Power

Statistical Inference

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

Power

- Power is the probability of rejecting the null hypothesis when it is false
- Ergo, power (as its name would suggest) is a good thing; you want more power
- A type II error (a bad thing, as its name would suggest) is failing to reject the null hypothesis when it's false; the probability of a type II error is usually called β
- Note Power $= 1 - \beta$

Notes

- Consider our previous example involving RDI
- $H_0 : \mu = 30$ versus $H_a : \mu > 30$
- Then power is

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

- Note that this is a function that depends on the specific value of μ_a!
- Notice as μ_a approaches 30 the power approaches α

Calculating power for Gaussian data

- We reject if $\frac{\bar{X}-30}{\sigma/\sqrt{n}} > z_{1-\alpha}$
    - Equivalently if $\bar{X} > 30 + Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}$
- Under $H_0 : \bar{X} \sim N(\mu_0, \sigma^2/n)$
- Under $H_a : \bar{X} \sim N(\mu_a, \sigma^2/n)$
- So we want

```
alpha = 0.05
z = qnorm(1 - alpha)
pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)
```

Example continued

- $\mu_a = 32$, $\mu_0 = 30$, $n = 16$, $\sigma = 4$

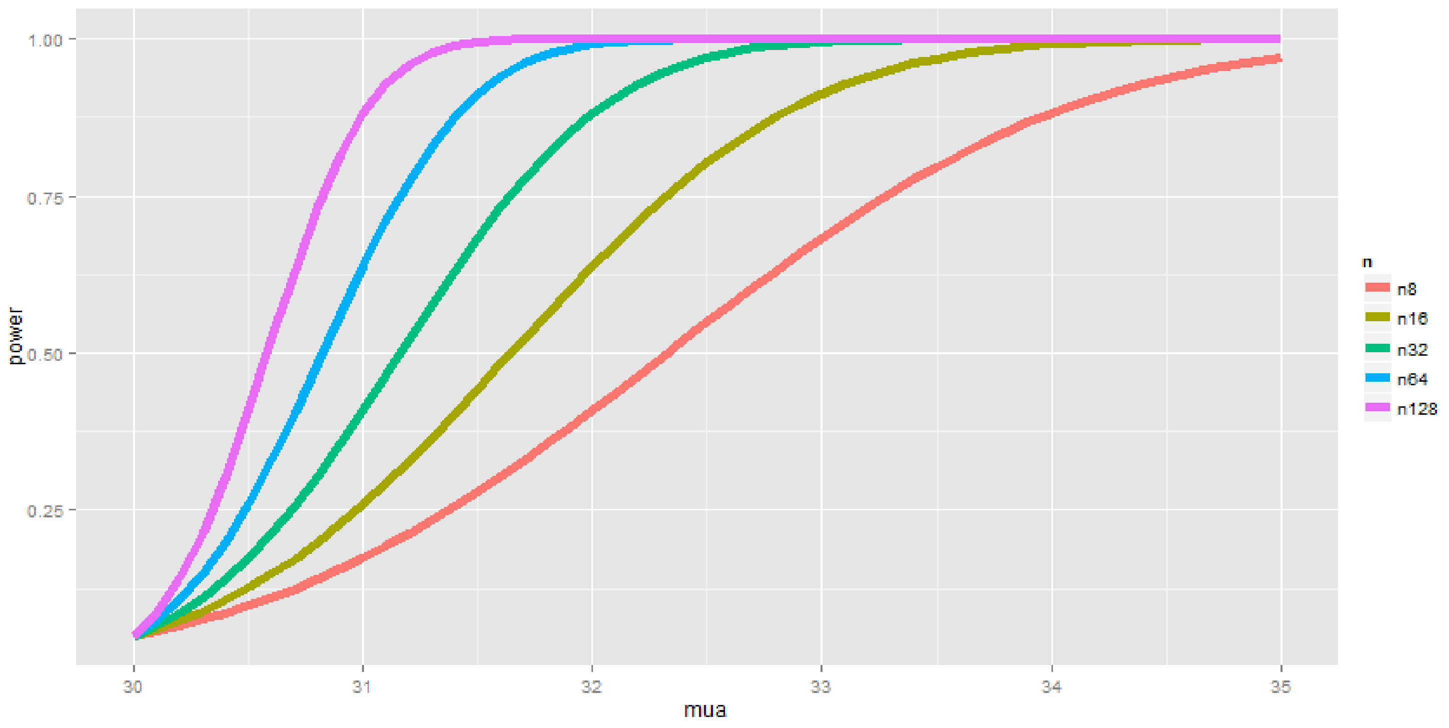
```
mu0 = 30
mua = 32
sigma = 4
n = 16
z = qnorm(1 - alpha)
pnorm(mu0 + z * sigma/sqrt(n), mean = mu0, sd = sigma/sqrt(n), lower.tail = FALSE)
```

```
## [1] 0.05
```

```
pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)
```

```
## [1] 0.6388
```

Plotting the power curve



Graphical Depiction of Power

```r
library(manipulate)
mu0 = 30
myplot <- function(sigma, mua, n, alpha) {
    g = ggplot(data.frame(mu = c(27, 36)), aes(x = mu))
    g = g + stat_function(fun = dnorm, geom = "line", args = list(mean = mu0,
        sd = sigma/sqrt(n)), size = 2, col = "red")
    g = g + stat_function(fun = dnorm, geom = "line", args = list(mean = mua,
        sd = sigma/sqrt(n)), size = 2, col = "blue")
    xitc = mu0 + qnorm(1 - alpha) * sigma/sqrt(n)
    g = g + geom_vline(xintercept = xitc, size = 3)
    g
}
manipulate(myplot(sigma, mua, n, alpha), sigma = slider(1, 10, step = 1, initial = 4),
    mua = slider(30, 35, step = 1, initial = 32), n = slider(1, 50, step = 1,
        initial = 16), alpha = slider(0.01, 0.1, step = 0.01, initial = 0.05))
```

Question

- When testing $H_a : \mu > \mu_0$, notice if power is $1 - \beta$, then

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

- 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

Notes

- The calculation for $H_a : \mu < \mu_0$ is similar
- For $H_a : \mu \neq \mu_0$ calculate the one sided power using $\alpha / 2$ (this is only approximately right, it excludes the probability of getting a large TS in the opposite direction of the truth)
- Power goes up as α gets larger
- Power of a one sided test is greater than the power of the associated two sided test
- Power goes up as μ_1 gets further away from μ_0
- Power goes up as n goes up
- Power doesn't need μ_a, σ and n, instead only $\frac{\sqrt{n}(\mu_a - \mu_0)}{\sigma}$
  - The quantity $\frac{\mu_a - \mu_0}{\sigma}$ is called the effect size, the difference in the means in standard deviation units.
  - Being unit free, it has some hope of interpretability across settings

T-test power

- Consider calculating power for a Gossett's T test for our example
- The power is

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

- Calcuting this requires the non-central t distribution.
- `power.t.test` does this very well
  - Omit one of the arguments and it solves for it

Example

```
power.t.test(n = 16, delta = 2/4, sd = 1, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

```
power.t.test(n = 16, delta = 2, sd = 4, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

```
power.t.test(n = 16, delta = 100, sd = 200, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

Example

```
power.t.test(power = 0.8, delta = 2/4, sd = 1, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

```
power.t.test(power = 0.8, delta = 2, sd = 4, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

```
power.t.test(power = 0.8, delta = 100, sd = 200, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

Power

Statistical Inference

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

Power

- Power is the probability of rejecting the null hypothesis when it is false
- Ergo, power (as its name would suggest) is a good thing; you want more power
- A type II error (a bad thing, as its name would suggest) is failing to reject the null hypothesis when it's false; the probability of a type II error is usually called β
- Note Power $= 1 - \beta$

Notes

- Consider our previous example involving RDI
- $H_0 : \mu = 30$ versus $H_a : \mu > 30$
- Then power is

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

- Note that this is a function that depends on the specific value of μ_a!
- Notice as μ_a approaches 30 the power approaches α

Calculating power for Gaussian data

- We reject if $\frac{\bar X - 30}{\sigma/\sqrt{n}} > z_{1-\alpha}$
  - Equivalently if $\bar X > 30 + Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}$
- Under $H_0 : \bar X \sim N(\mu_0, \sigma^2/n)$
- Under $H_a : \bar X \sim N(\mu_a, \sigma^2/n)$
- So we want

```
alpha = 0.05
z = qnorm(1 - alpha)
pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)
```

Example continued

- $\mu_a = 32$, $\mu_0 = 30$, $n = 16$, $\sigma = 4$

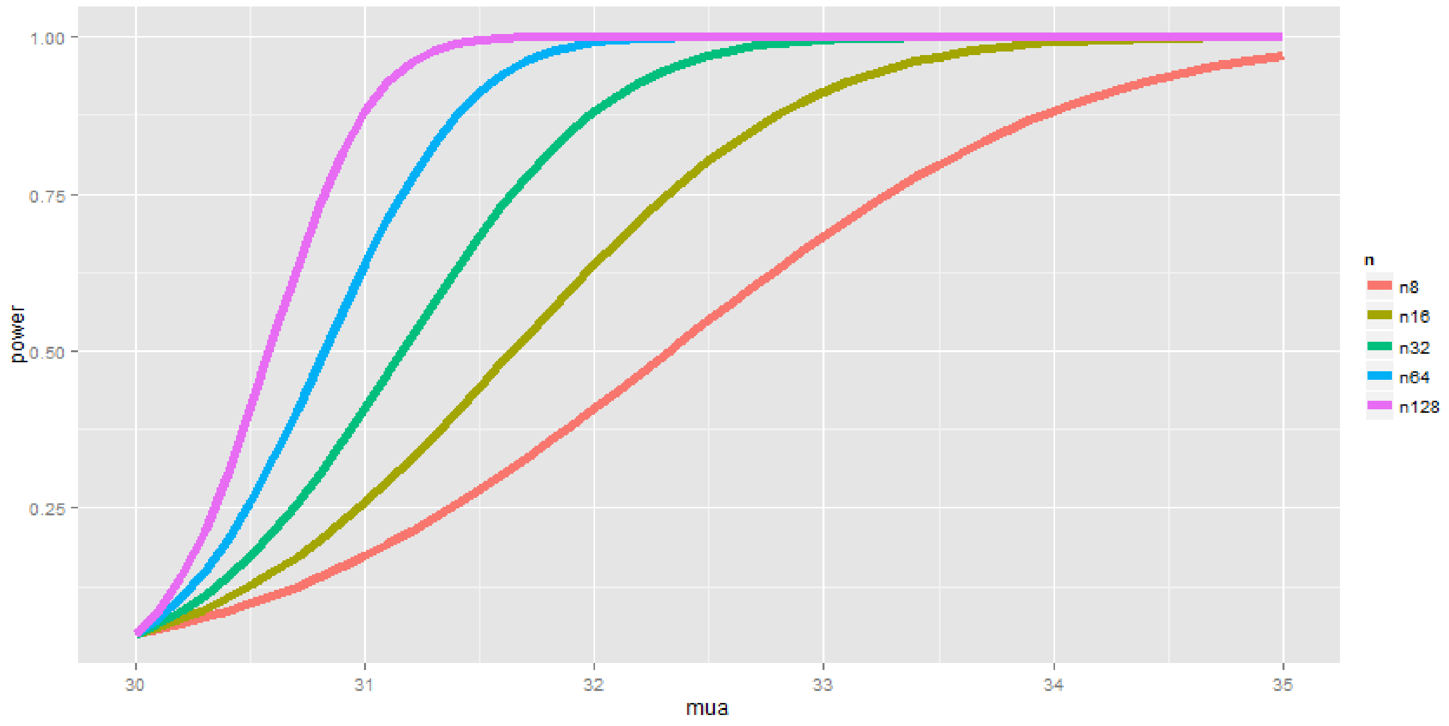
```
mu0 = 30
mua = 32
sigma = 4
n = 16
z = qnorm(1 - alpha)
pnorm(mu0 + z * sigma/sqrt(n), mean = mu0, sd = sigma/sqrt(n), lower.tail = FALSE)
```

```
## [1] 0.05
```

```
pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)
```

```
## [1] 0.6388
```

