BMEG 802 – Advanced Biomedical Experimental Design and Analysis

Bootstrapping

Joshua G. A. Cashaback, PhD

Recap

- MCMC
  - Estimating the Posterior Distribution
  - Sampling Method
    - Breaking down multivariate into univariate densities
  - Markov Chain
    - the process of sampling a new value from the posterior distribution, given the previous value
  - Monte Carlo
    - refers to the random simulation process (random walk)
    - Gibbs, Metropolis-Hastings

Today

Bootstrapping

- Population Parameter Estimates
    - doesn't rely on parametric assumptions (e.g. normality)
    - means, medians, IQR... anything you want!
    - confidence intervals
- Hypothesis Testing
    - non-parametric, by simulating the null
        - Between Group
        - Paired Differences
- Power Analyses
    - not restricted by assumptions (can perform on non-parametric tests)
- Model Fitting

General Problem

- what if assumptions are violated?
- data are not normally distributed
  - heteroskedastic, variances unequal, sample size unequal
- ceilings, floors
- nonlinear model
- etc.

Population Parameter Estimates

We saw earlier:

- best estimate of a population mean is the sample mean (assuming normality)

$$\mu = \bar{X} = \frac{\Sigma x_i}{n}$$

- estimate of sd of sampling distribution of means is standard error of mean:

$$s_{\bar{x}} = \frac{s_x}{\sqrt{n}}$$

- can use this to generate 95% CIs of population mean:

$$\bar{X} \pm t_{\alpha}(s_{\bar{x}})$$

Population Parameter Estimates

- bootstrapping can estimate sampling distribution of means (or any other statistic)
- no need to assume any particular theoretical distribution
- use resampling with replacement to simulate repeatedly sampling from the population
- uses sample as proxy for population

Population Parameter Estimates

- assume you have a sample $X_1, X_2, ..., X_n$ and a statistic of interest (e.g. the mean) repeat M times (where M is large, e.g. 10,000)
    - generate a new sample of size n by resampling, with replacement, from X1..Xn
        - compute the statistic based on the new sample
        - set that statistic aside (e.g. save it in a list)
- now you have a list of M versions of the statistic, one for each resampling
- that list represents an empirical bootstrap distribution of the statistic of interest
- now you can compute relevant quantities of that distribution (e.g. 95% CIs)

Population Parameter Estimates

- e.g. we have a sample of size 20:
- 66 79 93 86 69 79 101 97 91 95 72 106 105 75 70 85 92 74 88 93
- estimate of population mean (using sample mean) is 85.8
- how precise is that estimate?

Population Parameter Estimates

```
X = c(66, 79, 93, 86, 69, 79, 101, 97, 91, 95, 72,
      106, 105, 75, 70, 85, 92, 74, 88, 93)
(Xm = mean(X)) # compute a statistic of interest
```

```
## [1] 85.8
```

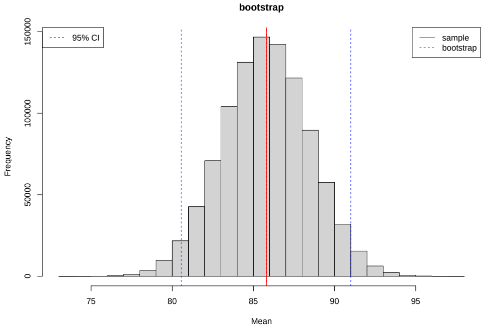
```
boot_m = 1000000 # how many simulated experiments?
Xm_boot = array(NA, boot_m) # create a list to store our bootstrap values

for (i in 1:boot_m) {
  Xb = sample(X, length(X), replace=TRUE) # generate new sample
  Xm_boot[i] = mean(Xb) # compute statistic of interest
}
```

Population Parameter Estimates

```
# display results
hist(Xm_boot, xlab="Mean", main="bootstrap")
abline(v=Xm, col="red")
abline(v=mean(Xm_boot), col="red", lty=2)
legend(x="topright", lty=c(1,2), col=c("red","red"), legend=c("sample","bootstrap"))

# compute 95% CI
CI95 = quantile(Xm_boot, probs=c(.025,.975))
abline(v=CI95[1], lty=2, col="blue")
abline(v=CI95[2], lty=2, col="blue")
legend(x="topleft", lty=2, col="blue", legend="95% CI")
```



Population Parameter Estimates

- here we used a bootstrap to estimate the sampling distribution of the mean
- we can do the same procedure to estimate the sampling distribution of any statistic we want
    - e.g. variance, median, IQR, skew, etc. . .

