Statistical Inference Course Notes

Xing Su

Contents

Overview

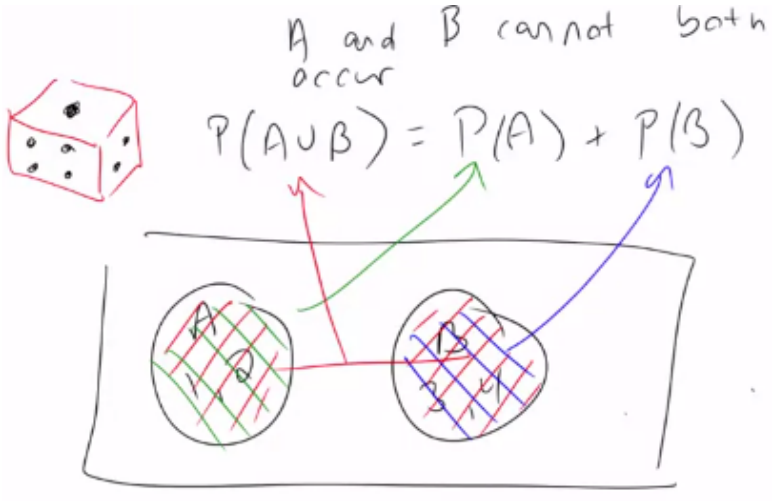
- **Statistical Inference** = generating conclusions about a population from a noisy sample
- Goal = extend beyond data to population
- Statistical Inference = only formal system of inference we have
- many different modes, but **two** broad flavors of inference (inferential paradigms): ***Bayesian*** vs ***Frequencist***
  - **Frequencist** –> uses long run proportion of times an event occurs independent identically distributed repetitions
    - frequentist is what this class is focused on
    - believes if an experiment is repeated many many times, the resultant percentage of success/something happening defines that population parameter
  - **Bayesian** –> probability estimate for a hypothesis is updated as additional evidence is acquired
- **statistic** = number computed from a sample of data
  - statistics are used to infer information about a population
- **random variable** = outcome from an experiment
  - deterministic processes (variance/means) produce additional random variables when applied to random variables, and they have their own distributions

Probability

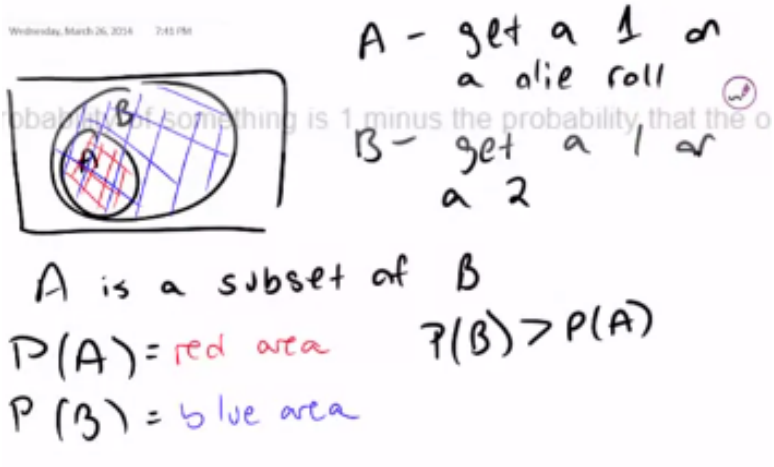
- **Probability** = the study of quantifying the likelihood of particular events occurring
  - given a random experiment, ***probability*** = population quantity that summarizes the randomness
    - not in the data at hand, but a conceptual quantity that exist in the population that we want to estimate

General Probability Rules

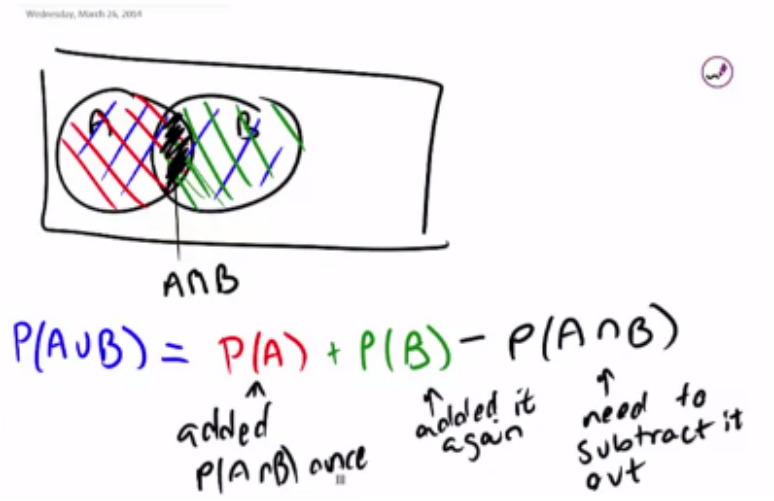
- discovered by Russian mathematician Kolmogorov, also known as "Probability Calculus"
- probability = function of any set of outcomes and assigns it a number between 0 and 1
  - $0 \leq P(E) \leq 1$, where E = event
- probability that nothing occurs = 0 (impossible, have to roll dice to create outcome), that something occurs is 1 (certain)
- probability of outcome or event E, ***P(E)*** = ratio of ways that E could occur to number of all possible outcomes or events
- probability of something = 1 - probability of the opposite occurring
- probability of the **union** of any two sets of outcomes that have nothing in common (mutually exclusive) = sum of respective probabilities



- if A implies occurrence of B, then P(A) occurring < P(B) occurring



- for any two events, probability of at least one occurs = the sum of their probabilities - their intersection (in other words, probabilities can not be added simply if they have non-trivial intersection)



- for independent events A and B, $P(A \cup B) = P(A) \times P(B)$
- for outcomes that can occur with different combination of events and these combinations are mutually exclusive, the $P(E_{total}) = \sum P(E_{part})$

Conditional Probability

- let B = an event so that $P(B) > 0$
- **conditional probability** of an event A, given B is defined as the probability that BOTH A and B occurring divided by the probability of B occurring

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

- if A and B are ***independent***, then

$$P(A \mid B) = \frac{P(A)P(B)}{P(A)} = P(A)$$

- ***example***
  - for die roll, $A = \{1\}$, $B = \{1, 3, 5\}$, then

$$P(1 \mid Odd) = P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)}{P(B)} = \frac{1/6}{3/6} = \frac{1}{3}$$

Baye's Rule

- definition

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A \mid B)P(B) + P(A \mid B^c)P(B^c)}$$

  where B^c = corresponding probability of event B, $P(B^c) = 1 - P(B)$

Random Variables

- **random variable** = numeric outcome of experiment
- **discrete** (what you can count/categories) = assign probabilities to every number/value the variable can take
  - coin flip, rolling a die, web traffic in a day
- **continuous** (any number within a continuum) = assign probabilities to the range the variable can take
  - BMI index, intelligence quotients
  - ***Note****: limitations of precision in taking the measurements may imply that the values are discrete, but we in fact consider them continuous*
- `rbinom()`, `rnorm()`, `rgamma()`, `rpois()`, `runif()` = functions to generate random variables from the binomial, normal, Gamma, Poisson, and uniform distributions
- density and mass functions (population quantities, not what occurs in data) for random variables = best starting point to model/think about probabilities for numeric outcome of experiments (variables)
  - use data to estimate properties of population –> linking sample to population

Probability Mass Function (PMF)

- evaluates the probability that the **discrete random variable** takes on a specific value
  - measures the chance of a particular outcome happening
  - always ≥ 0 for every possible outcome
  - $\sum$ possible values that the variable can take = 1
- ***Bernoulli distribution example***
  - X = 0 –> tails, X = 1 –> heads
    - X here represents potential outcome
  - $p(X = x) = (\frac{1}{2})^x(\frac{1}{2})^{1-x}$ for $X = 0, 1$
    - x here represents a value we can plug into the PMF
    - general form –> $p(x) = (\theta)^x(1-\theta)^{1-x}$
- `dbinom(k, n, p)` = return the probability of getting `k` successes out of `n` trials, given probability of success is `p`

Probability Density Function (PDF)

- evaluates the probability that the **continuous random variable** takes on a specific value
  - always $\geq$ everywhere
  - total area under the must = 1
- **areas under PDFs** correspond to the probabilities for that random variable taking on that range of values (PMF)

- but the probability of the variable taking a specific value = 0 (area of a line is 0)

- ***Note**: the above is true because it is modeling random variables as if they have infinite precision, when in reality they do not*
- `dnorm()`, `dgamma()`, `dpois()`, `dunif()` = return probability of a certain value from the normal, Gamma, Poisson, and uniform distributions

Cumulative Distribution Function (CDF)

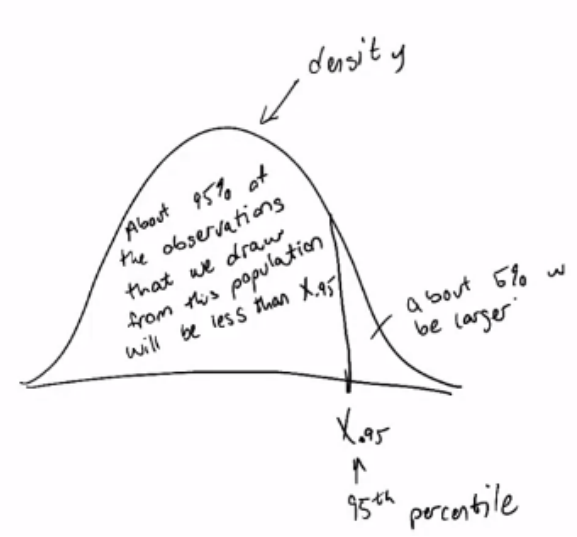
- CDF of a random variable X = probability that the random variable is $\leq$ value x
  - $F(x) = P(X \leq x)$ <- applies when X is discrete/continuous
- PDF = derivative of CDF
  - integrate PDF –> CDF
    - `integrate(function, lower=0, upper=1)` –> can be used to evaluate integrals for a specified range
- `pbinom()`, `pnorm()`, `pgamma()`, `ppois()`, `punif()` = returns the cumulative probabilities from 0 up to a specified value from the binomial, normal, Gamma, Poisson, and uniform distributions

Survival Function

- survival function of a random variable X = probability the random variable > x, complement of CDF
  - $S(x) = P(X > x) = 1 - F(x)$, where $F(x)$ = CDF

Quantile

- the α^{th} quantile of a distribution with distribution function F = point x_α
  - $F(x_\alpha) = \alpha$
  - percentile = quantile with α expressed as a percent
  - median = 50th percentile
  - α% of the possible outcomes lie below it



- `qbeta(quantileInDecimals, 2, 1)` = returns quantiles for beta distribution
  - works for `qnorm()`, `qbinom()`, `qgamma()`, `qpois()`, etc.
- median estimated in this fashion = a population median
- probability model connects data to population using assumptions
  - population median = ***estimand***, sample median = ***estimator***

Independence

- two events A and B are ***independent*** if the following is true
  - $P(A \cap B) = P(A)P(B)$
  - $P(A \mid B) = P(A)$
- two random variables X and Y are ***independent***, if for any two sets, **A** and **B**, the following is true
  - $P([X \in A] \cap [Y \in B]) = P(X \in A)P(Y \in B)$
- **independence** = statistically unrelated from one another
- if A is ***independent*** of B, then the following are true
  - A^c is independent of B
  - A is independent of B^c
  - A^c is independent of B^c

IID Random Variables

- random variables are said to be **IID** if they are ***independent and identically distributed***
  - **independent** = statistically unrelated from each other
  - **identically distributed** = all having been drawn from the same population distribution
- IID random variables = default model for random samples = default starting point of inference

Diagnostic Test

- Let + and − be the results, positive and negative respectively, of a diagnostic test
- Let D = subject of the test has the disease, D^c = subject does not
- **sensitivity** = $P(+ \mid D)$ = probability that the test is positive given that the subject has the disease (the higher the better)
- **specificity** = $P(- \mid D^c)$ = probability that the test is negative given that the subject does not have the disease (the higher the better)
- **positive predictive value** = $P(D \mid +)$ = probability that that subject has the disease given that the test is positive
- **negative predictive value** = $P(D^c \mid -)$ = probability that the subject does not have the disease given the test is negative
- **prevalence of disease** = $P(D)$ = marginal probability of disease

Example

- specificity of 98.5%, sensitivity = 99.7%, prevalence of disease = .1%

$$\begin{aligned} P(D \mid +) &= \frac{P(+ \mid D)P(D)}{P(+ \mid D)P(D) + P(+ \mid D^c)P(D^c)} \\ &= \frac{P(+ \mid D)P(D)}{P(+ \mid D)P(D) + \{1 - P(- \mid D^c)\}\{1 - P(D)\}} \\ &= \frac{.997 \times .001}{.997 \times .001 + .015 \times .999} \\ &= .062 \end{aligned}$$

- low positive predictive value –> due to low prevalence of disease and somewhat modest specificity
  - suppose it was know that the subject uses drugs and has regular intercourse with an HIV infect partner (his probability of being + is higher than suspected)
  - evidence implied by a positive test result

Likelihood Ratios

- **diagnostic likelihood ratio** of a **positive** test result is defined as

$$DLR_+ = \frac{sensitivity}{1 - specificity} = \frac{P(+ \mid D)}{P(+ \mid D^c)}$$

- **diagnostic likelihood ratio** of a **negative** test result is defined as

$$DLR_- = \frac{1 - sensitivity}{specificity} = \frac{P(- \mid D)}{P(- \mid D^c)}$$

- from Baye's Rules, we can derive the *positive predictive value* and *false positive value*

$$P(D \mid +) = \frac{P(+ \mid D)P(D)}{P(+ \mid D)P(D) + P(+ \mid D^c)P(D^c)} \qquad (1)$$

$$P(D^c \mid +) = \frac{P(+ \mid D^c)P(D^c)}{P(+ \mid D)P(D) + P(+ \mid D^c)P(D^c)} \qquad (2)$$