Plotting the power curve



Graphical Depiction of Power

```r
library(manipulate)
mu0 = 30
myplot <- function(sigma, mua, n, alpha) {
    g = ggplot(data.frame(mu = c(27, 36)), aes(x = mu))
    g = g + stat_function(fun = dnorm, geom = "line", args = list(mean = mu0,
        sd = sigma/sqrt(n)), size = 2, col = "red")
    g = g + stat_function(fun = dnorm, geom = "line", args = list(mean = mua,
        sd = sigma/sqrt(n)), size = 2, col = "blue")
    xitc = mu0 + qnorm(1 - alpha) * sigma/sqrt(n)
    g = g + geom_vline(xintercept = xitc, size = 3)
    g
}
manipulate(myplot(sigma, mua, n, alpha), sigma = slider(1, 10, step = 1, initial = 4),
    mua = slider(30, 35, step = 1, initial = 32), n = slider(1, 50, step = 1,
        initial = 16), alpha = slider(0.01, 0.1, step = 0.01, initial = 0.05))
```

Question

- When testing $H_a : \mu > \mu_0$, notice if power is $1 - \beta$, then

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

- 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

Notes

- The calculation for $H_a : \mu < \mu_0$ is similar
- For $H_a : \mu \neq \mu_0$ calculate the one sided power using $\alpha / 2$ (this is only approximately right, it excludes the probability of getting a large TS in the opposite direction of the truth)
- Power goes up as α gets larger
- Power of a one sided test is greater than the power of the associated two sided test
- Power goes up as μ_1 gets further away from μ_0
- Power goes up as n goes up
- Power doesn't need μ_a, σ and n, instead only $\frac{\sqrt{n}(\mu_a - \mu_0)}{\sigma}$
  - The quantity $\frac{\mu_a - \mu_0}{\sigma}$ is called the effect size, the difference in the means in standard deviation units.
  - Being unit free, it has some hope of interpretability across settings

T-test power

- Consider calculating power for a Gossett's T test for our example
- The power is

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

- Calcuting this requires the non-central t distribution.
- `power.t.test` does this very well
  - Omit one of the arguments and it solves for it

Example

```
power.t.test(n = 16, delta = 2/4, sd = 1, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

```
power.t.test(n = 16, delta = 2, sd = 4, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

```
power.t.test(n = 16, delta = 100, sd = 200, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

Example

```
power.t.test(power = 0.8, delta = 2/4, sd = 1, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

```
power.t.test(power = 0.8, delta = 2, sd = 4, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

```
power.t.test(power = 0.8, delta = 100, sd = 200, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

Power

Statistical Inference

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

Power

- Power is the probability of rejecting the null hypothesis when it is false
- Ergo, power (as its name would suggest) is a good thing; you want more power
- A type II error (a bad thing, as its name would suggest) is failing to reject the null hypothesis when it's false; the probability of a type II error is usually called β
- Note Power $= 1 - \beta$

Notes

- Consider our previous example involving RDI
- $H_0 : \mu = 30$ versus $H_a : \mu > 30$
- Then power is

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

- Note that this is a function that depends on the specific value of μ_a!
- Notice as μ_a approaches 30 the power approaches α

Calculating power for Gaussian data

- We reject if $\frac{\bar{X}-30}{\sigma/\sqrt{n}} > z_{1-\alpha}$
  - Equivalently if $\bar{X} > 30 + Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}$
- Under $H_0 : \bar{X} \sim N(\mu_0, \sigma^2/n)$
- Under $H_a : \bar{X} \sim N(\mu_a, \sigma^2/n)$
- So we want

```
alpha = 0.05
z = qnorm(1 - alpha)
pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)
```

Example continued

- $\mu_a = 32$, $\mu_0 = 30$, $n = 16$, $\sigma = 4$

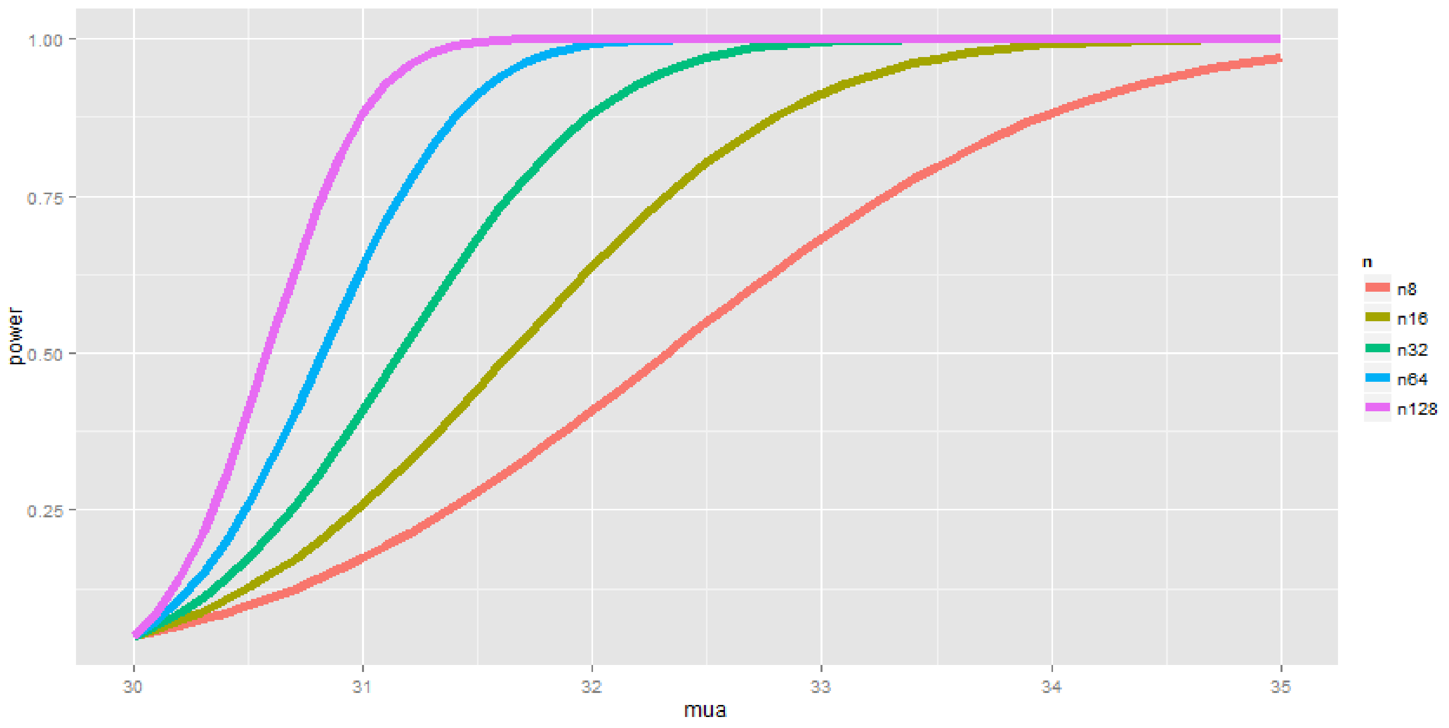
```
mu0 = 30
mua = 32
sigma = 4
n = 16
z = qnorm(1 - alpha)
pnorm(mu0 + z * sigma/sqrt(n), mean = mu0, sd = sigma/sqrt(n), lower.tail = FALSE)
```

```
## [1] 0.05
```

```
pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)
```

```
## [1] 0.6388
```

Plotting the power curve



Graphical Depiction of Power

```r
library(manipulate)
mu0 = 30
myplot <- function(sigma, mua, n, alpha) {
    g = ggplot(data.frame(mu = c(27, 36)), aes(x = mu))
    g = g + stat_function(fun = dnorm, geom = "line", args = list(mean = mu0,
        sd = sigma/sqrt(n)), size = 2, col = "red")
    g = g + stat_function(fun = dnorm, geom = "line", args = list(mean = mua,
        sd = sigma/sqrt(n)), size = 2, col = "blue")
    xitc = mu0 + qnorm(1 - alpha) * sigma/sqrt(n)
    g = g + geom_vline(xintercept = xitc, size = 3)
    g
}
manipulate(myplot(sigma, mua, n, alpha), sigma = slider(1, 10, step = 1, initial = 4),
    mua = slider(30, 35, step = 1, initial = 32), n = slider(1, 50, step = 1,
        initial = 16), alpha = slider(0.01, 0.1, step = 0.01, initial = 0.05))
```

Question

- When testing $H_a : \mu > \mu_0$, notice if power is $1 - \beta$, then

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

- 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

Notes

- The calculation for $H_a : \mu < \mu_0$ is similar
- For $H_a : \mu \neq \mu_0$ calculate the one sided power using $\alpha / 2$ (this is only approximately right, it excludes the probability of getting a large TS in the opposite direction of the truth)
- Power goes up as α gets larger
- Power of a one sided test is greater than the power of the associated two sided test
- Power goes up as μ_1 gets further away from μ_0
- Power goes up as n goes up
- Power doesn't need μ_a, σ and n, instead only $\frac{\sqrt{n}(\mu_a - \mu_0)}{\sigma}$
  - The quantity $\frac{\mu_a - \mu_0}{\sigma}$ is called the effect size, the difference in the means in standard deviation units.
  - Being unit free, it has some hope of interpretability across settings

T-test power

- Consider calculating power for a Gossett's T test for our example
- The power is

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

- Calcuting this requires the non-central t distribution.
- `power.t.test` does this very well
  - Omit one of the arguments and it solves for it

Example

```
power.t.test(n = 16, delta = 2/4, sd = 1, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

```
power.t.test(n = 16, delta = 2, sd = 4, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

```
power.t.test(n = 16, delta = 100, sd = 200, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

Example

```
power.t.test(power = 0.8, delta = 2/4, sd = 1, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

```
power.t.test(power = 0.8, delta = 2, sd = 4, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

```
power.t.test(power = 0.8, delta = 100, sd = 200, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

