey2 <- sample(y, 200, prob=under100000)
+   X2[i] <- mean(survey2)
+ }
> mean(y) # Population mean

[1] 49977.65

> mean(X1) # Unbiased estimator

[1] 50029.27

> mean(X2) # Biased estimator; tends to underestimate

[1] 48163.43
```



## 3.5 Confidence Intervals

### 3.5.1 Normal Distribution

- Assume that we have a sample which has the sample mean of 5, the standard deviation of 2, and the sample size is 20. In the example below, we will use a 95% confidence level and wish to find the confidence interval for the population mean.
- Recall that when we use a normal distribution for confidence intervals, we are relying on the approximate (asymptotic) distribution of the sample mean, based on the Central Limit Theorem.

```
> a <- 5 # sample mean
> s <- 2 # standard deviation
> n <- 20 # sample size
> MoE <- qnorm(0.975)*s/sqrt(n) # margin of error
> left <- a - MoE
> right <- a + MoE
> left; right
```

```
[1] 4.123477

[1] 5.876523
```

### 3.5.2 *t* Distribution

- Now, assume that the sample is drawn from a normal population. The sample mean is then exactly distributed according to the t distribution.

```
> MoE.t <- qt(0.975, df = n-1)*s/sqrt(n) # margin of error; t
> left.t <- a - MoE.t
> right.t <- a + MoE.t
> left.t; right.t # notice the interval is slightly wider

[1] 4.063971

[1] 5.936029
```

## 3.6 Coverage Probability of Confidence Intervals

- In repeated sampling, we expect our confidence intervals to contain the truth a set proportion of the time. For example, ninety-five percent of the 95%CIs should contain the true value.
- Biased estimators, however, will in repeated sampling contain the true value less often than expected given the size of the intervals.

```
> # Generate the population distribution
> y <- rgamma(20000, 5)*10000
> under100000 <- ifelse(y<100000, 1, 0) # Indicator variable
> in.interval.ub <- rep(NA, 1000)
> in.interval.b <- rep(NA, 1000)
> for (i in 1:1000){
+   # All people have equal chance of being surveyed
+   survey <- sample(y,200)
+   low <- mean(survey) - qnorm(0.975)*sd(survey)/sqrt(200)
+   hi <- mean(survey) + qnorm(0.975)*sd(survey)/sqrt(200)
+   in.interval.ub[i] <- ifelse(low <= mean(y) & mean(y) <= hi, 1, 0)
+
+   # People with incomes over $100,000 aren't sampled
+   survey2 <- sample(y, 200, prob=under100000)
+   low2 <- mean(survey2) - qnorm(0.975)*sd(survey2)/sqrt(200)
+   hi2 <- mean(survey2) + qnorm(0.975)*sd(survey2)/sqrt(200)
+   in.interval.b[i] <- ifelse(low2 <= mean(y) & mean(y) <= hi2, 1, 0)
+ }
> # Proportion of times confidence interval contains truth
> sum(in.interval.ub)/length(in.interval.ub) # unbiased estimator

[1] 0.944

> sum(in.interval.b)/length(in.interval.b) # biased estimator

[1] 0.699
```

## 3.7 Hypothesis Testing

### 3.7.1 *t*-Test

- We conduct a **one-sample *t*-test** when we want to compare the sample mean from a random sample with its population mean. The null hypothesis for a *two-sided test* takes the form of $H_0 : \mu = \mu_0$, where $\mu$ is the population mean and $\mu_0$ is its hypothesized value.

- We conduct a **two-sample *t*-test** when we want to compare the sample means from two random samples. The null hypothesize for a *two-sided test* takes the form of $H_0 : \mu_1 = \mu_2$ (or equivalently $\mu_1 - \mu_2 = 0$), where $\mu_1$ and $\mu_2$ indicate the population mean of the first and second samples, respectively.

- When we have a strong prior belief about the direction of the gap between $\mu$ and $\mu_0$ (or $\mu_1$ and $\mu_2$ in the two-sample case), we conduct a *one-sided test*, for which we change the equal sign in $H_1$ to an inequality sign (either $>$ or $<$, depending on the direction of the prior belief).

- For a given significance level, the critical value is smaller in a one-sided text than in a two-sided test.

- Note that we need to assume a normal population to justify the use of a *t*-test.

- To conduct a *t*-test, we can use the `t.test()` function in **R**.

- The general syntax for the function is

