Introductory Statistics with R: Simple Inferences for continuous data

Statistical Packages

STAT 1301 / 2300, Fall 2014

Sungkyu Jung
Department of Statistics
University of Pittsburgh

E-mail: sungkyu@pitt.edu
http://www.stat.pitt.edu/sungkyu/stat1301/

1. Data Exploration and Graphics
2. One- and Two-Sample Tests
3. Power and the computation of sample size

Section 1

Data Exploration and Graphics

Summary Statistics

- Data analysis ALWAYS begins with an inspection of the data entry (e.g., using `head(dataset.name)` and `str(dataset.name)`)
- To calculate the mean, standard deviation, variance, median and quantiles.

```
> x <- rnorm(50)
> mean(x)
[1] 0.03301363
> sd(x)
[1] 1.069454
> var(x)
[1] 1.143731
> median(x)
[1] -0.08682795
> quantile(x)
          0%          25%          50%          75%         100%
-2.60741896 -0.54495849 -0.08682795 0.70018536 2.98872414
```

Exercise: Obtain all *deciles* and *percentiles*.

Summary Statistics

use `na.rm` argument (if *n*ot *a*vailable, then *rem*ove) for missing values

```
> attach(juul)
> mean(igf1)
[1] NA
> mean(igf1,na.rm=T)
[1] 340.168
```

`summary` function: to obtain summaries of any objects

```
> summary(igf1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   25.0   202.2   313.5   340.2   462.8   915.0     321
> summary(juul)
      age            menarche          sex             igf1
 Min.   : 0.170   Min.   :1.000   Min.   :1.000   Min.   : 25.0
 1st Qu.: 9.053   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:202.2
 Median :12.560   Median :1.000   Median :2.000   Median :313.5
 Mean   :15.095   Mean   :1.476   Mean   :1.534   Mean   :340.2
 3rd Qu.:16.855   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:462.8
 Max.   :83.000   Max.   :2.000   Max.   :2.000   Max.   :915.0
 NA's   :5        NA's   :635     NA's   :5       NA's   :321
    tanner        testvol
 Min.   :1.00   Min.   : 1.000
 1st Qu.:1.00   1st Qu.: 1.000
 Median :2.00   Median : 3.000
 Mean   :2.64   Mean   : 7.896
 3rd Qu.:5.00   3rd Qu.:15.000
 Max.   :5.00   Max.   :30.000
 NA's   :240    NA's   :859
```

summary function for factor data type

The data set has menarche, sex, and tanner coded as numeric variables even though they are clearly categorical. This can be mended as follows:

```
> detach(juul)
> juul$sex <- factor(juul$sex,labels=c("M","F"))
> juul$menarche <- factor(juul$menarche,labels=c("No","Yes"))
> juul$tanner <- factor(juul$tanner,
+                       labels=c("I","II","III","IV","V"))
> attach(juul)
> summary(juul)
      age          menarche     sex           igf1         tanner
 Min.   : 0.170   No  :369   M   :621   Min.   : 25.0   I   :515
 1st Qu.: 9.053   Yes :335   F   :713   1st Qu.:202.2   II  :103
 Median :12.560   NA's:635   NA's:  5   Median :313.5   III : 72
 Mean   :15.095                         Mean   :340.2   IV  : 81
 3rd Qu.:16.855                         3rd Qu.:462.8   V   :328
 Max.   :83.000                         Max.   :915.0   NA's:240
 NA's   :5                              NA's   :321
```

Exercise: Use the `transform` function (or `within`) to convert menarche, sex, and tanner variables in the `juul` data set as factors.

Histograms: `hist` function

```
> hist(numeric.vector)
> hist(numeric.vector, ...)
```

- option `breaks = n`, you get *approximately n* cells (bars)
- option `breaks = vector`, `vector` giving the breakpoints between histogram cells
- option `freq = TRUE`, you get frequencies (counts), `freq = FALSE`, you get densities ($count/n$)

Smoothed Histograms: `density` function

Smoothed Histgram is an estimate of the true density function

```
> xd = density(juul$igf1,na.rm = T)
> str(xd)
List of 7
 $ x         : num [1:512] -90.6 -88.4 -86.2 -84 -81.8 ...
 $ y         : num [1:512] 4.84e-07 5.84e-07 7.03e-07 8.43e-07 1.01
 $ bw        : num 38.5
 $ n         : int 1018
 $ call      : language density.default(x = juul$igf1, na.rm = T)
 $ data.name: chr "juul$igf1"
 $ has.na    : logi FALSE
 - attr(*, "class")= chr "density"
> plot(xd$x,xd$y)
```

Comparing Distributions

Quantile-Quantile plot: To directly compare data dist'n with theoretical normal dist'n

```
> qqnorm(numeric.vector)
```

Exercise: Compare 100 random normal (or exponential) numbers with the normal distribution.