Power

Statistical Inference

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

Power

- Power is the probability of rejecting the null hypothesis when it is false
- Ergo, power (as its name would suggest) is a good thing; you want more power
- A type II error (a bad thing, as its name would suggest) is failing to reject the null hypothesis when it's false; the probability of a type II error is usually called β
- Note Power $= 1 - \beta$

Notes

- Consider our previous example involving RDI
- $H_0 : \mu = 30$ versus $H_a : \mu > 30$
- Then power is

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

- Note that this is a function that depends on the specific value of μ_a!
- Notice as μ_a approaches 30 the power approaches α

Calculating power for Gaussian data

- We reject if $\frac{\bar{X}-30}{\sigma/\sqrt{n}} > z_{1-\alpha}$
  - Equivalently if $\bar{X} > 30 + Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}$
- Under $H_0 : \bar{X} \sim N(\mu_0, \sigma^2/n)$
- Under $H_a : \bar{X} \sim N(\mu_a, \sigma^2/n)$
- So we want

```
alpha = 0.05
z = qnorm(1 - alpha)
pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)
```

Example continued

- $\mu_a = 32$, $\mu_0 = 30$, $n = 16$, $\sigma = 4$

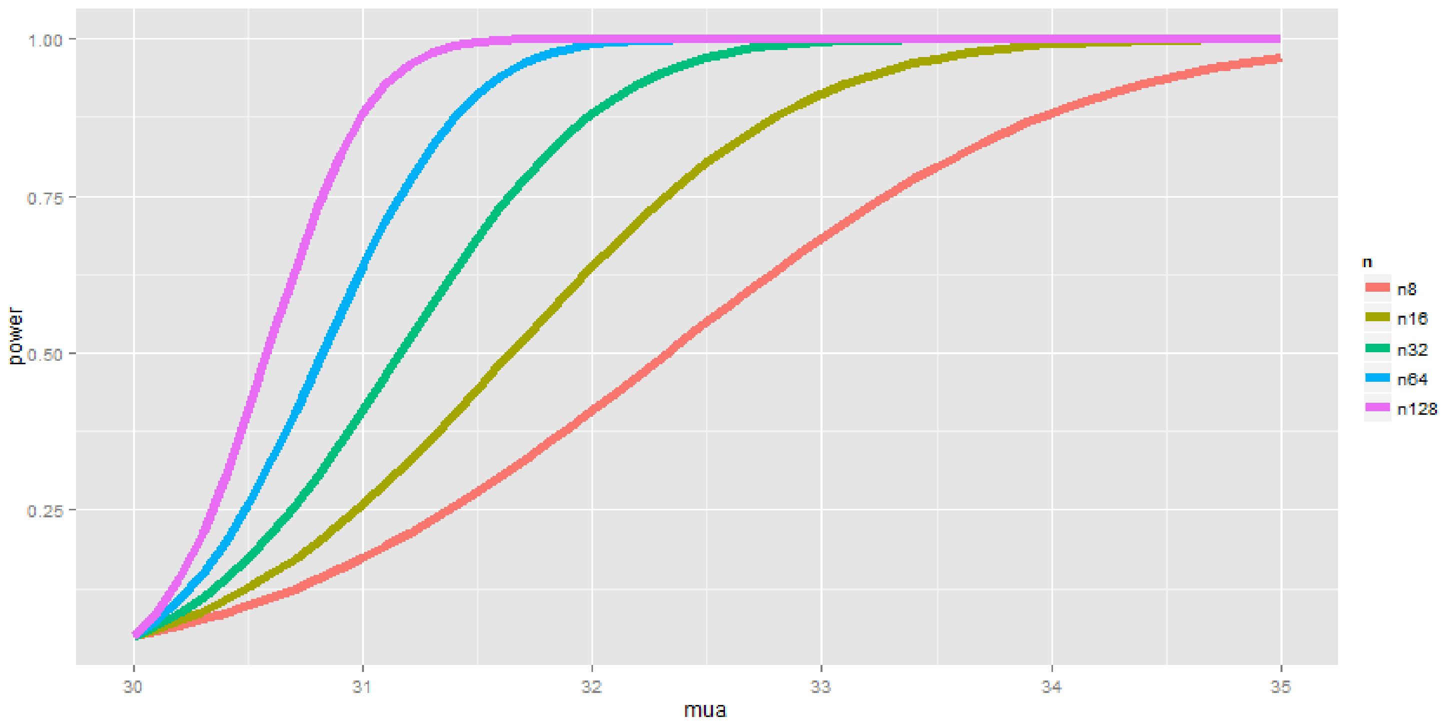
```
mu0 = 30
mua = 32
sigma = 4
n = 16
z = qnorm(1 - alpha)
pnorm(mu0 + z * sigma/sqrt(n), mean = mu0, sd = sigma/sqrt(n), lower.tail = FALSE)
```

```
## [1] 0.05
```

```
pnorm(mu0 + z * sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = FALSE)
```

```
## [1] 0.6388
```

Plotting the power curve



Graphical Depiction of Power

```
library(manipulate)
mu0 = 30
myplot <- function(sigma, mua, n, alpha) {
    g = ggplot(data.frame(mu = c(27, 36)), aes(x = mu))
    g = g + stat_function(fun = dnorm, geom = "line", args = list(mean = mu0,
        sd = sigma/sqrt(n)), size = 2, col = "red")
    g = g + stat_function(fun = dnorm, geom = "line", args = list(mean = mua,
        sd = sigma/sqrt(n)), size = 2, col = "blue")
    xitc = mu0 + qnorm(1 - alpha) * sigma/sqrt(n)
    g = g + geom_vline(xintercept = xitc, size = 3)
    g
}
manipulate(myplot(sigma, mua, n, alpha), sigma = slider(1, 10, step = 1, initial = 4),
    mua = slider(30, 35, step = 1, initial = 32), n = slider(1, 50, step = 1,
        initial = 16), alpha = slider(0.01, 0.1, step = 0.01, initial = 0.05))
```

Question

- When testing $H_a : \mu > \mu_0$, notice if power is $1 - \beta$, then

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

- 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

Notes

- The calculation for $H_a : \mu < \mu_0$ is similar
- For $H_a : \mu \neq \mu_0$ calculate the one sided power using $\alpha / 2$ (this is only approximately right, it excludes the probability of getting a large TS in the opposite direction of the truth)
- Power goes up as α gets larger
- Power of a one sided test is greater than the power of the associated two sided test
- Power goes up as μ_1 gets further away from μ_0
- Power goes up as n goes up
- Power doesn't need μ_a, σ and n, instead only $\frac{\sqrt{n}(\mu_a - \mu_0)}{\sigma}$
  - The quantity $\frac{\mu_a - \mu_0}{\sigma}$ is called the effect size, the difference in the means in standard deviation units.
  - Being unit free, it has some hope of interpretability across settings

T-test power

- Consider calculating power for a Gossett's T test for our example
- The power is

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

- Calcuting this requires the non-central t distribution.
- `power.t.test` does this very well
  - Omit one of the arguments and it solves for it

Example

```
power.t.test(n = 16, delta = 2/4, sd = 1, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

```
power.t.test(n = 16, delta = 2, sd = 4, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

```
power.t.test(n = 16, delta = 100, sd = 200, type = "one.sample", alt = "one.sided")$power
```

```
## [1] 0.604
```

Example

```r
power.t.test(power = 0.8, delta = 2/4, sd = 1, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

```r
power.t.test(power = 0.8, delta = 2, sd = 4, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

```r
power.t.test(power = 0.8, delta = 100, sd = 200, type = "one.sample", alt = "one.sided")$n
```

```
## [1] 26.14
```

Multiple testing

Statistical Inference

Brian Caffo, Jeffrey Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

Key ideas

- Hypothesis testing/significance analysis is commonly overused
- Correcting for multiple testing avoids false positives or discoveries
- Two key components
    - Error measure
    - Correction

Three eras of statistics

The age of Quetelet and his successors, in which huge census-level data sets were brought to bear on simple but important questions: Are there more male than female births? Is the rate of insanity rising?

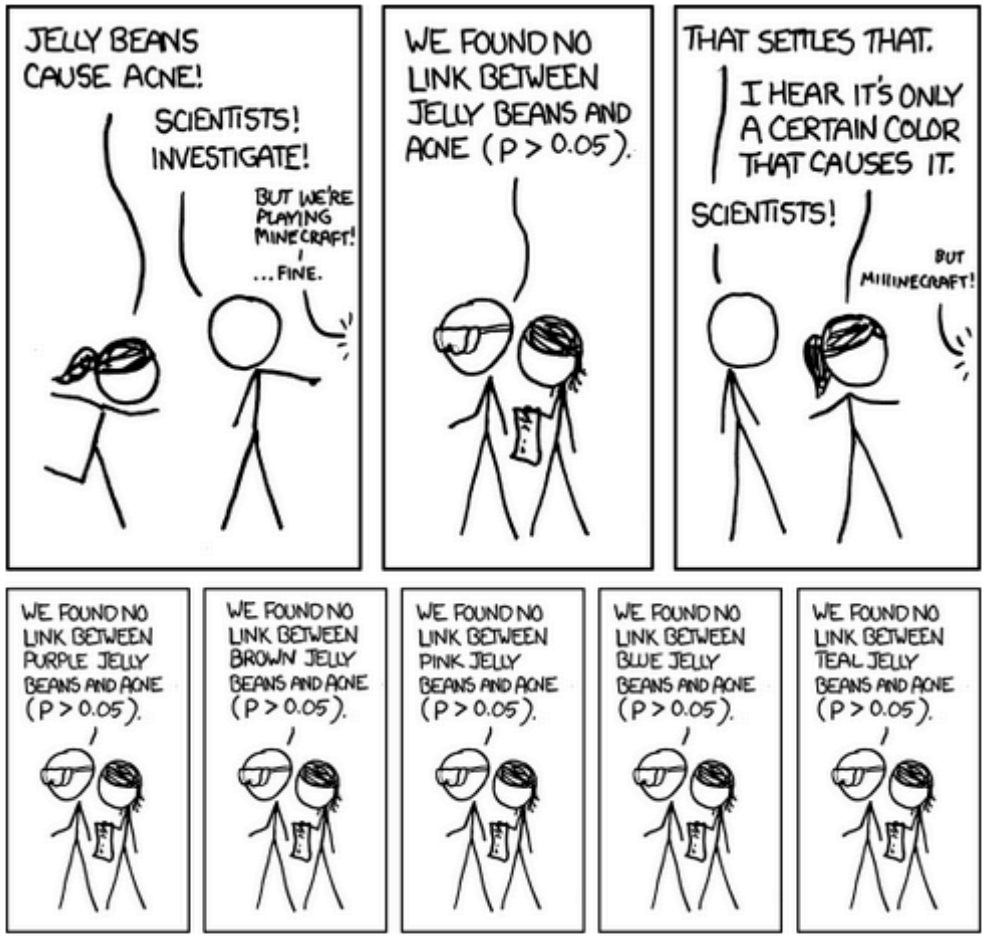
The classical period of Pearson, Fisher, Neyman, Hotelling, and their successors, intellectual giants who **developed a theory of optimal inference capable of wringing every drop of information out of a scientific experiment**. The questions dealt with still tended to be simple Is treatment A better than treatment B?