- if we divide equation (1) over (2), the quantities over have the same denominator so we get the following

$$\frac{P(D \mid +)}{P(D^c \mid +)} = \frac{P(+ \mid D)}{P(+ \mid D^c)} \times \frac{P(D)}{P(D^c)}$$

which can also be written as

$$\text{post-test odds of D} = DLR_+ \times \text{pre-test odds of D}$$

  - **odds** $= p/(1-p)$
  - $\frac{P(D)}{P(D^c)}$ = **pre-test odds**, or odds of disease in absence of test
  - $\frac{P(D \mid +)}{P(+ \mid D^c)}$ = **post-test odds**, or odds of disease given a positive test result
  - DLR_+ = factor by which the odds in the presence of a positive test can be multiplied to obtain the post-test odds
  - DLR_- = relates the decrease in odds of disease after a negative result

- following the previous example, for sensitivity of 0.997 and specificity of 0.985, so the diagnostic likelihood ratios are as follows

$$DLR_+ = .997/(1-.985) = 66 \qquad DLR_- = (1-.997)/.985 = 0.003$$

  - this indicates that the result of the positive test is the odds of disease is 66 times the pretest odds

Expected Values/Mean

- useful for characterizing a distribution (properties of distributions)
- **mean** = characterization of the center of the distribution = *expected value*
- expected value operation = ***linear*** –> $E(aX + bY) = aE(X) + bE(Y)$
- **variance/standard deviation** = characterization of how spread out the distribution is
- *sample* expected values for sample mean and variance will estimate the *population* counterparts
- **population mean**
  - expected value/mean of a random variable = center of its distribution (center of mass)
  - ***discrete variables***
    - for X with PMF $p(x)$, the population mean is defined as

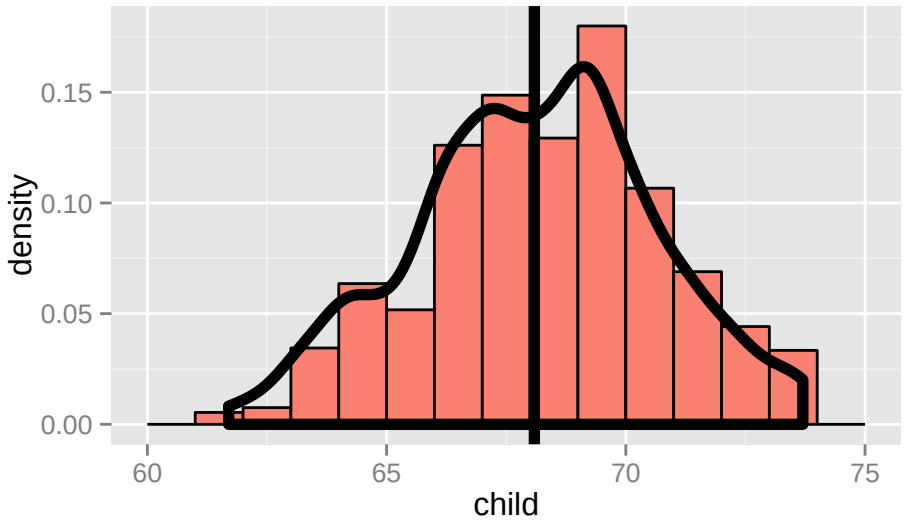
$$E[X] = \sum_x xp(x)$$

      where the sum is taken over ***all*** possible values of x
    - $E[X]$ = center of mass of a collection of location and weights $x, p(x)$
    - *coin flip example*: $E[X] = 0 \times (1 - p) + 1 \times p = p$
  - ***continuous variable***
    - for X with PDF $f(x)$, the expected value = the center of mass of the density
    - instead of summing over discrete values, the expectation ***integrates*** over a continuous function
      - PDF = $f(x)$
      - $\int xf(x)$ = area under the PDF curve = mean/expected value of X
- **sample mean**
  - sample mean estimates the population mean
    - sample mean = center of mass of observed data = empirical mean

$$\bar{X} = \sum_x^n x_i p(x_i)$$

      where $p(x_i) = 1/n$

```
# load relevant packages
library(UsingR); data(galton); library(ggplot2)
# plot galton data
g <- ggplot(galton, aes(x = child))
# add histogram for children data
g <- g + geom_histogram(fill = "salmon", binwidth=1, aes(y=..density..), colour="black")
# add density smooth
g <- g + geom_density(size = 2)
# add vertical line
g <- g + geom_vline(xintercept = mean(galton$child), size = 2)
# print graph
g
```

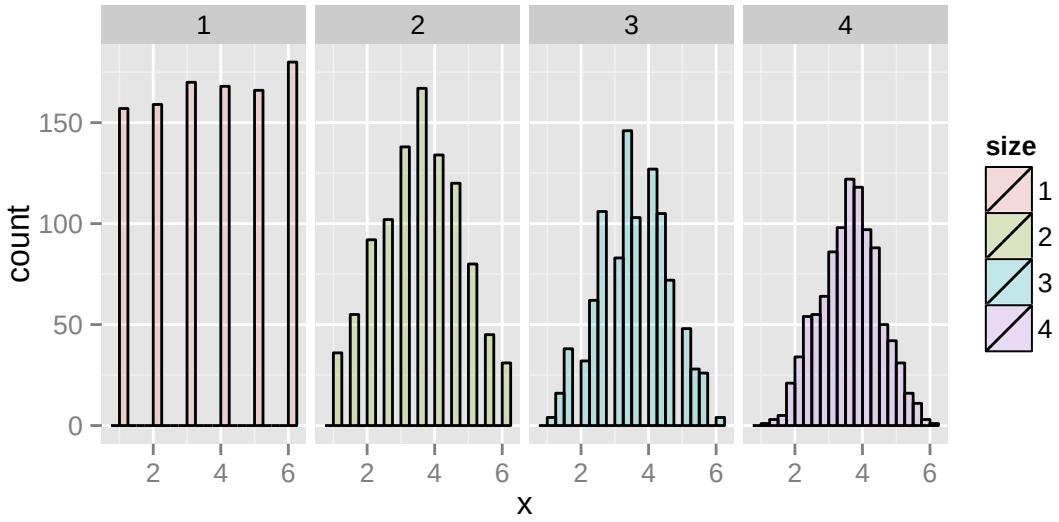


- **average of random variables** = a new random variable where its distribution has an expected value that is the **same** as the original distribution (centers are the same)
  - the mean of the averages = average of the original data –> estimates average of the population
  - if $E[\textit{sample mean}]$ = population mean, then estimator for the sample mean is **unbiased**
    - [**derivation**] let X_1, X_2, X_3, ... X_n be a collection of n samples from the population with mean μ
    - mean of this sample
    $$\bar{X} = \frac{X_1 + X_2 + X_3 + . + X_n}{n}$$
    - since $E(aX) = aE(X)$, the expected value of the mean is can be written as
    $$E\left[\frac{X_1 + X_2 + X_3 + ... + X_n}{n}\right] = \frac{1}{n} \times [E(X_1) + E(X_2) + E(X_3) + ... + E(X_n)]$$
    - since each of the $E(X_i)$ is drawn from the population with mean μ, the expected value of each sample should be
    $$E(X_i) = \mu$$
    - therefore
    $$\begin{aligned} E\left[\frac{X_1 + X_2 + X_3 + ... + X_n}{n}\right] &= \frac{1}{n} \times [E(X_1) + E(X_2) + E(X_3) + ... + E(X_n)] \\ &= \frac{1}{n} \times [\mu + \mu + \mu + ... + \mu] \\ &= \frac{1}{n} \times n \times \mu \\ &= \mu \end{aligned}$$

- ***Note**: the more data that goes into the sample mean, the more concentrated its density/mass functions are around the population mean*

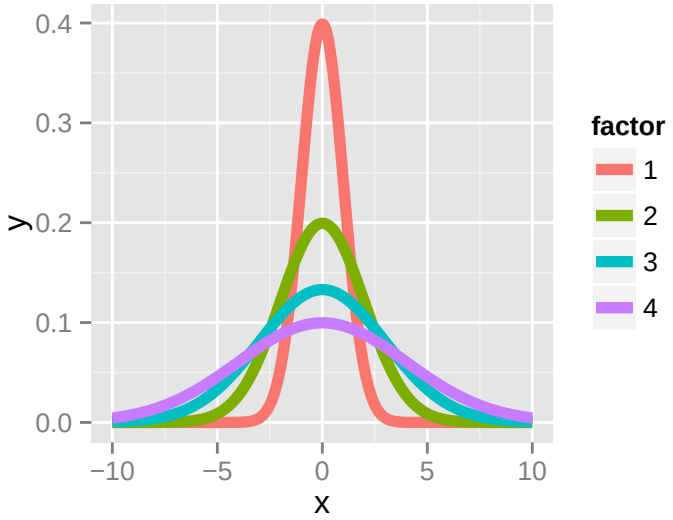
```
nosim <- 1000
# simulate data for sample size 1 to 4
dat <- data.frame(
    x = c(sample(1 : 6, nosim, replace = TRUE),
        apply(matrix(sample(1 : 6, nosim * 2, replace = TRUE), nosim), 1, mean),
```

```
        apply(matrix(sample(1 : 6, nosim * 3, replace = TRUE), nosim), 1, mean),
        apply(matrix(sample(1 : 6, nosim * 4, replace = TRUE), nosim), 1, mean)),
    size = factor(rep(1 : 4, rep(nosim, 4))))
# plot histograms of means by sample size
g <- ggplot(dat, aes(x = x, fill = size)) + geom_histogram(alpha = .20, binwidth=.25, colour = "black")
g + facet_grid(. ~ size)
```



Variance

```
# generate x value ranges
xvals <- seq(-10, 10, by = .01)
# generate data from normal distribution for sd of 1 to 4
dat <- data.frame(
    y = c(dnorm(xvals, mean = 0, sd = 1),
        dnorm(xvals, mean = 0, sd = 2),
        dnorm(xvals, mean = 0, sd = 3),
        dnorm(xvals, mean = 0, sd = 4)),
    x = rep(xvals, 4),
    factor = factor(rep(1 : 4, rep(length(xvals), 4)))
)
# plot 4 lines for the different standard deviations
ggplot(dat, aes(x = x, y = y, color = factor)) + geom_line(size = 2)
```

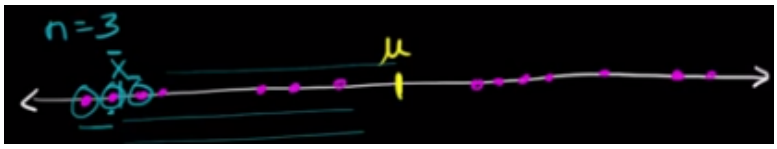


- **variance** = measure of spread, the square of expected distance from the mean (expressed in X's units^2)
  - as we can see from above, higher variances –> more spread, lower –> smaller spread
  - $Var(X) = E[(X - \mu)^2] = E[X^2] - E[X]^2$
  - **standard deviation** = $\sqrt{Var(X)}$ –> has same units as X
  - ***example***
    - for die roll, $E[X] = 3.5$
    - $E[X^2] = 1^2 \times 1/6 + 2^2 \times 1/6 + 3^2 \times 1/6 + . + 6^2 \times 1/6 = 15.17$
    - $Var(X) = E[X^2] - E[X]^2 \approx 2.92$
  - ***example***
    - for coin flip, $E[X] = p$
    - $E[X^2] = 0^2 \times (1 - p) + 1^2 \times p = p$
    - $Var(X) = E[X^2] - E[X]^2 = p - p^2 = p(1 - p)$

