Week 4: Sampling Distributions, Error and Estimation

D. ODay

2022-07-06

Sampling Error

	Sample	Population
Definitions	Subgroup or portion of the population chosen for evaluation or study	Collection of all items produced or considered
Characteristics	Statistics	Parameters
Size	n	N
Mean	$\bar{X}$	μ
Median	$\tilde{X}$	M
Standard Deviation	s	σ
Variance	s^2	σ^2
Skewness	g_3	γ_3
Kurtosis	g_4	γ_4
Proportion	p	π
Rate	$\bar{c}$	λ

To create random number is R from a specific distribution we use `rnorm()`, `rexp()`, `rpois()`, `rbinom()`.

```
require(lolcat)
```

```
## Loading required package: lolcat
```

```
## lolcat 2.0.0
```

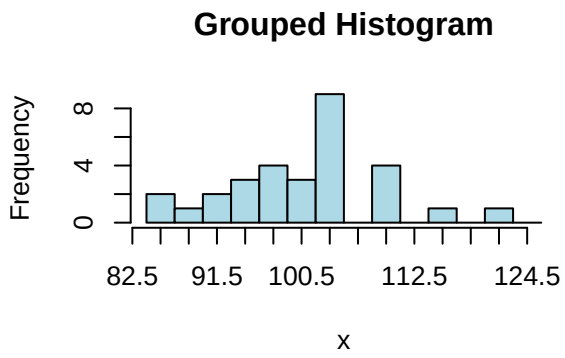
```
# Create four random samples of size n=30 with Mu = 100, sigma = 10
d1<-rnorm(n = 30,mean = 100, sd = 10)
d2<-rnorm(n = 30,mean = 100, sd = 10)
d3<-rnorm(n = 30,mean = 100, sd = 10)
d4<-rnorm(n = 30,mean = 100, sd = 10)

# Create a dataframe of all variables
normdata<-data.frame(d1,d2,d3,d4)

# Review summary statistics of all variables
summary.all.variables(normdata, stat.sd=T)
```

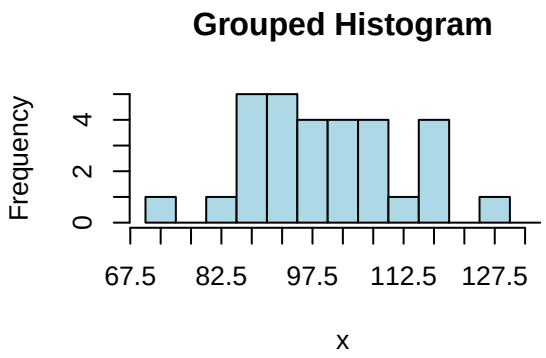
```
##   dv.name  n missing     mean       var        sd g3.skewness   g3test.p
## 1      d1 30       0 100.9108  68.29416  8.264028   0.2033879 0.61488748
## 2      d2 30       0 100.0783 159.64302 12.634992   0.1506734 0.70875097
## 3      d3 30       0 102.9199 112.88729 10.624843  -0.3160431 0.43745180
## 4      d4 30       0 103.5996 124.43291 11.154950   0.7592867 0.07564044
##   g4.kurtosis  g4test.p
## 1  0.60999318 0.3632300
## 2 -0.05923686 0.8779961
## 3  0.69574846 0.3227384
## 4  1.22409943 0.1558770
```

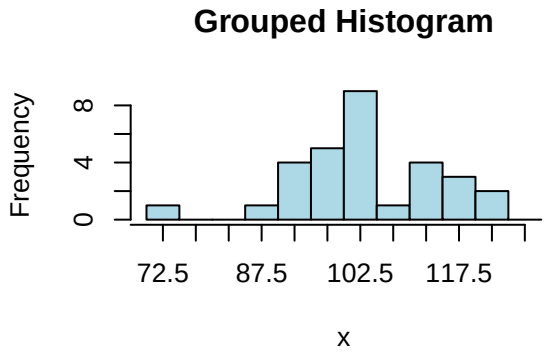
```
# Make output easier to read
nqtr<-function(x,d){noquote(t(round.object(x, d)))}
nqtr(summary.all.variables(normdata, stat.sd=T),3)
```

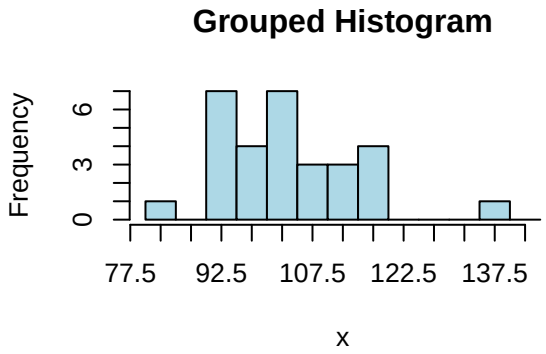
```
##             1       2       3       4
## dv.name     d1      d2      d3      d4
## n           30      30      30      30
## missing     0       0       0       0
## mean        100.911 100.078 102.920 103.600
## var          68.294 159.643 112.887 124.433
## sd           8.264  12.635  10.625  11.155
## g3.skewness  0.203   0.151  -0.316   0.759
## g3test.p    0.615   0.709   0.437   0.076
## g4.kurtosis  0.610  -0.059   0.696   1.224
## g4test.p    0.363   0.878   0.323   0.156
```

```
# Create histograms of each variable in one plot
# Set parameters for graphical output, and create a matrix of nrows x ncols
par(mfrow=c(2,2))
hist.grouped(normdata$d1)
hist.grouped(normdata$d2)
hist.grouped(normdata$d3)
hist.grouped(normdata$d4)
```

