#install.packages("beanplot")

#install.packages("orddom")

#install.packages("compute.es")

#install.packages("pwr")

library(beanplot)

library(orddom)

library(compute.es)

library(pwr)

###Plots: d and t with increasing sample size.

set.seed(1234)

#parameters

a <- 30

b <- 2.5

err.sd <- 1.5

sim.y <- function(x, a, b, err.sd){

 y <- a + b\*x + rnorm(length(x), 0, err.sd)

 return(y)

}

plot.unex.var <- function(x,y,...){

 plot(x, y, xlab = "x", ylab = "y", pch = 19,...)

 fit <- lm(y~x)

 a <- fit$coefficients[1]

 b <- fit$coefficients[2]

 x.line <- seq(min(x)-5, max(x)+5, length.out=10)

 y.line <- a+b\*x.line

 lines(x.line,y.line)

 points(x, a + b\*x, pch = 4)

 for(i in 1:length(x)){

 lines(c(x[i],x[i]),c(a + b\*x[i], y[i]), lty = 2)

 }

}

plot.tot.var <- function(x,y,...){

 plot(x, y, xlab = "x", ylab = "y", pch = 19,...)

 a <- mean(y)

 b <- 0

 x.line <- seq(min(x)-5, max(x)+5, length.out=10)

 y.line <- a+b\*x.line

 lines(x.line,y.line)

 points(x, a + b\*x, pch = 4)

 for(i in 1:length(x)){

 lines(c(x[i],x[i]),c(a + b\*x[i], y[i]), lty = 2)

 }

}

x <- c(6,5,6,5)

ns <- numeric(0)

ds <- numeric(0)

ts <- numeric(0)

for(i in 1:998){

 ns[i] <- length(x)

 y <- sim.y(x,a, b = 0.5, err.sd = 1)

 fit <- t.test(y[x==x[1]], y[x==x[2]], var.equal = TRUE)

 ts[i] <- fit$statistic

 ds[i] <- (mean(y[x==x[1]]) - mean(y[x==x[2]]))/sqrt( mean(var(y[x==x[1]]), var(y[x==x[2]])) )

 x <- c(x, x[1:2])

}

par(mar = c(3,3,1,1), mgp = c(2,0.7,0), las = 1)

plot(ns, ts, xlab = "Total n", ylab = "t", pch = 20)

plot(ns, ds, xlab = "Total n", ylab = "Estimated d", pch = 20)

#Plot: example effect size measurement.

x <- rep(c(6,5), 100)

y <- sim.y(x, a, b, err.sd)

pal <- c('#a6cee3','#1f78b4','#b2df8a','#33a02c')

beanplot(y ~ x, col = pal, what = c(0,1,1,1), xlab = "Age", ylab = "Height (in.)")

mean(y[x ==6]) - mean(y[x ==5])

(mean(y[x ==6]) - mean(y[x ==5]))/sqrt(mean(var(y[x ==6]), var(y[x ==5])))

orddom(y[x ==6], y[x ==5])

cor(x,y)

fit <- lm(y~x)

plot.unex.var(x,y, xlim = c(4,8), ylim = c(35,55))

plot.tot.var(x,y, xlim = c(4,8), ylim = c(35,55))

#plot: effect size measurement with four ages.

x4 <- rep(c(4,5,6,7,8), 20)

y4 <- sim.y(x4, a, b, err.sd)

fit.4 <- lm(y4~x4)

plot(x4,y4, pch = 20, xlab = "Age", ylab = "Height (in.)", )

abline(fit)

cor(x4,y4)

plot.unex.var(x4,y4, xlim = c(4,8), ylim = c(35,55))

plot.tot.var(x4,y4, xlim = c(4,8), ylim = c(35,55))

beanplot(y, y4, what = c(0,0,0,1), col = pal, ylim = c(35,55), ylab = "y", horizontal = TRUE)

beanplot(fit$residuals, fit.4$residuals, what = c(0,0,0,1), col = pal, ylim = c(-10,10), ylab = "y", horizontal = TRUE)

var(y)

var(y4)

var(fit$residuals)

var(fit.4$residuals)

#Plot: example effect size measurement with unequal sampling.

xu <- rep(c(6,rep(5,19)), 100)

yu <- sim.y(xu, a, b, err.sd)

fit.u <- lm(yu~xu)

pal <- c('#a6cee3','#1f78b4','#b2df8a','#33a02c')

beanplot(yu ~ xu, col = pal, what = c(0,1,1,1), xlab = "Age", ylab = "Height (in.)")

mean(yu[xu ==6]) - mean(yu[xu ==5])

(mean(yu[xu ==6]) - mean(yu[xu ==5]))/sqrt(mean(var(yu[xu ==6]), var(yu[xu ==5])))

orddom(yu[xu ==6], yu[xu ==5])

cor(xu,yu)

plot.unex.var(xu,yu, xlim = c(4,8), ylim = c(35,55))

plot.tot.var(xu,yu, xlim = c(4,8), ylim = c(35,55))

var(yu)

var(fit.u$residuals)

#Plot: scatterplots for explaining coefficient of determination.