  ```
  t.test(x, y=NULL, alternative = c("two.sided", "less", "greater"),
  mu = 0, conf.level = 0.95),
  ```

  where

  - `x` is the vector of data — we test whether its mean is statistically different from the hypothesized population parameter `mu`;
  - `conf.level` is the test's level of confidence (defaults to `0.95`);
  - `alternative` specifies a direction of the test (defaults to `two.sided`). One-sided tests correspond to strong prior beliefs regarding the direction of change (e.g. higher or lower) we expect to observe;
  - If users wish to conduct a two-sample t-test, include a second vector of data values, `y`, and `mu` will be the hypothesized difference between the means of `x` and `y`. Hence the options are to compare a sample average to a set proportion, or to compare two distinct samples against one another.

- *Example*: Political opinion of Middle Eastern Americans. We test whether Middle Eastern Americans favored or opposed setting a timetable for withdrawal from Iraq.

- We conduct a one-sample *t*-test because we compare a sample mean to a hypothesized value of the population mean, 0.5.

  ```
  > # T-test for mean: one-sample
  > # 1 = support setting timetable; 0 = oppose setting timetable
  > middle.e <- c(rep(1,47), rep(0,36))
  > # H0: support = 0.5 vs. Ha: support != 0.5
  > t.test(middle.e, alternative="two.sided", mu=0.5, conf.level=0.95)
  ```

```
        One Sample t-test

data:  middle.e
t = 1.2108, df = 82, p-value = 0.2295
alternative hypothesis: true mean is not equal to 0.5
95 percent confidence interval:
 0.4573922 0.6751379
sample estimates:
mean of x
0.5662651
```

- *Example*: Political opinion of Middle Eastern Americans and Native Americans. We test whether the level of support from Middle Eastern Americans differed from that of Native Americans with regard to stem cell research.
- We conduct a two-sample $t$-test because we compare two sample means.

```
> # T-test for mean: two-sample
> # 1 = for government funding stem cell research
> # 0 = against government funding stem cell research
> # H0: middle.e = native.a vs. Ha: middle.e != native.a
> middle.e <- c(rep(1,43), rep(0,40))
> native.a <- c(rep(1,136), rep(0,81))
> t.test(middle.e, native.a, conf.level=0.95)

        Welch Two Sample t-test

data:  middle.e and native.a
t = -1.6912, df = 143.804, p-value = 0.09297
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.23564901  0.01833737
sample estimates:
mean of x mean of y
0.5180723 0.6267281
```

- Suppose we have a strong belief that Middle Eastern Americans are on average less supportive of stem cell research than Native Americans.

```
> # Same framework, except now a one-sided test
> # H0: middle.e = native.a vs. Ha: middle.e < native.a
> t.test(middle.e, native.a, alternative="less", conf.level=0.95)

        Welch Two Sample t-test

data:  middle.e and native.a
t = -1.6912, df = 143.804, p-value = 0.04648
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
        -Inf -0.002291438
sample estimates:
mean of x mean of y
0.5180723 0.6267281
```

### 3.7.2 *z*-test

- When sample size is large enough, we can conduct a $z$-test using the standard normal reference distribution, instead of the $t$ distribution.
- Note that we no longer need to assume a normal population.
- To conduct a $z$-test in R, we need to compute necessary statistics manually. That is,
  - For a one-sample test, use $\overline{X} = \texttt{mean}(X)$ and $s.e. = \sqrt{\texttt{var}(X)/n}$, and compute the $z$-statistic, $(\overline{X} - \mu_0)/s.e.$.
  - For a two-sample test, use $\overline{X}_1 = \texttt{mean}(X_1)$, $\overline{X}_2 = \texttt{mean}(X_2)$ and $s.e. = \sqrt{(\texttt{var}(X_1)/n_1) + (\texttt{var}(X_2)/n_2)}$, and compute the $z$-statistic, $(\overline{X}_1 - \overline{X}_2)/s.e.$
  - Calculate the critical value for the $\alpha$ significance level by $\texttt{qnorm(}1-\alpha/2\texttt{)}$ for a two-sided test and $\texttt{qnorm(}1-\alpha\texttt{)}$ for a one-sided test.

