

1 Introduction

First class stuff

1.1 Syllabus and Survey

Go over syllabus

Survey

About the course:

1. The course is not about theory! It will have both theory and practice in it. We will do things that are truly useful.
2. The course is not about computing alone! Computing needs intelligence and knowledge of statistics to be productive.

1.1.1 Results of Monday's survey:

1. Expectation:
 - a. Most answered R and bootstrap. (correct answer!:) simulation technique, programming in S/S-plus.
 - b. Biostatistics
 - c. SAS : can't help. Mastering one package helps learning new ones.
 - d. 'Real-world' statistics : depending on what real-world means. R is flexible and can serve both basic and complicated tasks well. (scalable!)
 - e. 'acquire what stated in the syllabus' (my favorite answer)
2. Computing Background : > 2/3 have no or little R exposure, so I'm comfortable doing somewhat basic introduction

2 R Basic

2.1 About R

Why R

- Compact (~ 10 mb)
- Available for Various OS
- Powerful with lots of stat functions
- Open source (Politically correct)
- De facto standard (at least among statisticians)
- Fast update (evolving)
- Very extensible (improve it yourself!)
- Free!

More on R

- Brief history of R: S at Bell Labs -> S-Plus (commercial) and R (Open Source)
- R is : tools; statistical models; graphics; object-oriented programming language

- R is 'environment', not 'package'
- R : console, not GUI (may be difficult for some)

2.2 Simple R-session

First, a few basics:

1. R : getting it; installing it on Windows, Linux, MacOS, etc. (www.r-project.org)
2. starting, ending, bailing out
3. on-line help (HTML version, search function)

Then go over the first half of A Sample Session of "An Introduction of R" (Appendix A).

```

help.start()
x <- rnorm(50)
y <- rnorm(x)
  #Generate two pseudo-random normal vectors of x- and y-coordinates.
plot(x, y)
  #Plot the points in the plane. A graphics window will appear automatically.
ls() #See which R objects are now in the R workspace.
rm(x, y) #Remove objects no longer needed. (Clean up).
x <- 1:20 #Make x = (1, 2, . . . , 20).
w <- 1 + sqrt(x)/2
  #A 'weight' vector of standard deviations.
dummy <- data.frame(x=x, y= x + rnorm(x)*w)
  #Can you relate this data with model formula?
dummy #Make a data frame of two columns, x and y, and look at it.
fm <- lm(y ~ x, data=dummy)
summary(fm)
  #Fit a simple linear regression of y on x and look at the analysis.
fm1 <- lm(y ~ x, data=dummy, weight=1/w^2)
summary(fm1)
  #Since we know the standard deviations, we can do a weighted regression.
attach(dummy)
  #Make the columns in the data frame visible as variables.
lrf <- lowess(x, y)
  #Make a nonparametric local regression function.
plot(x, y)
  #Standard point plot.
lines(x, lrf$y)
  #Add in the local regression.
abline(0, 1, lty=3)
  #The true regression line: (intercept 0, slope 1).
abline(coef(fm))
  #Unweighted regression line.
abline(coef(fm1), col = "red")
  #Weighted regression line.
detach() #Remove data frame from the search path.
plot(fitted(fm), resid(fm),
xlab="Fitted values",
ylab="Residuals",
main="Residuals vs Fitted")
  #A standard regression diagnostic plot to
  #check for heteroscedasticity. Can you
  #see it?
qqnorm(resid(fm), main="Residuals Rankit Plot")
  #A normal scores plot to check for skewness,
  #kurtosis and outliers.

```

```
rm(fm, fml, lrf, x, dummy)
#Clean up again.
```

Also, various tips and tricks for efficient usage of R.

2.2.1 Preview of ECDF and Q-Q plot

2.2.2 Air conditioning data homework

HW: for the air-conditioner data, (3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)

1. draw empirical distribution function (ecdf) and overlay cdf of exp with 'correct parameter' (qexp).
2. Plot Q-Q plot of the data and the corresponding exponential distribution

Solutions to homework:

```
y <- c(3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)
n <- length(y)

plot(y)
plot(ecdf(y))
curve(pexp(x, 1/mean(y)), add=TRUE, col='red')

qqplot(qexp(seq(1, n, length=n)/(n+1), 1/mean(y)), y)
# or, alternatively,
plot(qexp(ppoints(y), 1/mean(y)), sort(y))
abline(0,1)
```

3 Monte-Carlo Simulation

3.1 What is Monte-Carlo simulation

When we use the word simulation, we refer to any analytical method meant to imitate a real-life system, especially when other analyses are too mathematically complex or too difficult to reproduce.

One type of simulation is Monte Carlo simulation, which randomly generates values for uncertain variables over and over to simulate a model.

