Statistics, Random Numbers and Stochastic Simulation

Richard Upton
7 August 2015

Statistics in R

There are plenty of textbooks on this!

Here are the very basics

```
> statdata <- read.csv("stats_data.csv")
> statdata$AGEBIN <- as.factor(statdata$AGEBIN)
> head(statdata)
```

```
  ID DOSE ROUTE AGE AGEBIN WEIGHT
1  1   50     0  32      0     75
2  2   50     0  79      1     54
3  3   50     0  60      1     90
4  4   50     0  79      1     63
5  5   50     0  25      0     66
6  6   50     1  64      1     88
```

Statistics - Linear regression

The variables are both continuous

```
> result1 <- lm(WEIGHT~AGE, data=statdata)
```

Statistics - Linear regression

```
Call:
lm(formula = WEIGHT ~ AGE, data = statdata)

Residuals:
   Min     1Q Median     3Q    Max
 -9.29  -7.56  -1.47   4.32  22.79

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   80.039      6.360   12.58  2.3e-10 ***
AGE           -0.214      0.116   -1.85    0.081 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.16 on 18 degrees of freedom
Multiple R-squared:  0.16,  Adjusted R-squared:  0.113
F-statistic: 3.42 on 1 and 18 DF,  p-value: 0.0808
```

Statistics - Analysis of Variance 1

Fact: ANOVA is a type of linear regression

The independent variable is a factor

It represented internally by "contrasts" - 0 or 1

```
> result2 <- lm(WEIGHT~AGEBIN, data=statdata)
```

Statistics - Analysis of Variance 1

```
Call:
lm(formula = WEIGHT ~ AGEBIN, data = statdata)

Residuals:
   Min     1Q Median     3Q    Max
-14.00  -6.00  -4.29   4.93  22.00

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    70.57       3.74   18.85  2.7e-13 ***
AGEBIN1        -2.57       4.64   -0.55     0.59
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.9 on 18 degrees of freedom
Multiple R-squared:  0.0168,    Adjusted R-squared:  -0.0379
F-statistic: 0.307 on 1 and 18 DF,  p-value: 0.587
```

Statistics - Analysis of Variance 2

This produces the same result

```
> result3 <- aov(WEIGHT~AGEBIN, data=statdata)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
AGEBIN        1     30    30.1    0.31   0.59
Residuals    18   1766    98.1
```

Statistical output

The statistical output is stored as a list object

Lists can be nested structures of mixed data types

```
> names(result1)
```

```
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

```
> result1$coefficients["AGE"]
```

```
    AGE
-0.2138
```

Repeated measures data

Use mixed-effect linear regression (lme)

This is analogous to NONMEM with fixed and random effects