Sample Variance

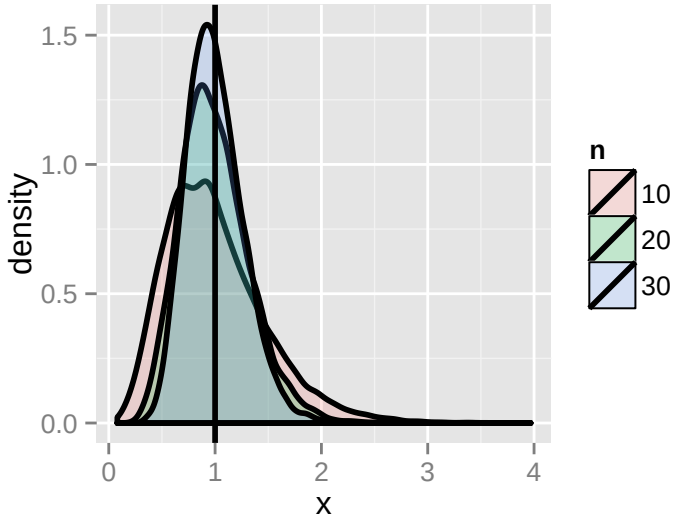
- the **sample variance** is defined as

$$S^2 = \frac{\sum_{i=1}(X_i - \bar{X})^2}{n - 1}$$

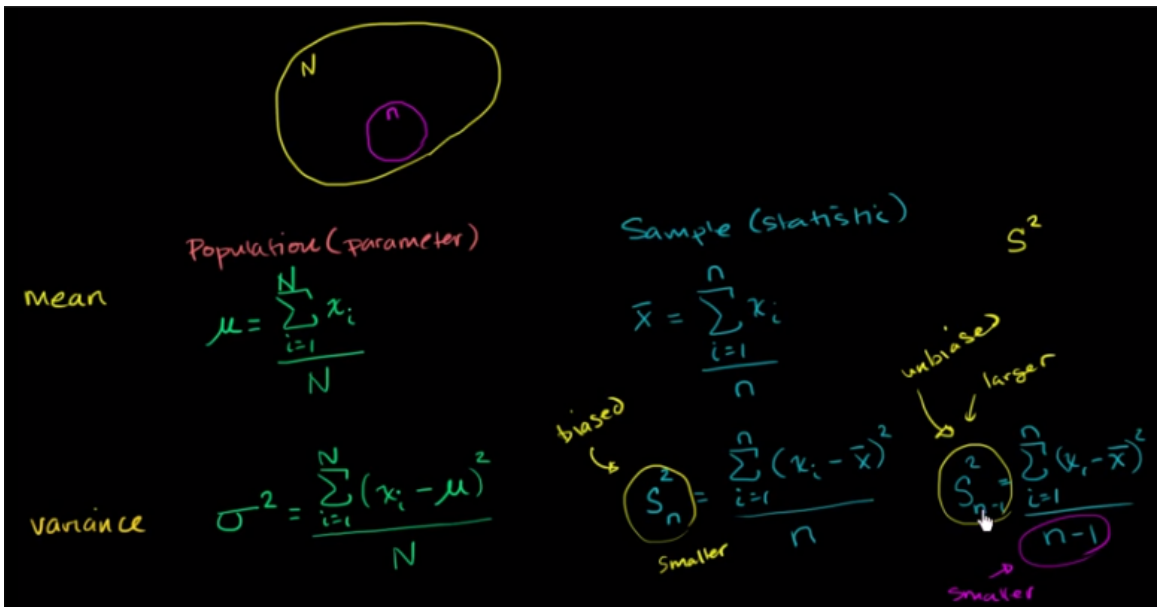


- on the above line representing the population (in magenta), any subset of data (3 of 14 selected, marked in blue) will most likely have a variance that is ***lower than*** the population variance
- dividing by $n-1$ will make the variance estimator ***larger*** to adjust for this fact –> leads to more accurate estimation –> S^2 = so called ***unbiased estimate of population variance***
  - S^2 is a random variable, and therefore has an associated population distribution
    - $E[S^2]$ = population variance, where S = sample standard deviation
    - as we see from the simulation results below, with more data, the distribution for S^2 gets more concentrated around population variance

```
# specify number of simulations
nosim <- 10000;
# simulate data for various sample sizes
dat <- data.frame(
    x = c(apply(matrix(rnorm(nosim * 10), nosim), 1, var),
          apply(matrix(rnorm(nosim * 20), nosim), 1, var),
          apply(matrix(rnorm(nosim * 30), nosim), 1, var)),
    n = factor(rep(c("10", "20", "30"), c(nosim, nosim, nosim))) )
# plot density function for different sample size data
ggplot(dat, aes(x = x, fill = n)) + geom_density(size = 1, alpha = .2) +
    geom_vline(xintercept = 1, size = 1)
```



- ***Note****: for any variable, properties of the population =* ***parameter****, estimates of properties for samples =* ***statistic***
  - below is a summary for the mean and variance for population and sample



- **distribution for mean of random samples**
  - expected value of the **mean** of distribution of means = expected value of the sample = population mean
    - $E[\bar{X}] = \mu$
  - expected value of the variance of distribution of means
    - $Var(\bar{X}) = \sigma^2/n$
    - as **n** becomes larger, the mean of random sample –> more concentrated around the population mean –> variance approaches 0
      - this again confirms that sample mean estimates population mean
  - ***Note**: normally we only have 1 sample mean (from collected sample) and can estimate the variance σ^2 –> so we know a lot about the **distribution of the means** from the data observed*
- **standard error (SE)**
  - the standard error of the mean is defined as

$$SE_{mean} = \sigma/\sqrt{n}$$

  - this quantity is effectively the standard deviation of the distribution of a statistic (i.e. mean)
  - represents variability of means

Entire Estimator-Estimation Relationship

- Start with a sample
- S^2 = sample variance
  - estimates how variable the population is
  - estimates population variance σ^2
  - S^2 = a random variable and has its own distribution centered around σ^2
    - more concentrated around σ^2 as n increases
- $\bar{X}$ = sample mean
  - estimates population mean μ

- $\bar{X}$ = a random variable and has its own distribution centered around μ
  - more concentrated around μ as n increases
  - variance of distribution of $\bar{X} = \sigma^2/n$
  - estimate of variance = S^2/n
  - estimate of standard error = $S/\sqrt{n}$ –> "sample standard error of the mean"
    - estimates how variable sample means (n size) from the population are

Example - Standard Normal

- variance = 1
- means of **n** standard normals (sample) have standard deviation = $1/\sqrt{n}$

```
# specify number of simulations with 10 as number of observations per sample
nosim <- 1000; n <-10
# estimated standard deviation of mean
sd(apply(matrix(rnorm(nosim * n), nosim), 1, mean))
```

```
## [1] 0.3153689
```

```
# actual standard deviation of mean of standard normals
1 / sqrt(n)
```

```
## [1] 0.3162278
```

- `rnorm()` –> generate samples from the standard normal
- `matrix()` –> puts all samples into a nosim by n matrix, so that each row represents a simulation with `nosim` observations
- `apply()` –> calculates the mean of the n samples
- `sd()` –> returns standard deviation

Example - Standard Uniform

- standard uniform –> triangle straight line distribution –> mean = 1/2 and variance = 1/12
- means of random samples of n uniforms have have standard deviation of $1/\sqrt{12 \times n}$

```
# estimated standard deviation of the sample means
sd(apply(matrix(runif(nosim * n), nosim), 1, mean))
```

```
## [1] 0.09147552
```

```
# actual standard deviation of the means
1/sqrt(12*n)
```

```
## [1] 0.09128709
```

Example - Poisson

- $Poisson(x^2)$ have variance of x^2
- means of random samples of n $Poisson(4)$ have standard deviation of $2/\sqrt{n}$

```r
# estimated standard deviation of the sample means
sd(apply(matrix(rpois(nosim * n, lambda=4), nosim), 1, mean))
```

```
## [1] 0.6449735
```

```r
# actual standard deviation of the means
2/sqrt(n)
```

```
## [1] 0.6324555
```

Example - Bernoulli

- for $p = 0.5$, the Bernoulli distribution has variance of 0.25
- means of random samples of n coin flips have standard deviations of $1/(2\sqrt{n})$

```r
# estimated standard deviation of the sample means
sd(apply(matrix(sample(0 : 1, nosim * n, replace = TRUE), nosim), 1, mean))
```

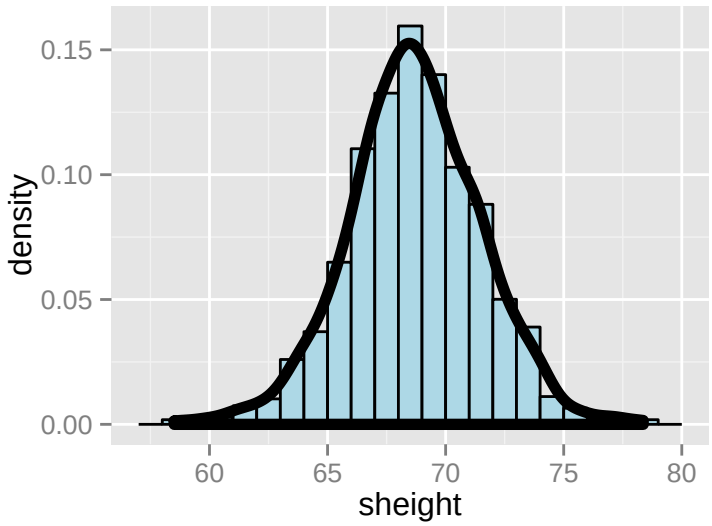
```
## [1] 0.1590323
```

```r
# actual standard deviation of the means
1/(2*sqrt(n))
```

```
## [1] 0.1581139
```

Example - Father/Son

```r
# load data
library(UsingR); data(father.son);
# define son height as the x variable
x <- father.son$sheight
# n is the length
n<-length(x)
# plot histogram for son's heights
g <- ggplot(data = father.son, aes(x = sheight))
g <- g + geom_histogram(aes(y = ..density..), fill = "lightblue", binwidth=1, colour = "black")
g <- g + geom_density(size = 2, colour = "black")
g
```




```
# we calculate the parameters for variance of distribution and sample mean,
round(c(sampleVar = var(x),
    sampleMeanVar = var(x) / n,
    # as well as standard deviation of distribution and sample mean
    sampleSd = sd(x),
    sampleMeanSd = sd(x) / sqrt(n)),2)
```

```
##     sampleVar sampleMeanVar      sampleSd  sampleMeanSd
##          7.92          0.01          2.81          0.09
```

Binomial Distribution

- **binomial random variable** = sum of **n** Bernoulli variables

$$X = \sum_{i=1}^{n} X_i$$

  where $X_1, \ldots, X_n = Bernoulli(p)$
  - PMF is defined as

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

    where $\binom{n}{x}$ = number of ways selecting x items out of n options without replacement or regard to order and for $x = 0, \ldots, n$
  - **combination** or "n choose x" is defined as

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

  - the base cases are

$$\binom{n}{n} = \binom{n}{0} = 1$$

- **Bernoulli distribution** –> binary outcome
  - only possible outcomes
    - 1 = "success" with probability of p
    - 0 = "failure" with probability of $1 - p$
  - PMF is defined as

$$P(X = x) = p^x (1-p)^{1-x}$$

  - mean = p
  - variance = $p(1-p)$

Example

- of 8 children, whats the probability of 7 or more girls (50/50 chance)?

$$\binom{8}{7} .5^7 (1-.5)^1 + \binom{8}{8} .5^8 (1-.5)^0 \approx 0.04$$

```
# calculate probability using PMF
choose(8, 7) * .5 ^ 8 + choose(8, 8) * .5 ^ 8
```

```
## [1] 0.03515625
```

```
# calculate probability using CMF from distribution
pbinom(6, size = 8, prob = .5, lower.tail = FALSE)
```

```
## [1] 0.03515625
```

- `choose(8, 7)` = R function to calculate n choose x
- `pbinom(6, size=8, prob =0.5, lower.tail=TRUE)` = probability of 6 or less successes out of 8 samples with probability of 0.5 (CMF)
  - `lower.tail=FALSE` = returns the complement, in this case it's the probability of greater than 6 successes out of 8 samples with probability of 0.5

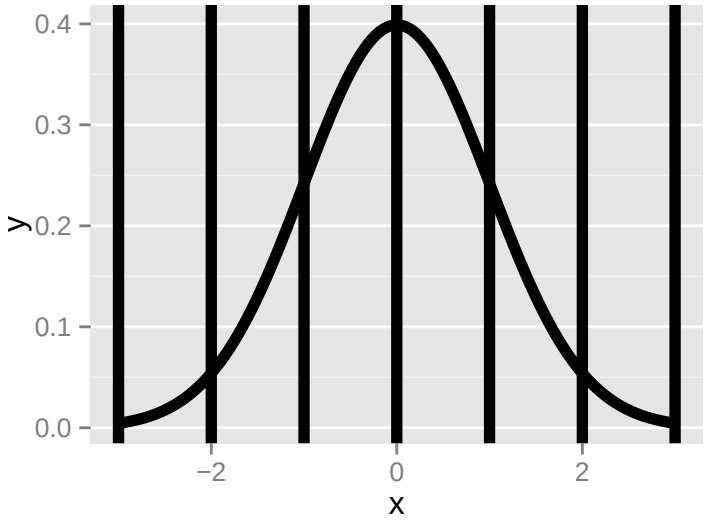
Normal Distribution

- normal/Gaussian distribution for random variable X
  - notation = $X \sim N(\mu, \sigma^2)$
  - mean = $E[X] = \mu$
  - variance = $Var(X) = \sigma^2$
  - PMF is defined as

$$f(x) = (2\pi\sigma^2)^{-1/2} e^{-(x-\mu)^2/2\sigma^2}$$

- $X \sim N(0,1)$ = **standard normal distribution** (standard normal random variables often denoted using $Z_1, Z_2, \ldots$)
  - ***Note**: see below graph for reference for the following observations*
  - ~68% of data/normal density –> between $\pm$ 1 standard deviation from μ
  - ~95% of data/normal density –> between $\pm$ 2 standard deviation from μ
  - ~99% of data/normal density –> between $\pm$ 3 standard deviation from μ
  - $\pm$ 1.28 standard deviations from μ –> 10^{th} (-) and 90^{th} (+) percentiles
  - $\pm$ 1.645 standard deviations from μ –> 5^{th} (-) and 95^{th} (+) percentiles
  - $\pm$ 1.96 standard deviations from μ –> 2.5^{th} (-) and 97.5^{th} (+) percentiles
  - $\pm$ 2.33 standard deviations from μ –> 1^{st} (-) and 99^{th} (+) percentiles

```
# plot standard normal
x <- seq(-3, 3, length = 1000)
g <- ggplot(data.frame(x = x, y = dnorm(x)),
            aes(x = x, y = y)) + geom_line(size = 2)
g <- g + geom_vline(xintercept = -3 : 3, size = 2)
g
```



- for any $X \sim N(\mu, \sigma^2)$, calculating the number of standard deviations each observation is from the mean ***converts*** the random variable to a ***standard normal*** (denoted as Z below)

$$Z = \frac{X - \mu}{\sigma} \sim N(0,1)$$

- conversely, a standard normal can then be converted to ***any normal distribution*** by multiplying by standard deviation and adding the mean

$$X = \mu + \sigma Z \sim N(\mu, \sigma^2)$$

- `qnorm(n, mean=mu, sd=sd)` = returns the n^{th} percentiles for the given normal distribution
- `pnorm(x, mean=mu, sd=sd, lower.tail=F)` = returns the probability of an observation drawn from the given distribution is larger in value than the specified threshold x