x <- anscombe$x1

y <- anscombe$y1

#y - y-hat

plot.unex.var(x,y)

#y - y-bar

plot.tot.var(x,y)

### Plots with jitter added to x for better visualization

jitter.x <- runif(length(x), -0.25, 0.25)

plot.unex.var(x + jitter.x,y, xlim = c(4,8), ylim = c(35,55))

plot.tot.var(x + jitter.x,y, xlim = c(4,8), ylim = c(35,55))

jitter.x4 <- runif(length(x4), -0.25, 0.25)

plot.unex.var(x4 + jitter.x4,y4, xlim = c(4,8), ylim = c(35,55))

plot.tot.var(x4 + jitter.x4,y4, xlim = c(4,8), ylim = c(35,55))

jitter.xu <- runif(length(xu), -0.25, 0.25)

plot.unex.var(xu + jitter.xu, yu, xlim = c(4,8), ylim = c(35,55))

plot.tot.var(xu + jitter.xu, yu, xlim = c(4,8), ylim = c(35,55))

plot.unex.var(x+ rnorm(length(x), 0 ),y, xlim = c(4,8), ylim = c(35,55))

plot.tot.var(x,y, xlim = c(4,8), ylim = c(35,55))

#####Another example: R^2 can depend hugely on sampling scheme.

#R^2 and variance in X.

#Imagine reading level is determined by

#reading level = grade + error,

#with the error distributed as Normal(0, 1)

#We sample 120 students in one of three ways:

#60 from 1st grade, 60 from 2nd grade, or

#10 from each grade, 1st through 12th

#60 from 1st grade, 60 from 12th grade.

#First and second graders

n.sims <- 1000 # number of simulations to run

grade <- rep(c(1,2), each = 60)

# a matrix holding 120 students' grade level for n.sims simulations

grade.mat <- matrix(rep(grade, n.sims), ncol = n.sims)

# a matrix of reading levels, generated by adding independent N(0,1) variables to grade matrix.

reading.level.mat <- grade.mat + matrix(rnorm(120\*n.sims, 0 , 1), ncol = n.sims)

#Compute the squared correlation for each of the 1000 simulated datasets.

r2.func <- function(y,x){cor(y,x)^2}

R2.dist <- apply(reading.level.mat, 2, r2.func, x = grade)

#Plot and summarize the squared correlations

hist(R2.dist)

summary(R2.dist)

#10 from each grade, 1 to 12.

grade <- rep(1:12, each = 10) #Now we pick 10 students from each of 12 grades.

grade.mat <- matrix(rep(grade, n.sims), ncol = n.sims)

reading.level.mat <- grade.mat + matrix(rnorm(120\*n.sims, 0 , 1), ncol = n.sims)

R2.dist <- apply(reading.level.mat, 2, r2.func, x = grade)

hist(R2.dist)

summary(R2.dist)

#100 First graders and 20 second graders

n.sims <- 1000 # number of simulations to run

grade <- c(rep(1,100), rep(2,20))

# a matrix holding 120 students' grade level for n.sims simulations

grade.mat <- matrix(rep(grade, n.sims), ncol = n.sims)

# a matrix of reading levels, generated by adding independent N(0,1) variables to grade matrix.

reading.level.mat <- grade.mat + matrix(rnorm(120\*n.sims, 0 , 1), ncol = n.sims)

#Compute the squared correlation for each of the 1000 simulated datasets.

r2.func <- function(y,x){cor(y,x)^2}

R2.dist <- apply(reading.level.mat, 2, r2.func, x = grade)

#Plot and summarize the squared correlations

hist(R2.dist)

summary(R2.dist)

###########################################################################

###########################################################################

###########################################################################

#Power

t1 <- 0.05

na <- 5

nb <- 25

nc <- 100

line.w <- 1.5

d <- seq(-2,2, length.out = 10000)

#this function assumes d as in eq. 6.4

pow.1samp.z.2side <- function(t1, n, d){

 crit.z <- abs(qnorm(t1/2))

 crit.val <- crit.z / sqrt(n)

 power <- 1 - pnorm(crit.val, mean = abs(d), sd = 1/sqrt(n))

 return(power)

}

pow.a <- pow.1samp.z.2side(t1, na, d)

pow.b <- pow.1samp.z.2side(t1, nb, d)

pow.c <- pow.1samp.z.2side(t1, nc, d)

plot(d, pow.a, type = "l", xlab = expression(italic(d)), ylab = "Power", lwd = line.w,

 las = 1)

lines(d, pow.b, lty = 2, lwd = line.w)

lines(d, pow.c, lty = 3, lwd = line.w)

legend("bottomright", legend = c(paste("n =", as.character(na)),

 paste("n =", as.character(nb)),

 paste("n =", as.character(nc))),

 lty = c(1,2,3), lwd = line.w)

###############################################

###############################################

#Simulations for power on the basis of d and winner's curse.

#Winner's curse (can be viewed as an instance of regression to the mean)

#Suppose that studies only get published if they reject a hypothesis of non-zero effect.