The era of scientific mass production, in which new technologies typified by the microarray allow a single team of scientists to produce data sets of a size Quetelet would envy. But now the flood of data is accompanied by a deluge of questions, perhaps thousands of estimates or hypothesis tests that the statistician is charged with answering together; not at all what the classical masters had in mind. Which variables matter among the thousands measured? How do you relate unrelated information?

http://www-stat.stanford.edu/~ckirby/brad/papers/2010LSIexcerpt.pdf

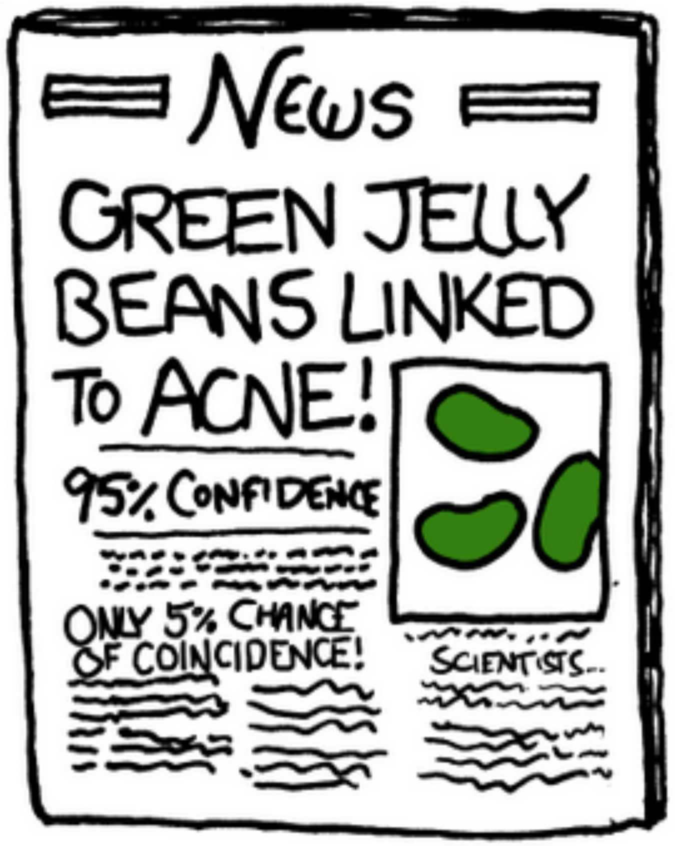
Reasons for multiple testing

Why correct for multiple tests?


JELLY BEANS CAUSE ACNE!
SCIENTISTS! INVESTIGATE!
BUT WE'RE PLAYING MINECRAFT!
... FINE.
WE FOUND NO LINK BETWEEN JELLY BEANS AND ACNE (P > 0.05).
THAT SETTLES THAT.
I HEAR IT'S ONLY A CERTAIN COLOR THAT CAUSES IT.
SCIENTISTS!
BUT MIIINECRAFT!
WE FOUND NO LINK BETWEEN PURPLE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BROWN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PINK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLUE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TEAL JELLY BEANS AND ACNE (P > 0.05).

http://xkcd.com/882/

Why correct for multiple tests?


NEWS
GREEN JELLY BEANS LINKED TO ACNE!
95% CONFIDENCE
ONLY 5% CHANCE OF COINCIDENCE!
SCIENTISTS...

http://xkcd.com/882/

Types of errors

Suppose you are testing a hypothesis that a parameter β equals zero versus the alternative that it does not equal zero. These are the possible outcomes.

	$\beta = 0$	$\beta \neq 0$	HYPOTHESES
Claim $\beta = 0$	U	T	$m - R$
Claim $\beta \neq 0$	V	S	R
Claims	m_0	$m - m_0$	m

Type I error or false positive (V) Say that the parameter does not equal zero when it does

Type II error or false negative (T) Say that the parameter equals zero when it doesn't

Error rates

False positive rate - The rate at which false results ($\beta = 0$) are called significant: $E\left[\frac{V}{m_0}\right]$*

Family wise error rate (FWER) - The probability of at least one false positive $\Pr(V \geq 1)$

False discovery rate (FDR) - The rate at which claims of significance are false $E\left[\frac{V}{R}\right]$

- The false positive rate is closely related to the type I error rate http://en.wikipedia.org/wiki/False_positive_rate

Controlling the false positive rate

If P-values are correctly calculated calling all $P < \alpha$ significant will control the false positive rate at level α on average.

Problem: Suppose that you perform 10,000 tests and $\beta = 0$ for all of them.

Suppose that you call all $P < 0.05$ significant.

The expected number of false positives is: $10,000 \times 0.05 = 500$ false positives.

How do we avoid so many false positives?

Controlling family-wise error rate (FWER)

The Bonferroni correction is the oldest multiple testing correction.

Basic idea:

- Suppose you do m tests
- You want to control FWER at level α so $Pr(V \geq 1) < \alpha$
- Calculate P-values normally
- Set $\alpha_{fwer} = \alpha/m$
- Call all P-values less than α_{fwer} significant

Pros: Easy to calculate, conservative **Cons**: May be very conservative

Controlling false discovery rate (FDR)

This is the most popular correction when performing *lots* of tests say in genomics, imaging, astronomy, or other signal-processing disciplines.

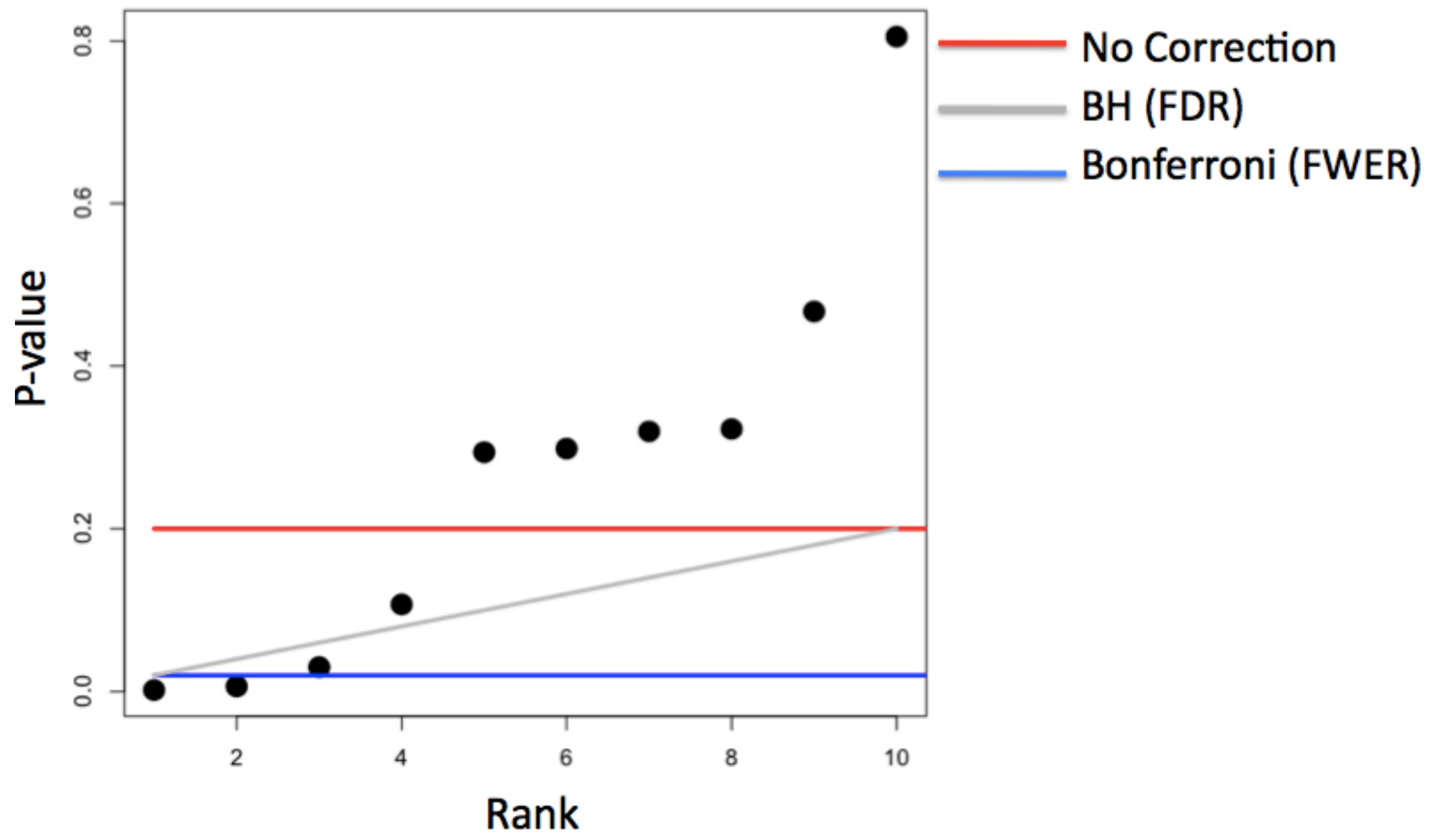
Basic idea:

- Suppose you do m tests
- You want to control FDR at level α so $E\left[\frac{V}{R}\right]$
- Calculate P-values normally
- Order the P-values from smallest to largest $P_{(1)}, \ldots, P_{(m)}$
- Call any $P_{(i)} \leq \alpha \times \frac{i}{m}$ significant

Pros: Still pretty easy to calculate, less conservative (maybe much less)

Cons: Allows for more false positives, may behave strangely under dependence

Example with 10 P-values



Controlling all error rates at $\alpha = 0.20$

Adjusted P-values

- One approach is to adjust the threshold α
- A different approach is to calculate "adjusted p-values"
- They *are not p-values* anymore
- But they can be used directly without adjusting α

Example:

- Suppose P-values are $P_1, \ldots, P_m$
- You could adjust them by taking $P_i^{fwer} = \max m \times P_i, 1$ for each P-value.
- Then if you call all $P_i^{fwer} < \alpha$ significant you will control the FWER.