Example

- the number of daily ad clicks for a company is (approximately) normally distributed with a mean of 1020 and a standard deviation of 50
- What's the probability of getting more than 1,160 clicks in a day?

```
# calculate number of standard deviations from the mean
(1160 - 1020) / 50
```

```
## [1] 2.8
```

```
# calculate probability using given distribution
pnorm(1160, mean = 1020, sd = 50, lower.tail = FALSE)
```

```
## [1] 0.00255513
```

```
# calculate probability using standard normal
pnorm(2.8, lower.tail = FALSE)
```

```
## [1] 0.00255513
```

- therefore, it is not very likely (0.255513% chance), since 1,160 is 2.8 standard deviations from the mean

- What number of daily ad clicks would represent the one where 75% of days have fewer clicks (assuming days are independent and identically distributed)?

```
qnorm(0.75, mean = 1020, sd = 50)
```

```
## [1] 1053.724
```

- therefore, 1053.7244875 would represent the threshold that has more clicks than 75% of days

Poisson Distribution

- used to model counts
  - mean = λ
  - variance = λ
  - PMF is defined as

$$P(X = x; \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

    where $X = 0, 1, 2, ...\infty$
- modeling uses for Poisson distribution
  - count data
  - event-time/survival –> cancer trials, some patients never develop and some do, dealing with the data for both ("censoring")
  - contingency tables –> record results for different characteristic measurements
  - approximating binomials –> instances where **n** is large and **p** is small (i.e. pollution on lung disease)
    - $X \sim Binomial(n, p)$
    - $\lambda = np$
  - rates –> $X \sim Poisson(\lambda t)$
    - $\lambda = E[X/t]$ –> expected count per unit of time
    - t = total monitoring time
- `ppois(n, lambda = lambda*t)` = returns probability of n or fewer events happening given the rate λ and time t

Example

- number of people that show up at a bus stop can be modeled with Poisson distribution with a mean of 2.5 per hour
- after watching the bus stop for 4 hours, what is the probability that 3 or fewer people show up for the whole time?

```
# calculate using distribution
ppois(3, lambda = 2.5 * 4)
```

```
## [1] 0.01033605
```

- as we can see from above, there is a 1.0336051% chance for 3 or fewer people show up total at the bus stop during 4 hours of monitoring

Example - Approximating Binomial Distribution

- flip a coin with success probability of 0.01 a total 500 times (low p, large n)
- what's the probability of 2 or fewer successes?

```
# calculate correct probability from Binomial distribution
pbinom(2, size = 500, prob = .01)
```

```
## [1] 0.1233858
```

```r
# estimate probability using Poisson distribution
ppois(2, lambda=500 * .01)
```

```
## [1] 0.124652
```

- as we can see from above, the two probabilities (12.3385774% vs 12.3385774%) are extremely close

Asymptotics

- **asymptotics** = behavior of statistics as sample size –> ∞
- useful for simple statistical inference/approximations
- form basis for frequentist interpretation of probabilities ("Law of Large Numbers")

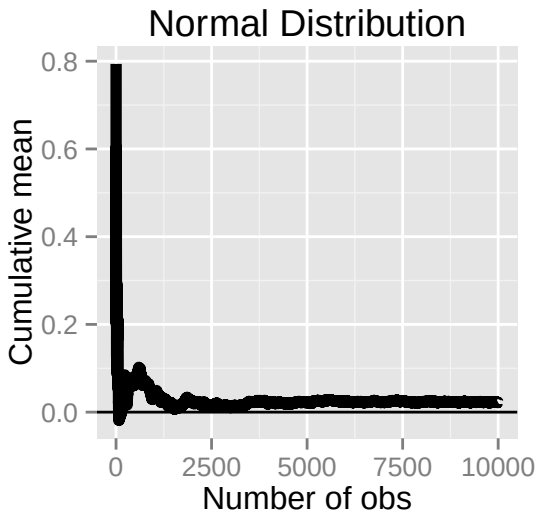
Law of Large Numbers (LLN)

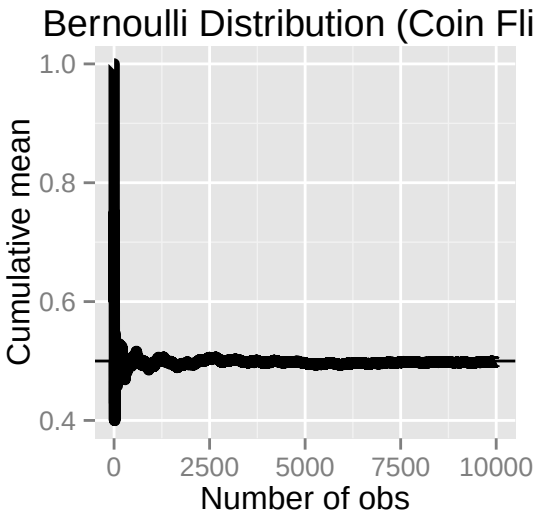
- IID sample statistic that estimates property of the sample (i.e. mean, variance) ***becomes*** the population statistic (i.e. population mean, population variance) as n increases
- ***Note**: an estimator is **consistent** if it converges to what it is estimating*
- sample mean/variance/standard deviation are all ***consistent estimators*** for their population counterparts
  - $\bar{X}_n$ is average of the result of n coin flips (i.e. the sample proportion of heads)
  - as we flip a fair coin over and over, it ***eventually converges*** to the true probability of a head

Example - LLN for Normal and Bernoulli Distribution

- for this example, we will simulate 10000 samples from the normal and Bernoulli distributions respectively
- we will plot the distribution of sample means as n increases and compare it to the population means

```r
# load library
library(gridExtra)
# specify number of trials
n <- 10000
# calculate sample (from normal distribution) means for different size of n
means <- cumsum(rnorm(n)) / (1  : n)
# plot sample size vs sample mean
g <- ggplot(data.frame(x = 1 : n, y = means), aes(x = x, y = y))
g <- g + geom_hline(yintercept = 0) + geom_line(size = 2)
g <- g + labs(x = "Number of obs", y = "Cumulative mean")
g <- g + ggtitle("Normal Distribution")
# calculate sample (coin flips) means for different size of n
means <- cumsum(sample(0 : 1, n , replace = TRUE)) / (1  : n)
# plot sample size vs sample mean
p <- ggplot(data.frame(x = 1 : n, y = means), aes(x = x, y = y))
p <- p + geom_hline(yintercept = 0.5) + geom_line(size = 2)
p <- p + labs(x = "Number of obs", y = "Cumulative mean")
p <- p + ggtitle("Bernoulli Distribution (Coin Flip)")
# combine plots
grid.arrange(g, p, ncol = 2)
```





- as we can see from above, for both distributions the sample means undeniably approach the respective population means as n increases

Central Limit Theorem

- one of the most important theorems in statistics
- distribution of means of IID variables approaches the standard normal as sample size n increases
- in other words, for large values of n,

$$\frac{\text{Estimate} - \text{Mean of Estimate}}{\text{Std. Err. of Estimate}} = \frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} = \frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma} \longrightarrow N(0,1)$$

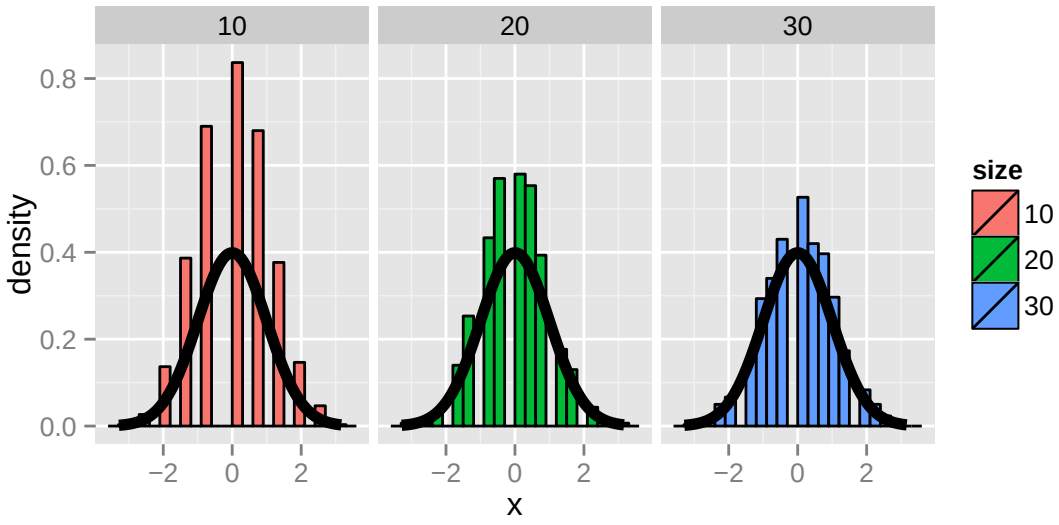
- this translates to the distribution of the sample mean $\bar{X}_n$ is approximately $N(\mu, \sigma^2/n)$
  - distribution is centered at the population mean
  - with standard deviation = standard error of the mean
- typically the Central Limit Theorem can be applied when $n \geq 30$

Example - CLT with Bernoulli Trials (Coin Flips)

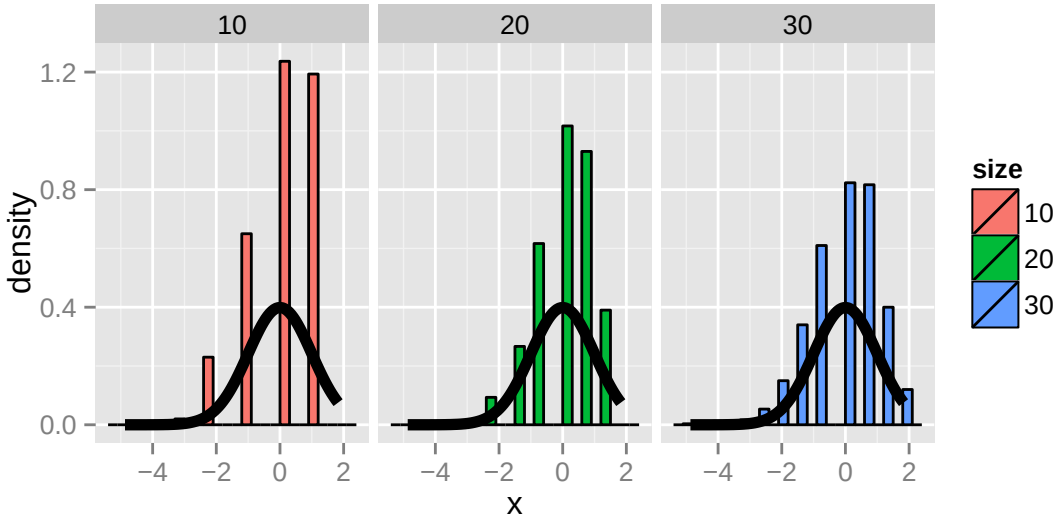
- for this example, we will simulate n flips of a possibly unfair coin
  - X_i be the 0 or 1 result of the i^{th} flip of a possibly unfair coin
  - sample proportion , $\hat{p}$, is the average of the coin flips
  - $E[X_i] = p$ and $Var(X_i) = p(1-p)$
  - standard error of the mean is $SE = \sqrt{p(1-p)/n}$
- in principle, normalizing the random variable X_i, we should get an approximately standard normal distribution

$$\frac{\hat{p} - p}{\sqrt{p(1-p)/n}} \sim N(0,\ 1)$$

- therefore, we will flip a coin n times, take the sample proportion of heads (successes with probability p), subtract off 0.5 (ideal sample proportion) and multiply the result by divide by $\frac{1}{2\sqrt{n}}$ and compare it to the standard normal



- now, we can run the same simulation trials for an extremely unfair coin with $p = 0.9$



- as we can see from both simulations, the converted/standardized distribution of the samples convert to the standard normal distribution
- ***Note**: speed at which the normalized coin flips converge to normal distribution depends on how biased the coin is (value of p)*
- ***Note**: does not guarantee that the normal distribution will be a good approximation, but just that eventually it will be a good approximation as n –> ∞*

Confidence Intervals - Normal Distribution/Z Intervals

- **Z confidence interval** is defined as

$$Estimate \pm ZQ \times SE_{Estimate}$$

  where ZQ = quantile from the standard normal distribution
- according to CLT, the sample mean, $\bar{X}$, is approximately normal with mean μ and sd $\sigma/\sqrt{n}$

- **95% confidence interval for the population mean** μ is defined as

$$\bar{X} \pm 2\sigma/\sqrt{n}$$

  for the sample mean $\bar{X} \sim N(\mu, \sigma^2/n)$
  - you can choose to use 1.96 to be more accurate for the confidence interval
  - $P(\bar{X} > \mu + 2\sigma/\sqrt{n}$ *or* $\bar{X} < \mu - 2\sigma/\sqrt{n}) = 5\%$
  - **interpretation**: if we were to repeated samples of size n from the population and construct this confidence interval for each case, approximately 95% of the intervals will contain μ
- confidence intervals get **narrower** with less variability or larger sample sizes
- ***Note**: Poisson and binomial distributions have exact intervals that don't require CLT*

- ***example***
  - for this example, we will compute the 95% confidence interval for sons height data in inches

```
# load son height data
data(father.son); x <- father.son$sheight
# calculate confidence interval for sons height in inches
mean(x) + c(-1, 1) * qnorm(0.975) * sd(x)/sqrt(length(x))
```

```
## [1] 68.51605 68.85209
```

