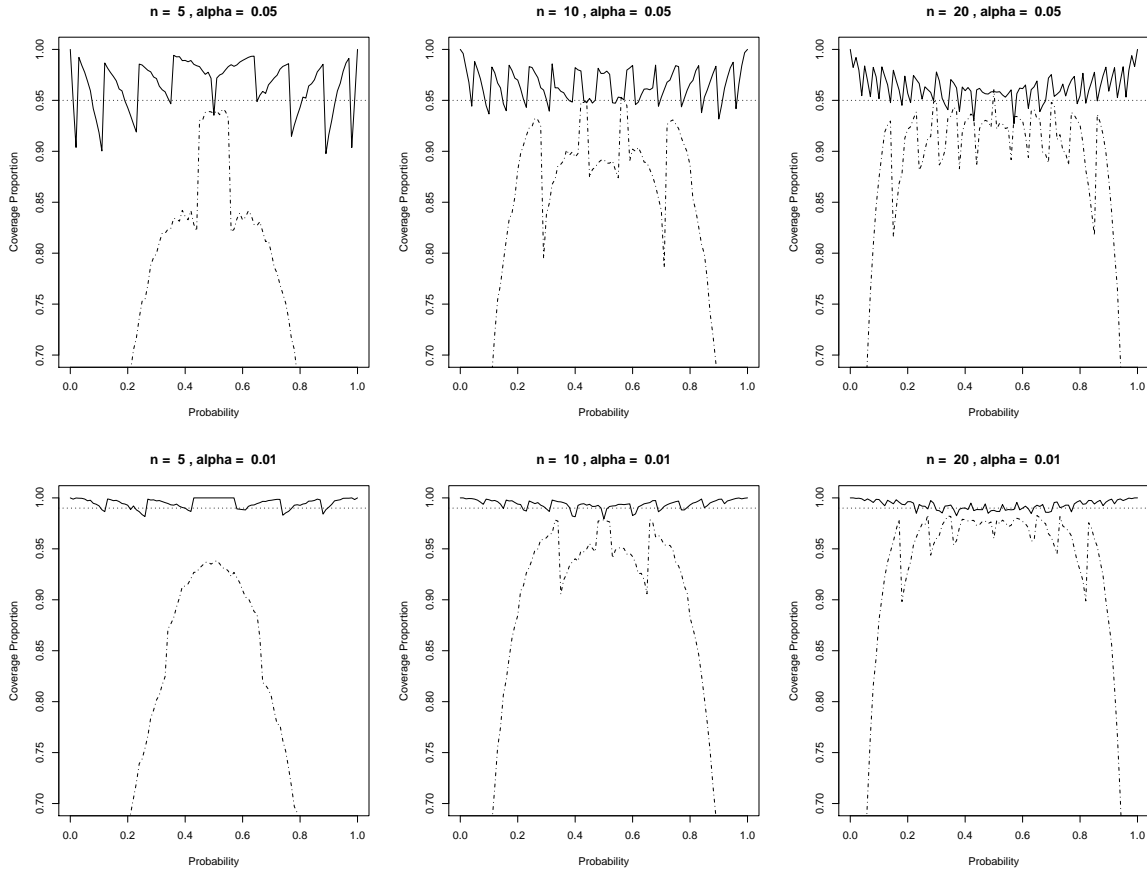


Stat 505 Assignment 8 Solutions

1. Recreate Figure 1 from Agresti and Caffo.



Reproduction of confidence interval comparison plot in Agresti and Caffo, 2000, *The American Statistician* **54**:4 280-288. Dashed lines show coverage rates of the conventional Wald intervals, solid lines those of the Wilson (add 2 successes and 2 failures) CI. They much prefer the Wilson CI.

2. Table 1 from *ibid*, rows 1 to 4, columns 1 to 6 using 10,000 draws.

		Number of Added Pseudo Observations				
n_1	n_2	0	2	4	6	8
10	10	0.895	0.947	0.958	0.960	0.945
20	20	0.927	0.949	0.952	0.955	0.948
30	30	0.934	0.948	0.956	0.952	0.948
30	10	0.894	0.952	0.959	0.958	0.950

In the same article, Agresti and Caffo looked into adding extra observations for the CI of the difference in two proportions. They conclude that adding $t = 4$ pseudo observations gives intervals with good properties.

3. Exercise 1 p 165. x_1 goes from 1 to 100, x_2 is 100 draws from Bernoulli(.5). Use model:
 $y_i = 3 + .1x_{i1} + .5x_{i2} + \epsilon_i$

(a) Generate one set of errors from scaled t_4 , multiplying each by 5. Check to see if the estimated coefficients fall within ± 2 SE's of the true values.

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.07152	2.19621	-0.032	9.74e-01
x1	0.12198	0.02297	5.309	6.97e-07
x2	2.00370	1.53228	1.307	1.94e-01

(Intercept)	x1	x2
FALSE	TRUE	TRUE

Two were included, one not.