Case study I: no true positives

```r
set.seed(1010093)
pValues <- rep(NA, 1000)
for (i in 1:1000) {
    y <- rnorm(20)
    x <- rnorm(20)
    pValues[i] <- summary(lm(y ~ x))$coeff[2, 4]
}

# Controls false positive rate
sum(pValues < 0.05)
```

```
## [1] 51
```

Case study I: no true positives

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

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

```
# Controls FDR
sum(p.adjust(pValues, method = "BH") < 0.05)
```

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

Case study II: 50% true positives

```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)
    }
    pValues[i] <- summary(lm(y ~ x))$coeff[2, 4]
}
trueStatus <- rep(c("zero", "not zero"), each = 500)
table(pValues < 0.05, trueStatus)
```

```
##          trueStatus
##           not zero zero
##     FALSE        0  476
```

Case study II: 50% true positives

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

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

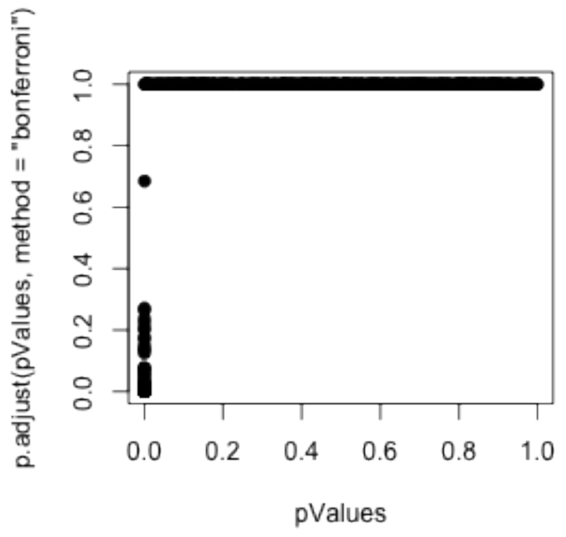
```
# Controls FDR
table(p.adjust(pValues, method = "BH") < 0.05, trueStatus)
```

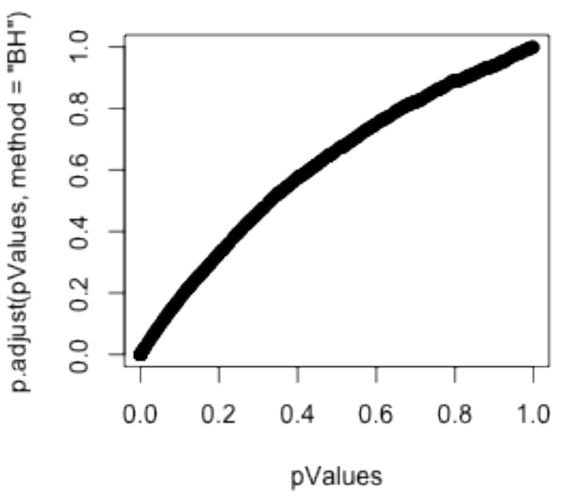
```
##          trueStatus
##           not zero zero
##     FALSE        0  487
##     TRUE       500   13
```

Case study II: 50% true positives

P-values versus adjusted P-values

```r
par(mfrow = c(1, 2))
plot(pValues, p.adjust(pValues, method = "bonferroni"), pch = 19)
plot(pValues, p.adjust(pValues, method = "BH"), pch = 19)
```





Notes and resources

Notes:

- Multiple testing is an entire subfield
- A basic Bonferroni/BH correction is usually enough
- If there is strong dependence between tests there may be problems
    - Consider method="BY"

Further resources:

- Multiple testing procedures with applications to genomics
- Statistical significance for genome-wide studies
- Introduction to multiple testing

Resampled inference

Statistical Inference

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

The jackknife

- The jackknife is a tool for estimating standard errors and the bias of estimators
- As its name suggests, the jackknife is a small, handy tool; in contrast to the bootstrap, which is then the moral equivalent of a giant workshop full of tools
- Both the jackknife and the bootstrap involve *resampling* data; that is, repeatedly creating new data sets from the original data

The jackknife

- The jackknife deletes each observation and calculates an estimate based on the remaining $n-1$ of them
- It uses this collection of estimates to do things like estimate the bias and the standard error
- Note that estimating the bias and having a standard error are not needed for things like sample means, which we know are unbiased estimates of population means and what their standard errors are

The jackknife

- We'll consider the jackknife for univariate data
- Let $X_1, \ldots, X_n$ be a collection of data used to estimate a parameter θ
- Let $\hat{\theta}$ be the estimate based on the full data set
- Let $\hat{\theta}_i$ be the estimate of θ obtained by *deleting observation* i
- Let $\bar{\theta} = \frac{1}{n}\sum_{i=1}^n \hat{\theta}_i$

Continued

- Then, the jackknife estimate of the bias is

$$(n-1)\left(\bar{\theta} - \hat{\theta}\right)$$

  (how far the average delete-one estimate is from the actual estimate)

- The jackknife estimate of the standard error is

$$\left[\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_i - \bar{\theta})^2\right]^{1/2}$$

  (the deviance of the delete-one estimates from the average delete-one estimate)

Example

We want to estimate the bias and standard error of the median

```r
library(UsingR)
data(father.son)
x <- father.son$sheight
n <- length(x)
theta <- median(x)
jk <- sapply(1:n, function(i) median(x[-i]))
thetaBar <- mean(jk)
biasEst <- (n - 1) * (thetaBar - theta)
seEst <- sqrt((n - 1) * mean((jk - thetaBar)^2))
```

Example test

```
c(biasEst, seEst)
```

```
## [1] 0.0000 0.1014
```

```
library(bootstrap)
temp <- jackknife(x, median)
c(temp$jack.bias, temp$jack.se)
```

```
## [1] 0.0000 0.1014
```

Example

- Both methods (of course) yield an estimated bias of 0 and a se of 0.1014
- Odd little fact: the jackknife estimate of the bias for the median is always 0 when the number of observations is even
- It has been shown that the jackknife is a linear approximation to the bootstrap
- Generally do not use the jackknife for sample quantiles like the median; as it has been shown to have some poor properties

Pseudo observations

- Another interesting way to think about the jackknife uses pseudo observations
- Let

$$\text{Pseudo Obs} = n\hat{\theta} - (n-1)\hat{\theta}_i$$

- Think of these as ``whatever observation i contributes to the estimate of θ''
- Note when $\hat{\theta}$ is the sample mean, the pseudo observations are the data themselves
- Then the sample standard error of these observations is the previous jackknife estimated standard error.
- The mean of these observations is a bias-corrected estimate of θ

The bootstrap

- The bootstrap is a tremendously useful tool for constructing confidence intervals and calculating standard errors for difficult statistics
- For example, how would one derive a confidence interval for the median?
- The bootstrap procedure follows from the so called bootstrap principle

The bootstrap principle

- Suppose that I have a statistic that estimates some population parameter, but I don't know its sampling distribution
- The bootstrap principle suggests using the distribution defined by the data to approximate its sampling distribution

The bootstrap in practice

- In practice, the bootstrap principle is always carried out using simulation
- We will cover only a few aspects of bootstrap resampling
- The general procedure follows by first simulating complete data sets from the observed data with replacement
    - This is approximately drawing from the sampling distribution of that statistic, at least as far as the data is able to approximate the true population distribution
- Calculate the statistic for each simulated data set
- Use the simulated statistics to either define a confidence interval or take the standard deviation to calculate a standard error

Nonparametric bootstrap algorithm example

- Bootstrap procedure for calculating confidence interval for the median from a data set of n observations

  i. Sample n observations **with replacement** from the observed data resulting in one simulated complete data set

  ii. Take the median of the simulated data set

  iii. Repeat these two steps B times, resulting in B simulated medians

  iv. These medians are approximately drawn from the sampling distribution of the median of n observations; therefore we can

    - Draw a histogram of them
    - Calculate their standard deviation to estimate the standard error of the median
    - Take the 2.5^{th} and 97.5^{th} percentiles as a confidence interval for the median

Example code

```
B <- 1000
resamples <- matrix(sample(x, n * B, replace = TRUE), B, n)
medians <- apply(resamples, 1, median)
sd(medians)
```

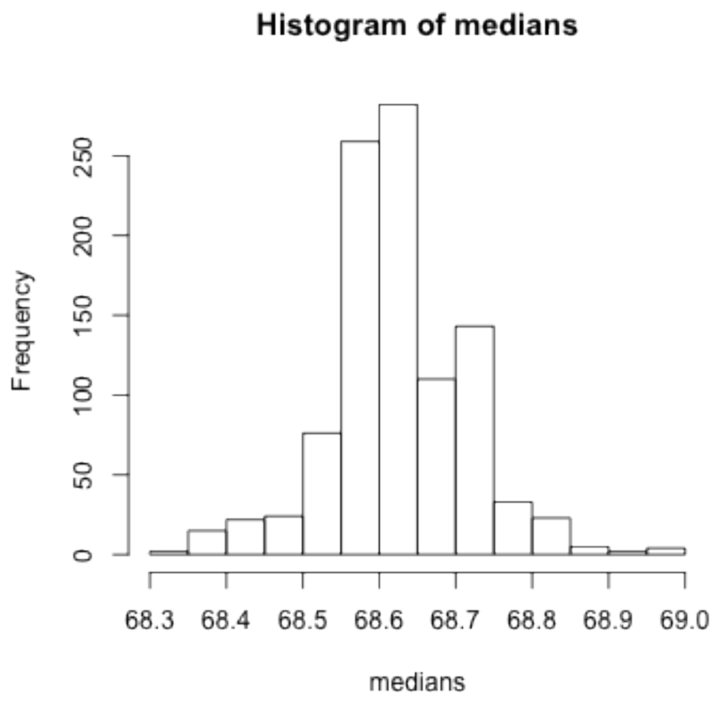
```
## [1] 0.08834
```

```
quantile(medians, c(0.025, 0.975))
```

```
##  2.5% 97.5%
## 68.41 68.82
```

Histogram of bootstrap resamples

```
hist(medians)
```



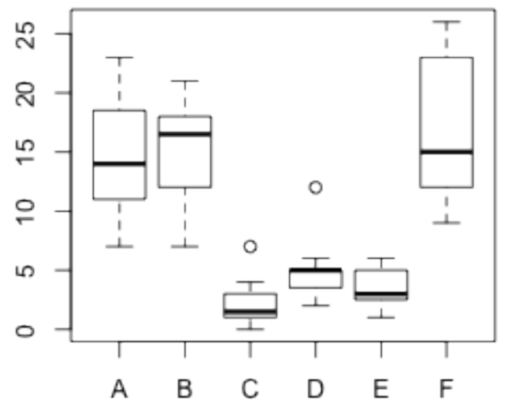
Notes on the bootstrap

- The bootstrap is non-parametric
- Better percentile bootstrap confidence intervals correct for bias
- There are lots of variations on bootstrap procedures; the book "An Introduction to the Bootstrap"" by Efron and Tibshirani is a great place to start for both bootstrap and jackknife information

Group comparisons

- Consider comparing two independent groups.
- Example, comparing sprays B and C

```
data(InsectSprays)
boxplot(count ~ spray, data = InsectSprays)
```



Permutation tests

- Consider the null hypothesis that the distribution of the observations from each group is the same
- Then, the group labels are irrelevant
- We then discard the group levels and permute the combined data
- Split the permuted data into two groups with n_A and n_B observations (say by always treating the first n_A observations as the first group)
- Evaluate the probability of getting a statistic as large or large than the one observed
- An example statistic would be the difference in the averages between the two groups; one could also use a t-statistic

Variations on permutation testing

DATA TYPE	STATISTIC	TEST NAME
Ranks	rank sum	rank sum test
Binary	hypergeometric prob	Fisher's exact test
Raw data		ordinary permutation test

- Also, so-called *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 as well
    - Permuting a regressor of interest
- Permutation tests work very well in multivariate settings

