**23**

**Hàm R dùng cho phân tích**

Đối với các bạn mới làm quen với R, có lẽ vấn đề nổi cộm là lúng túng, không nhớ hàm và lệnh. Tôi hiểu khó khăn đó. Dưới đây là một bảng tổng hợp các hàm cần thiết cho đọc dữ liệu, biên tập dữ liệu, phân tích mô tả, và mô hình thống kê. Tôi sắp xếp theo thứ tự của chương sách. Mô hình sắp xếp hàm là việc làm và tên hàm, một số nơi tôi cho ví dụ cụ thể. Gần như tất cả các hàm chính yếu trong sách đều được tóm lược trong bảng này. Bạn đọc có thể dùng bảng này như là một tài liệu tham khảo khi cần biết hàm R nào để phân tích dữ liệu.

**Môi trường vận hành**

|  |  |
| --- | --- |
| **Phân tích / việc** | **Hàm R** |
| Cho biết directory hiện hành là gì | getwd() |
| Chuyển directory vận hành về c:\works | setwd(c:/works) |
| Đổi chiều rộng cửa sổ R thành 100 characters | options(width=100) |
| Đổi số thành 3 số thập phân (thay vì kiểu 1.2E-04) | options(scipen=3) |
| Cho biết các thông số về môi trường của R | options() |
| Liệt kê các đối tượng (objects) trong memory | ls() |
| Tìm kiếm | search() |
| Cộng, trừ, nhân, chia, luỹ thừa | + - \* / ^ |
| Chia số nguyên | %/% |
| Số dư từ chia 2 số nguyên | %% |
| **Kí hiệu logic** |  |
| Bằng | == |
| Không bằng | != |
| Nhỏ / lớn hơn | <, > |
| Nhỏ hoặc bằng, lớn hoặc bằng | <=, >= |
| Có phải là số missing value | is.na(x) |
| và (AND) | & |
| hoặc (OR) | | |
| Không là (NOT) | ! |
| **Ma trận** |  |
| Số đầu tiên của biến số x | x[1] |
| Năm số đầu tiên của biến số x | x[1:5] |
| Chọn x sao cho y nhỏ hơn hoặc bằng 30 | x[y<=30] |
| Chọn x sao cho sex bằng male | x[sex=="male"] |

**Đọc dữ liệu**

|  |  |
| --- | --- |
| **Phân tích / việc** | **Hàm R** |
| Đọc dữ liệu từ file dạng csv | read.csv() |
| Đọc dữ liệu từ file dạng text/ascii | read.table() |
| Đọc / nhập số liệu dạng tab delimited | read.delim() |
| Đọc từ SAS | library(foreign);  read.ssd() |
| Đọc từ SPSS | library(foreign);  read.spss() |
| Đọc từ Stata | library(foreign);  read.dta() |
| Tạo data frame | dat=data.frame(x1,x2,x3) |

**Mô phỏng**

|  |  |
| --- | --- |
| **Phân tích / việc** | **Hàm R** |
| Tạo dãy số -4, -3.5, -3.0, …, 3.0 | seq(-4, 3, 0.5) |
| Tạo dãy số 1, 2, 3, …, 10 | 1:10 |
| Nhập số 5, 7, 8, 1 vào biến x | x = c(5, 7, 8, 1) |
| Lặp lại số “1” 5 lần | rep(1, 5) |
| Lặp “A” 4 lần | rep("A", 4) |
| Yếu tố 3 bậc, lặp lại 2 lần, tổng cộng 12 số | gl(3, 2, 12) |
| **Mô phỏng** |  |
| Phân bố chuẩn (normal distribution) 1000 đối tượng, với trung bình = 0 và độ lệch chuẩn = 1 | rnorm(n=1000, mean=0, sd=1) |
| Phân bố mũ (exponential distribution) | rexp(n, rate=1) |
| Phân bố gamma | rgamma(n,shape,scale=1) |
| Phân bố Poisson | rpois(n, lambda) |
| Phân bố Weibull | rweibull(n,shape,scale=1) |
| Phân bố Cauchy | rcauchy(n,location=0,scale=1) |
| Phân bố beta | rbeta(n, shape1, shape2) |
| Phân bố t | rt(n, df) |
| Phân bố Khi bình phương | rchisq(n, df) |
| Phân bố nhị phân (binomial) | rbinom(n, size, prob) |
| Phân bố geometric | rgeom(n, prob) |
| Phân bố hypergeometric | rhyper(nn, m, n, k) |
| Phân phối log normal | rlnorm(n,meanlog=0,sdlog=1) |
| Phân phối logistic | rlogis(n,location=0,scale=1) |
| Phân phối negative Binomial | rnbinom(n,size,prob) |
| Phân phối uniform | runif(n,min=0,max=1) |

