1/19/2010 36-402/608 ADA-II H. Seltman
Handout #3: Simulation

1. Uses of simulation
   (a) Gain insight into statistical testing
   (b) Evaluate robustness of tests to assumption violation
   (c) Power calculation
   (d) Evaluate the performance of (new) tests

2. Power definition: For a particular model, sample size, α value, and set of true parameter values, the power is the probability that H_0 will be rejected, i.e, that we will obtain a p.value less than α (over repeated experiments).

3. Breakout (Questions 1-4)

4. My little R "simulation class"
   (a) The (S3) class mechanism in R

```
> methods(summary)
[1] summary.aov           summary.aovlist       summary.aspell*
[4] summary.connection    summary.data.frame    summary.Date
[7] summary.default       summary.ecdf*         summary.factor
10] summary.glm           summary.infl          summary.lm
13] summary.loess*        summary.manova        summary.matrix
16] summary.mlm           summary.nls*          summary.packageStatus*
19] summary.POSIXct       summary.POSIXlt       summary.ppr*
22] summary.prcomp*       summary.princomp*     summary.stepfun
25] summary.stl*          summary.table         summary.tukeysmooth*
  Non-visible functions are asterisked
```

When you call function `foo()` on an object of class "bar", R automatically runs `foo.bar()` if it exists, or `foo.default()` otherwise. E.g.,

```
> m0 = lm(y~x)
> class(m0)
 [1] "lm"
> summary(m0)
```

causes `summary.lm(m0)` to be run.

5. You can `source()` SimClass.R to load the functions `print.simulation()`, `summary.simulation()`, and `plot.simulation()`.

   If you create an appropriate simulation object, it is easy to examine it, by just entering `print(foo)`, `summary(foo)` or `plot(foo)`.

   Because the "simulation" class is based on the simpler "S3" class type (as opposed to the more sophisticated S4 class type used by some newer R packages), you can turn an object into a simulation class object just by using a command like `class(foo) = "simulation"`.

6. Characteristics of a simulation class object:

   (a) Concept: contains some results obtained by repeatedly applying a statistical method to different simulated data sets.

   (b) Technical detail: It is a list whose class has been set to "simulation".

   (c) Components of the list (normally your components will be "options" plus at least one of "p.values", "CIs", and/or "params" and each corresponding names component):

      i. params: a matrix with one row per simulation and one column per estimated parameter
      ii. names: a string vector of human-readable names (ids) for the estimated parameters
      iii. p.values: a matrix with one row per simulation and one column per p.value calculated
      iv. pnames: a string vector of human-readable names (ids) for the p.values
      v. CIs: a matrix with one row per simulation and two columns per confidence interval calculated
      vi. CInames: a string vector of human-readable names (ids) for the CIs
      vii. alpha: the alpha defining the width of the CIs and also used as the p.value cutoff in the power calculation
      viii. options: a (named) list containing any other information about how the simulation was performed for record keeping purposes. "nsim" is standard for the number of simulations.

7. Example

```
# Example: Errors in (x) variables simulation
# Arguments: nsim = number of simulated datasets
#             n = number of subjects per simulation
#             b = true intercept and slope
#             sdx = s.d. of error in x (0 allowed)
#             sdy = s.d. of error in y
# Note: x is simulated as uniform(0,1)
# Value: an object of the simulation class
eiv <- function(nsim=1000, n=100, b=c(0,1), sdx=0.2, sdy=0.2) {
  doOne <- function(n) {
    x <- runif(n, 0, 1)
    xpn <- x+rnorm(n, 0, sdx)
    y <- b[1] + b[2]*x + rnorm(n, 0, sdy)
    rslt <- lm(y˜xpn)
    return(c(as.numeric(rslt$coef),
             as.numeric(t(confint(rslt))),
             as.numeric(summary(rslt)$coef[,4])
             ))
  }
  values <- t(sapply(rep(n,nsim), doOne))
  rtn <- list(estimates=values[,1:2], names=c("Intercept","Slope"),
              params=b,
              alpha=0.05,
              CIs=values[,3:6], CInames=c("CIb0","CIb1"),
              p.values=values[,7:8], pnames=c("p.b0","p.b1"),
              CIparams=b,
              options=list(nsim=nsim, n=n, sdx=sdx, sdy=sdy))
  class(rtn) <- "simulation"
  return(rtn)
}
```

8. Breakout (Questions 5 and 6)

9. Power discussion