Confidence Interval - Bernoulli Distribution/Wald Interval

- for Bernoulli distributions, X_i is 0 or 1 with success probability p and the variance is $\sigma^2 = p(1-p)$
- the confidence interval takes the form of

$$\hat{p} \pm z_{1-\alpha/2}\sqrt{\frac{p(1-p)}{n}}$$

- since the population proportion p is unknown, we can use $\hat{p} = X/n$ as estimate
- $p(1-p)$ is largest when $p = 1/2$, so 95% confidence interval can be calculated by

$$\begin{aligned}\hat{p} \pm Z_{0.95}\sqrt{\frac{0.5(1-0.5)}{n}} &= \hat{p} \pm 1.96\sqrt{\frac{1}{4n}} \\ &= \hat{p} \pm \frac{1.96}{2}\sqrt{\frac{1}{n}} \\ &\approx \hat{p} \pm \frac{1}{\sqrt{n}}\end{aligned}$$

  - this is known as the **Wald Confidence Interval** and is useful in ***roughly estimating*** confidence intervals
  - generally need $n = 100$ for 1 decimal place, 10,000 for 2, and 1,000,000 for 3

- ***example***
  - suppose a random sample of 100 likely voters, 56 intent to vote for you, can you secure a victory?
  - we can use the Wald interval to quickly estimate the 95% confidence interval
  - as we can see below, because the interval [0.46, 0.66] contains values below 50%, victory is not guaranteed
  - `binom.test(k, n)$conf` = returns confidence interval binomial distribution (collection of Bernoulli trial) with `k` successes in `n` draws

```r
# define sample probability and size
p = 0.56; n = 100
# Wald interval
c("WaldInterval" = p + c(-1, 1) * 1/sqrt(n))
```

```
## WaldInterval1 WaldInterval2
##          0.46          0.66
```

```r
# 95% confidence interval
c("95CI" = p + c(-1, 1) * qnorm(.975) * sqrt(p * (1-p)/n))
```

```
##     95CI1     95CI2
## 0.4627099 0.6572901
```

```r
# perform binomial test
binom.test(p*100, n*100)$conf.int
```

```
## [1] 0.004232871 0.007265981
## attr(,"conf.level")
## [1] 0.95
```

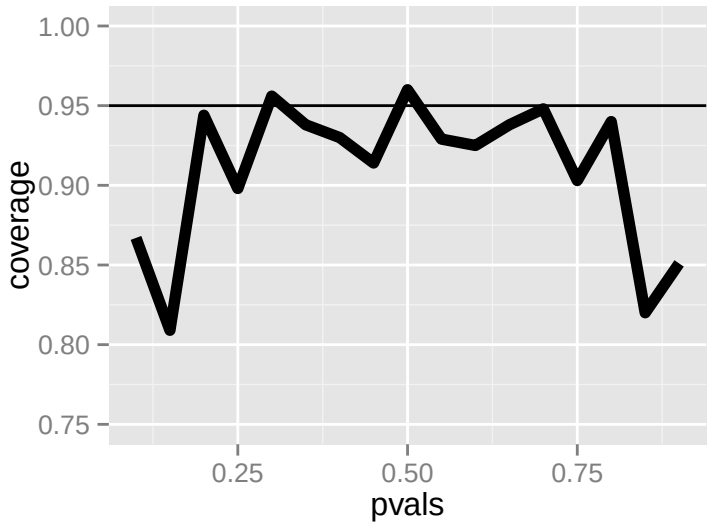
Confidence Interval - Binomial Distribution/Agresti-Coull Interval

- for a binomial distribution with smaller values of n (when $n < 30$, thus not large enough for CLT), often time the normal confidence intervals, as defined by

$$\hat{p} \pm z_{1-\alpha/2}\sqrt{\frac{p(1-p)}{n}}$$

do not provide accurate estimates

```r
# simulate 1000 samples of size 20 each
n <- 20; nosim <- 1000
# simulate for p values from 0.1 to 0.9
pvals <- seq(.1, .9, by = .05)
# calculate the confidence intervals
coverage <- sapply(pvals, function(p){
    # simulate binomial data
    phats <- rbinom(nosim, prob = p, size = n) / n
    # calculate lower 95% CI bound
    ll <- phats - qnorm(.975) * sqrt(phats * (1 - phats) / n)
    # calculate upper 95% CI bound
    ul <- phats + qnorm(.975) * sqrt(phats * (1 - phats) / n)
    # calculate percent of intervals that contain p
    mean(ll < p & ul > p)
})
# plot CI results vs 95%
ggplot(data.frame(pvals, coverage), aes(x = pvals, y = coverage)) + geom_line(size = 2) + geom_hline(yi
```



- as we can see from above, the interval do not provide adequate coverage as 95% confidence intervals (frequently only provide 80 to 90% coverage)
- we can construct the **Agresti-Coull Interval**, which is defined uses the adjustment

$$\hat{p} = \frac{X+2}{n+4}$$

where we effectively ***add 2*** to number of successes, X, and ***add 2*** to number of failure

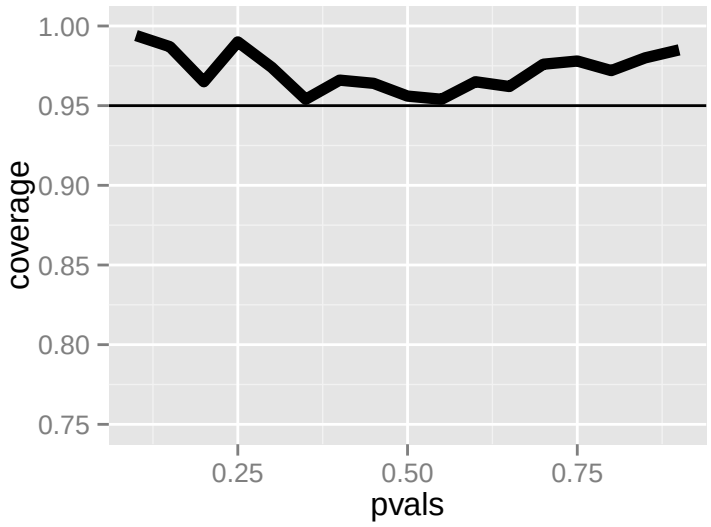
- therefore the interval becomes

$$\frac{X+2}{n+4} \pm z_{1-\alpha/2}\sqrt{\frac{p(1-p)}{n}}$$

- ***Note****: interval tend to be* ***conservative***

- ***example***

```
# simulate 1000 samples of size 20 each
n <- 20; nosim <- 1000
# simulate for p values from 0.1 to 0.9
pvals <- seq(.1, .9, by = .05)
# calculate the confidence intervals
coverage <- sapply(pvals, function(p){
    # simulate binomial data with Agresti/Coull Interval adjustment
    phats <- (rbinom(nosim, prob = p, size = n) + 2) / (n + 4)
        # calculate lower 95% CI bound
    ll <- phats - qnorm(.975) * sqrt(phats * (1 - phats) / n)
    # calculate upper 95% CI bound
    ul <- phats + qnorm(.975) * sqrt(phats * (1 - phats) / n)
    # calculate percent of intervals that contain p
    mean(ll < p & ul > p)
})
# plot CI results vs 95%
ggplot(data.frame(pvals, coverage), aes(x = pvals, y = coverage)) + geom_line(size = 2) + geom_hline(yi
```



- as we can see from above, the coverage is much better for the 95% interval
- in fact, all of the estimates are more conservative as we previously discussed, indicating the Agresti-Coull intervals are ***wider*** than the regular confidence intervals

Confidence Interval - Poisson Interval

- for $X \sim Poisson(\lambda t)$
  - estimate rate $\hat{\lambda} = X/t$
  - $var(\hat{\lambda}) = \lambda/t$
  - variance estimate $= \hat{\lambda}/t$
- so the confidence interval is defined as

$$\hat{\lambda} \pm z_{1-\alpha/2}\sqrt{\frac{\lambda}{t}}$$

  - however, for small values of λ (few events larger time interval), we **should not** use the asymptotic interval estimated
  - ***example***
    - for this example, we will go through a specific scenario as well as a simulation exercise to demonstrate the ineffectiveness of asymptotic intervals for small values of λ
    - nuclear pump failed 5 times out of 94.32 days, give a 95% confidence interval for the failure rate per day?
    - `poisson.test(x, T)$conf` = returns Poisson 95% confidence interval for given `x` occurrence over `T` time period

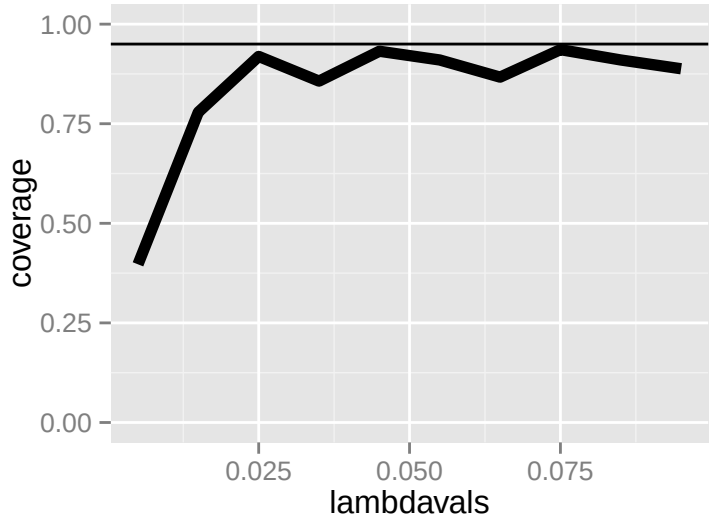
```
# define parameters
x <- 5; t <- 94.32; lambda <- x / t
# calculate confidence interval
round(lambda + c(-1, 1) * qnorm(.975) * sqrt(lambda / t), 3)
```

```
## [1] 0.007 0.099
```

```
# return accurate confidence interval from poisson.test
poisson.test(x, T = 94.32)$conf
```

```
## [1] 0.01721254 0.12371005
## attr(,"conf.level")
## [1] 0.95
```

```r
# small lambda simulations
lambdavals <- seq(0.005, 0.10, by = .01); nosim <- 1000; t <- 100
# calculate coverage using Poisson intervals
coverage <- sapply(lambdavals, function(lambda){
    # calculate Poisson rates
    lhats <- rpois(nosim, lambda = lambda * t) / t
    # lower bound of 95% CI
    ll <- lhats - qnorm(.975) * sqrt(lhats / t)
    # upper bound of 95% CI
    ul <- lhats + qnorm(.975) * sqrt(lhats / t)
    # calculate percent of intervals that contain lambda
    mean(ll < lambda & ul > lambda)
})
# plot CI results vs 95%
ggplot(data.frame(lambdavals, coverage), aes(x = lambdavals, y = coverage)) + geom_line(size = 2) + geo
```

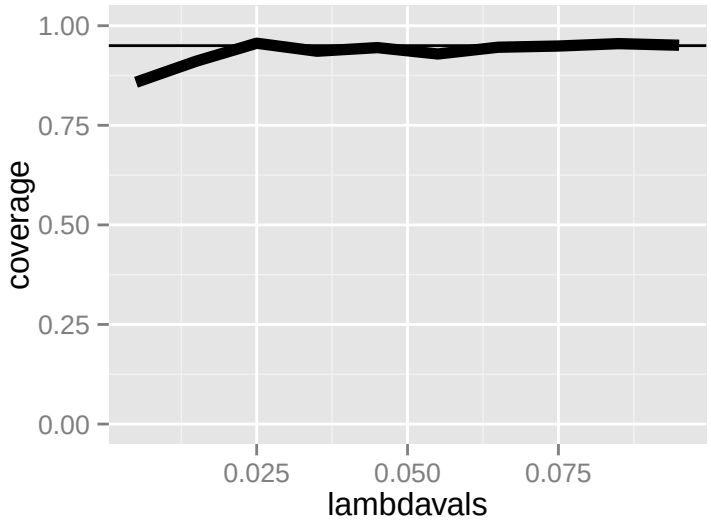


- as we can see above, for small values of $\lambda = X/t$, the confidence interval produced by the asymptotic interval is ***not*** an accurate estimate of the actual 95% interval (not enough coverage)

- however, as $t \to \infty$, the interval becomes the ***true 95% interval***

```r
# small lambda simulations
lambdavals <- seq(0.005, 0.10, by = .01); nosim <- 1000; t <- 1000
# calculate coverage using Poisson intervals
coverage <- sapply(lambdavals, function(lambda){
    # calculate Poisson rates
    lhats <- rpois(nosim, lambda = lambda * t) / t
    # lower bound of 95% CI
    ll <- lhats - qnorm(.975) * sqrt(lhats / t)
    # upper bound of 95% CI
    ul <- lhats + qnorm(.975) * sqrt(lhats / t)
    # calculate percent of intervals that contain lambda
```

```
    mean(ll < lambda & ul > lambda)
})
# plot CI results vs 95%
ggplot(data.frame(lambdavals, coverage), aes(x = lambdavals, y = coverage)) + geom_line(size = 2) + geo
```



- as we can see from above, as t increases, the Poisson intervals become closer to the actual 95% confidence intervals

Confidence Intervals - T Distribution(Small Samples)