3.1.1 Monte-Carlo computation of π

Generate many (N) uniform random points in $[0,1] \times [0,1]$. Count how many of them fall into the quarter-circle with center 0 and radius 1. The ratio of "hits" ($\pi/4$) to "throws" will be one-fourth the value of π .

```
N <- 1000
x <- runif(N)
y <- runif(N)
idx <- x^2 + y^2 <= 1
length(which(idx))/N*4

plot(x,y)
points(x[idx],y[idx], col='red')
# What does this plot illustrate?
length(which(idx))/N*4 # estimate of pi
```

HW: play around with different values of repetition N. Run the above many times. Do you observe changes in accuracy in the approximation? (The large N is, the more accurate the approximation tend to be)

3.2 Generating random data

For Monte-Carlo simulation, good, reliable random number generator is the key and R has that. But one needs to know

- 1) the pseudo-nature of the any computer generated random number and
- 2) the efficient way of programming R by utilizing vector parameter values.

3.2.1 Pseudo- nature of random number generation

From <http://www.stats.ox.ac.uk/pub/MASS4/>

In the simplest form, congruential generator

$$X_{i+1} = 69069 X_i \bmod 2^{32}$$

Use `set.seed` to get 'reproducible' result

period $2^{19937} - 1$

Uniform random number is MOARNG (mother of all random number generator) **Q** Why?

3.2.2 More on generating random data

See table 5.1 (p.108) of V&R for available distribution.

Distribution	R name	Parameters
Exponential	exp	Rate
Normal	norm	Mean, sd

...
-----	-----	-----

$rdist(n,...)$: independent random samples from the prob dist $dist$. Parameters can be specified as vectors allowing the samples to be non-identically distributed.

Go over slowly.

Mixture model Example: 100 samples from the contaminated normal distribution in which sample is $N(0,1)$ with probability 0.95 and otherwise from $N(0,9)$

Start with inefficient, loop version

```
Contam <- numeric(n)
for(i in 1:n){...}
```

Finally, an efficient version

```
contam <- rnorm(n, 0, (1+2*rbinom(n, 1, 0.05)))
```

3.3 Resampling using 'sample' function

It samples from a data vector, with or without replacement.

```
x <- -10:10
#round(rnorm(20), 2)
n <- 10
x
sample(n) # select a random permutation from 1,...,n
sample(x) # randomly permute x, for length(x)>1
table(sample(x)) # no repetition
table(table(sample(x))) # confirm that!
sample(x, replace=T) # a bootstrap sample
table(sample(x, replace=T)) # effect of replacement
table(table(sample(x, replace=T))) # confirm that
sample(x, n) # sample n items from x WOR
sample(x, n, replace=T) # sample n items from x WR
sample(x, n, replace=T, prob=1:21) # probability sample of n items from x
# a way to sample from an arbitrary, finite discrete distribution (name your
own distribution)
```

3.3.1 Simulation Example: Distribution of t-statistic

Classical one-sample t-test:

$X_1, \dots, X_n \sim \text{iid } N(\mu, \sigma^2)$

What's the distribution of the statistic $T=(\bar{X} - \mu)/(\text{sqrt}(S/n))$? How close is it to $N(0,1)$ distribution? Study by simulation!

```
N <- 10000
mu <- 100
sigma <- 10
t.stats <- rep(NA, N)
```

```
n <- 5

for(i in 1:N){
  x <- rnorm(n, mu, sigma)
  t.stats[i] <-
    (mean(x)-mu)/sqrt(var(x)/(n-1))
}

hist(t.stats, prob=TRUE, nclass=100)
hist(t.stats, prob=TRUE, nclass=100, xlim=c(-10,10))
curve(dnorm, add=TRUE)
```

From theory, follows $t(n-1)$ distribution. Does the simulation result match the expectation?

```
curve(dt(x, df=n-1), add=TRUE, col='blue')

qqnorm(t.stats)
abline(0,1)
plot(qt(ppoints(t.stats), df=n-1), sort(t.stats))
abline(0,1)
```

HW: Play around with different values of n and degrees of freedom, reducing n to 4, 3, even down to 2. What do you observe?

By now, from this and previous examples, one needs to know about:

- Loops (for)
- Random number generation (rnorm)
- Drawing function (hist, curve)
- Density functions (dnorm, dt)
- Comparing two distribution (qqnorm, qqline)
- The concept of simulation! (Had we not known about t -distribution)

4 Q-Q Plot, Histogram and Kernel Density Estimate

4.1 Summary of Q-Q plot (V&R 5.1)

For a sample x , the quantile function is the inverse of the empirical CDF

Quantile(p) = $\min\{z \mid \text{proportion } p \text{ of the data } \leq z\}$

R function `quantile` calculates the quantiles of a single set of data.

Function `qqplot(x,y,...)` plots the quantile functions of two samples x and y against each other.

Sort the data, 'compress' the long data x (length m) to the short data y (length $n < m$) by linear interpolation at n equally spaced points spanning the interval $[\min(x), \max(x)]$.

illustration

function `qqnorm(x)` replaces one of the samples by the quantiles of a standard normal distribution.