**Biên tập dữ liệu**

|  |  |
| --- | --- |
| **Phân tích / việc** | **Hàm R** |
| Tạo biến mới ($) | dat=data.frame(x1,x2);  dat$x3 = dat$x1 + dat$x2 |
| Mã hoá dữ liệu | dat$sex[gender=="male"] <- 1;  dat$sex[gender=="female"] <- 2 |
| Hoán chuyển dữ liệu | as.numeric(), as.character(), as.vector(), as.matrix(), as.data.frame |
| Data.frame như là ma trận | dat[,1]  dat[,2:3]  dat[2,]  dat[3:8,]  dat[1:5, 2:3] |
| Subset dữ liệu | dat2 = subset(dat, id<=5) |
| Sort dữ liệu | id = c(1:5)  name = c("A","B","C","D","E")  x = c(12, 45, 67, 32, 26)  dat = data.frame(id, name, x)  **new.dat = dat[order(x), ]**  **new.dat = dat[-order(dat$x), ]** |
| Merge dữ liệu | dat = merge(dat1, dat2, by="id");  dat = merge(dat1, dat2, by="id", all.x=T, all.y=T) |
| Chuyển từ cột sang dòng | id=c(1:4)  sex=c("M","F","M","F")  group=c(1,1,2,2)  day1=c(15,16,21,31)  day2=c(17,15,23,35)  day3=c(19,20,19,33)  dat=data.frame(id,sex,group,day1,day2,day3)  **require(reshape2)**  **dat1 = melt(dat,**  **id=c("id", "sex","group"),**  **measure.vars=c("day1", "day2", "day3"))** |
| Chuyển từ dòng sang cột | **require(reshape)**  **dat2 = cast(dat1,**  **id+sex+group ~ variable)** |

**Biểu đồ cơ bản trong R**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Biểu đồ phân bố (histogram) | hist(x, breaks=20, col="blue", border="white")  # Thêm density  lines(density(na.omit(x)), col="red", lwd=2) |
| Biểu đồ hộp (box plot) | boxplot(hdl, notch=T, col="blue")  # Biểu đồ cho theo nhóm sex  boxplot(x ~ sex, col="blue", notch=T, names=c("Men", "Women") |
| Biểu đồ tương quan (scatterplot) | plot(y ~ x, pch=16)  # dùng package car  scatterplot(y ~ x, pch=16, smooth=F, col="blue", lwd=2)  # Biểu đồ cho theo nhóm group  scatterplot(y ~ x | group, pch=c(16, 14), smooth=F, col=c("blue","red"), lwd=c(1,2) |
| Biểu đồ tương quan đa biến (package psych) | library(psych);  vars = cbind(age, bmi, whr, hdl, ldl, insulin, ir);  pairs.panels(vars) |
| Phân chia cửa sổ | par(mfrow=c(rows, cols)) |

**Biểu đồ dùng package chuyên dụng ggplot2**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Biểu đồ phân bố (histogram) | qplot(x) + geom\_histogram(col="white", fill="blue")  # Thêm density |
| Biểu đồ phân bố có thêm đường density | p = ggplot(data=pisa, aes(MATH));  p + geom\_histogram(bindwith=20, aes(y=..density..), col="white", fill="blue", lwd=0.5)  + geom\_density()  + labs(x="MATH SCORE", y="Frequency") |
| Biểu đồ phân bố cho nhiều nhóm; mỗi nhóm một box riêng | qplot(MATH, col=AREA, geom="histogram", facets=~AREA) |
| Biểu đồ phân bố cho nhiều nhóm; mỗi nhóm một box riêng; hai biến phân nhóm | p = ggplot(data=pisa, aes(MATH))  p + geom\_histogram(binwidth=20, aes(y=..density..), col="white", fill="blue", lwd=0.5) + geom\_density() + facet\_grid(REGION ~ AREA) |
| Biểu đồ phân bố cho nhiều nhóm; thêm đường biểu diễn và density | qplot(MATH, geom="density", fill=AREA, alpha=I(0.5)) + opts(legend.position="top") |
| Biểu đồ tương quan (xy) phân theo nhóm | qplot(x=READING, y=MATH, col=AREA) + stat\_smooth(method="lm") + opts(legend.position="top") |
| Biểu đồ xy cho từng đối tượng; vẽ chồng lên nhau | p = ggplot(data=long, aes(x=visit, y=L1L4, group=id, colour = factor(id)));  p + geom\_line() + opts(legend.text=theme\_text(size=5)) |
| Biểu đồ xy cho từng đối tượng; mỗi đối tượng một box | p = ggplot(data=long, aes(x=visit, y=L1L4))    p + geom\_point() + geom\_smooth(method=lm) + facet\_wrap(~id) |

