Supplementary Materials: R codes

The following R codes replicate Figure 1 and Table 1:

```
n=10; sigma=1; alpha=0.05
x=seq(-4,10,0.01);
theta=0.5
NCP=sqrt(n)*theta/sigma

density0=dt(x,df=n-1,ncp=0)     # density under H0
density1=dt(x,df=n-1,ncp=NCP)   # density under H1

plot(x,density0,type="l",lwd=3,ylab="density",main="Density Functions")
points(x,density1,type="l",col="red",lwd=3)
abline(v=0,h=0);

cr0=qt(1-alpha,df=n-1)          # Critical value

# Region for Prob(Type I error)
region.x=x[ x  > cr0]
region.y=density0[ x > cr0]
region.x=c(cr0,region.x,tail(region.x,1))
region.y=c(0,region.y,0)
polygon(region.x,region.y,density=-1,col="gray")

# Region for Prob(Type II error)
region.x=x[ x  < cr0]
region.y=density1[ x < cr0]
region.x=c(head(region.x,1),region.x,cr0)
region.y=c(0,region.y,0)
polygon(region.x,region.y,density=10,col="red")
```

```
alpha=c(0.01, 0.05, 0.20,0.60)
cr=round(qt(1-alpha,df=n-1),digit=2)
beta=round(pt(cr,df=n-1,ncp=NCP),digit=2)
power=1-beta

mat=cbind(alpha,cr,beta,power);
colnames(mat)=c("alpha", "critical value","beta", "Power")
knitr::kable(mat,caption = "Alpha, Beta, and Power",format = "latex")
```

The following R codes replicate Figure 2:

```
alphavec=seq(0.00001,1,0.00001)
crvec=qt(1-alphavec,df=n-1)
betavec= pt(crvec,df=n-1,ncp=NCP)
plot(betavec,alphavec,type="l",xlim=c(0,1),col=1,lwd=2,ylab="alpha",xlab="beta")
for(i in 1:nrow(mat)) points(mat[i,3],mat[i,1],col="red",pch=16,cex=1)
abline(v=seq(0,1,0.1), h=seq(0,1,0.1),col="light gray", lty="dotted")
```

The following R codes replicate Table 2 and Figure 3:

```
# The line of enlightened judgement
alphavec=seq(0.00001,1,0.00001)
```

```r
crvec=qt(1-alphavec,df=n-1)
betavec= pt(crvec,df=n-1,ncp=NCP)
plot(betavec,alphavec,type="l",xlim=c(0,1),col=1,lwd=2,ylab="alpha",xlab="beta")
abline(v=seq(0,1,0.1), h=seq(0,1,0.1), col="lightgray", lty="dotted")

# Find the points of minimum expected loss for different values P and k
kvec=c(1,1/5,5,1,1/5)
Pvec=c(0.5,0.5,0.5,0.2,0.2)
pchvec=c(15,16,17,1,4)

table.2 = matrix(NA,nrow=5,ncol=4)

for(i in 1:length(kvec)){
p=Pvec[i]; k=kvec[i]
  LOSS=cbind(alphavec,betavec,abs(p*alphavec+(1-p)*k*betavec))
  MIN= LOSS[which.min(LOSS[,3]),]
  alphas=MIN[1];betas=MIN[2];
  points(betas,alphas,col="red",pch=pchvec[i],cex=1.25)
  table.2[i,] = c(p,k,betas,alphas)
}

table.2 = round(table.2,digits = 2)
colnames(table.2)=c("P", "k","beta*","alpha*")
knitr::kable(table.2,caption = "Optimal Significance Levels",format = "latex")
```

Feasibility and profitability of constructing a wind turbine

```r
# Calculation of t-statistic
xbar=38.33; mu=32; s = 19.2; n=50
tstat = (xbar-mu)/(s/sqrt(n)); tstat=round(tstat,digits = 5)
print(tstat)

# Calculation of p-value
pval= pnorm(tstat,lower.tail = FALSE); pval=round(pval,digits = 5)
print(pval)
```

```r
# Optimal level of significance
library(OptSig)
Opt.sig.t.test(d=(40-32)/19.2,n=50,p=0.5,k= 1,alternative="greater")
```

```r
NCP=sqrt(50)*(40-32)/19.2
Opt.sig.t.test(ncp=NCP,n=50,p=0.5,k=1,alternative="greater",Figure=FALSE)
```

```r
Opt.sig.t.test(d=(40-32)/19.2,n=50,p=0.5,k=0.1,alternative="greater")
```

```r
Opt.sig.t.test(d=(40-32)/19.2,n=50, p=0.5,k=0.05,alternative="greater")
```