Box plots: Useful for side-by-side comparison

```
> par(mfrow=c(1,2))
> boxplot(juul$igf1)
> boxplot(log(juul$igf1))
> par(mfrow=c(1,1))
> boxplot(iris[,1:4])
```

Summary Statistics by Groups

Grouped Data

- Numeric data grouped (categorized) by levels of factors
- Typical for data aimed for two-sample t-test and ANOVA

```
> str(red.cell.folate)
'data.frame': 22 obs. of  2 variables:
 $ folate     : num  243 251 275 291 347 354 380 392 206 210 ...
 $ ventilation: Factor w/ 3 levels "N2O+O2,24h","N2O+O2,op",..: 1
```

tapply function

Work with numeric (or integer) data type as a grouping variable

```
> attach(red.cell.folate);attach(juul)
> tapply(folate,ventilation,mean)
 N2O+O2,24h  N2O+O2,op     O2,24h
   316.6250   256.4444   278.0000
> tapply(age, tanner, mean, na.rm=T)
        1         2         3         4         5
 8.195456 12.316019 12.989306 14.746667 17.878872
```

aggregate function

Splits the data into subsets, computes summary statistics for each, and returns the result in a convenient form.

```
> aggregate(object, by = list.object, FUN, ...)
> aggregate(formula, data = data.frame, FUN, ...)
```

- `object`: a vector or data frame
- `list.object`: a `list` or `data frame` of grouping elements
- `FUN`: `mean`, `median`, etc.
- `formula`: R formula, see example below.

```
> aggregate(juul, list(Sex = sex,Tanner = tanner),
            mean, na.rm = T)
> aggregate(igf1 ~ sex + tanner, data = juul, mean, na.rm = T)
```

by function: Apply a Function to a Data Frame Split by Factors

```
> by(data.frame, INDICES, FUN, ...)
```

- `INDICES`: factor indices (need not to be factor data type)
- `FUN`: function name, which can be applied to the data frame

Examples:

```
by(juul, as.factor(tanner), summary)
by(juul, tanner, summary)
by(juul, juul["tanner"], summary)
```

Top-Bottom Histograms

```
> attach(energy)
> expend.lean <- expend[stature=="lean"]
> expend.obese <- expend[stature=="obese"]
> par(mfrow=c(2,1))
> hist(expend.lean,breaks=10,xlim=c(5,13),ylim=c(0,4),col="white")
> hist(expend.obese,breaks=10,xlim=c(5,13),ylim=c(0,4),col="grey")
> par(mfrow=c(1,1))
```

Parallel Boxplot

```
> boxplot(expend.lean,expend.obese)
> boxplot(expend ~ stature, data = energy)
> boxplot(expend ~ stature)
```

Section 2

One- and Two-Sample Tests

One Sample T-Test: `t.test` function

Assume $X_1, \ldots, X_n \sim$ i.i.d. $N(\mu, \sigma^2)$,

```
> daily.intake <- c(5260,5470,5640,6180,6390,6515,
+ 6805,7515,7515,8230,8770)
> t.test(daily.intake)
> t.test(daily.intake,mu=7725)
> t.test(daily.intake,mu=7725,alt = "l")
```

The t tests are fairly robust against departures from the normal distribution especially in larger samples. For small sample size, assuming only that the distribution is symmetric around μ,

Wilcoxon signed-rank test: `wilcox.test` function

```
> wilcox.test(daily.intake, mu=7725)
> wilcox.test(daily.intake, mu=7725, alt = "g")
```

Two-Sample t test: `t.test`, `var.test` functions

- For comparison of two vectors

```
t.test(expend.lean,expend.obese, var.equal = TRUE)
t.test(expend.lean,expend.obese, var.equal = FALSE)
t.test(expend.lean,expend.obese, var.equal = FALSE, alt = "g")
var.test(expend.lean,expend.obese)
```

- Using the model formula

```
t.test(expend ~ stature, var.equal = T)
var.test(expend ~ stature)
```

Wilcoxon rank sum test: `wilcox.test` function

```
wilcox.test(expend.lean,expend.obese)
wilcox.test(expend~stature)
```

Paired difference test

Paired tests are used when there are two measurements on the same experimental unit. The theory is essentially based on taking differences and thus reducing the problem to that of a *one-sample test*.