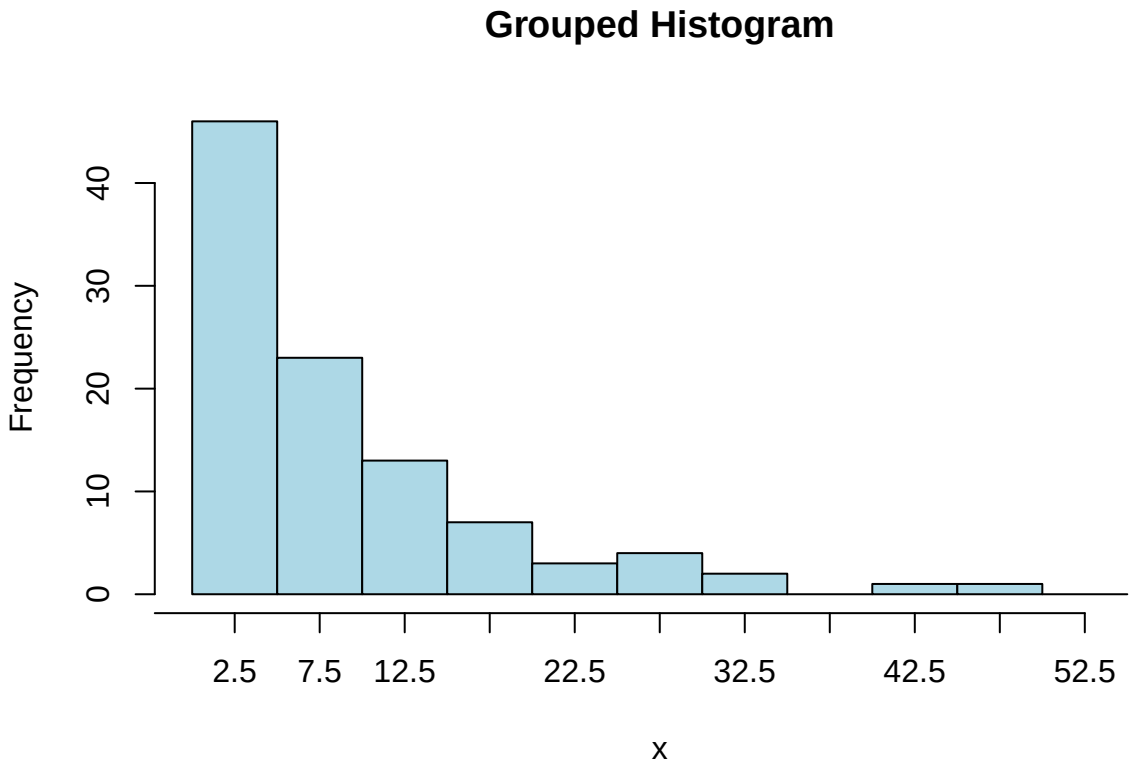









```
# Set parameters back to one graph
par(mfrow=c(1,1))
randoexp<-rexp(n = 100, rate = 1/10)
hist.grouped(randoexp)
```



Sampling Error

- An expected phenomenon since we are not measuring all of the subjects or units for the entire population
- Statistical Methods allow us to account for sampling error, and make appropriate decisions
- In spite of the presence of sampling error, random sampling allow us to use sample statistics as point estimators of population parameters; however, even when unbiased. It will not exactly equal the true value
- An observed difference between a true parameter value and its associated sample descriptive statistic is caused by **sampling error**
- The expected and quantifiable discrepancy between a population parameters and its associated descriptive statistic due to the sample size employed, and in the case of some descriptive statistics, the variability of the population.

-Sampling error is quantifiable using Random Sampling Distributions (RSDs)

- These distributions, like al probability distributions, are based on the principles of classical probability

Random Sampling Distributions and CLT

Random Sampling Distributions

- A RSD is the distribution of a sample statistic calculated from all possible random samples of the given (fixed) size from a given population
  - It is a population distribution and foundational for understanding statistical inference
- To create a RSD
  - Draw all possible random samples of size n from a given research population
  - Calculate descriptive statistics for each of the samples
  - Construct a distribution for each of the sampled descriptive statistics
  - Each of thje resultant distributions constitutes the RSD of the statistics

```
# Random Sampling Distribution Simulation
nqtr<-function(x,d){noquote(t(round.object(x, d)))}

# First, set seed so we will get the same results
set.seed(133)

# Create a distribution with mean of 10 and standard deviation of 2
pop<-rnorm(n = 50000, mean = 10, sd = 2)

# Create a histogram of the population distribution
hist.grouped(pop, anchor.value = 0)
hist.grouped(pop, anchor.value=0)
```