**Phân tích mô tả**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Trung bình; phương sai; độ lệch chuẩn; trung vị; giá trị thấp nhất; giá trị cao nhất; range; số đối tượng | mean(x); var(x); sd(x); median(x); min(x); max(x); range(x); length(x) |
| Tính bách phân vị 2.5, 50, 97.5% | quantile(median, probs=c(0.025, 0.975)) |
| Tóm tắt dữ liệu | summary(dataset) |
| Tóm tắt dữ liệu cho nhiều biến hay dataset dùng describe (trong psych) | library(psych);  summary(dataset, skew=F, range=F) |
| Tóm tắt dữ liệu cho nhiều biến hay dataset; theo nhóm | library(psych);  describeBy(dataset, skew=F, range=F) |
| Tóm tắt bằng hàm tabular (trong tables); biến phân nhóm | tabular(REGION ~ 1);  tabular(REGION~1\*(n=1 + Percent("col")));  tabular(REGION ~ AREA\*(n=1 + Percent("row")));  tabular(REGION ~ AREA\*(n=1 + Percent("col")));  tabular(REGION\*AREA ~ TYPE) |
| Tóm tắt bằng hàm CrossTable (trong gmodels); biến phân nhóm | CrossTable(REGION, TYPE, prop.c=T, prop.r=T, prop.t=F, prop.chisq=F, chisq=T) |
| Tóm tắt bằng hàm tabular (trong tables); biến phân nhóm | tabular(REGION ~ MATH\*(n=1+mean+sd));  tabular(REGION\*TYPE ~ MATH\*(n=1+mean+sd));  tabular(REGION\*TYPE ~ (MATH + READING)\*(n=1+mean+sd)) |
| Kiểm định xem biến x có tuân theo luật phân bố chuẩn | shapiro.test(x) |
| Kiểm định t xem trung bình mẫu bằng trung bình quần thể (mu) | t.test(x, mu=30) |
| Kiểm định t hai nhóm độc lập (unpaired t-test) | t.test(x ~ group);  hoặc t.test (group1, group2) |
| Kiểm định t hai nhóm không độc lập (paired t-test) | t.test(x ~ group, paired=T);  hoặc t.test (group1, group2, paired=T) |
| Kiểm định phương sai hai nhóm | var.test(x ~ group) |
| Kiểm định Wilcoxon cho 2 nhóm độc lập | wilcox.test(x ~ group) |
| Kiểm định Wilcoxon cho 2 nhóm không độc lập | wilcox.test(group1, group2, paired=TRUE) |
| Kiểm định nhị phân cho một tỉ lệ (binomial test) | # kiểm định xem tần số x xày ra trong số n có bằng 50%  prop.test(x, n, 0.50) |
| So sánh 2 tỉ lệ; kiểm định nhị phân | fracture = c(7, 20)  total = c(100, 110  prop.test(fracture, total) |
| Kiểm định Ki bình phương (χ2 test) 2 nhóm | freq = table(sex, ethnicity);  prop.test(freq) |
| Kiểm định Ki bình phương (χ2 test) nhiều nhóm | chisq.test(sex, ethnicity) |
| Kiểm định chính xác Fisher (Fisher’s exact test) | fisher.test(sex, ethnicity) |
| Khoảng tin cậy 95% cho một tỉ lệ | library(binom);  x = c(3, 5, 7);  n = rep(10, length(x));  binom.confint(x, n, conf.level=0.95, methods="all") |
| Khoảng tin cậy 95% cho một tỉ lệ gần 100% hay gần 0 (Phương pháp Cai & DasGupta 2002) | n=40; x=39; alpha=0.05;  qbeta(c(alpha/2,1-alpha/2),x+0.5,n-x+0.5) |
| Khoảng tin cậy 95% *chính xác* cho 2 tỉ lệ | require("ExactCIdiff");  # binormCI(n1,n2,x1,x2,conf.level,CItype)  binormCI(10,15,2,4, conf.level=0.95,CItype="Two.sided") |

**Mô hình hồi qui tuyến tính**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Hệ số tương quan Pearson (coefficient of correlation) giữa x và y | cor(x, y);  cor.test(x, y) |
| Hệ số tương quan Spearman | cor.test(x, y, method="spearman") |
| Hệ số tương quan Kendall | cor.test(x, y, method="kendall") |
| Mô hình hồi qui tuyến tính đơn giản y = a + bx | m = lm(y ~ x);  summary(m);  anova(m) |
| Mô hình hồi qui tuyến tính đa biến y = a + b1x1 + b2x2 + b3x3 + … | m = lm(y ~ x1 + x2 + x3 + ...);  summary(m);  anova(m) |
| Mô hình hồi đa thức y = a + b1x1 + b2x2 + … + bpxp | # đa thức bậc 2  m = lm(y ~ poly(x, 2))  # đa thức bậc k  m = lm(y ~ poly(x, k)) |
| Tìm mô hình “tối ưu” bằng tiêu chuẩn AIC | reg = lm(y ~ x1+x2+x3+x4+x5+x6+x7, data=REGdata);  reg = lm(y ~ ., data=REGdata);  step(reg, direction="both") |
| Tìm mô hình “tối ưu” bằng tiêu chuẩn BIC (Bayesian Model Average) | library(BMA);  xvars = REGdata[,-1];  yvar = REGdata[,1];  bma = bicreg(xvars, y, strict=FALSE, OR=20)  imageplot.bma(bma) |