```
> # Normal test for mean: one-sample
> # Same setup, now using a standard normal reference distribution
> middle.e <- c(rep(1,47), rep(0,36))
> mean.ME <- mean(middle.e)
> n.ME <- length(middle.e)
> se.1 <- sqrt(var(middle.e)/n.ME)
> z.1 <- (mean.ME - 0.5)/se.1
> z.1 # z-value

[1] 1.210792

> abs(z.1) > qnorm(0.975) # two-sided test with 0.05 significance level

[1] FALSE

> abs(z.1) > qnorm(0.95) # one-sided test with 0.05 significance level

[1] FALSE

> #make sure sample mean is in hypothesized direction for one-sample test
>
> # Normal test for mean: two-sample
> middle.e <- c(rep(1,43), rep(0,40))
> native.a <- c(rep(1,136), rep(0,81))
> mean.NA <- mean(native.a)
> n.NA <- length(native.a)
> se.2 <- sqrt(var(middle.e)/n.ME + var(native.a)/n.NA)
> z.2 <- ((mean.ME - mean.NA) - 0)/se.2
> z.2 # z-value
```

```
[1] -0.941083

> abs(z.2) > qnorm(0.975) # two-sided test with 0.05 significance level

[1] FALSE

> abs(z.2) > qnorm(0.95) # one-sided test with 0.05 significance level

[1] FALSE
```

## 3.8 $p$-Values

### 3.8.1 Normal Distribution

- Example: A random sample of size 20 with mean 6 and standard deviation 2.
- We find the $p$-value for the null hypothesis of $\mu = 5$ — that is, we compute the probability of having a sample mean of 6 when the true mean was 5.

```
> a <- 5 # mean under H0
> s <- 2 # standard deviation
> n <- 20 # sample size
> xbar <- 6 # sample mean
> z <- (xbar - a)/(s/sqrt(n)) # Z-statistic
> z

[1] 2.236068

> 2*(1 - pnorm(abs(z))) # abs(x) returns the absolute value of x

[1] 0.02534732

> # Altenative method: Compute directly using the upper tail
> 2*(1 - pnorm(xbar, mean = a, sd = s/sqrt(n)))

[1] 0.02534732
```

- To obtain $p$-value for a one-sided test, omit "2*" in the last step of calculation.

```
> 1 - pnorm(abs(z)) # p-value for the one-sided test of H0: mu <= 5 vs. Ha: mu > 5

[1] 0.01267366
```

- To obtain $p$-value for a two-sample test, we follow similar steps except that we must use the appropriate formula for computing the $z$-statistic.

```
> xbar1 <- 5 # sample mean of group 1
> s1 <- 2 # standard deviation of group 1
> n1 <- 20 # sample size of group 1
> xbar2 <- 6 # sample mean of group 2
> s2 <- 2 # standard deviation of group 2
> n2 <- 20 # sample size of group 2
> z <- ((xbar1 - xbar2) - 0) / sqrt(s1^2/n1 + s2^2/n2) # z-score
> z
```

```
[1] -1.581139

> 2 * ( 1 - pnorm(abs(z)) ) # p-value for the two-sided test

[1] 0.1138463
```

### 3.8.2 $t$ Distribution

- The `t.test()` function also reports the $p$-value corresponding to any type of null hypothesis (one-sided or two-sided, one-sample or two-sample).
- Finding the $p$-value using a $t$ distribution manually is very similar to using the $z$-score as demonstrated above. The only difference is that you have to specify the degrees of freedom, $n-1$.