The paired t test

```
attach(intake)
pre - post
t.test(pre,post,paired = T)
t.test(pre,post,mu = 1100,paired = T)
t.test(pre,post,mu = 1100,paired = T,alternative = "g")
t.test(pre-post, mu = 1100, alternative = "g")
```

The matched-pairs Wilcoxon test

```
wilcox.test(pre,post,paired = T)
wilcox.test(pre,post, mu = 1100,paired = T)
wilcox.test(pre,post, mu = 1100,paired = T, alternative = "g")
```

Saving the result of analysis

```
> ttestresult <- t.test(pre,post,paired = T)
> ttestresult

Paired t-test

data:  pre and post
t = 11.9414, df = 10, p-value = 3.059e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1074.072 1566.838
sample estimates:
mean of the differences
                1320.455

> ttestresult$statistic
       t
11.94139
> ttestresult$p.value
[1] 3.059021e-07
```

Section 3

Power and the computation of sample size

The principles of power calculations

When designing experiments, the experimenter should try to ensure that a sufficient amount of data are collected to be reasonably sure that a difference of a specified size will be detected.

A simple example: Z-test

Test $H_0 : \mu = \mu_0$ based on the sample $X_1, \ldots, X_n \sim N(\mu, 1)$, $\sigma = 1$ known. With alternative $H_1 : \mu > \mu_0$, a sensible test is the Z-test, where H_0 is rejected if

$$\sqrt{n}(\bar{X} - \mu_0) > z_{1-\alpha}$$

Then

- Prob. of Type I error: $P_{\mu_0}(\sqrt{n}(\bar{X} - \mu_0) > z_{1-\alpha}) = \alpha$
- Power at $\mu > \mu_0$ ($\pi(\mu)$): Probability that the test detects the difference when the true difference is μ

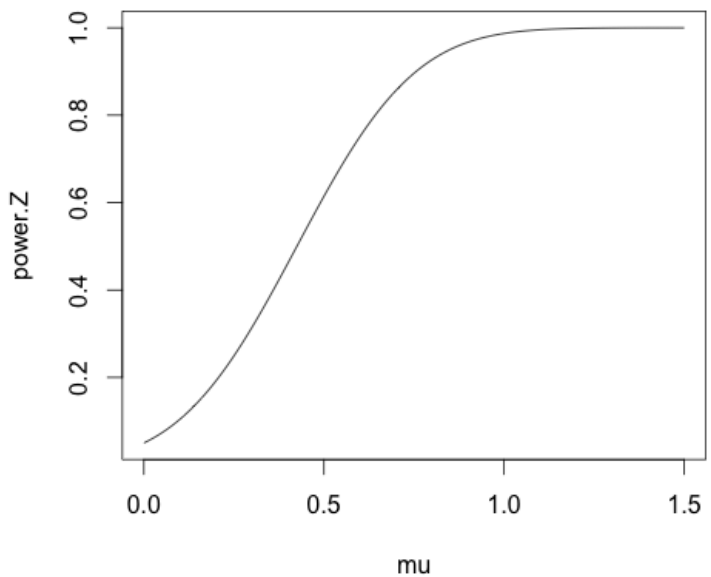
$$\begin{aligned}
\pi(\mu) &= P_\mu(\sqrt{n}(\bar{X} - \mu_0) > z_{1-\alpha}) \\
&= P(Z > z_{1-\alpha} - \sqrt{n}(\mu - \mu_0)), \ Z \sim N(0,1) \\
&= 1 - \Phi(z_{1-\alpha} - \sqrt{n}(\mu - \mu_0))
\end{aligned}$$