Permutation test for pesticide data

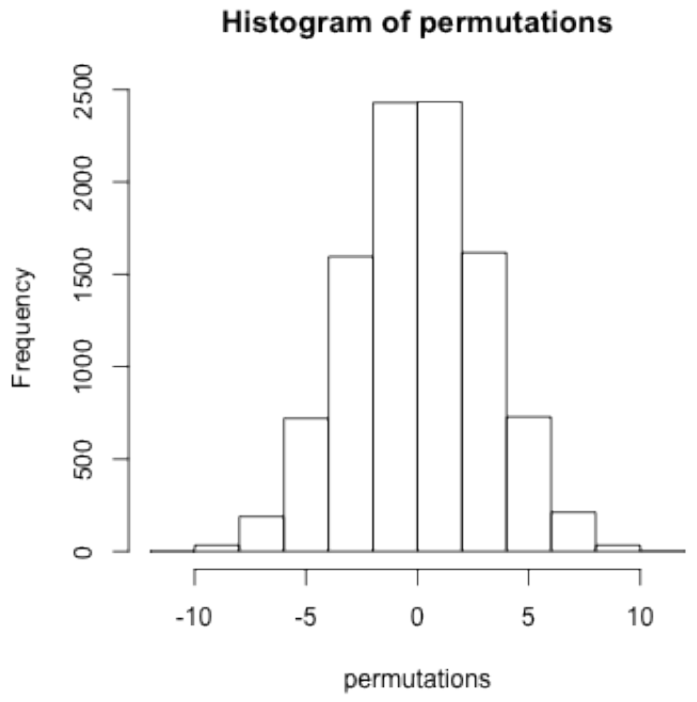
```r
subdata <- InsectSprays[InsectSprays$spray %in% c("B", "C"), ]
y <- subdata$count
group <- as.character(subdata$spray)
testStat <- function(w, g) mean(w[g == "B"]) - mean(w[g == "C"])
observedStat <- testStat(y, group)
permutations <- sapply(1:10000, function(i) testStat(y, sample(group)))
observedStat
```

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

```r
mean(permutations > observedStat)
```

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

Histogram of permutations



Resampled inference

Statistical Inference

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

The jackknife

- The jackknife is a tool for estimating standard errors and the bias of estimators
- As its name suggests, the jackknife is a small, handy tool; in contrast to the bootstrap, which is then the moral equivalent of a giant workshop full of tools
- Both the jackknife and the bootstrap involve *resampling* data; that is, repeatedly creating new data sets from the original data

The jackknife

- The jackknife deletes each observation and calculates an estimate based on the remaining $n-1$ of them
- It uses this collection of estimates to do things like estimate the bias and the standard error
- Note that estimating the bias and having a standard error are not needed for things like sample means, which we know are unbiased estimates of population means and what their standard errors are

The jackknife

- We'll consider the jackknife for univariate data
- Let $X_1, \ldots, X_n$ be a collection of data used to estimate a parameter θ
- Let $\hat{\theta}$ be the estimate based on the full data set
- Let $\hat{\theta}_i$ be the estimate of θ obtained by *deleting observation* i
- Let $\bar{\theta} = \frac{1}{n}\sum_{i=1}^n \hat{\theta}_i$

Continued

- Then, the jackknife estimate of the bias is

$$(n-1)\left(\bar{\theta}-\hat{\theta}\right)$$

(how far the average delete-one estimate is from the actual estimate)

- The jackknife estimate of the standard error is

$$\left[\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_i-\bar{\theta})^2\right]^{1/2}$$

(the deviance of the delete-one estimates from the average delete-one estimate)

Example

We want to estimate the bias and standard error of the median

```
library(UsingR)
data(father.son)
x <- father.son$sheight
n <- length(x)
theta <- median(x)
jk <- sapply(1:n, function(i) median(x[-i]))
thetaBar <- mean(jk)
biasEst <- (n - 1) * (thetaBar - theta)
seEst <- sqrt((n - 1) * mean((jk - thetaBar)^2))
```

Example test

```
c(biasEst, seEst)
```

```
## [1] 0.0000 0.1014
```

```
library(bootstrap)
temp <- jackknife(x, median)
c(temp$jack.bias, temp$jack.se)
```

```
## [1] 0.0000 0.1014
```

Example

- Both methods (of course) yield an estimated bias of 0 and a se of 0.1014
- Odd little fact: the jackknife estimate of the bias for the median is always 0 when the number of observations is even
- It has been shown that the jackknife is a linear approximation to the bootstrap
- Generally do not use the jackknife for sample quantiles like the median; as it has been shown to have some poor properties

Pseudo observations

- Another interesting way to think about the jackknife uses pseudo observations
- Let

$$\text{Pseudo Obs} = n\hat{\theta} - (n-1)\hat{\theta}_i$$

- Think of these as ``whatever observation i contributes to the estimate of θ''
- Note when $\hat{\theta}$ is the sample mean, the pseudo observations are the data themselves
- Then the sample standard error of these observations is the previous jackknife estimated standard error.
- The mean of these observations is a bias-corrected estimate of θ

The bootstrap

- The bootstrap is a tremendously useful tool for constructing confidence intervals and calculating standard errors for difficult statistics
- For example, how would one derive a confidence interval for the median?
- The bootstrap procedure follows from the so called bootstrap principle

The bootstrap principle

- Suppose that I have a statistic that estimates some population parameter, but I don't know its sampling distribution
- The bootstrap principle suggests using the distribution defined by the data to approximate its sampling distribution

The bootstrap in practice

- In practice, the bootstrap principle is always carried out using simulation
- We will cover only a few aspects of bootstrap resampling
- The general procedure follows by first simulating complete data sets from the observed data with replacement
    - This is approximately drawing from the sampling distribution of that statistic, at least as far as the data is able to approximate the true population distribution
- Calculate the statistic for each simulated data set
- Use the simulated statistics to either define a confidence interval or take the standard deviation to calculate a standard error

Nonparametric bootstrap algorithm example

- Bootstrap procedure for calculating confidence interval for the median from a data set of n observations

  i. Sample n observations **with replacement** from the observed data resulting in one simulated complete data set

  ii. Take the median of the simulated data set

  iii. Repeat these two steps B times, resulting in B simulated medians

  iv. These medians are approximately drawn from the sampling distribution of the median of n observations; therefore we can

    - Draw a histogram of them
    - Calculate their standard deviation to estimate the standard error of the median
    - Take the 2.5^{th} and 97.5^{th} percentiles as a confidence interval for the median

Example code

```r
B <- 1000
resamples <- matrix(sample(x, n * B, replace = TRUE), B, n)
medians <- apply(resamples, 1, median)
sd(medians)
```

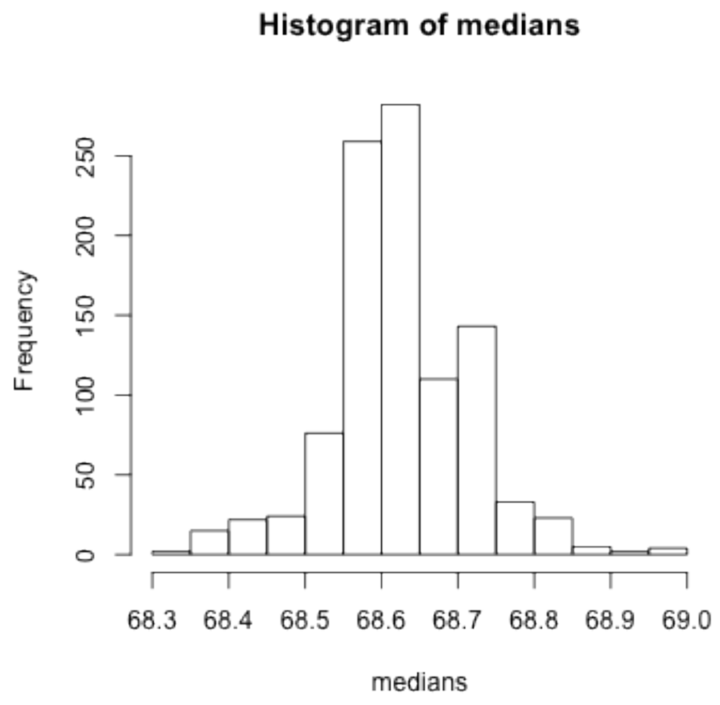
```
## [1] 0.08834
```

```r
quantile(medians, c(0.025, 0.975))
```

```
##  2.5% 97.5%
## 68.41 68.82
```

Histogram of bootstrap resamples

```
hist(medians)
```



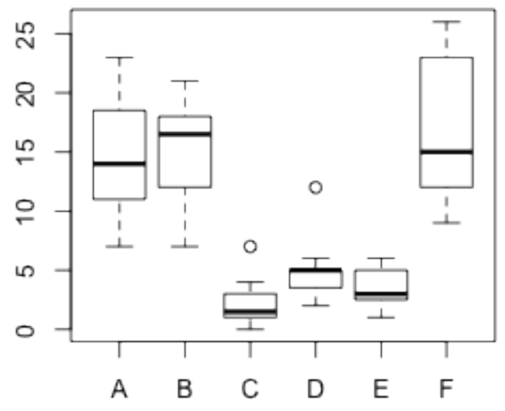
Notes on the bootstrap

- The bootstrap is non-parametric
- Better percentile bootstrap confidence intervals correct for bias
- There are lots of variations on bootstrap procedures; the book "An Introduction to the Bootstrap"" by Efron and Tibshirani is a great place to start for both bootstrap and jackknife information

Group comparisons

- Consider comparing two independent groups.
- Example, comparing sprays B and C

```
data(InsectSprays)
boxplot(count ~ spray, data = InsectSprays)
```



Permutation tests

- Consider the null hypothesis that the distribution of the observations from each group is the same
- Then, the group labels are irrelevant
- We then discard the group levels and permute the combined data
- Split the permuted data into two groups with n_A and n_B observations (say by always treating the first n_A observations as the first group)
- Evaluate the probability of getting a statistic as large or large than the one observed
- An example statistic would be the difference in the averages between the two groups; one could also use a t-statistic

Variations on permutation testing

DATA TYPE	STATISTIC	TEST NAME
Ranks	rank sum	rank sum test
Binary	hypergeometric prob	Fisher's exact test
Raw data		ordinary permutation test

- Also, so-called *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 as well
    - Permuting a regressor of interest
- Permutation tests work very well in multivariate settings