```
> library(nlme)
> #lme(WEIGHT~AGEBIN, random= ~1|ID, data=repdata)
```

Stochastic simulation

The simulation of a system that has random components

Randomness comes from a random number generator

The random numbers can be modified into various distributions

- Normal
- Log-normal
- Uniform
- Bionomial

Simulation in pharmacometrics

All simulations from population models are stochastic

The stochastic components (ETA, EPS) of the model are recreated by sampling from distributions of random numbers

Examples

- Visual Predictive Check
- Simulations of model prediction intervals
- Clinical trial simulation
- Simulations of study power

Generating random numbers

Many computer algorithms have been developed

Pseudo-random numbers will eventually repeat

Testing for randomness is complicated

- plot(runif(10000) ~ runif(10000))
- estimate pi

Bad algorithims repeat quickly and may be autocorrelated

- Excel had problems

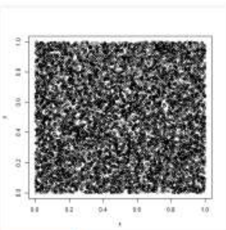
Random Number Table

	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21 22 23 24	25 26 27 28	29 30 31 32
1	8 0 9 4	2 5 2 5	8 2 4 7	1 3 4 7	7 4 3 8	3 8 2 0	1 8 9 7	2 1 3 4
2	3 5 6 0	2 1 9 8	8 2 1 1	0 0 4 5	2 6 1 8	2 7 9 1	2 0 2 7	1 0 9 5
3	1 3 3 0	6 3 3 1	3 7 6 3	9 6 9 3	8 7 3 8	8 8 1 5	1 3 3 6	8 5 4 3
4	3 6 6 5	0 0 1 6	2 2 4 3	6 4 3 2	4 7 9 6	6 0 9 5	5 2 0 3	1 0 2 0
5	7 8 5 0	5 9 2 6	5 6 6 8	7 9 1 1	2 1 9 2	4 5 4 5	3 5 3 0	5 5 9 9
6	4 4 9 0	5 4 1 7	9 7 2 7	6 1 6 3	5 9 9 1	4 8 7 8	9 9 8 0	9 8 7 7
7	6 6 4 6	9 1 0 4	9 3 1 8	8 8 1 9	7 5 3 7	2 7 8 8	8 3 7 9	2 4 4 5
8	9 6 2 6	5 8 9 6	1 2 1 5	9 7 5 3	9 2 2 3	6 6 5 6	2 9 4 4	2 8 9 9
9	4 6 6 5	4 8 2 0	7 5 5 4	0 6 1 2	9 6 8 3	4 2 5 1	9 1 3 6	1 7 0 9
10	6 4 9 8	7 6 1 9	0 4 7 4	7 8 1 8	6 8 3 2	9 6 8 3	9 8 7 2	4 0 9 0
11	6 7 2 2	8 8 6 9	9 3 6 1	7 6 7 5	4 8 8 3	1 3 1 5	9 6 7 9	8 8 3 4
12	9 7 4 8	5 6 3 2	5 1 1 5	2 7 2 1	0 0 3 3	9 3 0 3	9 7 1 3	4 0 1 2
13	5 6 4 1	1 4 1 7	1 4 1 9	7 4 3 4	8 1 4 6	7 3 6 8	1 2 1 6	5 0 3 9
14	7 4 4 4	9 2 0 0	8 8 4 0	5 8 8 2	4 3 9 8	3 9 0 4	9 1 9 9	9 3 3 8
15	8 2 7 9	3 0 1 9	4 6 7 2	3 7 4 3	9 9 7 9	4 6 6 9	9 0 2 1	6 9 9 0
16	0 1 6 1	7 8 1 7	1 0 2 4	2 3 6 7	2 8 9 1	8 6 7 7	1 5 8 5	2 4 8 2
17	7 3 6 8	9 7 5 9	7 5 5 5	6 6 2 4	9 9 7 7	2 0 0 8	5 5 9 6	9 7 4 0
18	7 8 3 0	4 7 1 4	3 6 9 6	2 9 1 9	1 8 0 4	4 0 4 4	1 0 3 4	2 6 9 7
19	9 8 8 7	4 2 1 6	8 5 2 6	4 5 3 5	8 4 3 0	5 2 7 0	9 6 0 5	0 7 5 8
20	1 2 6 1	2 5 1 6	8 5 6 9	2 3 1 0	3 9 3 9	8 7 0 3	9 8 4 1	0 3 5 3
21	3 9 4 7	4 9 3 7	7 6 3 4	2 5 4 3	6 2 3 9	7 4 9 5	2 0 5 5	7 7 9 5
22	4 5 5 6	0 1 0 3	1 2 5 0	2 3 0 4	1 1 2 8	9 7 8 0	5 1 4 4	4 5 2 6
23	1 3 4 4	0 6 9 7	2 3 6 3	8 9 7 6	6 2 5 1	4 2 0 1	2 0 3 8	8 6 5 2
24	8 8 7 6	5 8 2 3	8 4 8 7	0 4 5 0	3 1 0 0	9 1 6 0	2 7 1 7	7 6 0 1
25	7 7 1 0	9 9 4 3	6 9 7 8	8 2 7 3	9 7 1 4	9 7 0 0	1 5 6 6	2 8 6 9
26	0 9 5 5	6 0 0 8	8 4 4 2	2 2 8 2	1 5 2 4	2 5 1 7	5 8 1 8	0 0 8 1
27	7 9 4 1	2 3 1 2	2 4 3 1	6 7 0 2	9 9 8 4	5 4 6 0	3 0 8 8	4 7 6 2
28	2 2 8 4	0 8 9 8	9 1 0 7	5 6 4 2	7 3 1 0	3 7 8 2	1 0 6 8	9 5 7 4
29	8 5 9 4	7 4 1 8	8 3 6 5	6 0 4 5	1 1 6 3	6 9 1 6	9 5 9 0	1 1 4 3
30	4 6 1 0	8 5 4 9	6 3 6 9	3 2 0 8	5 1 0 9	9 6 8 0	1 1 0 8	6 1 3 3

A test of randomness

x and y are 10,000 independent random uniform numbers

```
x <- runif(10000)
y <- runif(10000)
plot(y ~ x)
```



Why simulate in R rather than NONMEM?

In NONMEM, doses and covariates are coded in the database

If you want to change these in a simulation, you make a new database

If it's complex simulation, this is usually done with R

And R might be used to plot the simulated data anyway

So cut out the middle man and use R for everything!

It may (sometimes!) be faster and easier

Generating random numbers in R

R has a family of functions for random numbers
see ?Distributions

function	distribution	examples
rnorm	normal	additive residual error, PD baseline
rlnorm	log-normal	clearance, distribution volume
runif	uniform	age
rbinom	binomial	sex, genotype

Related functions in R

For each class of distribution

function	distribution	comment
dnorm(x, mean=0, sd=1)	gives the density of x	used in maximum likelihood estimation
pnorm(x, mean=0, sd=1)	gives the distribution function	turns x into a probability
qnorm(p, mean=0, sd=1)	gives the quantile function	turns probability into a quantile
rnorm(n, mean=0, sd=1)	generate n random numbers	from a distribution with mean=0 and sd=1

Using the rnorm function