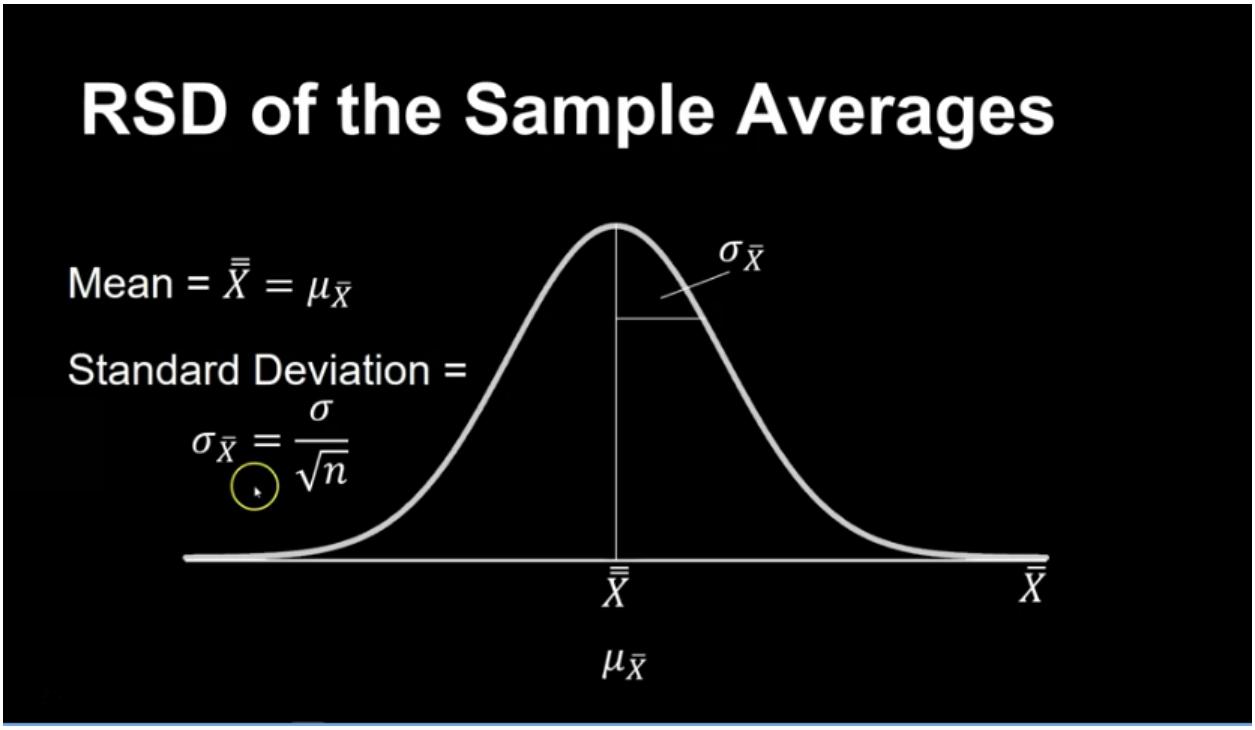
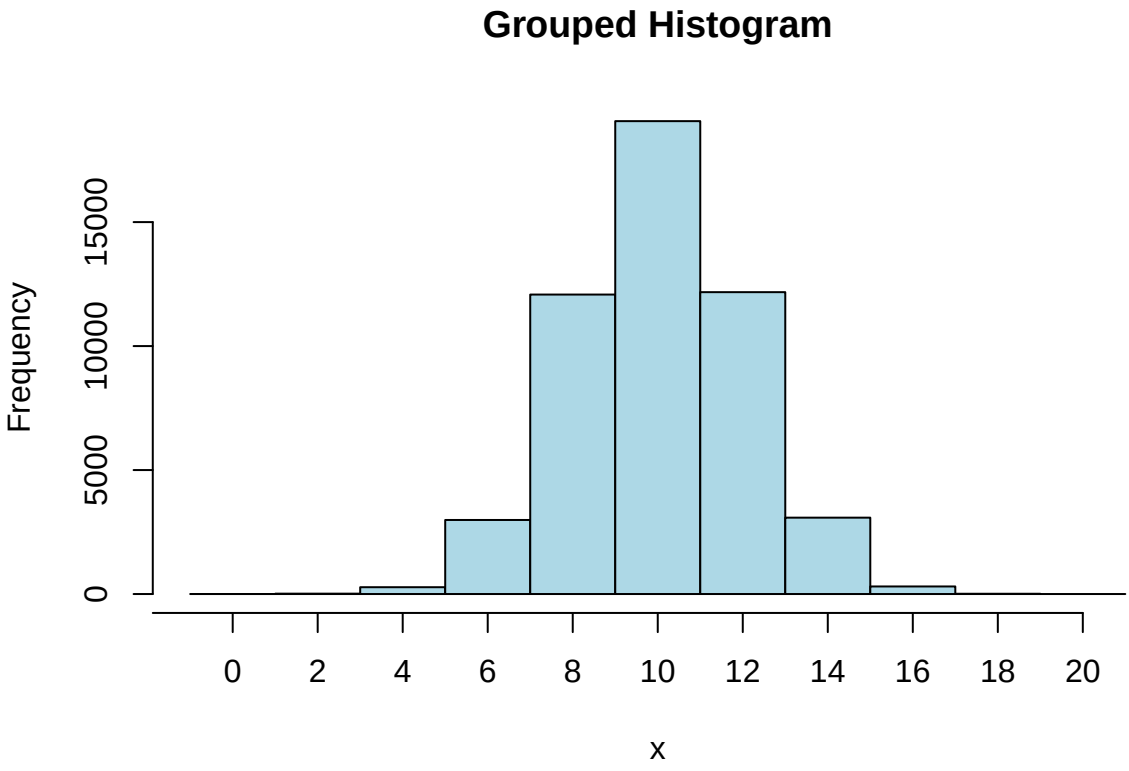