**Phân tích phương sai**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Mô hình phân tích phương sai đơn giản hay ANOVA: y và x (x là factor) | m = lm(y ~ x);  m = aov(y ~ x);  summay(x);  anova(m) |
| So sánh giữa các nhóm (posthoc comparison) từ ANOVA, phương pháp Bonferroni | pairwise.t.test(galactose, group, p.adj="bonferroni") |
| So sánh giữa các nhóm (posthoc comparison) từ ANOVA, phương pháp Tukey | m = aov(y ~ x)  TukeyHSD(m) |
| Phân tích phương sai phi tham số (Kruskal test) | kruskal.test(y ~ x) |
| Phân tích phương sai với nhiều factor g1, g2, g3. | m = aov(y ~ g1 + g2 + g3)  TukeyHSD(m)  plot(TukeyHSD(m), ordered=TRUE) |
| Phân tích phương sai giai thừa (factorial experiment) | m = aov(y ~ factor1 + factor2 + factor3);  anova(m);  plot(TukeyHSD(m), ordered=T) |
| Phân tích phương sai thí nghiệm giao chéo (cross-over experiment) | m = lm(y ~ treat + seq + period);  anova(m);  TukeyHSD(aov(y ~ treat+seq+period+id)) |
| Phân tích hiệp biến (analysis of covariance); x là biến liên tục | m = lm(y ~ x + factor)  # có interaction  m = lm(y ~ x + factor + factor:x) |

**Mô hình hồi qui logistic**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Mô hình hồi qui logistic: biến y là nhị phân, biến x có thể là nhị phân hay liên tục | m = glm(y ~ x, family="binomial");  summary(m) |
| Tính toán giá trị tiên lượng;vẽ biểu đồ | predict(m, type="response");  plot(x, fitted(glm(y ~ x, family="binomial"))) |
| Mô hình hồi qui logistic đa biến với 3 biến x1, x2, x3 | m = glm(y ~ x1 + x2 + x3, family="binomial");  summary(m) |
| Chọn mô hình “tối ưu” bằng phương pháp Bayesian Model Average | library(BMA);  xvars = cbind(x1,x2,x3,x4,x5);  bma = bic.glm(xvars, y, strict=F, OR=20,glm.family="binomial");  imageplot.bma(bma) |
| Mô hình hồi qui Poisson | age=c(19.5,29.5,39.5,49.5,59.5,69.5,79.5);  cases = c(1, 16, 30, 71, 102, 130, 133);  pop = c(172675, 123065, 96216, 92051, 72159, 54722, 32185);  dat = data.frame(age, cases, pop)  library(MASS)  m = glm(cases ~ age + offset(log(pop)), family=poisson,data=dat)  summary(m) |
| Mô hình hồi qui Binomial | library(MASS)  m = glm.nb(y ~ x1 + x2 + x3))  # chú ý: y là biến phụ thuộc, x1, x2 và x3 là biến tiên lượng.  summary(m) |

**Mô hình hồi qui Cox và phân tích sống còn**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Phương pháp Kaplan-Meier | library(survival);  # status=1 là có biến cố, 0 là censored  survtime = Surv(weeks, status==1));  kp = survfit(Surv(weeks, status==1)**~1**)**;**  plot(kp) |
| Phương pháp Kaplan-Meier, so sánh 2 nhóm | bygroup = survfit(Surv(time, status==1)~group);  summary(bygroup);  plot(bygroup) |
| Mô hình Cox đơn biến | m = coxph(Surv(time, status==1) ~ group);  summary(m) |
| Mô hình Cox đa biến, với biến tiên lượng x1, x2 | m = coxph(Surv(time, status==1) ~ group+x1+x2);  summary(m) |
| Tìm mô hình Cox “tối ưu” bằng phương pháp AIC | cox = coxph(Surv(y, status) ~ ., data=dat);  summary(cox);  search = step(cox, direction="both");  summary(search) |
| Tìm mô hình Cox “tối ưu” bằng phương pháp Bayesian Model Average | library(BMA);  bma = bic.surv(xvars, time, death);  summary(bma);  imageplot.bma(bma) |

**Phân tích đa biến (multivariate analysis)**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Phân tích thành phần (principal component analysis) | pca = princomp(~factor1 + factor2, data=temp);  summary(pca);  loadings(pca) |
| Phân tích cụm (cluster analysis): đo lường khoảng cách | dat = read.table(text="  x1 x2  4 8  6 6  10 11  11 8  17 5  19 3  20 11  21 2  ", header=T)  distance = dist(dat) |
| Phân tích cụm (cluster analysis): phân tích | library(cluster);  hc.average = hclust(dist, method = "average");  plot(hc.average, hang = -1, main = "average") |
| Phân tích cụm (cluster analysis): xác định số cluster | library(NbClust);  teeth = read.table(teeth.txt", header=T)  teeth1 = as.numeric(as.matrix(teeth[,-1]))  NbClust(teeth1, method = "average", index = "all") |

**Mô hình mixed effects**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Biểu đồ xyplot cho từng cá nhân: y là biến phân tích và time là thời gian , id là mã cá nhân | library(rms);  library(lattice);  library(lme4);  xyplot(y ~ time | id, type=c("p","r"), as.table=time) |
| Biểu đồ interaction cho từng cá nhân: y là biến phân tích và time là thời gian | fit = by(y, id,  function(data) fitted.values(lm(y ~ time, data=data)));  fit = unlist(fit);  names(fit) = NULL;  interaction.plot(time, id, fit) |
| Mô hình unconditional growth:  ;  *ai* = *A + u;*  *bi* = *B + v;*  *yi* = (*A + u*) + (*B + v*)*T* + *ei;*  *yi* = (*A + BT*) +(*u + vT* + *ei*) | library(lme4)  fit = lmer(y ~ 1 + time + (1 + time | id), data=glucose, REML=0)  summary(fit) |
| Mô hình conditional growth:  *ai* = *A*0 *+ A1 x* treatment *+ u;*  *bi* = *B0 + B1 x* treatment *+ v;*  *yi* = (*A*0 *+ A*1*x*treatment *+ B*0*T + B*1*x*treatment*xT*) *+* (*u + vT* + *ei*) | fit = lmer(y ~ 1 + treatment + time + time:treatment + (1+time|id), data=glucose, REML=0)  summary(fit) |