Hypothesis Testing - Two Sample

- example: comparing two populations
- drug vs control
- null hypothesis: drug has no effect
  - drug and control sampled from same population
- alternate hypothesis: drug has an effect
  - drug and control not sampled from same population

Hypothesis Testing - Two Sample

- choose a test statistic (e.g. the difference between means... but could be anything; t, F, sd, whatever your scientific question calls for)
- do many many times (at least 10,000):
- simulate the null hypothesis (that drug and control labels are random)
- how many times did you get a test statistic as large or larger as the original one? $<$ 5%? then reject H_0

Hypothesis Testing - Two Sample

- simulate the null hypothesis (that drug and control labels are random)
    - throw both groups into a bucket
    - randomly reconstitute the two groups, disregarding their original group membership *(resample without replacement)*
    - recompute the statistic of interest

Hypothesis Testing - Two Sample

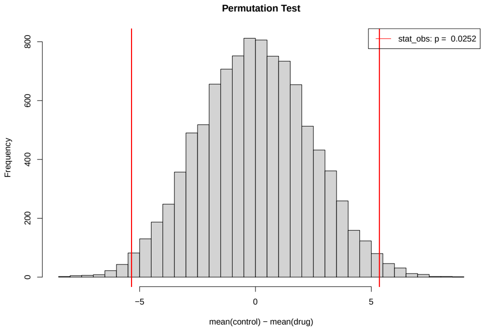
```r
g_control <- c(87,90,82,77,71,81,77,79,84,86,78,84,86,69,81,75,70,76,75,93)
g_drug <- c(74,67,81,61,64,75,81,81,81,67,72,78,83,85,56,78,77,80,79,74)
# our statistic of interest here is the difference between means
(stat_obs <- mean(g_control) - mean(g_drug))
```

```
## [1] 5.35
```

```r
n_perm = 10000 # how many simulated experiments?
stat_perm = array(NA, n_perm) # create a list to store our permutation test values
g_control_n = length(g_control)
g_drug_n = length(g_drug)
g_bucket = c(g_control, g_drug)
g_bucket_n = length(g_bucket)
for (i in 1:n_perm) {
  # reconstitute both groups, ignoring original labels
  permuted_bucket <- sample(g_bucket,g_bucket_n,replace=FALSE)
  perm_control <- permuted_bucket[1:g_control_n]
  perm_drug <- permuted_bucket[(g_control_n+1):(g_control_n+g_drug_n)]
  stat_perm[i] <- mean(perm_control) - mean(perm_drug)
}
```

Hypothesis Testing - Two Sample

```
# visualize the empirical permutation distribution of our statistic of interest
hist(stat_perm, 50, xlab="mean(control) - mean(drug)", main="Permutation Test")
abline(v=stat_obs, col="red", lwd=2)
abline(v=-stat_obs, col="red", lwd=2)
# how many times in the permutation tests did we observe a stat_perm bigger than or smaller than the stat_obs?
p_perm0 <- length(which(stat_perm >= stat_obs)) / n_perm
p_perm1 <- length(which(stat_perm <= -stat_obs)) / n_perm
p_perm2 <- p_perm0 + p_perm1
legend(x="topright", lty=1, col="red", legend=paste("stat_obs: p = ", p_perm2))
```



Note: can make this a one-tailed test by calculating observed permutations above OR below statobs

Hypothesis Testing - Two Sample

- here we tested the difference between means
- but we can apply this method to any statistic of interest that we can calculate
- no need to assume theoretical distribution
- compute probability under H_0 empirically by simulating the null hypothesis
- p-value = probability that our OBSERVED statistic is randomly sampled under the null hypothesis!

Hypothesis Testing - Paired Sample

Assignment question

Power Calculations

- We can use random resampling to simulate experiments not only under the null hypothesis but under any alternate hypothesis of our choosing
- we can use simulations to answer questions about statistical power
- We did a version of this in our effect size / power analysis lecture!
  - handy for calculating power for nonparametric statistics

Power Calculations

- what's the probability of detecting a given effect with a given number of subjects?
- how many subjects are required to detect a given effect 80% of the time? (or any other % of your choosing)
- again a bootstrapping/resampling approach doesn't require assumptions about a theoretical distribution