Figure 1: Ex Distribution




```
# Calculate the descriptive statistics of the population distribution
nqtr(summary.continuous(pop, stat.sd=T),5)
```

```
##             1
## dv.name     fx
## n           50000
## missing     0
## mean        10.00851
## var         4.00936
## sd          2.00234
## g3.skewness 0.00476
## g3test.p    0.66361
## g4.kurtosis -0.00231
## g4test.p    0.92471
```

```r
# Next, create / simulate a random sampling distribution
# Set the sample size equal to 4
n<-4

# Create the number of repetitions
# (number of times we will take a sample of size 4)
reps<-5000

# Take the random samples from a population with a mean of 10 and
# standard deviation of 2
samples <- replicate(reps, rnorm(n, mean = 10, sd = 2))

# Calculate the averages of each sample of 4
sampleavgs <- colMeans(samples)

# Calculate the descriptive statistics of the RSD
nqtr(summary.continuous(sampleavgs, stat.sd=T),5)
```

```
##             1
## dv.name     fx
## n           5000
## missing     0
## mean        9.97972
## var         1.02914
## sd          1.01447
## g3.skewness 0.01886
## g3test.p    0.58577
## g4.kurtosis -0.01807
## g4test.p    0.81852
```

```r
# Create 2 histograms of the population and sample averages
# using the same axis to compare

# Create a 1 x 2 matrix
dev.off() # turns off the default
```

```
## null device
##           1
```

```
layout(matrix(1:2, nrow=2))

# Create both histograms
hist.grouped(pop, xlim=c(0,22), xaxt='n', width.consider = 1)
axis(side = 1, at = seq(0,22,2), labels = seq(0,22,2))

hist.grouped(sampleavgs, xlim=c(0,22), xaxt='n', width.consider = 1)
axis(side = 1, at = seq(0,22,2), labels = seq(0,22,2))
```

Probability with RSDs

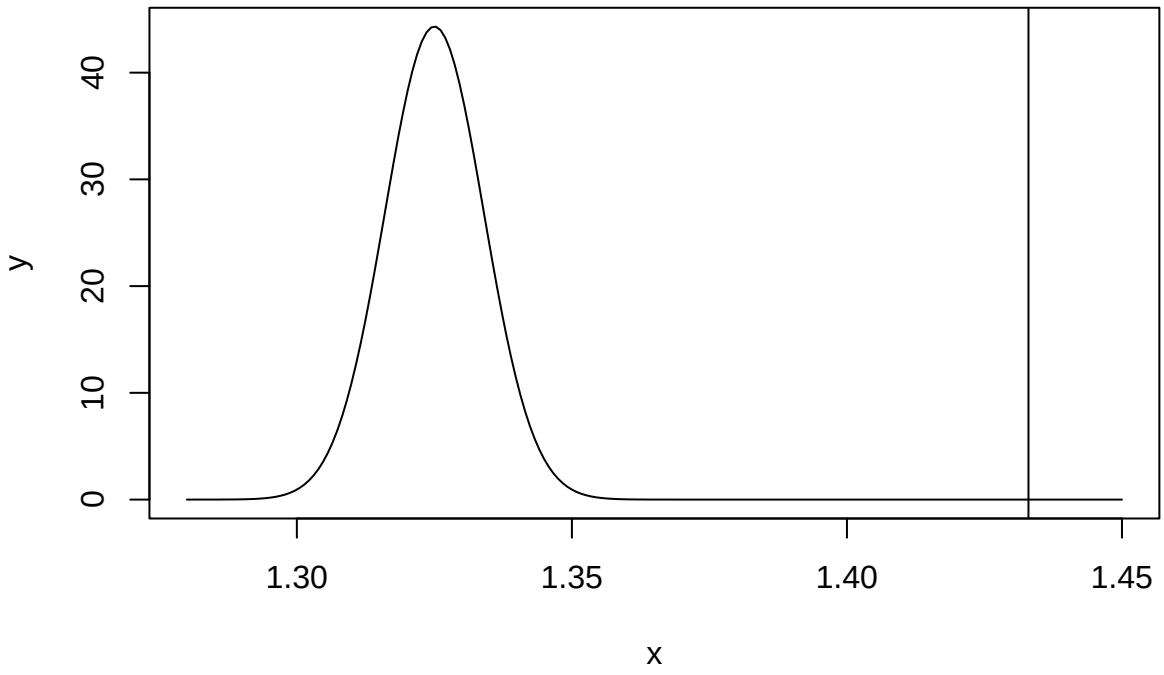
Can estimate probability with z-score

$$Z_{\bar{X}} = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

```
# Using the RSD to solve probability problems
# Example 1
# Define the variables in the problem
mu1<-1.325
sigma1<-0.045
n1<-25
xbar1<-1.433
stderror1<-sigma1/sqrt(n1)

# Calculate the area under the normal curve using the pnorm function
pnorm(q = xbar1, mean = mu1,sd = stderror1, lower.tail = F)
```

```
## [1] 1.776482e-33
```