- **t** confidence interval is defined as

$$Estimate \pm TQ \times SE_{Estimate} = \bar{X} \pm \frac{t_{n-1}S}{\sqrt{n}}$$

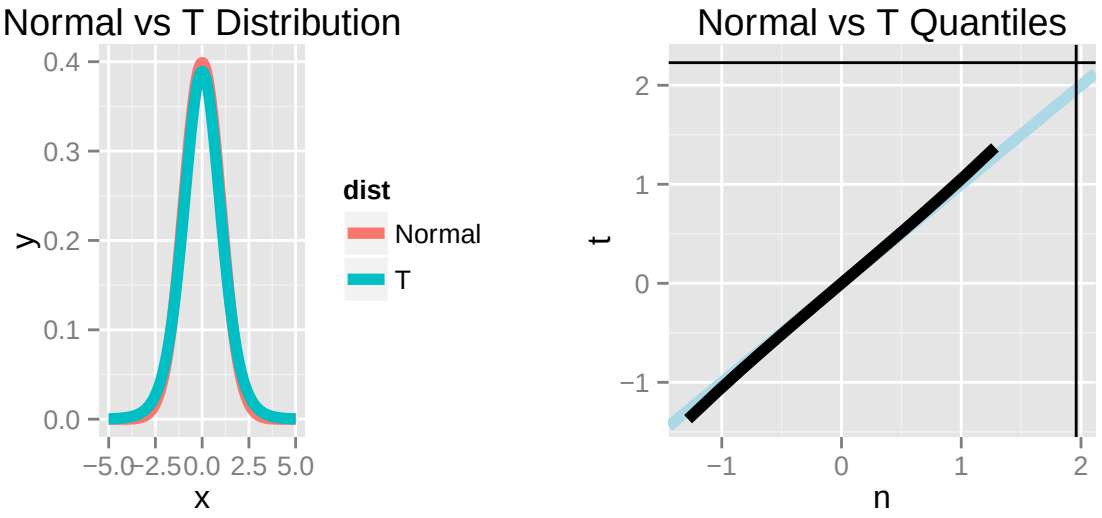
  - TQ = quantile from T distribution
  - t_{n-1} = relevant quantile
  - t interval assumes data is IID normal so that

$$\frac{\bar{X} - \mu}{S/\sqrt{n}}$$

    follows Gosset's t distribution with $n-1$ degrees of freedom
  - works well with data distributions that are roughly symmetric/mound shaped, and ***does not*** work with skewed distributions
    - skewed distribution –> meaningless to center interval around the mean $\bar{X}$
    - logs/median can be used instead
  - paired observations (multiple measurements from same subjects) can be analyzed by t interval of differences
  - as more data collected (large degrees of freedom), t interval –> z interval
  - `qt(0.975, df=n-1)` = calculate the relevant quantile using t distribution

```
# Plot normal vs t distributions
k <- 1000; xvals <- seq(-5, 5, length = k); df <- 10
d <- data.frame(y = c(dnorm(xvals), dt(xvals, df)),x = xvals,
```

```
              dist = factor(rep(c("Normal", "T"), c(k,k))))
g <- ggplot(d, aes(x = x, y = y))
g <- g + geom_line(size = 2, aes(colour = dist)) + ggtitle("Normal vs T Distribution")
# plot normal vs t quantiles
d <- data.frame(n= qnorm(pvals),t=qt(pvals, df),p = pvals)
h <- ggplot(d, aes(x= n, y = t))
h <- h + geom_abline(size = 2, col = "lightblue")
h <- h + geom_line(size = 2, col = "black")
h <- h + geom_vline(xintercept = qnorm(0.975))
h <- h + geom_hline(yintercept = qt(0.975, df)) + ggtitle("Normal vs T Quantiles")
# plot 2 graphs together
grid.arrange(g, h, ncol = 2)
```



- William Gosset's **t** Distribution ("Student's T distribution")
  - test = Gosset's pseudoname which he published under
  - indexed/defined by ***degrees of freedom***, and becomes more like standard normal as degrees of freedom gets larger
  - thicker tails centered around 0, thus confidence interval = ***wider*** than Z interval (more mass concentrated away from the center)
  - for ***small*** sample size (value of n), normalizing the distribution by $\frac{\bar{X}-\mu}{S/\sqrt{n}}$ –> t distribution, ***not*** the standard normal distribution
    - S = standard deviation may be inaccurate, as the std of the data sample may not be truly representative of the population std
    - using the Z interval here thus may produce an interval that is too ***narrow***

Confidence Interval - Paired T Tests

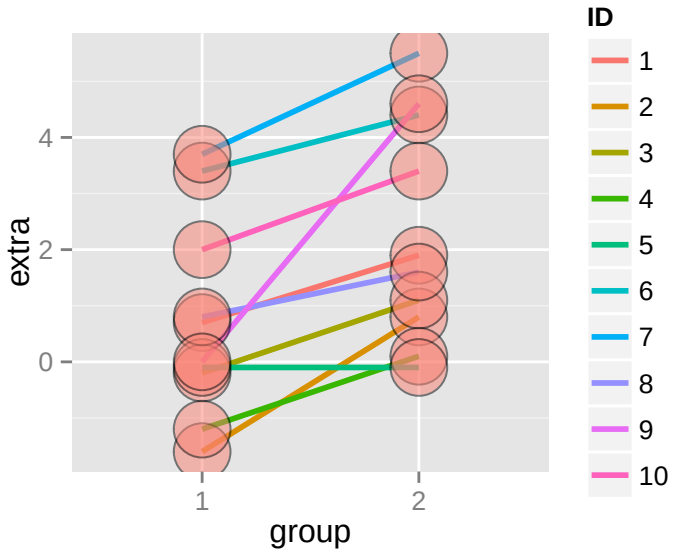
- compare observations for the same subjects over two different sets of data (i.e. different times, different treatments)
- the confidence interval is defined by

$$\bar{X}_1 - \bar{X}_2 \pm \frac{t_{n-1}S}{\sqrt{n}}$$

where $\bar{X}_1$ represents the first observations and $\bar{X}_2$ the second set of observations

- `t.test(difference)` = performs group mean t test and returns metrics as results, which includes the confidence intervals
  - `t.test(g2, g1, paired = TRUE)` = performs the same paired t test with data directly
- ***example***
  - the data used here is for a study of the effects of two soporific drugs (increase in hours of sleep compared to control) on 10 patients

```
# load data
data(sleep)
# plot the first and second observations
g <- ggplot(sleep, aes(x = group, y = extra, group = factor(ID)))
g <- g + geom_line(size = 1, aes(colour = ID)) + geom_point(size =10, pch = 21, fill = "salmon", alpha =
g
```




```
# define groups
g1 <- sleep$extra[1 : 10]; g2 <- sleep$extra[11 : 20]
# define difference
difference <- g2 - g1
# calculate mean and sd of differences
mn <- mean(difference); s <- sd(difference); n <- 10
# calculate intervals manually
mn + c(-1, 1) * qt(.975, n-1) * s / sqrt(n)
```

```
## [1] 0.7001142 2.4598858
```

```
# perform the same test to get confidence intervals
t.test(difference)
```

```
##
##  One Sample t-test
##
## data:  difference
```

```
## t = 4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.7001142 2.4598858
## sample estimates:
## mean of x
##      1.58
```

```
t.test(g2, g1, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  g2 and g1
## t = 4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.7001142 2.4598858
## sample estimates:
## mean of the differences
##                    1.58
```

Independent Group t Intervals - Same Variance

- compare two groups in randomized trial ("A/B Testing")
- cannot use the paired t test because the groups are independent and may have different sample sizes
- perform randomization to balance unobserved covariance that may otherwise affect the result
- t confidence interval for $\mu_y - \mu_x$ is defined as

$$\bar{Y} - \bar{X} \pm t_{n_x+n_y-2,1-\alpha/2} S_p \left(\frac{1}{n_x} + \frac{1}{n_y}\right)^{1/2}$$

  - $t_{n_x+n_y-2,1-\alpha/2}$ = relevant quantile
  - $n_x + n_y - 2$ = degrees of freedom
  - $S_p \left(\frac{1}{n_x} + \frac{1}{n_y}\right)^{1/2}$ = standard error
  - $S_p^2 = \{(n_x - 1)S_x^2 + (n_y - 1)S_y^2\}/(n_x + n_y - 2)$ = pooled variance estimator
    - this is effectively a weighted average between the two variances, such that different sample sizes are taken in to account
  - ***Note:*** *this interval assumes* ***constant variance*** *across two groups; if variance is different, use the next interval*

Independent Group t Intervals - Different Variance

- confidence interval for $\mu_y - \mu_x$ is defined as

$$\bar{Y} - \bar{X} \pm t_{df} \times \left(\frac{s_x^2}{n_x} + \frac{s_y^2}{n_y}\right)^{1/2}$$

  - t_{df} = relevant quantile with df as defined below
  - ***Note****: normalized statistic does not follow t distribution but can be approximated through the formula with df defined below*

* $$df = \frac{\left(S_x^2/n_x + S_y^2/n_y\right)^2}{\left(\frac{S_x^2}{n_x}\right)^2/(n_x-1) + \left(\frac{S_y^2}{n_y}\right)^2/(n_y-1)}$$
* $\left(\frac{s_x^2}{n_x} + \frac{s_y^2}{n_y}\right)^{1/2}$ = standard error

- Comparing other kinds of data
  - binomial –> relative risk, risk difference, odds ratio
  - binomial –> Chi-squared test, normal approximations, exact tests
  - count –> Chi-squared test, exact tests
- R commands
  - t Confidence Intervals
    * `mean + c(-1, 1) * qt(0.975, n - 1) * std / sqrt(n)`
      · ***c(-1, 1)*** = plus and minus, ±
  - Difference Intervals (all equivalent)
    * `mean2 - mean1 + c(-1, 1) * qt(0.975, n - 1) * std / sqrt(n)`
      · ***n*** = number of paired observations
      · ***qt(0.975, n - 1)*** = relevant quantile for paired
      · ***qt(0.975, n_x + n_y - 2)*** = relevant quantile for independent
    * `t.test(mean2 - mean1)`
    * `t.test(data2, data1, paired = TRUE, var.equal = TRUE)`
      · ***paired*** = whether or not the two sets of data are paired (same subjects different observations for treatment) <– `TRUE` for paired, `FALSE` for independent
      · ***var.equal*** = whether or not the variance of the datasets should be treated as equal <– `TRUE` for same variance, `FALSE` for unequal variances
    * `t.test(extra ~ I(relevel(group, 2)), paired = TRUE, data = sleep)`
      · ***relevel(factor, ref)*** = reorders the levels in the factor so that "ref" is changed to the first level –> doing this here is so that the second set of measurements come first (1, 2 –> 2, 1) in order to perform mean_2 - mean_1
      · ***I(object)*** = prepend the class "AsIs" to the object
      · ***Note****: I(relevel(group, 2)) = explanatory variable, must be* ***factor*** *and have* ***two levels***

Hypothesis Testing

- Hypothesis testing = making decisions using data
  - **null** hypothesis ($\mathbf{H}_0$) = status quo
  - assumed to be ***true*** –> statistical evidence required to reject it for **alternative** or "research" hypothesis ($\mathbf{H}_a$)
    - alternative hypothesis typically take form of $>$, $<$ or $\neq$
  - Results

Truth	Decide	Result
H_0	H_0	Correctly accept null
H_0	H_a	Type I error
H_a	H_a	Correctly reject null
H_a	H_0	Type II error

- α = Type I error rate
  - probability of ***rejecting*** the null hypothesis when the hypothesis is ***correct***
  - $\alpha = 0.5$ –> standard for hypothesis testing
  - ***Note****: as Type I error rate increases, Type II error rate decreases and vice versa*
- for large samples (large n), use the **Z Test** for $H_0 : \mu = \mu_0$
  - H_a**:**
    - $H_1 : \mu < \mu_0$
    - $H_2 : \mu \neq \mu_0$
    - $H_3 : \mu > \mu_0$
  - Test statistic $TS = \frac{\bar{X} - \mu_0}{S/\sqrt{n}}$
  - Reject the null hypothesis H_0 when
    - $H_1 : TS \leq Z_\alpha$ OR $-Z_{1-\alpha}$
    - $H_2 : |TS| \geq Z_{1-\alpha/2}$
    - $H_3 : TS \geq Z_{1-\alpha}$
  - ***Note****: In case of $\alpha = 0.5$ (most common), $Z_{1-\alpha} = 1.645$ (95 percentile)*
  - α = low, so that when H_0 is rejected, original model –> wrong or made an error (low probability)
- For small samples (small n), use the **T Test** for $H_0 : \mu = \mu_0$
  - H_a**:**
    - $H_1 : \mu < \mu_0$
    - $H_2 : \mu \neq \mu_0$
    - $H_3 : \mu > \mu_0$
  - Test statistic $TS = \frac{\bar{X} - \mu_0}{S/\sqrt{n}}$
  - Reject the null hypothesis H_0 when
    - $H_1 : TS \leq T_\alpha$ OR $-T_{1-\alpha}$
    - $H_2 : |TS| \geq T_{1-\alpha/2}$
    - $H_3 : TS \geq T_{1-\alpha}$
  - ***Note****: In case of $\alpha = 0.5$ (most common), $T_{1-\alpha}$ =* `qt(.95, df = n-1)`
  - R commands for T test:

    * `t.test(vector1 - vector2)`
    * `t.test(vector1, vector2, paired = TRUE)`
        · `alternative` argument can be used to specify one-sided tests: `less` or `greater`
        · `alternative` default = `two-sided`
    * prints test statistic (`t`), degrees of freedom (`df`), `p-value`, 95% confidence interval, and mean of sample
        · confidence interval in units of data, and can be used to intepret the practical significance of the results

- **rejection region** = region of TS values for which you reject H_0
- **power** = probability of rejecting H_0
  - power is used to calculate sample size for experiments
- **two-sided tests** –> $H_a : \mu \neq \mu_0$
  - reject H_0 only if test statistic is too larger/small
  - for $\alpha = 0.5$, split equally to 2.5% for upper and 2.5% for lower tails
    * equivalent to $|TS| \geq T_{1-\alpha/2}$
    * example: for T test, `qt(.975, df)` and `qt(.025, df)`
  - ***Note**: failing to reject one-sided test = fail to reject two-sided*
- **tests vs confidence intervals**
  - $(1-\alpha)$% confidence interval for μ = set of all possible values that fail to reject H_0
  - if $(1-\alpha)$% confidence interval contains μ_0, fail to reject H_0
- **two-group intervals/test**
  - Rejection rules the same
  - Test H_0: $\mu_1 = \mu_2$ –> $\mu_1 - \mu_2 = 0$
  - Test statistic:
$$\frac{Estimate - H_0 Value}{SE_{Estimate}} = \frac{\bar{X}_1 - \bar{X}_2 - 0}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$
  - R Command
    * `t.test(values ~ factor, paired = FALSE, var.equal = TRUE, data = data)`
        · `paired = FALSE` –> independent values
        · `factor` argument must have only two levels
- **p values**
  - most common measure of statistical significance
  - **p-value** = probability under the null hypothesis of obtaining evidence as extreme or more than that of the obtained
    * Given that H_0 is true, how likely is it to obtain the result (test statistic)?
  - **attained significance level** = smallest value for α for which H_0 is rejected –> equivalent to p-value
    * if p-value $< \alpha$, reject H_0
    * for two-sided tests, double the p-values
  - if p-value is small, either H_0 is true AND the obeserved is a rare event **OR** H_0 is false
  - R Command
    * p-value = `pt(statistic, df, lower.tail = FALSE)`
        · `lower.tail = FALSE` = returns the probability of getting a value from the t distribution that is larger than the test statistic