```
> randomn <- rnorm(n=10, mean=2, sd=0.5)
> randomn
```

```
 [1] 3.230 2.229 2.579 1.762 1.974 1.952 1.454 2.980 2.892 2.219
```

```
> mean(randomn)
```

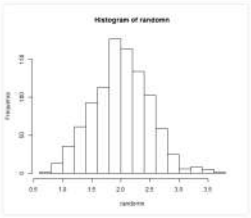
```
[1] 2.247
```

```
> sd(randomn)
```

```
[1] 0.5441
```

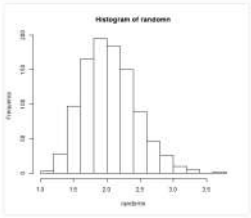
Using the rnorm function

```
> randomn <- rnorm(n=1000, mean=2, sd=0.5)
> hist(randomn)
```



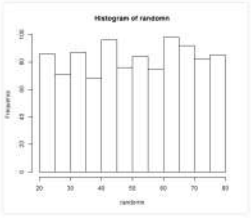
Using the rlnorm function

```
> randomn <- rlnorm(n=1000, mean=log(2), sd=0.2)
> #Note sd is now a ratio
> hist(randomn)
```



Using the runif function

```
> randomn <- runif(n=1000, min=20, max=80)
> hist(randomn, breaks=10)
```



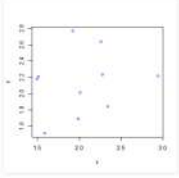
Using the rbinom function

```
> randomn <- rbinom(n=1000, size=1, prob=0.25)
> table(randomn)
```

```
randomn
  0   1
761 239
```

A simulation to understand randomness

```
> x <- rlnorm(n=10, mean=log(2), sd=0.2)
> y <- rlnorm(n=10, mean=log(2), sd=0.2)
> plot(y ~ x, col="blue")
```



We are programmed to see patterns!

Our job as scientists is to distinguish information from randomness

$p < 0.05$ means we are happy to fooled by randomness 1 time out of 20!

For low powered studies, focus on effect size and uncertainty

p-values are for *confirming* not *learning*

Setting a seed for reproducible random numbers!

```
> #No seed set
> rnorm(n=3, mean=2, sd=0.5)
```

```
[1] 2.162 1.952 1.797
```

```
> #No seed set
> rnorm(n=3, mean=2, sd=0.5)
```

```
[1] 2.218 2.603 1.539
```

Setting a seed for reproducible random numbers!

```
> set.seed(123)
> rnorm(n=3, mean=2, sd=0.5)
```

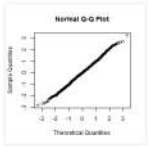
```
[1] 1.720 1.885 2.779
```

```
> set.seed(123)
> rnorm(n=3, mean=2, sd=0.5)
```

```
[1] 1.720 1.885 2.779
```

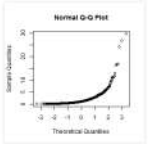
Testing normality - quantile-quantile plots

```
> x <- rnorm(1000)
> qqnorm(x)
```



Testing normality - quantile-quantile plots

```
> y <- rlnorm(1000)
> qqnorm(y)
```



Log-normal distributions in biology

Normal distributions arise from additive process
c(mean+1, mean+3, mean+0, mean-2, mean+2)

Log-normal distributions arise from multiplicative processes
c(mean*2, mean/3, mean*1, mean/2, mean*4)

Log-normal distributions are common in biological systems

No zero values, right skewed, occasional high values

Described by the geometric mean, the standard deviation is a ratio

If x is normal, log(x) is log-normal
If x is log-normal, exp(x) is normal

Truncated & Censored distributions

A metric may have normal distribution

But our ability to measure the metric might be censored:

- LLOQ of an assay
- Ethical limits for thermal pain tests

The distribution we measure is censored or truncated

This may need to replicated in a model

Non-normal distributions

Skewed

- Left (median > mean)
- Right (median < mean)

Kurtotic

- Unimodal (median = mean)
- Platykurtic, thin-tailed, peaky
- Leptokurtic, fat-tailed, flat

Multi-modal

- More than 1 peak

Transformed distributions

NONMEM simulations assume an underlying normal distribution for ETA

A transformation may replicate a skewed parameter distribution

Transformations try to "normalize" a distribution

- Log-normal
- Box-Cox
- Manly

See Petersson et al., *Pharm Res* 2009 26:2174-85

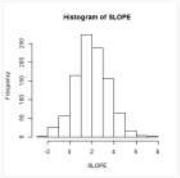
Additive Between Subject Variability

Effect = Baseline + Slope*Conc