```
> a <- 5 # mean under H0
> s <- 2 # standard deviation
> n <- 20 # sample size
> xbar <- 6 # sample mean
> t <- (xbar - a)/(s/sqrt(n)) # t-statistic
> t

[1] 2.236068

> 2 * ( 1 - pt(abs(t), df = n-1) )

[1] 0.03754055
```

- We can also generate a $p$-value when comparing two samples. However, note that a complex formula must be used to calculate the degrees of freedom.

```
> xbar1 <- 5 # sample mean
> s1 <- 2 # standard deviation
> n1 <- 20 # sample size
> xbar2 <- 6 # sample mean
> s2 <- 2 # standard deviation
> n2 <- 20 # sample size
> t <- (xbar1 - xbar2) / sqrt(s1^2/n1 + s2^2/n2) # t-statistic
> t

[1] -1.581139

> df.12 <- (s1^2/n1 + s2^2/n2)^2 / ((s1^2/n1)^2 / (n1-1) + (s2^2/n2)^2 / (n2-1))
> 2 * (1 - pt(abs(t), df = df.12) )

[1] 0.1221352
```

## 3.9 Power of a Statistical Test

- The power of a statistical test is the probability that the test will reject the null hypothesis when it is false, which equals one minus the probability of making a Type II error.
- In the example below, we will calculate the probability that we can reject the null $H_0 : \mu = 5$ where $\mu$ is the population mean, given a sample standard deviation of 2 and sample size of 20. We assume the normal distribution for the data, and use a 95% confidence level. We wish to find the power to reject when the actual population mean is 6.5. (Note that we need to fix the true parameter, which we never get to know in the real world, in order to calculate the power.)

```
> # H0: mu = 5,
> # Ha: mu != 5.
>
> a <- 5 # null value of parameter
> s <- 2 # sd
> n <- 20 # sample size
> MoE <- qt(0.975,df=n-1)*s/sqrt(n)
> left <- a - MoE
> right <- a + MoE
> left; right # 95% CI you would compute for testing H0

[1] 4.063971

[1] 5.936029

> # Next we find the t-statistics for the left and right
> # values assuming that the true mean is 5 + 1.5 = 6.5:
>
> assumed <- a + 1.5 # assumed true parameter
> tleft <- (left-assumed)/(s/sqrt(n))
> tright <- (right-assumed)/(s/sqrt(n))
> p <- pt(tright, df=n-1) - pt(tleft, df=n-1)
> p #probability that we make a type II error if mu=6.5

[1] 0.1112583

> 1-p #power of the test

[1] 0.8887417
```

- The `power.t.test()` function can help us automate the process of calculating the power of our tests.
- It also calculates the number of observations necessary to reach a certain degree of power. This is called **sample size calculation**.
- The syntax of the function is `power.t.test(n, delta, sd, sig.level, power, type, alternative, strict)`, where
  - `n` is the number of observations;

- `delta` is the true difference in means;
- `sd` is the standard deviation within the population;
- `sig.level` is the test's level of significance (Type I error probability);
- `power` is the power of the test (1 minus Type II error probability);
- `type` is the type of $t$-test (`"two.sample"`,`"one.sample"` or `"paired"`);
- `alternative` specifies a direction of the test (`"two.sided"` or `"one.sided"`);
- `strict = "TRUE"` allows for null hypothesis to be rejected by data in the opposite direction of the truth.

```
> test1 <- power.t.test(n=n,delta=1.5,sd=s,sig.level=0.05,
+     type="one.sample", alternative="two.sided", strict = TRUE)
> test1

     One-sample t test power calculation

              n = 20
          delta = 1.5
             sd = 2
      sig.level = 0.05
          power = 0.8888478
    alternative = two.sided

> names(test1) # shows what the output actually contains

[1] "n"           "delta"       "sd"          "sig.level"   "power"
[6] "alternative" "note"        "method"

> test1$power

[1] 0.8888478

> #sample size calculation using t-test
> power.t.test(delta=1, sd=s, sig=0.01, power=0.95,
+     type="one.sample", alternative="two.sided", strict=TRUE)

     One-sample t test power calculation

              n = 74.6109
          delta = 1
             sd = 2
      sig.level = 0.01
          power = 0.95
    alternative = two.sided
```