Name: _______________________________________

1 Objective: Bootstrap Resampling Method

We use statistical sampling to determine the value of a parameter of a population. We sample a population, measure a statistic of this sample, and then use this statistic to say something about the corresponding parameter of the population. Bootstrapping is a resampling method with replacement from the initial sample. Each iteration of bootstrap improves the order of accuracy for the parameter in question.

The name bootstrap is a metaphor for "a self-sustaining process that proceeds without external help"

Our goal is to produce graphs like those of Mammen's paper, distributed in class, when sampling from two populations:

2 Key Words

`rchisq, quintile, sample, t, apply, cbind, pnorm`, quantiles, plain bootstrap, wild bootstrap, bias.

3 Brief Description

Consider the following two populations:

- POP1 : CHI 8: X has chi-squared distribution, $df = 8$, so $E[X] = 8$, $V[X] = 16$.
  where df is degrees of freedom, $E[X] = df$ is expected value, and $V[X] = 2(df)$ is variance.
- POP2: EXP 1: X has exponential distribution, $\lambda = 1$, $E[X] = 1$, $V[X] = 1$.
  where λ is the rate, $E[X] = \lambda^{-1}$ is expected value, and $V[X] = \lambda^{-2}$ is variance.

The statistic we'll look at is **sample mean**. There are **two pivots** of interest (notation is given below):

Difference $Q = \widehat{M} - M$ and studentized difference $Q_s = (\widehat{M} - M)/\widehat{D}$, and the corresponding bootstrap resampling pivots Q^* and Q_s^*. The claim of bootstrap is BEHAVIOR OF Q^* MIMICS THAT OF Q. Mammen's graphs are an attempt to prove this claim.

Read the notation and definitions as needed from Mammen's.

4 Notation

- POP: P; mean: $M = M(P)$; var: $V = V(P)$; dev: $D = \sqrt{V(P)}$.

  SAMPLE: S; $n =$ `ns` $=$ number in sample; mean: $\widehat{M} = M(S)$; var: $\widehat{V} = V(S) \neq V$; dev: $\widehat{D} = D(\widehat{M}) = \sqrt{V(S)/n} \neq \sqrt{V/n}$

- BOOTSTRAP: given $S = \{X_1, \ldots, X_n\}$ (assumes independence)

  SAMPLE: S^*: simple random sample, with replacement, of size `ns` from S; mean: $M^* = M(S^*)$; dev: $D^* = \sqrt{V(S^*)/n}$.

- WILDBOOTSTRAP: given $S = \{X_1, \ldots, X_n\}$. Wildbootstrap resamples from S differently, by choosing a_1 x_1's, a_2 x_2's, ... where $a_1, a_2, \ldots$ are independent Poisson random variables (assumes heteroscedasticity):

  SAMPLE: $a_k =$ `ak` iid POISSON(1);
  $MW = \text{mean}(a_k(X_k - \widehat{M}))$;
  $DW = \sqrt{\text{Var}(a_k(X_k - \widehat{M}))}$.

[A] QUANTILES: The first step is to use MC = Monte Carlo simulation to estimate the sequence of 5% quantiles `uk` and `vk` for the two pivots:

$$P\left[\widehat{M} - M \le \texttt{uk}\right] = k/20, \qquad P\left[(\widehat{M} - M)/\widehat{D} \le \texttt{vk}\right] = k/20, \ \ k \ \text{ in } \ 1:19$$

respectively.

[1] Fix a function `q.chi8` with input `ns` and `nq` = number of quantile simulations, and output a vector of 5% quantiles:

```
q.chi8 <- function (ns, nq)
{
        df <- double(nq)
        for (k in 1:nq)         {
                s <- rchisq(ns, 8)      # mean=df=8 in Chi-square distr.
                df[k] <- mean(s) - 8    # change with other parameters
        }
        return(quantile(df, 1:19/20))
}
```

[2] Fix a function `qstu.chi8` like `q.chi8` but with `df` replaced by `dfs` and

```
dfs[k]<-(mean(s) - 8)/(sqrt(var(s)/ns))
```

[3] Fix two more functions `q.exp1` and `qstu.exp1` that work for the population `exp1`.

[4] Below we'll need `uk.8 <- q.chi8(20,1000)`, and similarly for `uks.8 <- qstu.chi8(20,1000)`. In the same fashion, generate `uk.1`, and `uks.1`.
Note: Mammen used `nq = 16,000` rather than 1000.

[B] PLAIN BOOTSTRAP: The second step is to use MC to estimate BIAS and MSE.

[1] Fix a function `bs.p.chi8` with inputs: `ns, nr` = number of resamples from S, `nb` = number of bootstrap samples of different S and vector of quantiles `uu`, and output: matrix of bias and `mse` = expected squared error. These involve

`pk(S) := P[M* - M^ <= uk | S ]` # inner Loop sampling within S.
BIAS : `E(pk(S) - k/20)` # outer Loop sampling different S.
MSE : `E(pk(S) - k/20 )^2`

```
bs.p.chi8 <- function{ns, nr, nb, uu)
{
        s <- double(ns)                                        # S
        ss <- double(ns)                                       # S*
        pp <- matrix(double(nb * 19),  nrow = nb)              # observed pk(S) across S's

        for(i in 1:nb) {
                s <- rchisq(ns, 8)                             # get S
                ms <- mean(s)                                  # M^
                fr <- rep(0, 19)                # place for frequencies, replicates 0 19-times
                for (j in 1:nr) {
                        ss <- sample(s, ns, rep = T)           # get S*
                        dm <- mean(ss) - ms                    # get M* - M^
                        fr <- fr + (dm < uu)                   # frequencies: WHY WORK?
                }
                pp[i, ] <- fr / nr                             # pk(S), k = 1:19
        }
        err <- t(pp) - 1:19/20                  # t:=transpose func., pk (S)-k/20, k=1:19
        bias <- apply(err, 1, mean)
        mse <- apply(err^2, 1, mean)
        cbind(bias, mse)
        }
```

[2] Use the function to get plot data

```
bsp8 <- bs.p.chi8(20,50,100,uk.8)               # uk.8 from [A4]
```

NOTE: Mammen used `nr = 1,000 & nb = 10,000` rather than 50 & 100.

[3] Make plots of `bias` and `mse` on y-axis versus `xax<-1:19/20` on x-axis. They should look like Mammen. SHOW ME SCREEN OUTPUT FOR LAB CREDIT.

[C] NORMAL APPROXIMATION: By the CLT. Distribution $(\widehat{M} - M)/\widehat{D} \neq N(0,1)$, $Z := N(0,1)$ random variable, so $k/20 = P(\widehat{M} - M \leq \mathtt{uk}) \neq P(\widehat{D}\, Z \leq \mathtt{uk}) = P(Z \leq \mathtt{uk}/\widehat{D})$

Modify the outer loop and omit the inner loop of `bs.p.chi8` to give normal approximation estimates `bias` & `mse` . Fix the function `bs.na.chi8`:

```
bs.na.chi8 <- function(ns, nb, uu)                           # NOTE: NO nr for inner loop
{
        s <- double(ns)                                      # S
        pp <- matrix(double(nb * 19), nrow = nb)
        for(i in 1:nb) {
                s <- rchisq(n, 8)
                sd <- sqrt(var(s)/n)                         # D^
                pp[i, ] <- pnorm(uu/sd)
        }
                                                     # FILL IN HERE AS BEFORE in bs.p.chi8
}
```