**Phân tích tổng hợp**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Phân tích biến nhị phân, 2 nhóm: chuẩn bị dữ liệu | Study = 1:13  N1 = số bệnh nhân nhóm 1  D1 = số biến cố nhóm 1  N0 = số bệnh nhân nhóm 2  D0 = số biến cố nhóm 2  S1 = N1-D1 # số không biến cố nhóm 1  S0 = N0-D0 # số không biến cố nhóm 2  bb = data.frame(Study,N1,D1,S1,N0,D0,S0) |
| Phân tích biến nhị phân, 2 nhóm: tính effect size | library(metafor)  es = escalc(measure = "RR", ai=D1, bi=S1, ci=D0, di=S0, data=bb, append=T) |
| Phân tích biến nhị phân, 2 nhóm: random effects model | re = rma(yi, vi, data = es);  summary(re);  plot(re) ;  forest(re, atransf=exp) |
| Phân tích biến liên tục, 2 nhóm: chuẩn bị dữ liệu (n=số đối tượng, m=trung bình, sd=độ lệch chuẩn) | n1 = c(155,31,75,18,8,57,34,110,60)  m1 = c(55,27,64,66,14,19,52,21,30)  sd1 = c(47,7,17,20,8,7,45,16,27)  n2 = c(156,32,71,18,13,52,33,183,52)  m2 = c(75,29,119,137,18,18,41,31,23)  sd2 = c(64,4,29,48,11,4,34,27,20)  los = data.frame(n1,m1,sd1,n2,m2,sd2) |
| Phân tích biến liên tục, 2 nhóm: tính effect size | library(metafor)  es = escalc(measure="MD", n1i=n1, m1i=m1, sd1i=sd1, n2i=n2, m2i=m2, sd2i=sd2, data=los, append=T) |
| Phân tích biến liên tục, 2 nhóm: random effects model | re = rma(yi, vi, data=es, method="FE")  plot(re)  forest(re) |

**Phân tích bootstrap**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Lấy mẫu ngẫu nhiên | x = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  sample(x, 5, replace=T)  sample(x, 5, replace=F) |
| Khoảng tin cậy 95% cho trung vị | x = c(0.05,0.15,0.35,0.25, ...);  n = length(pain);  B=1000;  median = numeric(B);  for (i in 1:B)  {  bs.sample = sample(x, n, replace=T)  median[i] = median(bs.sample)  }  quantile(median, probs=c(0.025, 0.975))  hist(median, col="blue", border="white", main="") |
| So sánh 2 số trung vị | control = c(52,104,146, ...);  treatment = c(94, 197, 16, ...);  B=1000;  con = rep(NA, B);  trt = rep(NA, B);  for (i in 1:B) {  con[i] = median(sample(control, replace=T))  trt[i] = median(sample(treatment, replace=T))  dif[i] = trt[i]-con[i]  }  quantile(dif, c(0.025, 0.5, 0.975)) |
| So sánh 2 tỉ lệ | n = c(11037, 11034);  disease = c(104, 189);  aspirin = c(rep(1, disease[1]), rep(0, n[1]-disease[1]));  control = c(rep(1, disease[2]), rep(0, n[2]-disease[2]));  B = 10000;  bs1 = rep(NA, B);  bs2 = rep(NA, B);  for (i in 1:B) {  resample1 = sample(aspirin, n[1], replace=T)  dis1 = sum(resample1)  resample2 = sample(control, n[2], replace=T)  dis2 = sum(resample2)  bs1[i] = dis1/n[1]  bs2[i] = dis2/n[2]  }  RR = bs1/bs2;  quantile(RR, c(0.025, 0.50, 0.975) |
| So sánh 2 tỉ lệ, phương pháp Bayes | x = 11; n1 = 20; alpha1 = 1; beta1 = 1;  y = 5; n2 = 20; alpha2 = 1; beta2 = 1;  p1 = rbeta(10000, x + alpha1, n1 - x + beta1);  p2 = rbeta(10000, y + alpha2, n2 - y + beta2);  rd = p2 - p1;  plot(density(rd));  quantile(rd, c(.025, 0.5, 0.975)) |