Power Calculations

- example: 2 groups, drug and control
- control: 87 90 82 87 71 81 77 79 84 86 78 84 86 69 81 75 70 76 75 93
- drug: 74 73 81 65 64 75 76 81 81 67 72 78 83 75 66 78 77 80 79 74
- Mann-Whitney U test: $p = 0.009556374$

Power Calculations

install.packages(“exactRankTests”)

```
library(exactRankTests)
```

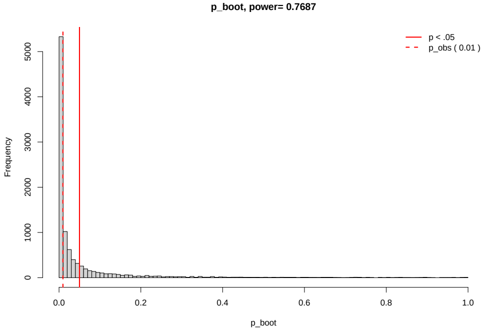
```
##  Package 'exactRankTests' is no longer under development.
##  Please consider using package 'coin' instead.
```

```
g_control = c(87,90,82,87,71,81,77,79,84,86,78,84,86,69,81,75,70,76,75,93)
g_drug = c(74,73,81,65,64,72,76,81,81,67,72,78,83,75,66,78,77,80,79,74)
p_obs = wilcox.exact(g_control, g_drug, alternative = "two.sided")$p.value
n_boot = 10000
p_boot = array(NA, n_boot)
for (i in 1:n_boot) {
  b_control = sample(g_control,length(g_control),replace=TRUE)
  b_drug = sample(g_drug,length(g_drug),replace=TRUE)
  p_boot[i] = wilcox.exact(b_control, b_drug, alternative = "two.sided")$p.value
}
(power <- length(which(p_boot <= .05)) / n_boot)
```

```
## [1] 0.7687
```

Power Calculations

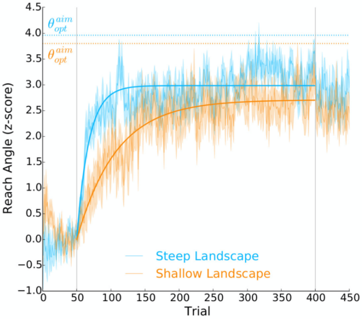
```r
hist(p_boot, 100, main=paste("p_boot, power=", power), xlab="p_boot")
abline(v=0.05, col="red", lty=1, lwd=2)
abline(v=p_obs, col="red", lty=2, lwd=2)
legend(x="topright", col="red", lty=c(1,2), lwd=2, legend=c("p < .05", paste("p_obs (",round(p_obs,3),")")), box.lty=0)
```

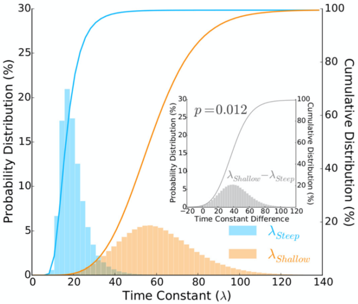


Model Fitting

Bootstrapping is a great way to fit models and perform hypothesis tests on these models.

- the probability of some parameter(s) given the data.





Cashaback et al 2019, PLoS Comp Bio: 15(3): e1006839.

Model Fitting

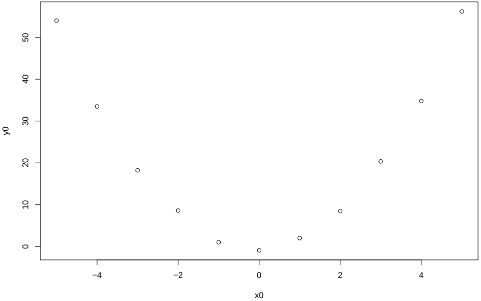
Let's pretend each data point represents a participant in a study. We know that the relationship follows a quadratic relation with Gaussian distributed noise:

$$y = c \cdot x^2 + \epsilon$$

Model Fitting - Data

First, let's use the following data

```
x0 = seq(-5,5,1)
## Can use commented out code to generate data
#noise = rnorm(11,0,1.0)
#c = 2.2 # pretend you don't know what the constant is (we are going to estimate this)
#y0 = c * x0^2 + noise
#y0
y0 = c(54.000534, 33.485031, 18.217317, 8.599760, 1.018073, -0.903174, 2.019343, 8.475278, 20.347628, 34.768279, 56.204352)
plot(x0,y0)
```