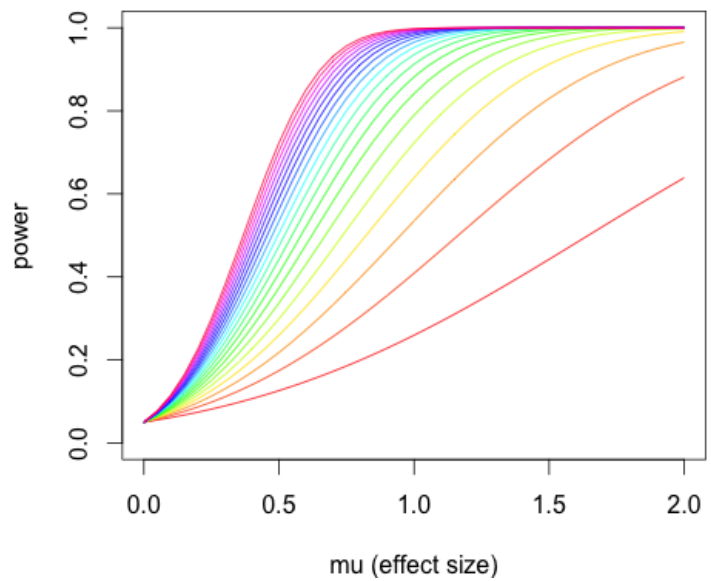
Notice that the power depends on $\mu - \mu_0$ (effect size), n (sample size) and α (significance level).

The principles of power calculations

For fixed sample size $n = 15$, the power $\pi(\mu)$ as a function of $\mu > \mu_0 = 0$ can be evaluated as follows:

```
n = 15
mu0 = 0
alpha = 0.05
mu = seq(0,1.5,by = 0.1)
power.Z = (1-pnorm(qnorm(1-alpha) -sqrt(n)*(mu-mu0)))
plot(mu, power.Z, type = "l")
```

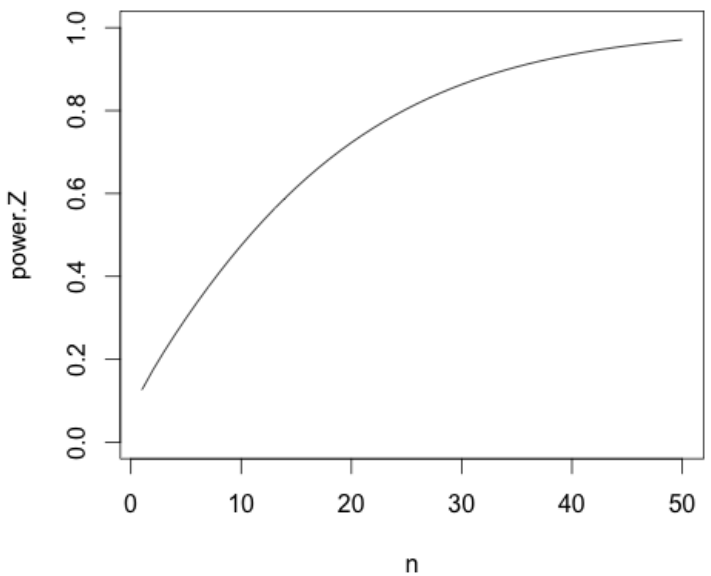




Power as a function of sample size

For a fixed (desired) effect size μ, power is also a function the sample size n.

```
mu = 0.5
mu0 = 0
alpha = 0.05
n = 1:50
power.Z = (1-pnorm(qnorm(1-alpha) -sqrt(n)*(mu-mu0)))
plot(n, power.Z, type = "l", ylim = c(0,1))
```



Sample size computation

For a fixed (desired) effect size $\mu - \mu_0$, how large n is needed to ensure that the power is greater than (say) 0.9?

$$\begin{aligned}
\pi(\mu) = 1 - \Phi(z_{1-\alpha} - \sqrt{n}(\mu - \mu_0)) &> 0.9 \\
\Leftrightarrow \Phi(z_{1-\alpha} - \sqrt{n}(\mu - \mu_0)) &< 0.1 \\
\Leftrightarrow z_{1-\alpha} - \sqrt{n}(\mu - \mu_0) &< \Phi^{-1}(0.1) = z_{0.1} \\
\Leftrightarrow n &> \left(\frac{z_{1-\alpha} - z_{0.1}}{\mu_1 - \mu_0} \right)^2
\end{aligned}$$

```
> mu = 0.5
> mu0 = 0
> alpha = 0.05
> nn<-((-qnorm(1-0.9)+ qnorm(1-alpha))/(mu-mu0))^2
> ceiling(nn)
[1] 35
```

Power and Sample Size Calculation for t test

One sample t test

Test $H_0 : \mu = \mu_0$ based on the sample $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$, σ unknown. With alternative $H_1 : \mu > \mu_0$, the one-sample t test rejects H_0 if

$$\frac{\bar{X} - \mu_0}{s/\sqrt{n}} > t_{n-1,1-\alpha}$$

Then

- Power at $\mu > \mu_0$ ($\pi(\delta)$): Probability that the test detects the positive effect size $\delta = \mu - mu_0$.