**Ước tính cỡ mẫu**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Ước tính cỡ mẫu (*n*) cho một tỉ lệ | alpha=0.05; z = qnorm(0.05/2)  p=0.10; e = 0.01  n = (z/e)^2\*p\*(1-p)  # Dùng package epicalc  library(epicalc);  n = n.for.survey(p=0.10, delta=0.01, alpha=0.05) |
| Ước tính *n* cho một số trung bình | alpha=0.05; z = qnorm(0.05/2)  sigma=12; e = 2.5  n = (z/e\*sigma)^2 ; n |
| Cỡ mẫu cho nghiên cứu so sánh 2 tỉ lệ | library(epicalc);  n = n.for.2p(p1=0.75, p2=0.90, power=0.9, alpha=0.05, ratio=1);  n |
| Cỡ mẫu cho nghiên cứu so sánh 2 số trung bình | library(epicalc);  n = n.for.2means(mu1=0.80, mu2=0.84, sd1=0.12, sd2=0.12, power=0.9, alpha=0.05, ratio=1);  n |
| Cỡ mẫu cho nghiên cứu so sánh 2 survival curves | library(epiR);  epi.studysize(treat=0.60, control=0.50, n=NA, sigma=NA, power=0.80, r=1, conf.level=0.95, sided.test=2, method = "survival") |
| Cỡ mẫu cho nghiên cứu bệnh chứng | library(epiR);  epi.studysize(treat = 2/100, control = 1/100, n = NA, sigma = 0.30, power = 0.90, r = 1, conf.level = 0.95, sided.test = 2, method = "case.control") |
| Cỡ mẫu cho nghiên cứu đoàn hệ | library(epiR);  smoke=1.4\*(5\*413)/100000  nonsmoke=(5\*413)/100000  epi.studysize(treat=smoke, control=nonsmoke, n=NA, sigma=NA, power=0.90, r=1, conf.level=0.95, sided.test=1, method="cohort.count") |

**Phân tích số liệu dãy thời gian (time series)**

|  |  |
| --- | --- |
| **Phân tích** | **Hàm R** |
| Sắp xếp dữ liệu bằng hàm ts | births = scan("~/Documents/births.txt");  ts.births = ts(births, frequency=12, start=c(1946, 1)) |
| Vẽ biểu đồ time series | plot.ts(ts.births) |
| Trung bình động (moving average) với bậc *k* | library(TTR);  SMA(ts.births, n=5) |
| Phân tích trend, season, và random: dữ liệu không có season | ma5 = SMA(ts.births, n=5)  plot.ts(ma5) |
| Phân tích trend, season, và random: dữ liệu có season | comp = decompose(ts.births);  comp;  plot(comp) |
| Điều chỉnh cho season | adj.births = ts.births-comp$seasonal;  plot(adj.births) |
| Tiên lượng bằng phương pháp simple exponential smoothing | library(forecast);  forecast = HoltWinters(ts.births, beta=F, gamma=F);  forecast;  plot(ts.births); plot(forecast) |
| Tiên lượng bằng phương pháp Holt exponential smoothing | forecast1 = HoltWinters(ts.births, gamma=F);  forecast1; plot(forecast1) |
| Tiên lượng bằng phương pháp Holt-Winters smoothing | forecast1 = HoltWinters(ts.births);  forecast1; plot(forecast1);  forecast2 = forecast.HoltWinters(forecast1, h=60)  plot.forecast(forecast2) |
| Mô hình ARIMA: sai phân (differencing), differences=k | skirts = scan("~/Documents/skirts.txt", skip=5);  ts.skirts = ts(skirts, start=c(1866));  # tính sai phân  dif.skirts = diff(ts.skirts, differences=1);  plot.ts(dif.skirts) |
| Autocorrelation và partial autocorrelation | acf(dif.kings, lag.max=20, plot=F);  acf(dif.kings, lag.max=20);  pacf(dif.kings, lag.max=20, plot=F);  pacf(dif.kings, lag.max=20) |
| Mô hình MA | kings = scan("~/Documents/kings.txt")  library(forecast);  auto.arima(kings) |
| Mô hình RA | vc = scan("~/Documents/vulcano.txt")  ts.vc = ts(vc, start=c(1500));  plot.ts(ts.vc);  auto.arima(vc);  auto.arima(vc, ic="bic") |
| Mô hình ARIMA(p, d, q) | arima1 = arima(ts.kings, order=c(0, 1, 1));  arima1 |
| Tiên lượng bằng mô hình ARIMA(p, d, q) | library(forecast);  forecast1 = forecast.Arima(arima1, h=5);  forecast1;  plot.forecast(forecast1) |

**WinBUGS**

**Bước 1: Viết mô hình**

MODEL {

# prior information

p1 ~ dbeta(0.5, 0.5)

p2 ~ dbeta(0.5, 0.5)

# data or likelihood

x1 ~ dbin(p1, n1)

x2 ~ dbin(p2, n2)

# effect size

diff = p1-p2

rr = p1/p2

or = (p1/(1-p1)) / (p2/(1-p2))

logOR = log(or)

ppos = step(OR - 1)

}

DATA list(x1=3, n1=18, x2=2, n2=23)

**Bước 2: Kiểm tra ngữ pháp**

Model 🡪 Specification 🡪 Specification Tool:

Highlight MODEL, nhấn **check model**

**Bước 3: Nhập dữ liệu**

Highlight **list.** Specification Tool 🡪 Load data

**Bước 4: Compile**

Specification Tool 🡪 Compile

**Bước 3: Initial value**

Specification Tool 🡪 gen inits

**Bước 6: Mô tả thông số (parameter)**

Inference 🡪 sample 🡪 Sample minitor tool

**node** trong **Sample Monotor Tool** và gõ **tên parameter**

**Bước 7: Mô phỏng (generating simulated values)**

Model 🡪 Update 🡪 Update Tool (from 5000, refresh 100)

**Bước 8: Xem xét kết quả thông tin hậu định (posterior information)**

**Sample Monotor Tool** 🡪 **beg**

Chọn thông số (trường hợp này là p) từ field **node**

**Bước 9: Chẩn đoán (diagnosis)**

**Sample Monitor Tool** và nhấn chuột vào nút **history**.

**Lập trình và hàm với** R

R được phát triển sao cho người sử dụng có thể phát triển những hàm thích hợp cho mục đích phân tích và tính toán của mình. Thật vậy, như đã đề cập trong phần đầu của sách, có thể xem R là một ngôn ngữ thống kê, và chúng ta có thể sử dụng ngôn ngữ để giải quyết các vấn đề không thường thấy trong sách giáo khoa. Phần này chỉ trình bày một vài hàm đơn giản để bạn đọc có thể hiểu cách vận hành của R và hi vọng giúp bạn đọc tự phát triển các hàm sau đó.

Hàm (hay có khi còn gọi là “macro” trong các phần mềm khác) thực chất là tập hợp một số lệnh được lưu trữ dưới một cái tên. Ở mức độ đơn giản nhất, hàm là “tốc kí” cho một nhóm lệnh.

**Ví dụ 1**. Trong các lệnh sau đây, chúng ta tạo hai dữ liệu (data1 và data2). Mỗi dữ liệu có hai cột số liệu được tạo ra bằng mô phỏng từ phân phối chuẩn. Sau đó, vẽ biểu đồ cho hai dữ liệu với ghi chú.

data1 = cbind(rnorm(100,1), rnorm(100,0))

data2 = cbind(rnorm(100,-1), rnorm(100,0))

xr = range(rbind(data1,data2)[,1])

yr = range(rbind(data1,data2)[,2])

plot(data1, xlim=xr, ylim=yr, col=1, xlab="", ylab="")

par(new=T)

plot(data2, xlim=xr, ylim=yr, col=2, xlab="", ylab="")

title(main="My simulated data", xlab="Weight", ylab="Yield")

legend(-3.0, -1.5, c("Big", "Small"), col=1:2, pch=1)

Một cách để nhớ tất cả các lệnh này là lưu trữ chúng trong một text file chẳng hạn. Mỗi lần muốn sử dụng, chúng ta chỉ đơn giản cắt và dán các lệnh này vào R. Một cách khác tốt hơn là tạo ra một hàm gồm các lệnh trên để có thể sử dụng nhiều lần.

Mỗi hàm R phải có tên. Tất cả các lệnh được chứa trong khu vực được giới hạn bằng hai kí hiệu { và }. Kí hiệu { cho biết tất cả các lệnh sau đó là nằm trong hàm; và kí hiệu } cho biết chấm dứt hàm. Trong ví dụ trên, chúng ta gọi hàm là plotfigure:

**plotfigure <- function()**

{

data1 = cbind(rnorm(100,1), rnorm(100,0))

data2 = cbind(rnorm(100,-1), rnorm(100,0))

xr = range(rbind(data1,data2)[,1])

yr = range(rbind(data1,data2)[,2])

plot(data1, xlim=xr, ylim=yr, col=1, xlab="", ylab="")

par(new=T)

plot(data2, xlim=xr, ylim=yr, col=2, xlab="", ylab="")

title(main="My simulated data", xlab="Weight",

ylab="Yield")

legend(-3.0, -1.5, c("Big", "Small"), col=1:2, pch=1)

}

Sau khi đã cho vào R, chúng ta chỉ đơn giản gọi hàm nhiều lần như sau:

> plotfigure()

> plotfigure()

và kết quả sẽ như sau:

|  |  |
| --- | --- |
|  |  |

Trong hàm plotfigure trên, chúng ta mô phỏng 100 số liệu từ phân phối chuẩn. Và cứ mỗi lần ứng dụng, hàm chỉ tạo ra 100 số liệu, chúng ta không thay đổi được (ngoại trừ phải thay đổi từ lúc biên tập, hay lập hàm). Nói cách khác, hàm trên không có thông số.