(b) Repeat in a loop (I did 10000 times) and obtain coverage rates.

```
[1] 0.682 0.680 0.684
```

Slight undercoverage on my intercept and slope for x_1 , slightly over on x_2 's slope.

(c) Repeat 10000 times using the `glm` function in the `hett` package instead of `lm`.

```
[1] 0.675 0.672 0.677
```

Overall, the `lm` results are slightly closer to 0.68.

R Code

```
#####
## code chunk number 2: fig1
#####
require(arm)
require(xtable)
set.seed(123456789)
Wald.cover1 <- function(p, n, nsim, alpha){
  ## function to compute coverage rate of Wald CI for p in binomial setting
  ## n = number of trials, nsim the number of times to simulate
  ## alpha the desired significance level
  phat <- rbinom(nsim,n,p)/n
  sum( abs(p-phat)/ sqrt(phat*(1-phat)/n) < qnorm(1 - alpha/2))/nsim
}
Wilson.cover1 <- function(p, n, nsim, alpha){
  ## function to compute coverage rate of Wilson CI for p in binomial setting
  ## n = number of trials, nsim the number of times to simulate
  ## alpha the desired significance level
  ptilde <- (2 + rbinom(nsim,n,p))/(n + 4)
  sum( abs(p-ptilde)/ sqrt(ptilde*(1-ptilde)/(n+4)) < qnorm(1 - alpha/2)) / nsim
}
##
probs <- seq(0,1,.01)
par(mfrow=c(2,3))
for(alpha in c(.05,.01)){
  for(sample.size in c(5,10,20)){
    plot( probs, sapply( probs, Wald.cover1, sample.size, 10000, alpha), type="l", lty=4,
          ylim=c(.7,1), xlab="Probability", ylab="Coverage Proportion", main = paste("n = ",
            sample.size, ", alpha = ", alpha))
    abline(h=1-alpha, lty=3)
    lines( probs, sapply( probs, Wilson.cover1, sample.size, 10000, alpha))
  }
}
#####
```

```

### code chunk number 3: tab1
#####
adj.cover2 ← function(n1, n2, t, nsim){
  # draw random p1 and p2 iid from U(0,1),
  # generate random binomial data, and
  # check to see if the std confidence interval contains p1-p2
  # inputs: n1, n2, sample sizes
  # nsims, number of replications to simulate
  # t is the number of successes and failures to add
  p1 ← runif(nsim,0 ,1)
  p2 ← runif(nsim, 0 ,1)
  phat1 ← (t/4 + rbinom(nsim,n1,p1))/(t/2 + n1)
  phat2 ← (t/4 + rbinom(nsim,n2,p2))/(t/2 + n2)
  sum( abs(p1 - p2 - phat1 + phat2)/ sqrt(phat1 * (1 - phat1)/(n1 + t/2) + phat2*(1 - phat2)
    /(n2 + t/2) ) < qnorm(.975)) / nsim
}

coverage.table ← cbind(c(10,20,30,30), c(10,20,30,10), matrix(0,4,5))
colnames(coverage.table) ← c("n_1", "n_2", "t=0", "t=2", "t=4", "t=6", "t=8")
for( rows in 1:4){
  n1=c(10,20,30,30)[rows]
  n2=c(10,20,30,10)[rows]
  for(cols in 1:5){
    t ← 2*(cols-1)
    coverage.table[rows, cols+2] ← adj.cover2(n1,n2,t,10000)
  }
}
xtable(coverage.table ,digits=3)

#####
### code chunk number 4: modelCheck
#####
## a generate the data and fit with lm
x1 ← 1:100
x2 ← rnorm(100, 1, .5)
beta ← c(3, 0.1, 0.5)
y ← cbind(1, x1, x2) %*% beta + 5 * rt(100, 4)
y.fit ← lm(y ~ x1 + x2)
(coef.table ← summary(y.fit)$coef)
abs(beta - coef.table[,1]) < coef.table[,2]

#####
### code chunk number 5: modelCheckLoop
#####
## a generate the data and fit with lm in a loop
results ← matrix(0,3,10000)
for(s in 1:10000){
  y ← cbind(1, x1, x2) %*% beta + 5 * rt(100, 4)
  y.fit ← lm(y ~ x1 + x2)
  results[,s] ← abs(beta - coef(y.fit)) < se.coef(y.fit)
}
rowSums(results)/10000

#####
### code chunk number 6: tlm
#####
require(hett)
results2 ← matrix(0,3,10000)
for(s in 1:10000){
  y ← cbind(1, x1, x2) %*% beta + 5 * rt(100, 4)
  y.fit ← tlm(y ~ x1 + x2, start = list(dof = 4))
  coef.table ← summary(y.fit)$loc.summary$coefficients
  results2[,s] ← abs(beta - coef.table[,1]) < coef.table[,2]
}
rowSums(results2)/10000

```