* Binomial (coin flips)
  - probability of getting x results out of n trials and event probability of p = `pbinom(x, size = n, prob = p, lower.tail = FALSE)`
  - two-sided interval (testing for $\neq$): find the smaller of two one-sided intervals (X < value, X > value), and double the result
  - ***Note:*** *`lower.tail = FALSE` = strictly greater*
* Poisson
  - probability of getting x results given the rate r = `ppois(x - 1, r, lower.tail = FALSE)`
  - `x - 1` is used here because the upper tail includes the specified number (since we want greater than x, we start at x - 1)
  - `r` = events that should occur given the rate (multiplied by 100 to yield an integer)
  - ***Note:*** *`lower.tail = FALSE` = strictly greater*

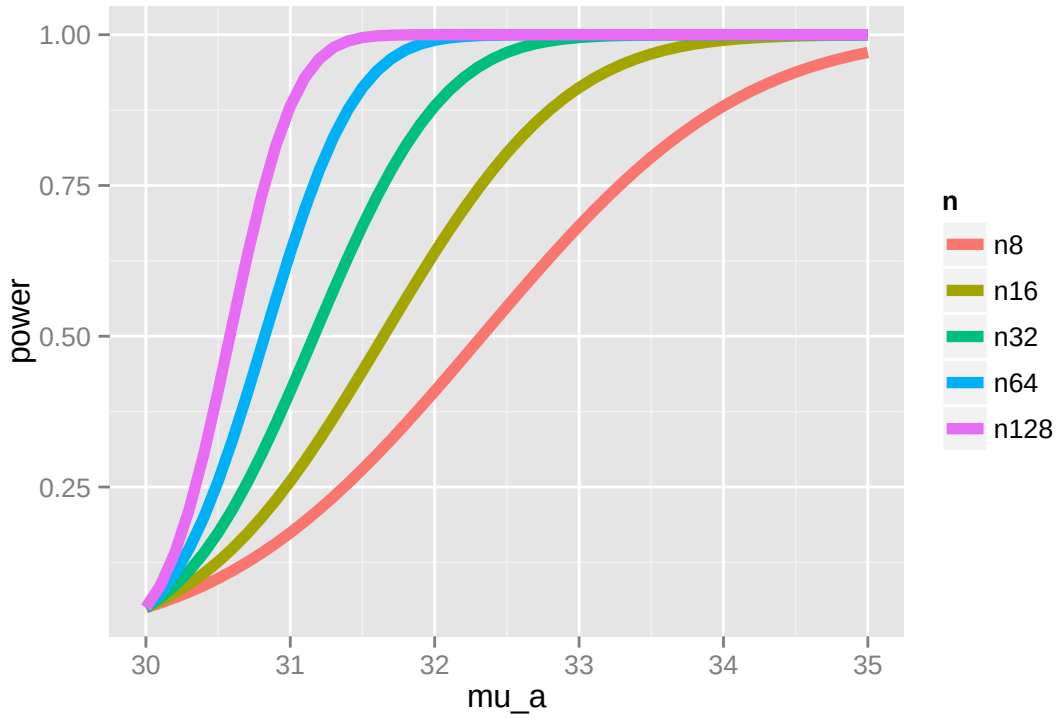
Power

- **Power** = probability of rejecting the null hypothesis when it is false (the more power the better)
  - most often used in designing studies so that there's a reasonable chance to detect the alternative hypothesis if the alternative hypothesis is true
- β = probability of type II error = failing to reject the null hypothesis when it's false
- power = $1 - \beta$
- ***example***
  - $H_0 : \mu = 30 \rightarrow \bar{X} \sim N(\mu_0, \sigma^2/n)$
  - $H_a : \mu > 30 \rightarrow \bar{X} \sim N(\mu_a, \sigma^2/n)$
  - Power:

$$Power = P\left(\frac{\bar{X} - 30}{s/\sqrt{n}} > t_{1-\alpha,n-1} \; ; \; \mu = \mu_a\right)$$

    - ***Note**: the above function depends on value of μ_a*
    - ***Note**: as μ_a approaches 30, power approaches α*
  - assuming the sample mean is normally distributed, H_0 is rejected when $\frac{\bar{X}-30}{\sigma/\sqrt{n}} > Z_{1-\alpha}$
  - or, $\bar{X} > 30 + Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}$
- R commands:
  - `alpha = 0.05; z = qnorm(1-alpha)` –> calculates $Z_{1-\alpha}$
  - `pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)` –> calculates the probability of getting a sample mean that is larger than $Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}$ given that the population mean is μ_a
    - ***Note**: using `mean = mu0` in the function would = alpha*
  - Power curve behavior
    - Power increases as mu_a increases –> we are more likely to detect the difference in mu_a and mu_0
    - Power increases as **n** increases –> with more data, more likely to detect any alternative mu_a

```
library(ggplot2)
mu0 = 30; mua = 32; sigma = 4; n = 16
alpha = 0.05
z = qnorm(1 - alpha)
nseq = c(8, 16, 32, 64, 128)
mu_a = seq(30, 35, by = 0.1)
power = sapply(nseq, function(n)
    pnorm(mu0 + z * sigma / sqrt(n), mean = mu_a, sd = sigma / sqrt(n),
          lower.tail = FALSE)
    )
colnames(power) <- paste("n", nseq, sep = "")
d <- data.frame(mu_a, power)
library(reshape2)
d2 <- melt(d, id.vars = "mu_a")
names(d2) <- c("mu_a", "n", "power")
g <- ggplot(d2,
            aes(x = mu_a, y = power, col = n)) + geom_line(size = 2)
g
```



- **Solving for Power**
  - When testing $H_a : \mu > \mu_0$ (or $<$ or $\neq$)

$$Power = 1 - \beta = P\left(\bar{X} > \mu_0 + Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}; \mu = \mu_a\right)$$

    where $\bar{X} \sim N(\mu_a, \sigma^2/n)$
  - Unknowns = μ_a, σ, n, β
  - Knowns = μ_0, α
  - Specify any 3 of the unknowns and you can solve for the remainder; most common are two cases
    1. Given power desired, mean to detect, variance that we can tolerate, find the **n** to produce desired power (designing experiment/trial)
    2. Given the size **n** of the sample, find the power that is achievable (finding the utility of experiment)
  - ***Note**: for $H_a : \mu \neq mu_0$, calculated one-sided power using $z_{1-\alpha/2}$; however, the power calculation here exclusdes the probability of getting a large TS in the opposite direction of the truth, but this is only applicable when μ_a and μ_0 are close together*
- **Power Behavior**
  - Power increases as α becomes larger
  - Power of one-sided test > power of associated two-sided test
  - Power increases as μ_a gets further away from μ_0
  - Power increases as **n** increases (sample mean has less variability)
  - Power increases as σ decreases (again less variability)
  - Power usually depends only $\frac{\sqrt{n}(\mu_a - \mu_0)}{\sigma}$, and not μ_a, σ, and n
    - **effect size** = $\frac{\mu_a - \mu_0}{\sigma}$ –> unit free, can be interpretted across settings
- **T-test Power**

- for Gossett's T test,

$$Power = P\left(\frac{\bar{X} - \mu_0}{S/\sqrt{n}} > t_{1-\alpha,n-1}; \mu = \mu_a\right)$$

  * $\frac{\bar{X}-\mu_0}{S/\sqrt{n}}$ does not follow a t distribution if the true mean is μ_a and NOT μ_0 –> follows a non-central t distribution instead

- `power.t.test` –> evaluates the non-central t distribution and solves for a parameter given all others are specified
  * `power.t.test(n = 16, delta = 0.5, sd = 1, type = "one.sample", alt = "one.sided")$power` –> calculates power with inputs of n, difference in means, and standard deviation
    - `delta` = argument for difference in means
    - ***Note**: since effect size = `delta/sd`, as `n`, `type`, and `alt` are held constant, any distribution with the same effect size will have the same power*
  * `power.t.test(power = 0.8, delta = 0.5, sd = 1, type = "one.sample", alt = "one.sided")$n` –> calculates size n with inputs of power, difference in means, and standard deviation
    - ***Note**: n should always be rounded up (ceiling)*

Multiple Testing

- Hypothesis testing/significant analysis commonly overused
- correct for multiple testing to avoid false positives/conclusions (two key components)
    1. error measure
    2. correction
- multiple testing is needed because of the increase in ubiquitous data collection technology and analysis
    - DNA sequencing machines
    - imaging patients in clinical studies
    - electronic medical records
    - individualized movement data (fitbit)

Type of Errors

| | Actual H_0 = True | Actual H_a = True | Total

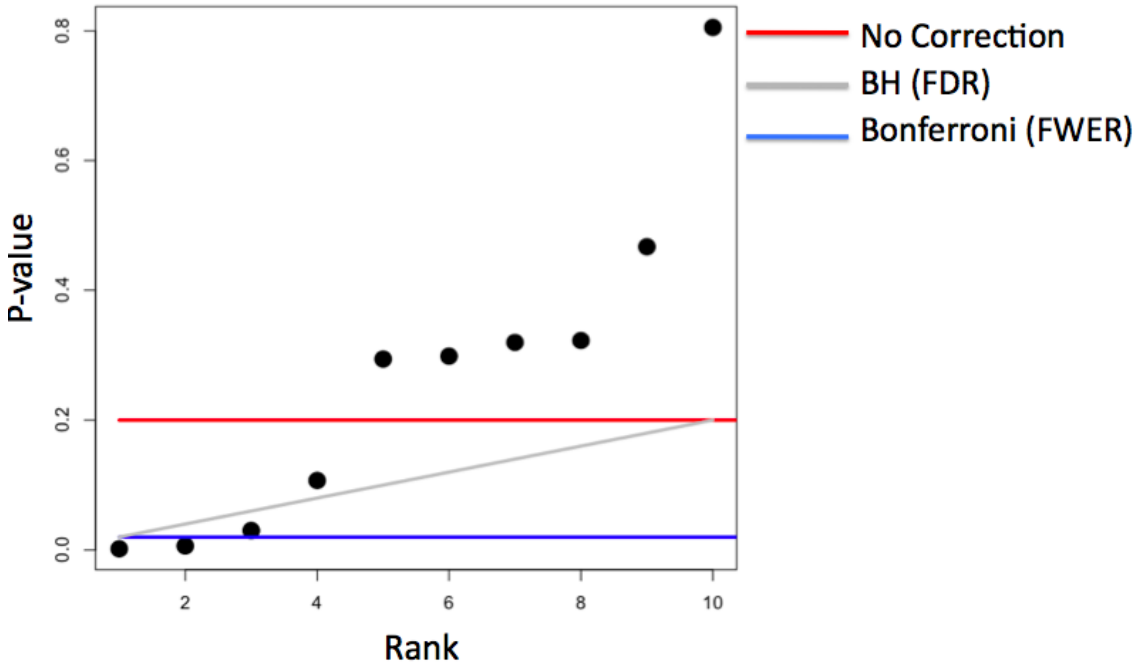
———————-|————|————|——— Conclude H_0 = True (non-significant) | U | T | $m-R$ Conclude H_a = True (significant) | V | S | R **Total** | m_0 | $m-m_0$ | m

- m_0 = number of true null hypotheses, or cases where H_0 = actually true (unknown)
- $m - m_0$ = number of true alternative hypotheses, or cases where H_a = actually true (unknown)
- R = number of null hypotheses rejected, or cases where H_a = concluded to be true (measurable)
- $m - R$ = number of null hypotheses that failed to be rejected, or cases where H_0 = concluded to be true (measurable)
- V = Type I Error / false positives, concludes H_a = True when H_0 = actually True
- T = Type II Error / false negatives, concludes H_0 = True when H_a = actually True
- S = true positives, concludes H_a = True when H_a = actually True
- U = true negatives, concludes H_0 = True when H_0 = actually True

Error Rates

- ***false positive rate*** = rate at which false results are called significant $E[\frac{V}{m_0}]$ –> average fraction of times that H_a is claimed to be true when H_0 is actually true
    - ***Note****: mathematically equal to type I error rate –> false positive rate is associated with a post-prior result, which is the expected number of false positives divided by the total number of hypotheses under the real combination of true and non-true null hypotheses (disregarding the "global null" hypothesis). Since the false positive rate is a parameter that is not controlled by the researcher, it cannot be identified with the significance level, which is what determines the type I error rate.*
- ***family wise error rate (FWER)*** = probabilit of at least one false positive $Pr(V \geq 1)$
- ***false discovery rate (FDR)*** = rate at which claims of significance are false $E[\frac{V}{R}]$
- **controlling error rates (adjusting α)**
    - false positive rate
        * if we call all $P < \alpha$ significant (reject H_0), we are expected to get $\alpha \times m$ false positives, where m = total number of hypothesis test performed
        * with high values of m, false positive rate is very large as well