```
> nsubs <- 1000
> SLOPEpop <- 2
> ETA <- rnorm(nsubs, mean=0, sd=1.5)
> SLOPE <- SLOPEpop + ETA
```

ETA is normally distributed
SLOPE is normally distributed
SLOPE can take negative values

```
> hist(SLOPE)
```



Exponential Between Subject Variability

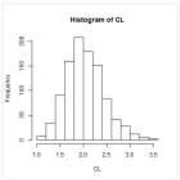
CL = CLpop*exp(ETA)

```
> nsubs <- 1000
> CLpop <- 2
> ETA <- rnorm(nsubs, mean=0, sd=0.2)
> CL <- CLpop*exp(ETA)
```

ETA is normally distributed
CL is log-normally distributed
CL can't take negative values
SD of CL is 0.2
VAR of CL is $0.2^2 = 0.04$

Exponential Between Subject Variability

```
> hist(CL)
```



Additive Residual Error

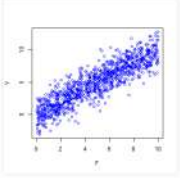
Y = F + EPS
F is model predicted concentration (fake with uniform random numbers!)

```
> nobs <- 1000
> F <- runif(nobs, min=0, max=10)
> EPS <- rnorm(nobs, mean=0, sd=1.5)
> Y <- F + EPS
```

Y (DV) is F with RUV
F is never negative
Y has negative values
shape = "tram tracks"

Additive Residual Error

```
> plot(Y ~ F, col="blue")
```



Proportional Residual Error

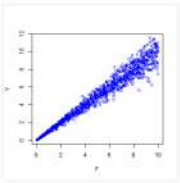
Y = F*(1 + EPS) or Y = F + F*EPS

```
> nobs <- 1000
> F <- runif(nobs, min=0, max=10)
> EPS <- rnorm(nobs, mean=0, sd=0.1)
> Y <- F*(1 + EPS)
```

Y (DV) is F with RUV
F is never negative
Y is never negative
(unless sd is large!)
shape = "cone"

Proportional Residual Error

```
> plot(Y ~ F, col="blue")
```



Additive and Proportional Residual Error

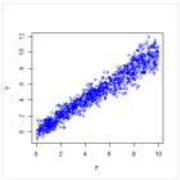
Y = F*(1 + EPS1) + EPS2

```
> nobs <- 1000
> F <- runif(nobs, min=0, max=10)
> EPS1 <- rnorm(nobs, mean=0, sd=0.1)
> EPS2 <- rnorm(nobs, mean=0, sd=0.5)
> Y <- F*(1 + EPS1) + EPS2
```

Y (DV) is F with RUV
F is never negative
Y can be negative
(at low concentrations)
shape = "tramtracks+cone"

Additive and Proportional Residual Error

```
> plot(Y ~ F, col="blue")
```



How many simulations?

Enough!

Simulations are repeated until the effect of randomness on summary statistics (e.g. mean and CI) are minimal

Some rules of thumb:

- 200 times for a mean
- 1,000 times for a confidence interval
- 10,000 times for study power

Capacity may be limited by computer memory for big problems(64 bit helps)

Covariance

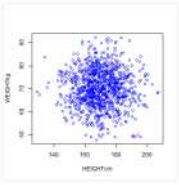
Covariance is a measure of how much two random variables change together

Not accounting for covariance may mean simulating implausible combinations of random numbers

```
> HEIGHTcm <- rnorm(1000, mean=170, sd=12)
> WEIGHTkg <- rnorm(1000, mean=70, sd=8)
```

Without Covariance

```
> plot(WEIGHTkg ~ HEIGHTcm, col="blue")
```



Covariance

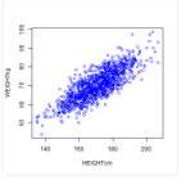
Covariance in NONMEM comes from $OMEGA BLOCK

Simulating correlated random numbers is complex

```
> library(MASS)
> OMEGA <- matrix(c(150,80,40,65),2,2)
> result <- mvrnorm(n=1000, mu=c(170,70), OMEGA)
> HEIGHTcm <- result[,1]
> WEIGHTkg <- result[,2]
```

With Covariance

```
> plot(WEIGHTkg ~ HEIGHTcm, col="blue")
```



Summary

Generating random numbers shows the influence of randomness on data

- Use simulation to educate yourself about how randomness affects your data

Random numbers are at the heart of every population model

- Use simulation to educate yourself about how randomness affects your model

Study design and study power are moving toward simulation based methods