Aspirin and heart attacks

```r
alpha=0.05
n1=11000; x1=104; p1=x1/n1
n2=11000; x2=189; p2=x2/n2
```

```r
# Confidence interval for p1-p2
p.diff= p1-p2; p.diff=round(p.diff,digits = 5)
print(p.diff)

cr = qnorm(0.5*alpha,lower.tail = FALSE)
se=sqrt( p1*(1-p1)/n1 + p2*(1-p2)/n2)

lower = p.diff - cr*se
upper = p.diff + cr*se
p.ci = c(lower,upper); p.ci=round(p.ci,digits = 5)
print(p.ci)
```

```r
# Z statistic and p-value
p = (x1+x2)/(n1+n2); q=1-p
Z = (p1-p2)/sqrt(p*q*(1/n1 + 1/n2))
pval= pnorm(Z,lower.tail = TRUE);
pval=round(pval,digits = 5); Z=round(Z,digits = 5)
print(c(Z,pval))
```

```r
# Distributions under H0 and H1 when n1=n2=11000
alpha=0.05
p_diff = -0.01
se = sqrt( p1*(1-p1)/n1 + p2*(1-p2)/n2 )
ncp = p_diff/se

x=seq(-10,4,0.01);
density0=dnorm(x,mean=0)
density1=dnorm(x,mean=ncp)

plot(x,density0,type="l",lwd=3,ylab="density",xlab="Z",cex.axis=0.7,main="Density Functions")
points(x,density1,type="l",col="red",lwd=3)
abline(v=0,h=0);

cr0=qnorm(alpha)                    # Critical value

# Region for Prob(Type I error)
region.x=x[ x  < cr0]
region.y=density0[ x < cr0]
region.x=c(cr0,region.x,tail(region.x,1))
region.y=c(0,region.y,0)
polygon(region.x,region.y,density=-1,col="gray")

# Region for Power
region.x=x[ x < cr0]
region.y=density1[ x < cr0]
region.x=c(head(region.x,1),region.x,cr0)
region.y=c(0,region.y,0)
polygon(region.x,region.y,density=10,col="red")
```

```r
# Distributions under H0 and H1 when n1=n2=2500
n1=2500; n2=2500
p1=0.01; p2=0.02
p_diff = -0.01
se = sqrt( p1*(1-p1)/n1 + p2*(1-p2)/n2 )
```

```
ncp = p_diff/se

x=seq(-8,4,0.01);
density0=dnorm(x,mean=0)
density1=dnorm(x,mean=ncp)

plot(x,density0,type="l",lwd=3,ylab="density",xlab="Z",cex.axis=0.7,main="Density Functions")
points(x,density1,type="l",col="red",lwd=3)
abline(v=0,h=0)

cr0=qnorm(0.072)                    # Critical value

# Region for Prob(Type I error)
region.x=x[ x  < cr0]
region.y=density0[ x < cr0]
region.x=c(cr0,region.x,tail(region.x,1))
region.y=c(0,region.y,0)
polygon(region.x,region.y,density=-1,col="gray")

# Region for Prob(Type II error)
region.x=x[ x > cr0]
region.y=density1[ x > cr0]
region.x=c(head(region.x,1),region.x,cr0)
region.y=c(0,region.y,0)
polygon(region.x,region.y,density=20,col="red")
```

```
OptSig.2p(ncp=ncp,p=0.5,k=1,alternative="less",Figure=TRUE)
```

Performance versus aptitude test scores

```
# data
score=c(59,47,58,66,77,57,62,68,69,36,48,65,51,61,40,67,60,56,76,71)
rating=c(3,2,4,3,2,4,3,3,5,1,3,3,2,3,3,4,2,3,3,5)
data=data.frame(score,rating)

# Plot and test statistic
fit=lm(rating ~ score, data = data)
plot(data$score,data$rating,xlab="Test score", ylab="Ratings")
abline(coef = coef(fit),col="red", lwd=2)
cor.test(data$score,data$rating,method="spearman")
```

```
n=nrow(data)
rs=cor(data$score,data$rating,method="spearman")
z=sqrt(n-1)*rs
pval= 2*pnorm(z,lower.tail = FALSE)
pval=round(pval,digits = 5); z=round(z,digits = 5)
print(c(z,pval))
```

```
# Optimal significance level
Opt.sig.norm.test(ncp=2.1794,p=0.5,k=1,alternative="greater",Figure = TRUE)
```

```
Opt.sig.norm.test(ncp=2.1794,p=0.2,k=5,alternative="greater",Figure=TRUE)
```