Permutation test for pesticide data

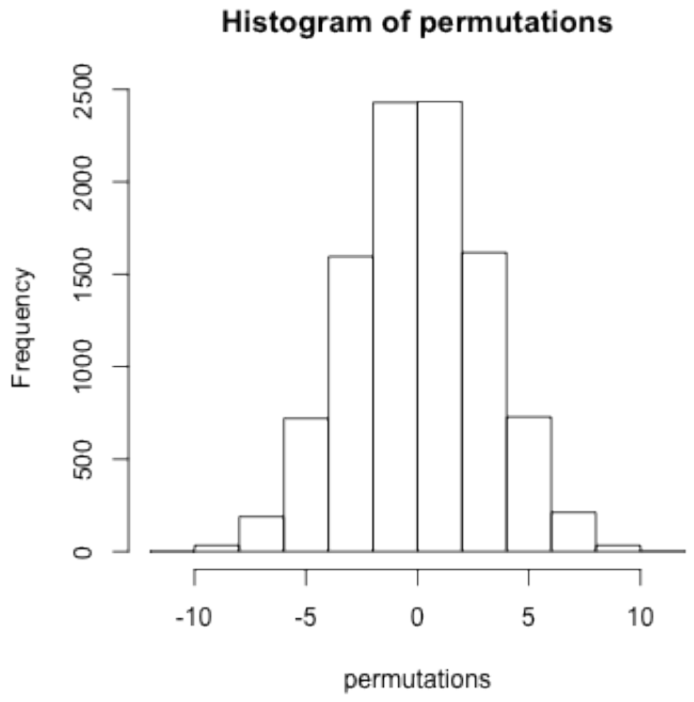
```r
subdata <- InsectSprays[InsectSprays$spray %in% c("B", "C"), ]
y <- subdata$count
group <- as.character(subdata$spray)
testStat <- function(w, g) mean(w[g == "B"]) - mean(w[g == "C"])
observedStat <- testStat(y, group)
permutations <- sapply(1:10000, function(i) testStat(y, sample(group)))
observedStat
```

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

```r
mean(permutations > observedStat)
```

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

Histogram of permutations



Resampled inference

Statistical Inference

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

The jackknife

- The jackknife is a tool for estimating standard errors and the bias of estimators
- As its name suggests, the jackknife is a small, handy tool; in contrast to the bootstrap, which is then the moral equivalent of a giant workshop full of tools
- Both the jackknife and the bootstrap involve *resampling* data; that is, repeatedly creating new data sets from the original data

The jackknife

- The jackknife deletes each observation and calculates an estimate based on the remaining $n-1$ of them
- It uses this collection of estimates to do things like estimate the bias and the standard error
- Note that estimating the bias and having a standard error are not needed for things like sample means, which we know are unbiased estimates of population means and what their standard errors are

The jackknife

- We'll consider the jackknife for univariate data
- Let $X_1, \ldots, X_n$ be a collection of data used to estimate a parameter θ
- Let $\hat{\theta}$ be the estimate based on the full data set
- Let $\hat{\theta}_i$ be the estimate of θ obtained by *deleting observation* i
- Let $\bar{\theta} = \frac{1}{n}\sum_{i=1}^n \hat{\theta}_i$

Continued

- Then, the jackknife estimate of the bias is

$$(n-1)\left(\bar{\theta} - \hat{\theta}\right)$$

(how far the average delete-one estimate is from the actual estimate)

- The jackknife estimate of the standard error is

$$\left[\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_i - \bar{\theta})^2\right]^{1/2}$$

(the deviance of the delete-one estimates from the average delete-one estimate)

Example

We want to estimate the bias and standard error of the median

```
library(UsingR)
data(father.son)
x <- father.son$sheight
n <- length(x)
theta <- median(x)
jk <- sapply(1:n, function(i) median(x[-i]))
thetaBar <- mean(jk)
biasEst <- (n - 1) * (thetaBar - theta)
seEst <- sqrt((n - 1) * mean((jk - thetaBar)^2))
```

Example test

```
c(biasEst, seEst)
```

```
## [1] 0.0000 0.1014
```

```
library(bootstrap)
temp <- jackknife(x, median)
c(temp$jack.bias, temp$jack.se)
```

```
## [1] 0.0000 0.1014
```

Example

- Both methods (of course) yield an estimated bias of 0 and a se of 0.1014
- Odd little fact: the jackknife estimate of the bias for the median is always 0 when the number of observations is even
- It has been shown that the jackknife is a linear approximation to the bootstrap
- Generally do not use the jackknife for sample quantiles like the median; as it has been shown to have some poor properties

Pseudo observations

- Another interesting way to think about the jackknife uses pseudo observations
- Let

$$\text{Pseudo Obs} = n\hat{\theta} - (n-1)\hat{\theta}_i$$

- Think of these as ``whatever observation i contributes to the estimate of θ''
- Note when $\hat{\theta}$ is the sample mean, the pseudo observations are the data themselves
- Then the sample standard error of these observations is the previous jackknife estimated standard error.
- The mean of these observations is a bias-corrected estimate of θ

The bootstrap

- The bootstrap is a tremendously useful tool for constructing confidence intervals and calculating standard errors for difficult statistics
- For example, how would one derive a confidence interval for the median?
- The bootstrap procedure follows from the so called bootstrap principle

The bootstrap principle

- Suppose that I have a statistic that estimates some population parameter, but I don't know its sampling distribution
- The bootstrap principle suggests using the distribution defined by the data to approximate its sampling distribution

The bootstrap in practice

- In practice, the bootstrap principle is always carried out using simulation
- We will cover only a few aspects of bootstrap resampling
- The general procedure follows by first simulating complete data sets from the observed data with replacement
  - This is approximately drawing from the sampling distribution of that statistic, at least as far as the data is able to approximate the true population distribution
- Calculate the statistic for each simulated data set
- Use the simulated statistics to either define a confidence interval or take the standard deviation to calculate a standard error

Nonparametric bootstrap algorithm example

- Bootstrap procedure for calculating confidence interval for the median from a data set of n observations

  i. Sample n observations **with replacement** from the observed data resulting in one simulated complete data set

  ii. Take the median of the simulated data set

  iii. Repeat these two steps B times, resulting in B simulated medians

  iv. These medians are approximately drawn from the sampling distribution of the median of n observations; therefore we can

    - Draw a histogram of them
    - Calculate their standard deviation to estimate the standard error of the median
    - Take the 2.5^{th} and 97.5^{th} percentiles as a confidence interval for the median

Example code

```r
B <- 1000
resamples <- matrix(sample(x, n * B, replace = TRUE), B, n)
medians <- apply(resamples, 1, median)
sd(medians)
```

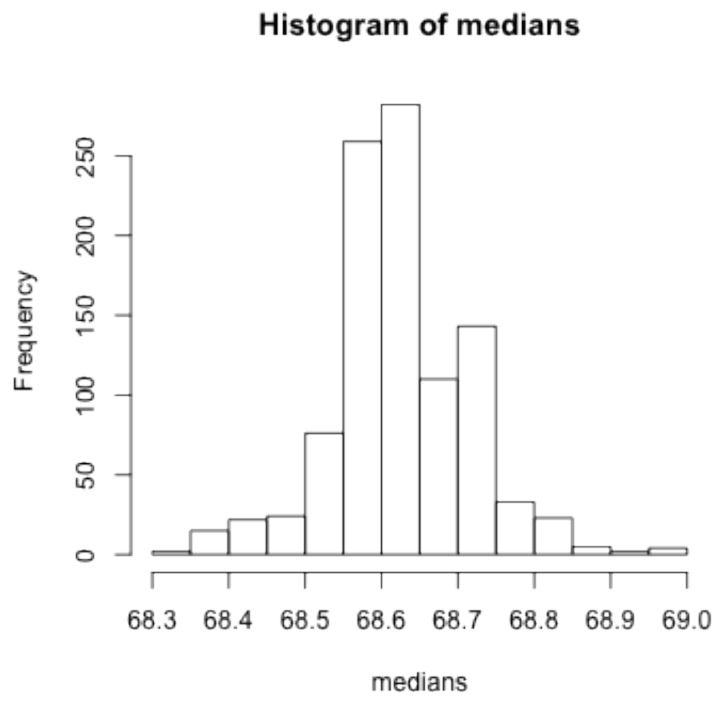
```
## [1] 0.08834
```

```r
quantile(medians, c(0.025, 0.975))
```

```
##  2.5% 97.5%
## 68.41 68.82
```

Histogram of bootstrap resamples

```
hist(medians)
```



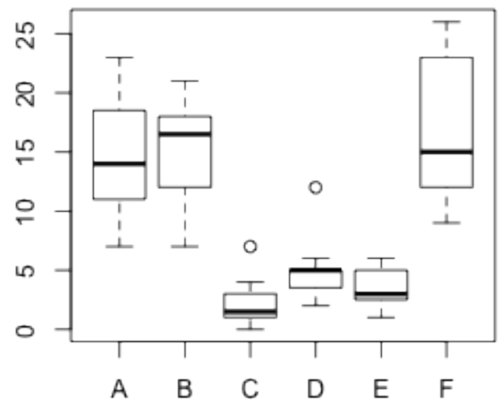
Notes on the bootstrap

- The bootstrap is non-parametric
- Better percentile bootstrap confidence intervals correct for bias
- There are lots of variations on bootstrap procedures; the book "An Introduction to the Bootstrap"" by Efron and Tibshirani is a great place to start for both bootstrap and jackknife information

Group comparisons

- Consider comparing two independent groups.
- Example, comparing sprays B and C

```
data(InsectSprays)
boxplot(count ~ spray, data = InsectSprays)
```



Permutation tests

- Consider the null hypothesis that the distribution of the observations from each group is the same
- Then, the group labels are irrelevant
- We then discard the group levels and permute the combined data
- Split the permuted data into two groups with n_A and n_B observations (say by always treating the first n_A observations as the first group)
- Evaluate the probability of getting a statistic as large or large than the one observed
- An example statistic would be the difference in the averages between the two groups; one could also use a t-statistic

Variations on permutation testing

DATA TYPE	STATISTIC	TEST NAME
Ranks	rank sum	rank sum test
Binary	hypergeometric prob	Fisher's exact test
Raw data		ordinary permutation test

- Also, so-called *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 as well
  - Permuting a regressor of interest
- Permutation tests work very well in multivariate settings

Permutation test for pesticide data

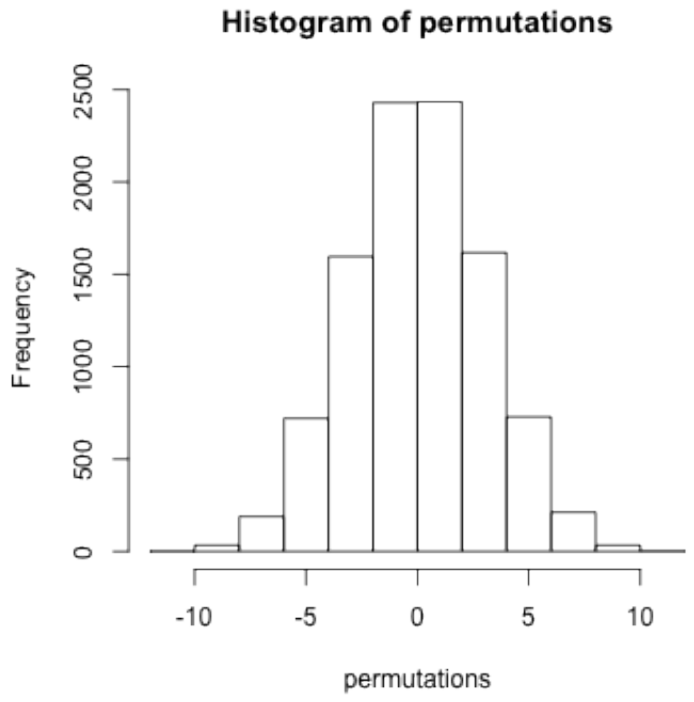
```r
subdata <- InsectSprays[InsectSprays$spray %in% c("B", "C"), ]
y <- subdata$count
group <- as.character(subdata$spray)
testStat <- function(w, g) mean(w[g == "B"]) - mean(w[g == "C"])
observedStat <- testStat(y, group)
permutations <- sapply(1:10000, function(i) testStat(y, sample(group)))
observedStat
```