Model Fitting - Loss Function

- We want to find c and the distribution of this estimate.
- Let's use the sum of least squares as our loss function. That is, let's find c that minimizes

$$min[\Sigma_{i=1}^{n}(y_i - c \cdot x_i^2)^2]$$

Model Fitting - Loss Function

Sum of least squares:

```r
# This is our sum of least squares loss function
#C = initial guess on c, X = xvals, Y = yvals
lse <- function(C,X,Y) {
  c <- C[1]
  x0 = X
  y0 = Y
  mindiff <- sum((y0 - c*x0^2)^2) # sum of least squares
  return(mindiff)
}

x0 = seq(-5,5,1) # x-values
y0 = c(54.000534, 33.485031, 18.217317, 8.599760, 1.018073, -0.903174,
          2.019343, 8.475278, 20.347628, 34.768279, 56.204352) #y-values
c_init = c(5) # initial guess of the constant, c
opt <- nlm(f = lse, c_init,x0,y0) # using the previously defined loss function
opt$estimate
```

```
## [1] 2.178547
```

Our least squares estimate is 2.1785 (actual is 2.2, so pretty darn close!). But we also want to know the probability of c given the data.

Model Fitting - Bootstrapping

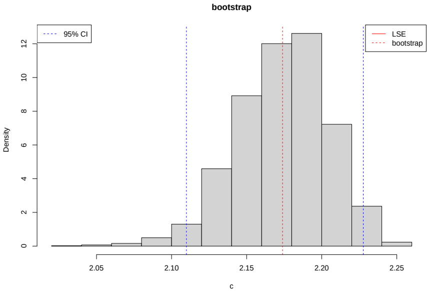
Resample, with replacement, while preserving X and Y pairs - remember, each X,Y pair corresponds to a participant

```r
x0 = seq(-5,5,1) # x-values
y0 = c(54.000534, 33.485031, 18.217317, 8.599760, 1.018073, -0.903174,
          2.019343, 8.475278, 20.347628, 34.768279, 56.204352) #y-values
boot_m = 10000 # how many simulated experiments?
C_boot = array(NA, boot_m) # create a list to store our bootstrap values
X = array(NA, length(x0))
Y = array(NA, length(y0))
c_init = c(5) # initial guess of the constant, c
for (i in 1:boot_m) {
  for (j in 1:length(x0)){# inner loop used to preserve x,y pairs
    k = sample(1:length(x0), 1) # sample random integer
    X[j] = x0[k] # use integer to sample x,y pair
    Y[j] = y0[k]
  }
  C_boot[i] = nlm(f = lse, c_init,X,Y)$estimate
}
```

Model Fitting - Bootstrapping

```r
# display results
hist(C_boot, xlab="c", main="bootstrap", freq=FALSE)
abline(v=mean(C_boot), col="red", lty=2)
legend(x="topright", lty=c(1,2), col=c("red","red"), legend=c("LSE","bootstrap"))

# compute 95% CI
CI95 = quantile(C_boot, probs=c(.025,.975))
abline(v=CI95[1], lty=2, col="blue")
abline(v=CI95[2], lty=2, col="blue")
legend(x="topleft", lty=2, col="blue", legend="95% CI")
```

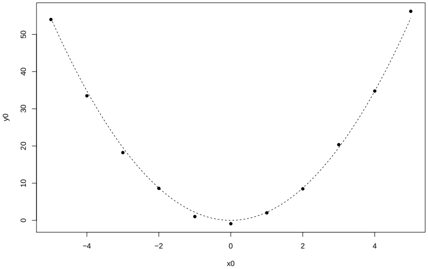


Now we also have confidence intervals for *c*!

Plotting Model Fits

Mean = least squares fit (LSE)

```
x0 = seq(-5,5,1)
y0 = c(54.000534, 33.485031, 18.217317, 8.599760, 1.018073, -0.903174, 2.019343, 8.475278, 20.347628, 34.768279, 56.204352)
xfit = seq(-5,5,0.1)
yfit =  mean(C_boot)*xfit^2
plot(x0,y0,pch=16)
lines(xfit,yfit,lty=2)
```



Can also use median and mode of the *c* posterior distribution

- Median = absolute difference estimate
- Mode = maximum a posterior (MAP) estimate

Model Fitting

- Linear

$$min[\Sigma_{i=1}^{n}(y_i - (\beta_0 + \beta_1 \cdot x_i))^2]$$

- Nonlinear
  - Exponential, Sigmoid, etc.
- Normal or nonnormal distributions.
- We get a distribution of possible parameters
- Does not account for priors

Next Class

Lab / Final Review