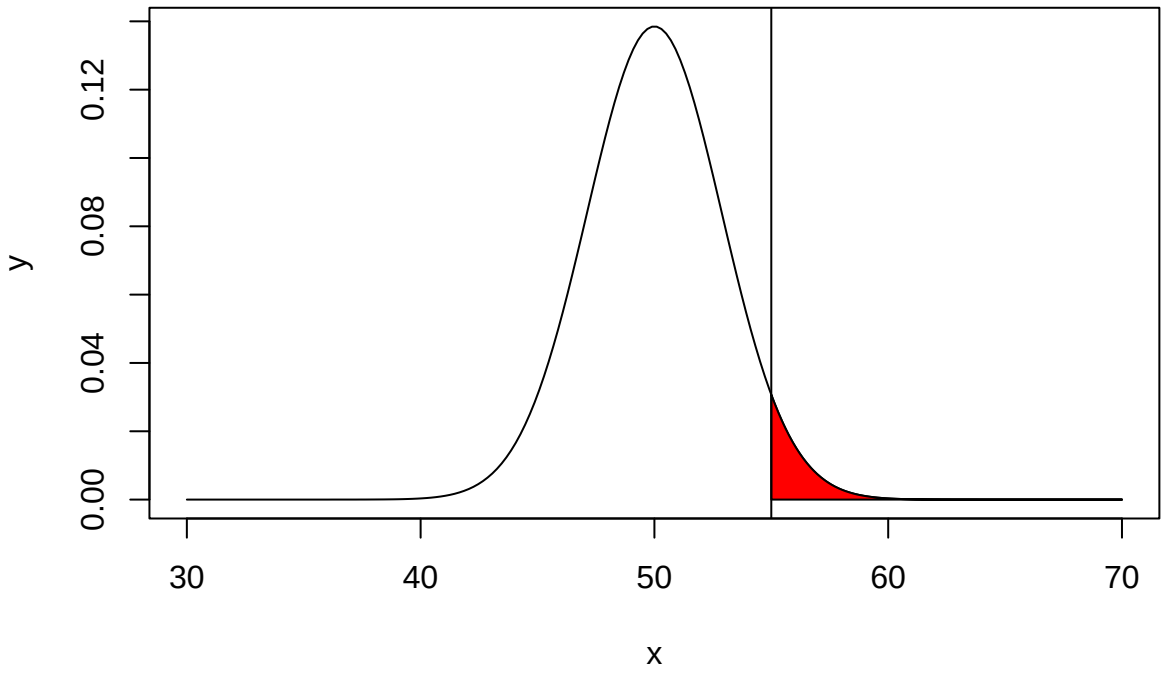
```
# Create the normal curve
x=seq(1.28,1.45,length=200)
y=dnorm(x,mean=mu1,sd=stderror1)
plot(x,y,type="l")

# Indicate the location of the xbar of 1.433
abline(v=1.433)
```




```
# Example 2
# Define the variables in the problem
mu2<-50
sigma2<-14.4
n2<-25
xbar2<-55
stderror2<-sigma2/sqrt(n2)

# Calculate the area under the normal curve using the pnorm function
pnorm(q = xbar2, mean = mu2,sd = stderror2, lower.tail = F)
```

```
## [1] 0.0412721
```

```
# Create the normal curve
x=seq(30,70,length=200)
y=dnorm(x,mean=mu2,sd=stderror2)
plot(x,y,type="l")

# Shade the upper tail area
x=seq(55,70,length=100)
y=dnorm(x,mean=mu2,sd=stderror2)
polygon(c(55,x,70),c(0,y,0),col="red")

# Indicate the location of the xbar of 55
abline(v=55)
```



Statistic	RSD	Standard Error
$\bar{X}$	RSD of the mean	of the mean
$\tilde{X}$	RSD of the median	of the median
p	RSD of the proportion	of the proportion
R	RSD of the range	of the range

Figure 2: RSD Example

Estimates and Estimators

- **Point Estimate**
  - single number used to estimate an unknown parameter
- **Interval Estimate**
  - Range of values to estimate a parameter
- **Estimator**
  - A sample statistic used to estimate a population parameter. An estimate is a specific observed value of a statistic
- Criteria for "Good" Estimators
  - Unbiased

    - the mean of the RSD of the estimator is equal to the parameter it estimates
- Efficiency
    - The standard error of the statistic RSD. The most efficient estimator is the one with the smallest standard err
- Consistency
    - Refers to the assumption that as n increases, the value of the statistic approaches the value of its associated population parameter
- Sufficient
    - Using all possible info the in the sample to estimate the corresponding parameter
    - example is using sd or var compared to the range since the range only uses two values

Point Estimate		Population Parameter	
Sample Mean	$\bar{X}$	Population Mean	μ
Sample Variance	s^2	Population Variance	σ^2
Sample Proportion	p	Population Proportion	π
Sample Count	c	Population Count	λ
Sample Skewness	g_3	Population Skewness	γ_3
Sample Kurtosis	g_4	Population Kurtosis	γ_4

Point Estimate	In RStudio
$\bar{X}$	`mean()`
s	`sd()`
p	`mean() # average proportion`
c	`mean() # average count per unit`

Confidence Intervals

Confidence Intervals

- The confidence level is the probability associated with an interval estimate
- This refers to the probability that the interval estimate includes the population parameter
- levels are 90, 95 and 99%

-The confidence interval is the range of the estimate. The range of values to find the true population parameter with a given level of confidence

- The interval estimate provides a way to qualify our estimate by indicating the magnitude of the sampling error and hence, the precision of our estimate
- To find this interval, we must look at the set of all possible parameters and assess each of those paramters for their probability of providing us with the sample statistic we observed

Confidence Intervals for the Mean and Variance

```
require(lolcat)
Point_Estimates <- read.delim("~/Documents/GitHub/school_cu/school_cu/methods for quality improvement/D

ro<-round.object
nqtr<-function(x,d){noquote(t(round.object(x, d)))}
options(scipen=999)
```

- Mean
  - When sd is known
  - When sd in unknown
- Standard Deviation / Variance