#Model: To study a group difference on a normally distributed trait, we sample

#2 groups of size n and conduct a 2-sample t-test. If significant,

#then we report the effect size in the literature. If someone does a meta-analysis,

#by how much will their effect size estimate be biased?

true.d <- 0.5

n <- 20 # n per group

n.studies <- 1000

#Function to simulate data from 1 "study"

gen.study <- function(n, d){

 y1 <- rnorm(n, d, 1)

 y2 <- rnorm(n, 0, 1)

 return(cbind(y1, y2))

}

#two-tailed permutatation test for mean.

perm.2means <- function(x1, x2, nperms = 1000){

 test.stat <- mean(x1) - mean(x2)

}

ds <- numeric(n.studies)

ds.l <- ds

ds.u <- ds

p.t <- ds

p.perm <- ds

p.wilcox <- ds

for(i in 1:n.studies){

 study <- gen.study(n, true.d)

 tt <- tt <- t.test(study[,1], study[,2])

 es <- tes(tt$statistic, n, n, verbose = FALSE)

 ds[i] <- es$d

 ds.l[i] <- es$l.d

 ds.u[i] <- es$u.d

 p.t[i] <- tt$p.value

 p.wilcox[i] <- wilcox.test(study[,1], study[,2])$p.value

 #p.perm <-

}

#plot of all effect size estimates

hist(ds, main = "Effect size estimates from all studies", xlab = "d",

 xlim = c(-max(abs(c(ds.l, ds.u))),max(abs(c(ds.l, ds.u))) ))

#plot of only significant effect size estimates

hist(ds[ds.l > 0 & ds.u > 0 | ds.l < 0 & ds.u < 0],

 main = "Effect size estimates from significant studies", xlab = "d",

 xlim = c(-max(abs(c(ds.l, ds.u))),max(abs(c(ds.l, ds.u))) ))

#mean(p.t < 0.05)

#mean(p.wilcox < 0.05)

sig.05 <- (ds.l > 0 & ds.u > 0 | ds.l < 0 & ds.u < 0)

pwr.t.test(n, true.d) #computed power from analytic power function

mean(sig.05) #estimated power at alpha = 0.05

mean(ds) #mean effect size estimate from all studies

mean(ds[sig.05]) #mean effect size estimates from published studies

#Plot: confidence intervals and the winner's curse

n.sp <- 100 #number of CIs to plot (more than 100 is unwieldy)

lbs <- ds.l[1:n.sp]

ubs <- ds.u[1:n.sp]

pal <- c("green", "blue")

plot(1:n.sp, ds[1:n.sp], pch = "", xlab = "Sample Number",

 ylab = expression(paste("Confidence Interval for d (true d = 0)")),

 ylim = c(min(c(ds.l, ds.u, 0) - 0.5),max(c(ds.l, ds.u,0))))

lines(c(-1000, 1000), c(0, 0), lty = 2)

lines(c(-1000, 1000), c(true.d, true.d), lty = 3)

for(i in 1:n.sp){

 lines(c(i,i), c(lbs[i], ubs[i]), col = pal[(ds.l[i] > 0 & ds.u[i] > 0 | ds.l[i] < 0 & ds.u[i] < 0) + 1])

}

legend("bottomright", lty = c(1,1,2,3), col = c(pal[1], pal[2], "black", "black"),

 legend = c("not significant","significant","null hypothesis d","true d"), cex = 0.8)

##Build power curve for Wilcoxon Signed Rank (Mann-Whitney) test compared with t.

n <- 50 #n in each group

ds\_to\_test <- seq(-20,20)/20

alpha = 0.05

n.sims<- 1000 # number of simulations per d.

power.t.ev

power.t <- numeric(length(ds\_to\_test))

power.t.ev <- power.t

power.wilcox <- power.t

for(i in 1:length(ds\_to\_test)){

 p.t <- numeric(n.sims)

 p.t.ev <- p.t

 p.wilcox <- p.t

 for(j in 1:n.sims){

 study <- gen.study(n, ds\_to\_test[i])

 tt <- t.test(study[,1], study[,2])

 tt.ev <- t.test(study[,1], study[,2], var.equal = TRUE)

 p.t[j] <- tt$p.value

 p.t.ev[j] <- tt.ev$p.value

 p.wilcox[j] <- wilcox.test(study[,1], study[,2])$p.value

 }

 power.t[i] <- mean(p.t < alpha)

 power.t.ev[i] <- mean(p.t.ev < alpha)

 power.wilcox[i] <- mean(p.wilcox < alpha)

}

plot(ds\_to\_test, power.t, ylim = c(0,1), type = "n", ylab = "Power", xlab = "effect size (d)",

 main = paste("Power of t-test vs. Wilcoxon (n = ", as.character(n),", alpha = ", as.character(alpha),")",sep = ""))

lines(ds\_to\_test, power.t, lty = 1)

#lines(ds\_to\_test, power.t.ev, lty = 2)

lines(ds\_to\_test, power.wilcox, lty = 3)

legend("bottomright", lty = c(1, 2, 3), legend = c("Welch's t-test","t-test (equal variances)","Wilcoxon test"))