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

```r
mean(permutations > observedStat)
```

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

Histogram of permutations



Resampled inference

Statistical Inference

Brian Caffo, Jeff Leek, Roger Peng
Johns Hopkins Bloomberg School of Public Health

The jackknife

- The jackknife is a tool for estimating standard errors and the bias of estimators
- As its name suggests, the jackknife is a small, handy tool; in contrast to the bootstrap, which is then the moral equivalent of a giant workshop full of tools
- Both the jackknife and the bootstrap involve *resampling* data; that is, repeatedly creating new data sets from the original data

The jackknife

- The jackknife deletes each observation and calculates an estimate based on the remaining $n - 1$ of them
- It uses this collection of estimates to do things like estimate the bias and the standard error
- Note that estimating the bias and having a standard error are not needed for things like sample means, which we know are unbiased estimates of population means and what their standard errors are

The jackknife

- We'll consider the jackknife for univariate data
- Let $X_1, \ldots, X_n$ be a collection of data used to estimate a parameter θ
- Let $\hat\theta$ be the estimate based on the full data set
- Let $\hat\theta_i$ be the estimate of θ obtained by *deleting observation* i
- Let $\bar\theta = \frac{1}{n}\sum_{i=1}^n \hat\theta_i$

Continued

- Then, the jackknife estimate of the bias is

$$(n-1)\left(\bar{\theta}-\hat{\theta}\right)$$

(how far the average delete-one estimate is from the actual estimate)

- The jackknife estimate of the standard error is

$$\left[\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_i-\bar{\theta})^2\right]^{1/2}$$

(the deviance of the delete-one estimates from the average delete-one estimate)

Example

We want to estimate the bias and standard error of the median

```r
library(UsingR)
data(father.son)
x <- father.son$sheight
n <- length(x)
theta <- median(x)
jk <- sapply(1:n, function(i) median(x[-i]))
thetaBar <- mean(jk)
biasEst <- (n - 1) * (thetaBar - theta)
seEst <- sqrt((n - 1) * mean((jk - thetaBar)^2))
```

Example test

```r
c(biasEst, seEst)
```

```
## [1] 0.0000 0.1014
```

```r
library(bootstrap)
temp <- jackknife(x, median)
c(temp$jack.bias, temp$jack.se)
```

```
## [1] 0.0000 0.1014
```

Example

- Both methods (of course) yield an estimated bias of 0 and a se of 0.1014
- Odd little fact: the jackknife estimate of the bias for the median is always 0 when the number of observations is even
- It has been shown that the jackknife is a linear approximation to the bootstrap
- Generally do not use the jackknife for sample quantiles like the median; as it has been shown to have some poor properties

Pseudo observations

- Another interesting way to think about the jackknife uses pseudo observations
- Let

$$\text{Pseudo Obs} = n\hat{\theta} - (n-1)\hat{\theta}_i$$

- Think of these as ``whatever observation i contributes to the estimate of θ"
- Note when $\hat{\theta}$ is the sample mean, the pseudo observations are the data themselves
- Then the sample standard error of these observations is the previous jackknife estimated standard error.
- The mean of these observations is a bias-corrected estimate of θ

The bootstrap

- The bootstrap is a tremendously useful tool for constructing confidence intervals and calculating standard errors for difficult statistics
- For example, how would one derive a confidence interval for the median?
- The bootstrap procedure follows from the so called bootstrap principle

The bootstrap principle

- Suppose that I have a statistic that estimates some population parameter, but I don't know its sampling distribution
- The bootstrap principle suggests using the distribution defined by the data to approximate its sampling distribution

The bootstrap in practice

- In practice, the bootstrap principle is always carried out using simulation
- We will cover only a few aspects of bootstrap resampling
- The general procedure follows by first simulating complete data sets from the observed data with replacement
    - This is approximately drawing from the sampling distribution of that statistic, at least as far as the data is able to approximate the true population distribution
- Calculate the statistic for each simulated data set
- Use the simulated statistics to either define a confidence interval or take the standard deviation to calculate a standard error

Nonparametric bootstrap algorithm example

- Bootstrap procedure for calculating confidence interval for the median from a data set of n observations

  i. Sample n observations **with replacement** from the observed data resulting in one simulated complete data set

  ii. Take the median of the simulated data set

  iii. Repeat these two steps B times, resulting in B simulated medians

  iv. These medians are approximately drawn from the sampling distribution of the median of n observations; therefore we can

    - Draw a histogram of them
    - Calculate their standard deviation to estimate the standard error of the median
    - Take the 2.5^{th} and 97.5^{th} percentiles as a confidence interval for the median

Example code

```r
B <- 1000
resamples <- matrix(sample(x, n * B, replace = TRUE), B, n)
medians <- apply(resamples, 1, median)
sd(medians)
```

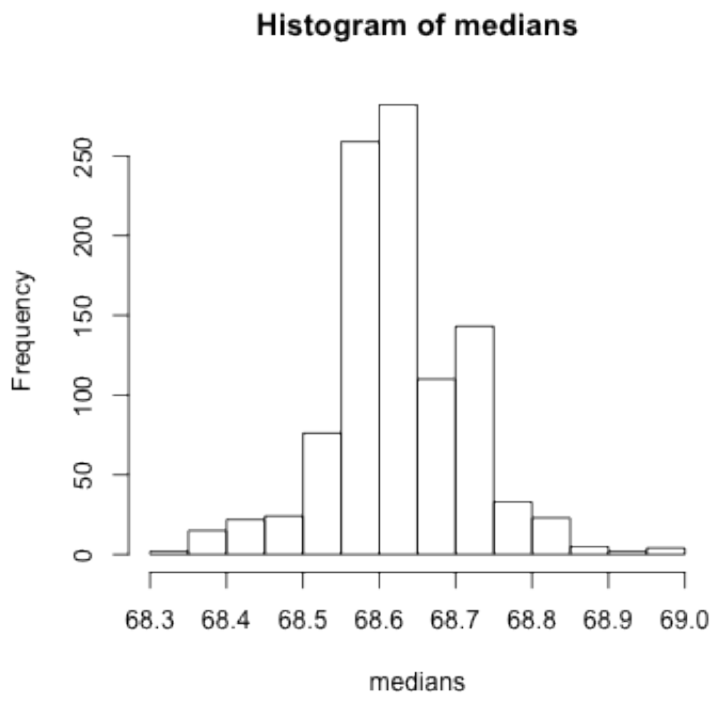
```
## [1] 0.08834
```

```r
quantile(medians, c(0.025, 0.975))
```

```
##  2.5% 97.5%
## 68.41 68.82
```

Histogram of bootstrap resamples

```
hist(medians)
```



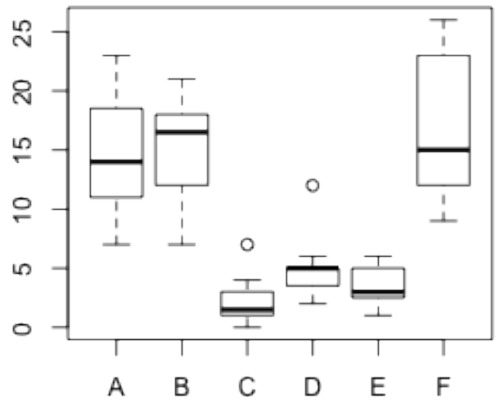
Notes on the bootstrap

- The bootstrap is non-parametric
- Better percentile bootstrap confidence intervals correct for bias
- There are lots of variations on bootstrap procedures; the book "An Introduction to the Bootstrap"" by Efron and Tibshirani is a great place to start for both bootstrap and jackknife information

Group comparisons

- Consider comparing two independent groups.
- Example, comparing sprays B and C

```
data(InsectSprays)
boxplot(count ~ spray, data = InsectSprays)
```



Permutation tests

- Consider the null hypothesis that the distribution of the observations from each group is the same
- Then, the group labels are irrelevant
- We then discard the group levels and permute the combined data
- Split the permuted data into two groups with n_A and n_B observations (say by always treating the first n_A observations as the first group)
- Evaluate the probability of getting a statistic as large or large than the one observed
- An example statistic would be the difference in the averages between the two groups; one could also use a t-statistic

Variations on permutation testing

DATA TYPE	STATISTIC	TEST NAME
Ranks	rank sum	rank sum test
Binary	hypergeometric prob	Fisher's exact test
Raw data		ordinary permutation test

- Also, so-called *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 as well
    - Permuting a regressor of interest
- Permutation tests work very well in multivariate settings

Permutation test for pesticide data

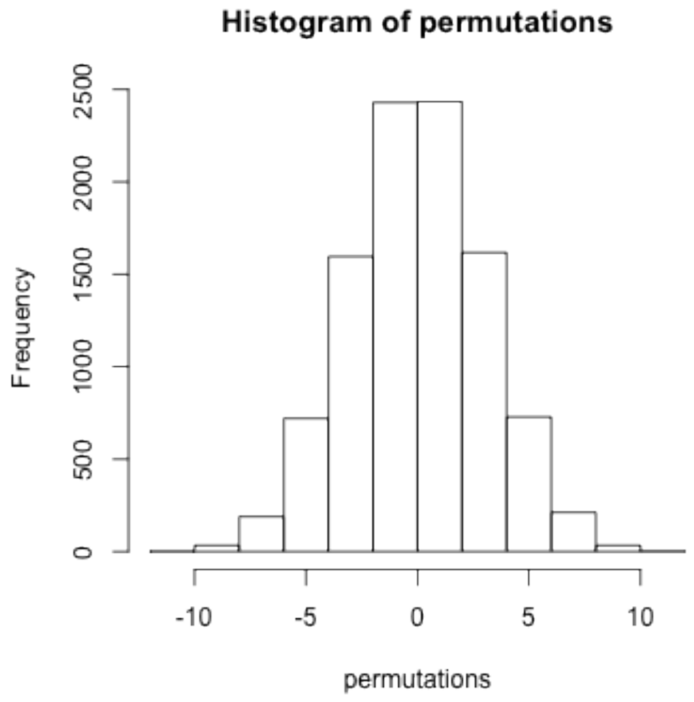
```r
subdata <- InsectSprays[InsectSprays$spray %in% c("B", "C"), ]
y <- subdata$count
group <- as.character(subdata$spray)
testStat <- function(w, g) mean(w[g == "B"]) - mean(w[g == "C"])
observedStat <- testStat(y, group)
permutations <- sapply(1:10000, function(i) testStat(y, sample(group)))
observedStat
```

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

```r
mean(permutations > observedStat)
```

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

Histogram of permutations


Histogram of permutations
Frequency
permutations