Mean (Sigma Known)

$$\mu_{CI} = \bar{X} \pm z_{\alpha/2} * \frac{\sigma}{\sqrt{n}}$$

```
# Confidence Interval for the Mean (Sigma is Known) ----------------------
# Example 1
n<-150
xbar<-20
sd<-5
conf<-0.95

#Use .simple syntax if not using dataframe. Otherwise, don't use the .simple
z.test.onesample.simple(sample.mean = 20
                        ,known.population.variance = 5^2
                        ,sample.size = 150
                        ,conf.level = 0.99)
```

```
##
##  One-Sample Z Test For Means
##
## data:  sample mean, population variance, sample size
## z statistic = 48.99, null hypothesis mean = 0, p-value <
## 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  18.94842 21.05158
```

```
## sample estimates:
## sample.mean sample.size      se.est       power
##  20.0000000 150.0000000   0.4082483   1.0000000
```

```
# Round the output
ro(z.test.onesample.simple(sample.mean = 20
                          ,known.population.variance = 5^2
                          ,sample.size = 150
                          ,conf.level = 0.95),2)
```

```
##
##  One-Sample Z Test For Means
##
## data:  sample mean, population variance, sample size
## z statistic = 48.99, null hypothesis mean = 0, p-value <
## 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  19.2 20.8
## sample estimates:
## sample.mean sample.size      se.est       power
##       20.00      150.00        0.41        1.00
```

Means (Sigma Unknown)

- Use t distribution
  - lower at the mean and lower at the tails compared to the normal distribution
- t-distribution considers the fact there is error associated with the use of s, teh sample standard deviation to estimate the population (s)
- This error increases the variability of the resulting statistic, the t, relative to a standard normal distribution

$$\mu_{CI} = \bar{X} \pm t_{\alpha/2,(n-1)df} * \frac{s}{\sqrt{n}}$$

```
# Confidence Interval for the Mean (Sigma is Unknown) ---------------------
# Example 2
n<-14
xbar<-15000
sd<-500
conf<-0.99

t.test.onesample.simple(sample.mean = xbar
                          ,sample.variance = sd^2
                          ,sample.size = n
                          ,conf.level = conf)
```

```
##
##  One-Sample t Test For Means
##
```

```
## data:  sample mean, sample size, and estimated variance
## t statistic = 112.25, null hypothesis mean = 0, p-value <
## 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  14597.47 15402.53
## sample estimates:
## sample.mean      se.est          df var.lowerci         var var.upperci
##  15000.0000    133.6306     13.0000 108989.1895 250000.0000 911632.1111
##  sd.lowerci          sd  sd.upperci       power
##    330.1351    500.0000    954.7943      1.0000
```