$$\begin{aligned}\pi(\delta) &= P_\delta\left(\frac{\bar{X} - \mu_0}{s/\sqrt{n}} > t_{n-1,1-\alpha}\right) \\ &= P\left(\frac{(\bar{X} - \mu) + \delta}{s/\sqrt{n}} > t_{n-1,1-\alpha}\right),\end{aligned}$$

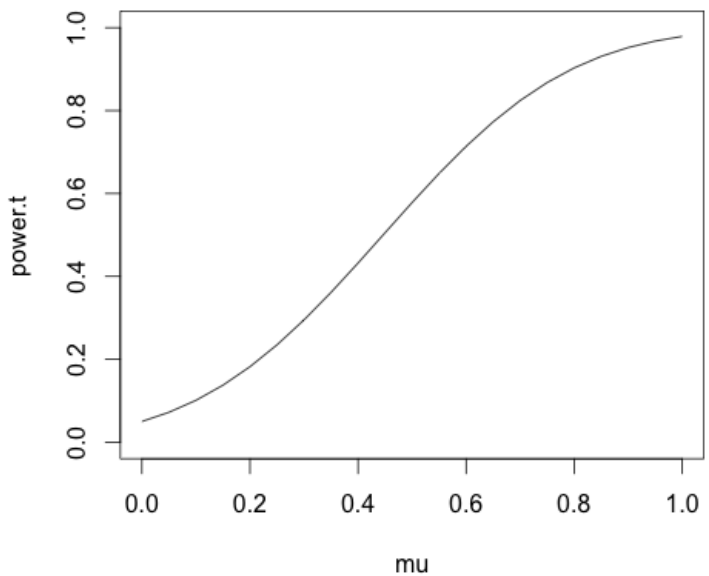
  where $T = \frac{(\bar{X}-\mu)+\delta}{s/\sqrt{n}}$ has the *non-central t distribution* with d.f. $n-1$ and noncentrality parameter $\nu = \sqrt{n}\delta/\sigma$.

Notice that the power depends on δ (effect size), σ (population s.d.), n (sample size) and α (significance level).

Power and Sample Size Calculation for t test

For fixed sample size $n = 15$, the power $\pi(\delta)$ as a function of $\delta > 0$ can be evaluated as follows:

```
n = 15
mu0 = 0
alpha = 0.05
mu = seq(0,1,by = 0.05)
sigma = 1;
t.ncp <- sqrt(n)*(mu-mu0)/sigma
power.t =  1- pt( qt(1-alpha,n-1), n-1 ,ncp = t.ncp )
plot(mu, power.t, type = "l", ylim = c(0,1))
```



Power calculation for one- and two-sample t test

power.t.test function

```
power.t.test(n = NULL, delta = NULL, sd = 1, sig.level = 0.05,
             power = NULL,
             type = c("two.sample", "one.sample", "paired"),
             alternative = c("two.sided", "one.sided"))
```

- Exactly one of the parameters `n`, `delta`, `power`, `sd` and `sig.level` must be passed as `NULL`, and that parameter is determined from the others.

Example:

```
power.t.test(n = 15, delta = 1, sd = 1,
             type = "one.sample", alternative = "one.sided")
power.t.test(power = 0.9, delta = 1, sd = 1,
             type = "one.sample", alternative = "two.sided")
power.t.test(power = 0.9, delta = 1, sd = 3,
             type = "paired", alternative = "two.sided")
```

Exercise

1. Use `power.t.test` function to reproduce the plot in page 26.

2. Use `power.t.test` function to draw the graph of one-sample t-test power as a function of sample size.

3. This example is from Altman (1991, p. 457) and concerns the influence of milk on growth. Two groups are to be given different diets, and their growth will be measured. We wish to compute the sample size that, with a power of 90%, a two-sided test at the 1% level can find a difference of 0.5 cm in a distribution with a standard deviation of 2 cm.

4. Under which assumptions is the use of `power.t.test` function valid?

Remark on Power calculation

- Basic R distribution offers two more power calculation functions. Find out by referring to their help pages.

```
> ?power.prop.test
> ?power.anova.test
```

- Power and sample size calculation in general cases may be handled by using independent packages (such as `pwr` package), or by using other specialized software (such as PASS Software, NCSS, LLC.)