- family-wise error rate (FWER)
  - controlling FWER = controlling the probability of even one false positive
  - *bonferroni* correction (oldest multiple testing correction)
    - for m tests, we want $Pr(V \geq 1) < \alpha$
    - calculate P-values normally, and deem them significant if and only if $P < \alpha_{fewer} = \alpha/m$
  - easy to calculate, but tend to be very ***conservative***
- false discovery rate (FDR)
  - most popular correction = controlling FDR
  - for m tests, we want $E[\frac{V}{R}] \leq \alpha$
  - calculate P-values normally and sort some from smallest to largest –> $P_{(1)}, P_{(1)}, ..., P_{(m)}$
  - deem the P-values significant if $P_{(i)} \leq \alpha \times \frac{i}{m}$
  - easy to calculate, less conservative, but allows for more false positives and may behave strangely under dependence (related hypothesis tests/regression with different variables)
- ***example***
  - 10 P-values with $\alpha = 0.20$



- **adjusting for p-values**
  - ***Note****: changing P-values will fundamentally change their properties but they can be used directly without adjusting /alpha*
  - *bonferroni* (FWER)
    - $P_i^{fewer} = max(mP_i, 1)$ –> since p cannot exceed value of 1
    - deem P-values significant if $P_i^{fewer} < \alpha$
    - similar to controlling FWER

Example

```r
set.seed(1010093)
pValues <- rep(NA,1000)
for(i in 1:1000){
  x <- rnorm(20)
  # First 500 beta=0, last 500 beta=2
  if(i <= 500){y <- rnorm(20)}else{ y <- rnorm(20,mean=2*x)}
  # calculating p-values by using linear model; the [2, 4] coeff in result = pvalue
  pValues[i] <- summary(lm(y ~ x))$coeff[2,4]
}
# Controls false positive rate
trueStatus <- rep(c("zero","not zero"),each=500)
table(pValues < 0.05, trueStatus)
```

```
##        trueStatus
##         not zero zero
##   FALSE        0  476
##   TRUE       500   24
```

```r
# Controls FWER
table(p.adjust(pValues,method="bonferroni") < 0.05,trueStatus)
```

```
##        trueStatus
##         not zero zero
##   FALSE       23  500
##   TRUE       477    0
```

```r
# Controls FDR (Benjamin Hochberg)
table(p.adjust(pValues,method="BH") < 0.05,trueStatus)
```

```
##        trueStatus
##         not zero zero
##   FALSE        0  487
##   TRUE       500   13
```

Resample Inference

- **Bootstrap** = useful tool for constructing confidence intervals and caclulating standard errors for difficult statistics
  - ***principle*** = if a statistic's (i.e. median) sampling distribution is unknown, then use distribution defined by the data to approximate it
  - ***procedures***
    1. simulate n observations **with replacement** from the observed data –> results in 1 simulated complete data set
    2. calculate desired statistic (i.e. median) for each simulated data set
    3. repeat the above steps B times, resulting in B simulated statistics
    4. these statistics are approximately drawn from the sampling distribution of the true statistic of n observations
    5. perform one of the following
       * plot a histogram
       * calculate standard deviation of the statistic to estimate its standard error
       * take quantiles (2.5^{th} and 97.5^{th}) as a confidence interval for the statistic ("*bootstrap CI*")
  - ***example***
    * Bootstrap procedure for calculating confidence interval for the median from a data set of n observations –> approximate sampling distribution

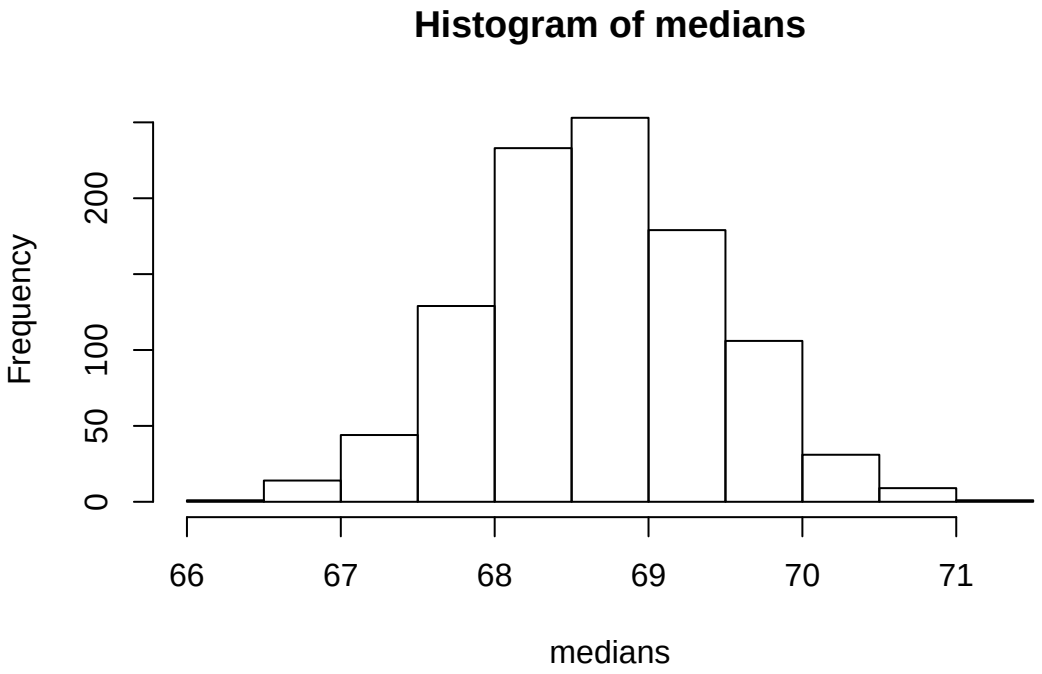
```
# load data
library(UsingR); data(father.son)
# observed dataset
x <- father.son$sheight
# number of simulated statistic
B <- 1000
# generate samples
resamples <- matrix(
    sample(x,                # sample to draw frome
           n * B,            # draw B datasets with n observations each
           replace = TRUE), # cannot draw n*B elements from x (has n elements) without replacement
    B, n)                    # arrange results into n x B matrix
                             # (every row = bootstrap sample with n observations)
# take median for each row/generated sample
medians <- apply(resamples, 1, median)
# estimated standard error of median
sd(medians)
```

```
## [1] 0.76595
```

```
# confidence interval of median
quantile(medians, c(.025, .975))
```

```
##     2.5%    97.5%
## 67.18292 70.16488
```

```
# histogram of bootstraped samples
hist(medians)
```

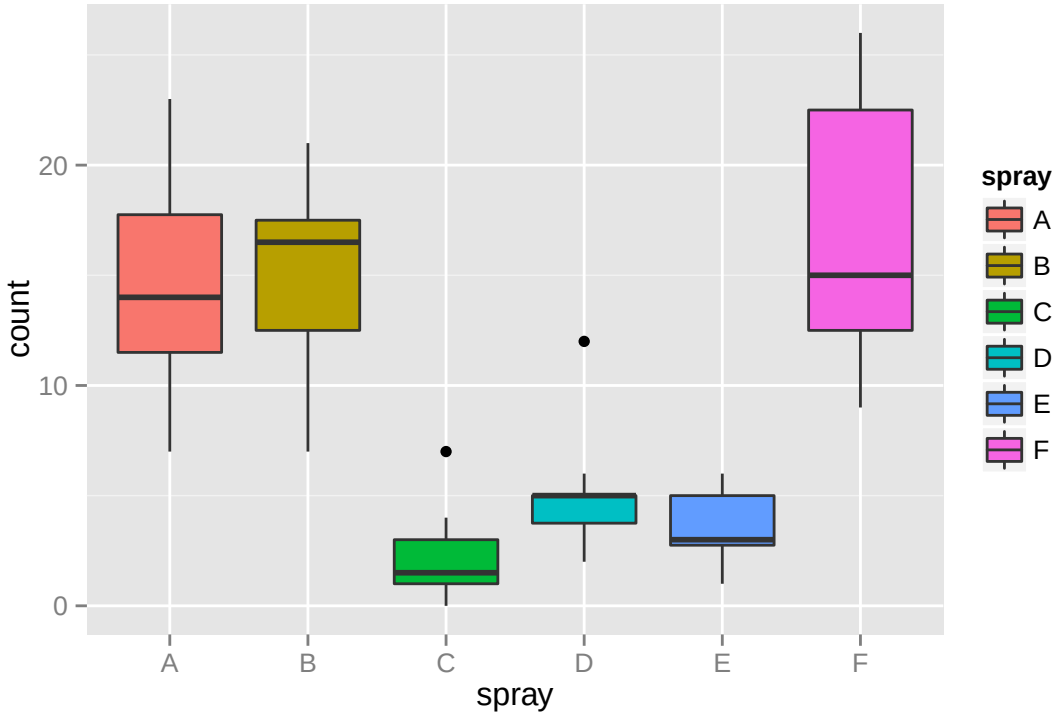


- ***Note:*** *better percentile bootstrap confidence interval = "bias corrected and accelerated interval" in* `bootstrap` *package*
- **Permutation Tests**
  - ***procedures***
    - compare groups of data and test the null hypothesis that the distribution of the observations from each group = same
      - ***Note****: if this is true, then group labels/divisions are irrelevant*
    - permute the labels for the groups
    - recalculate the statistic
      - Mean difference in counts
      - Geometric means
      - T statistic
    - Calculate the percentage of simulations wherethe simulated statistic was more extreme (toward the alternative) than the observed
  - ***variations***

Data type	Statistic	Test name
Ranks	rank sum	rank sum test
Binary	hypergeometric prob	Fisher's exact test
Raw data		ordinary permutation test

    - ***Note****: randomization tests are exactly permutation tests, with a different motivation*
    - For matched data, one can randomize the signs
    - For ranks, this results in the **signed rank test**
    - Permutation strategies work for regression by permuting a regressor of interest
    - Permutation tests work very well in multivariate settings

- ***example***
  - we will compare groups **B** and **C** in this dataset for null hypothesis H_0 : there are no difference between the groups



- we will compare groups **B** and **C** in this dataset for null hypothesis H_0 : there are no difference between the groups

```
# subset to only "B" and "C" groups
subdata <- InsectSprays[InsectSprays$spray %in% c("B", "C"),]
# values
y <- subdata$count
# labels
group <- as.character(subdata$spray)
# find mean difference between the groups
testStat <- function(w, g) mean(w[g == "B"]) - mean(w[g == "C"])
observedStat <- testStat(y, group)
observedStat
```

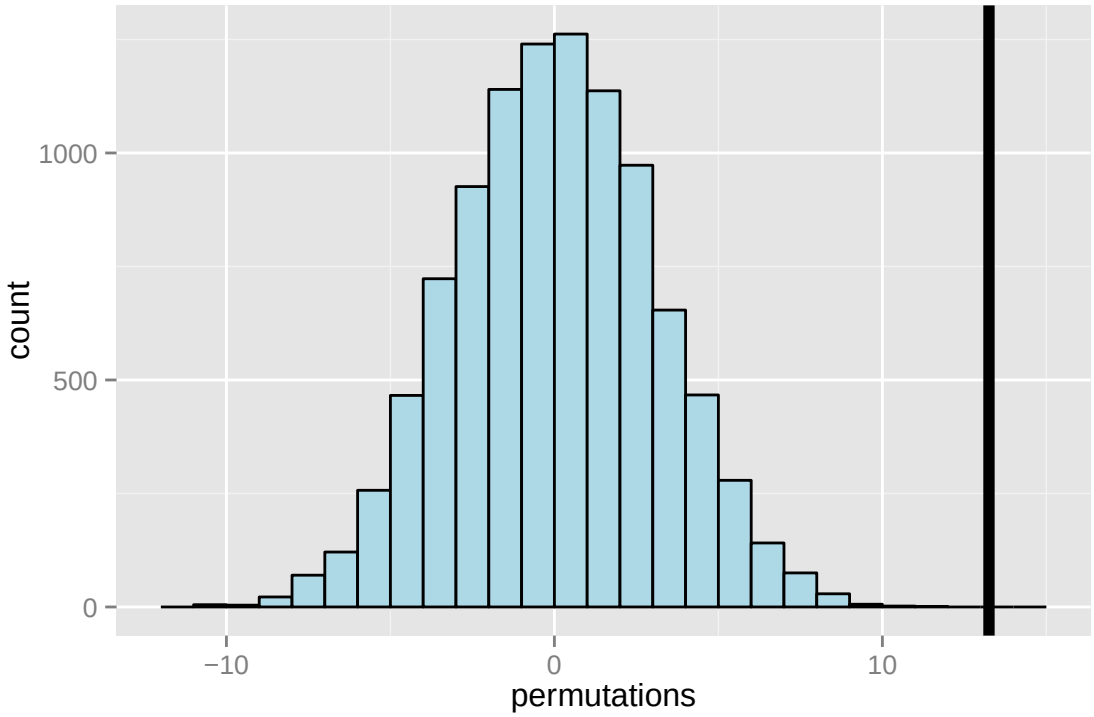
```
## [1] 13.25
```

- the observed difference between the groups is 13.25
- now we changed the resample the lables for groups **B** and **C**

```
# create 10000 permutations of the data with the labels' changed
permutations <- sapply(1 : 10000, function(i) testStat(y, sample(group)))
# find the number of permutations whose difference that is bigger than the observed
mean(permutations > observedStat)
```

```
## [1] 0
```

- we created 1000 permutations from the observed dataset, and found ***no datasets*** with mean differences between groups **B** and **C** larger than the original data
- therefore, p-value is very small and we can ***reject the null*** hypothesis with any reasonable α levels
- below is the plot for the null distribution/permutations



- as we can see from the black line, the observed difference/statistic is very far from the mean –> likely 0 is ***not*** the true difference
  - with this information, formal confidence intervals can be constructed and p-values can be calculated