Standard Deviation

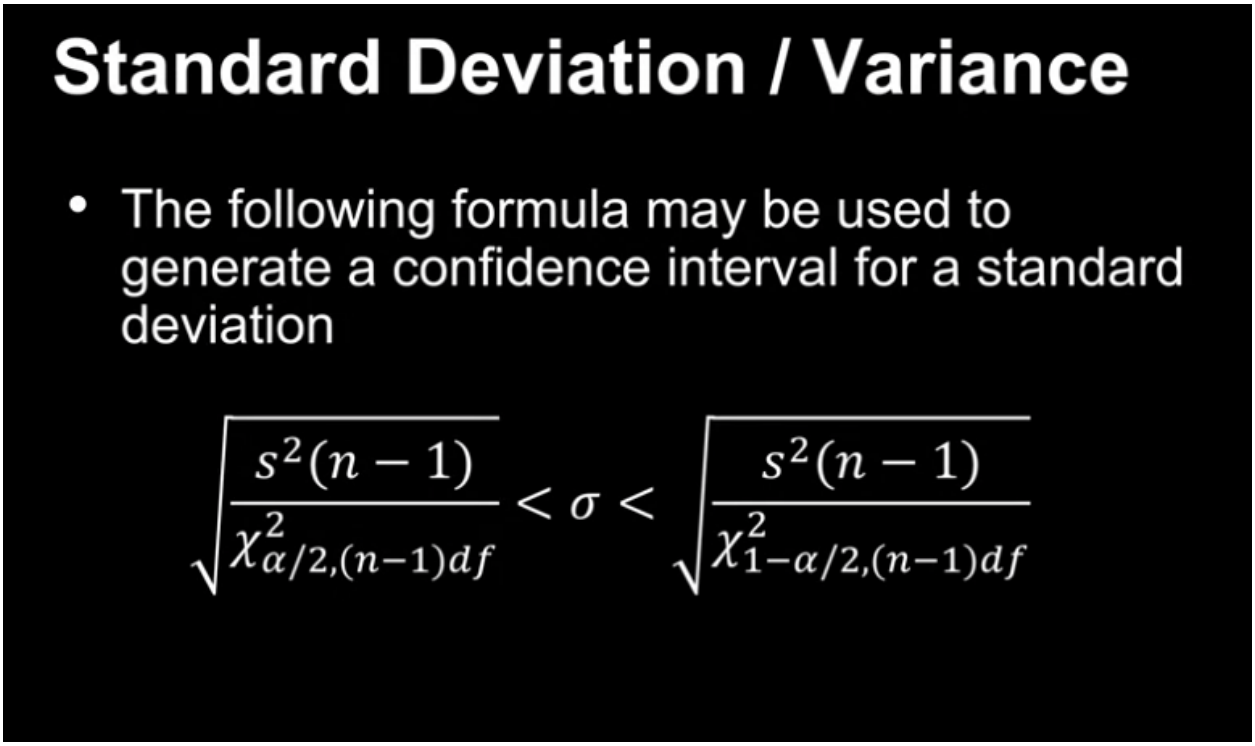


Figure 3: Std Dev Interval

```
# Confidence Interval for the Variance ----------------------------------
# Example 3
sd<-10
n<-25
conf<-0.95

ci.var<-variance.test.onesample.simple(sample.variance = 10^2
                                ,sample.size = 25
                                ,conf.level = 0.95)

# Confidence Interval for the Variance
ci.var$conf.int
```

```
## [1]  60.96929 193.53044
## attr(,"conf.level")
## [1] 0.95
```

```
# Confidence Interval for the Standard Deviation
sqrt(ci.var$conf.int)
```

```
## [1]  7.808284 13.911522
## attr(,"conf.level")
## [1] 0.95
```

```
# Generate Confidence Intervals using a file
# Using the point estimate file, calculate the 95% confidence interval
# estimates for the mean, variance and standard deviation for the Weight data

# Test for normality
summary.continuous(Point_Estimates$Weight)
```

```
##   dv.name  n missing  mean      var adtest.AA  adtest.p  swtest.W  swtest.p
## 1      fx 20       0 842.8 19.01053 0.3633424 0.4407723 0.9539818 0.4315902
##   g3.skewness  g3test.p g4.kurtosis  g4test.p
## 1   -0.273123 0.5732011  -0.5046044 0.6871433
```

```
# Calculate confidence intervals
t.test.onesample(Point_Estimates$Weight, conf.level = 0.95)
```

```
##
##  One-Sample t Test For Means
##
## data:  sample mean, sample size, and estimated variance
## t statistic = 864.46, null hypothesis mean = 0, p-value <
## 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  840.7594 844.8406
## sample estimates:
## sample.mean      se.est          df var.lowerci         var var.upperci
## 842.8000000   0.9749494  19.0000000  10.9946550  19.0105263  40.5545760
##  sd.lowerci          sd  sd.upperci       power
##   3.3158189   4.3601062   6.3682475   1.0000000
```

CI for Porportions and Poisson Counts

ex) - n = 100 - p = 0.12 - Confidence Level Desired = 95%

- mean = np = 12
- n = 100
- a = 0.05
- lower_ci = qbeta(0.025; 12,89) = 0.0636

- upper_ci = qbeta(0.975; 13, 88) = 0.2002

```
# Confidence Interval for a Proportion ----------------------
# Example 1
ro(proportion.test.onesample.exact.simple(sample.proportion = 0.12
                                          ,sample.size = 100
                                          , conf.level = 0.95),4)
```

```
##
##  One-Sample Proportion Test (Exact)
##
## data:  sample proportion and sample size
## p = 0.12, null hypothesis proportion = 0.5, p-value <
## 0.00000000000000022
## alternative hypothesis: true proportion is not equal to 0.5
## 95 percent confidence interval:
##  0.0636 0.2002
## sample estimates:
## sample.prop sample.size   n.times.p       power
##        0.12      100.00       12.00        1.00
```

```
# Using the Point Estimate File, calculate the 90% confidence interval for the
# Proportion

(prop<-mean(Point_Estimates$Proportion)) # average proportion
```

```
## [1] 0.03525
```

```
(n<-length(Point_Estimates$Proportion)) # sample size
```

```
## [1] 20
```

```
# Proportion (Exact)
ro(proportion.test.onesample.exact.simple(sample.proportion = 0.03525
                                          ,sample.size = 20
                                          ,conf.level = 0.90),4)
```

```
##
##  One-Sample Proportion Test (Exact)
##
## data:  sample proportion and sample size
## p = 0.0352, null hypothesis proportion = 0.5, p-value <
## 0.00000000000000022
## alternative hypothesis: true proportion is not equal to 0.5
## 90 percent confidence interval:
##  0.0006 0.1949
## sample estimates:
## sample.prop sample.size   n.times.p       power
##      0.0352     20.0000      0.7050      1.0000
```

Poission

- n = 20
- lambda = 25.05
- desired CI = 95%
- lambda * n = 501
- alpha = 0.05
- lambda_lower = qgamma(0.025; 501)/20 = 22.90
- lambda_upper = qgamma(0.975; 502)/20 = 27.34

```
# Confidence Interval for Poisson Counts ---------------------
# Using the Point Estimate File, calculate the 99% confidence interval for the
# Poisson Counts

# Make sure data are Poisson distributed
poisson.dist.test(Point_Estimates$Count)
```

```
##
##  Poisson Distribution Fit Test Using Variance and Mean
##
## data:  input data
## chi.square = 15.926, degrees of freedom = 19, p-value = 0.6756
## alternative hypothesis: true chi.square is not equal to 19
## sample estimates:
##      chi.square sample variance     sample mean
##        15.92615        20.99737        25.05000
```

```
# Get Total Counts in the Sample
(counts<-sum(Point_Estimates$Count))
```

```
## [1] 501
```

```
# Get Sample Size
(n<-length(Point_Estimates$Count))
```

```
## [1] 20
```

```
poisson.test.onesample.simple(sample.count = 501
                              ,sample.size = 20
                              ,conf.level = 0.90)
```

```
##
##  Exact Poisson test
##
## data:  sample.count time base: sample.size
## number of events = 501, time base = 20, p-value < 0.00000000000000022
## alternative hypothesis: true event rate is not equal to 1
```

```
## 90 percent confidence interval:
##  23.23802 26.97066
## sample estimates:
## event rate
##      25.05
```