Khía cạnh tiện lợi của hàm là chúng ta có thể làm cho thông số thay đổi theo ý muốn của người sử dụng. Chẳng hạn như chúng ta muốn thay đổi số số liệu mô phỏng và trung bình từ luật phân phối chuẩn, chúng ta chỉ cần cho hai con số này là hai thông số (parameters) để người sử dụng có thể thay đổi. Tạm gọi đó là thông số **n**, **mean1,** và **mean2**, thì hàm sẽ như sau:

**plotfigure <- function(n, mean1, mean2)**

{

data1 = cbind(rnorm(**n**,**mean1**), rnorm(**n**,0))

data2 = cbind(rnorm(**n**,**mean2**), rnorm(**n**,0))

xr = range(rbind(data1,data2)[,1])

yr = range(rbind(data1,data2)[,2])

plot(data1, xlim=xr, ylim=yr, col=1, xlab="", ylab="")

par(new=T)

plot(data2, xlim=xr, ylim=yr, col=2, xlab="", ylab="")

title(main="My simulated data", xlab="Weight",

ylab="Yield")

legend(-3.0, -1.5, c("Big", "Small"), col=1:2, pch=1)

}

Khi ứng dụng hàm, chúng ta chỉ đơn giản thay đổi n và mean. Trong hai lệnh sau đây, chúng ta đầu tiên vẽ một biểu đồ tán xạ với 200 số liệu, và số trung bình -2 và 2. Trong lệnh hai, chúng ta nâng số liệu lên 200, nhưng trung bình vẫn như lần mô phỏng trước:

> plotfigure(200, 2, -2)

> plotfigure(500, 2, -2)

Và kết quả sẽ khác trên:

|  |  |
| --- | --- |
|  |  |

**Ví dụ 2**. Chúng ta muốn viết một hàm để cộng hai số. (Tất nhiên R có khả năng làm “việc” này, nhưng vì lí do minh họa, chúng ta sẽ giả thiết đơn giản như thế). Gọi hàm đó là add. Hai thông số a và b là “arguments. Cách viết như sau:

**add <- function(a, b)**

{

sum = a+b

ans <- "Answer = "

cat(ans, sum, “\n”)

}

Như đã thấy, bước đầu tiên, chúng ta cho tên hàm là add và định nghĩa thông số a và b. Một hàm phải được mở đầu bằng kí hiệu { và chấm dứt bằng }. sum là một biến số cộng a và b. ans <- "Answer = " định nghĩa trả lời (có thể không cần). cat(ans, sum, “\n”) có chức năng thu thập số liệu và trình bày kết quả cho người sử dụng hàm, trong đó “\” có nghĩa là sau khi trình bày, cho người sử dụng một prompt khác. Bạn đọc có thể dán các lệnh trên vào R và thử cho lệnh:

> add(3, 9)

Answer = 12

> add(sqrt(5), exp(10))

Answer = 22028.7

**Ví dụ 3.** Hàm sau đây tiến hành nhiều tính toán hơn hàm trong ví dụ 1. Nếu chúng ta có một biến số gồm n phần tử  tuân theo luật phân phối chuẩn với trung bình ** và phương sai . Viết theo kí hiệu toán:



Nếu chúng ta có thông tin trước cho biết ** có luật phân phối chuẩn với trung bình ** và phương sai **2, hay:



Qua định lí Bayes, chúng ta có thể ước tính trung bình:



và phương sai

.

Trong đó, là số trung bình của mẫu *n*.  và  được gọi là “posterior”. Chúng ta có thể viết một hàm bằng R để tính hai số này như sau. Gọi tên hàm là bayes.

**bayes <- function(x, prior.mean, prior.var)**

{

n <- length(x)

sample.mean = mean(x)

sample.var = var(x)

numerator = (prior.mean/prior.var) + (n\*sample.mean/sample.var)

denominator = 1/prior.var + n/sample.var

posterior.mean = numerator/denominator

posterior.var = 1/denominator

a = "Posterior mean = "

b = "Posterior variance = "

cat("Sample size = ", n, "\n")

cat("Sample mean = ", sample.mean, "\n")

cat("Sample var = ", sample.var, "\n")

cat("Prior mean = ", prior.mean, "\n")

cat("Prior var = ", prior.var, "\n")

cat(a, posterior.mean, "\n")

cat(b, posterior.var, "\n")

}

**Ví dụ 4**. Mật độ chất khoáng trong xương (bone mineral density - bmd) trong một quần thể thường phân phối theo luật phân phối chuẩn, với giá trị trung bình khoảng 1.0 g/cm2 và phương sai 0.0144 g/cm4. Giả dụ chúng ta đo mật độ xương của một nhóm bệnh nhân như sau: 1.0, 1.5, 2.1, 1.7, 1.8, 0.9, 0.7. Chúng ta muốn biết giá trị trung bình và phương sai của mẫu này sau khi “điều chỉnh” cho trung bình và phương sai đã biết trước. Trước hết, chúng ta gọi nhóm số liệu này là bmd:

> bmd = c(1.0, 1.5, 2.1, 1.7, 1.8, 0.9, 0.7)

và sau đó “gọi” hàm bayes như sau:

> bayes(bmd, 1.0, 0.0144)

Sample size = 7

Sample mean = 1.385714

Sample var = 0.2747619

Prior mean = 1

Prior var = 0.0144

Posterior mean = 1.103525

Posterior variance = 0.01053507

Trên đây chỉ là một vài hàng giới thiệu cách lập trình và viết hàm bằng ngôn ngữ R. Trong thực tế, tất cả các hàm như survival, BMA, meta, metafor, rms, Hmisc, v.v. đều được phát triển bằng ngôn ngữ R. Bạn đọc có thể tham khảo tài liệu “Introduction to R” của W. Venables và B. Ripley (phần cuối của sách) để biết thêm chi tiết kĩ thuật.