Homework Quiz

*All wrong

```
#mean = 5.0061
#
set.seed(145)
d1<-rnorm(n = 10000,mean = 5, sd = 1)
normdata = data.frame(d1)
summary.all.variables(normdata, stat.sd=T)
```

```
##   dv.name     n missing     mean      var        sd g3.skewness  g3test.p
## 1      d1 10000       0 5.006063 0.974553 0.9871945 -0.03564934 0.1454511
##   g4.kurtosis g4test.p
## 1 -0.02753086 0.585028
```

```
#mean = 5.0061
#
set.seed(100)
vec = c()

for (x in 1:5000) {
  d1<-rexp(100, 1/5)
  mean_d1 = mean(d1)
  vec = c(vec, mean_d1)
}

round(skewness(vec), 4)
```

```
## [1] 0.2089
```

```
round(kurtosis(vec), 4)
```

```
## [1] 0.2043
```

```
  d1<-rexp(100, 1/5)
d1
```

```
##   [1]  2.12659542  1.25146881 10.03797713  2.31319254 15.36474003  7.98185118
##   [7]  5.24535073 11.90480753  2.11937809  7.70530766  0.11864879  2.20538920
##  [13]  7.00505605  8.56158744  3.22796692  6.94906911  1.37997024  4.76762712
##  [19]  2.51828152  1.95150748  0.96525154  3.81843423  7.08664824  2.67403585
##  [25]  0.89769581  2.93939372  7.14042965  3.28259843  7.51909691  1.05144754
##  [31] 16.48651023  3.28143381  5.99390319  2.99482517  9.69354290  1.34843090
```

```
##  [37] 17.37304739  7.21387358  3.68841067  2.23418916  2.75167462  8.52297955
##  [43]  7.80559372 13.11724692  8.30617893  1.32551096  0.93976122  7.02368305
##  [49]  0.75428501  5.57957686  5.30281518  2.27909274  2.23863152  7.76094224
##  [55]  0.59173383  9.50680024  6.47733470  0.02994499  0.09334880  0.98772388
##  [61]  2.99598439  0.61305107 15.20726375  6.58451323 14.46956675  0.52659853
##  [67]  2.81430567  2.64581506  1.24143679  2.59791133  0.45053206 27.43199014
##  [73]  0.02530980  1.24531822  2.45116074  4.32763489  1.14943928  5.79234493
##  [79]  0.95928872  6.53249697  4.52545986  4.56790269  2.58491491  2.24420809
##  [85]  9.64051549 12.34522761  7.10978365  1.74139609  0.49666949 14.04733442
##  [91] 14.18558523  9.87637504  8.41470282 15.62081091  1.75494538  4.26743344
##  [97]  3.92810639  0.39744026  7.92511449  0.37138166
```

4) Approximately normal
5) CLT, Sample Size

```
mu = 150
sd = 10
n = 15

stderror = sd / sqrt(n)
stderror
```

```
## [1] 2.581989
```

```
mu = 1.575
sigma = 0.01
n=5

stderror = sigma / sqrt(n)

pnorm(q=1.58, mean=mu, sd=stderror, lower.tail=F)
```

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

```
mu = 1.575
sigma = 0.01
n=10

stderror = sigma / sqrt(n)

pnorm(q=1.572, mean=mu, sd=stderror, lower.tail=T)
```

```
## [1] 0.1713909
```

9) Point and interval estimates are values, estimates are mathematical functions. Interval Estimates will vary based on the desired Confidence Level

10) Mean and std

```
preforms <- read.csv("~/Documents/GitHub/school_cu/school_cu/methods for quality improvement/DTSA5704_D
```

```r
mean(preforms$weight)
```

```
## [1] 42.8
```

```r
#95% CI

t.test.onesample(preforms$weight, conf.level = 0.95)
```

```
##
##  One-Sample t Test For Means
##
## data:  sample mean, sample size, and estimated variance
## t statistic = 43.9, null hypothesis mean = 0, p-value <
## 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  40.75941 44.84059
## sample estimates:
## sample.mean      se.est          df var.lowerci         var var.upperci
##  42.8000000   0.9749494  19.0000000  10.9946550  19.0105263  40.5545760
##  sd.lowerci          sd  sd.upperci       power
##   3.3158189   4.3601062   6.3682475   1.0000000
```

```r
#90% CI

t.test.onesample(preforms$weight, conf.level = 0.90)
```

```
##
##  One-Sample t Test For Means
##
## data:  sample mean, sample size, and estimated variance
## t statistic = 43.9, null hypothesis mean = 0, p-value <
## 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 90 percent confidence interval:
##  41.11418 44.48582
## sample estimates:
## sample.mean      se.est          df var.lowerci         var var.upperci
##  42.8000000   0.9749494  19.0000000  11.9826720  19.0105263  35.7022372
##  sd.lowerci          sd  sd.upperci       power
##   3.4615996   4.3601062   5.9751349   1.0000000
```

17) As the confidence level of an estimator increases, the confidence interval becomes wider

```r
# Poisson counts
n = 500
p = 15/500
conf = 0.95

proportion.test.onesample.exact.simple(sample.proportion=p, sample.size = n, conf.level = 0.95)
```

```
##
##  One-Sample Proportion Test (Exact)
##
## data:  sample proportion and sample size
## p = 0.03, null hypothesis proportion = 0.5, p-value <
## 0.00000000000000022
## alternative hypothesis: true proportion is not equal to 0.5
## 95 percent confidence interval:
##  0.01688598 0.04899823
## sample estimates:
## sample.prop sample.size   n.times.p       power
##        0.03      500.00       15.00        1.00
```

```
n_counts = 250
lambda = 2.58

sample_size = lambda * n

poisson.test.onesample.simple(sample.count = sample_size,sample.size = n_counts,conf.level = 0.90)
```

```
##
##  Exact Poisson test
##
## data:  sample.count time base: sample.size
## number of events = 1290, time base = 250, p-value < 0.00000000000000022
## alternative hypothesis: true event rate is not equal to 1
## 90 percent confidence interval:
##  4.925986 5.402654
## sample estimates:
## event rate
##       5.16
```

```
#lower= 2.4152
#higher = 2.7535
```