To test a sample against a distribution *dist*, use

`plot(qdist(ppoints(x), ...), sort(x))`

The function `ppoints` computes the appropriate set of probabilities for the plot: $(i-1/2)/n$. Compare this with what we used $i/(n+1)$

`plots(qdist((1:n)/(n+1), ...), sort(x))`

does approximately the same thing, more transparently.

The function `qqline(x)` helps assess how straight a qqnorm plot is by plotting a straight line.

4.2 Data summaries (V&R 5.3?)

Standard univariate summaries: mean, median, var, max, min, range

The `summary` function returns mean, quantiles and the # of missing values.

`quantile(x,p) ~ x([np])`

Many have `na.rm` option set to FALSE, but can remove missing values.

`var, cor`, : have many different ways to handle missing values

```
data(faithful)
plot(faithful)
class(faithful)
names(faithful)
dim(faithful)
x <- faithful$eruptions

boxplot(x)
hist(x)
hist(x, breaks=seq(0, 6, by=0.5))

par(mfrow=c(2,3))
for(shift in seq(0, 0.8, by=0.2)){
  hist(x, breaks=seq(0, 6, by=0.5)+shift, col='red')
}
```

4.2.1 histogram

`hist(x,...)`: conventional histogram

`probability=T` gives a plot of unit total area rather than total cell counts.

`nclass` suggests the n# of bins. Default to `upper[log2n+1]` called Sturges' formula. Q bin width becomes `range(x)/(log2n+1)` **Problem**: outliers can inflate the range dramatically...

`breaks` specifies the breakpoints b/w bins

4.2.2 boxplot

4.3 Kernel density estimation (V&R 5.6)

Non-parametric estimation of probability density function: Silverman (1986)

One way: `histogram` with `prob=TRUE` option. : depends on the starting point of the grid of bins

A better way: a kernel density estimate

$$\hat{f}_h(x) = 1/nb \sum_{j=1, \dots, n} w((x-x_j)/h)$$

for a sample x_1, \dots, x_n , a fixed kernel $K()$ and a bandwidth b

The kernel **kernel** is normally chosen to be a probability density function, symmetric with zero mean. Default is normal. Usually doesn't matter much.

The bandwidth **b** is specified by **bw**, a multiple of the sd of the kernel. Determines the smoothness of \hat{f} . The key is to produce a smooth result, while not flattening out significant **modes**.

Mathematically, there's a compromise between

- 1) the bias of $\hat{f}(x)$, increases as b increases
- 2) the variance, decreases as b increases

Density estimation is usually harder (**meaning**, need more sample!) than probability or quantile estimation.

Choosing bandwidth: theory suggests that bandwidth proportional to $n^{-1/5}$, but the constant depends on the unknown density. Silverman's "rule of thumb", Silverman (1986, page 48, eqn (3.31) implemented in R is

$$b\text{-hat} = 0.9 \min(\sigma\text{-hat}, Q/1.34) n^{-1/5}.$$

where Q is the IQR (interquartile range). (divide by 1.34 since $EQ=1.34 * \sigma$ for normal.)

```
density(x, bw = "nrd0", adjust = 1,
        kernel = c("gaussian", "epanechnikov", "rectangular", "triangular",
                  "biweight", "cosine", "optcosine"),
        window = kernel, width,
        give.Rkern = FALSE,
        n = 512, from, to, cut = 3, na.rm = FALSE)
```

where w is

5 Bootstrap

Explicit recognition of uncertainty is central to the statistical sciences

Sometimes can be handled by analytical calculation based on an assumed probability model, but often the reality is too complicated

Bootstrap method:

1. Use computer to resample from the original data (either directly or via a fitted model) to create MANY (computer-intensive) replicate datasets, from which the variability of the quantities of interest can be assessed
2. It's not fraud. There's theory behind it.
3. Even when the standard analysis based on a parametric model seems OK, it helps to see what can be inferred w/o particular parametric model assumptions.

Ideas:

1. Be lazy: Avoid tedious calculations
2. Be smart: Avoid relying too much on questionable assumptions
3. No free lunch: Need clear understanding about the problem, design and methodology

5.1 Short version or 'Baby bootstrap' (Computer-intensive Statistics, V&R 5.7)

Make use of extensive repeated calculations to explore the sampling distribution of a parameter estimate $\hat{\theta}$.

- Suppose $x_1, \dots, x_n \sim \text{iid}$ from one member of a parametric family $\{F_\theta: \theta \in \Theta\}$ of distributions.
- Suppose further that $\hat{\theta} = T(\mathbf{x})$ is a symmetric function of the sample, that is it does not depend on the sample order. **Q**. Example?

The bootstrap procedure is to take m samples from \mathbf{x} with replacement and to calculate $\hat{\theta}^*$ for these samples.

IDEA: to assess the variability of $\hat{\theta}$ about the unknown true θ by the variability of $\hat{\theta}^*$ about $\hat{\theta}$.

The bias of $\hat{\theta}$ may be estimated by the (bootstrap) mean of $\hat{\theta}^* - \hat{\theta}$.

5.1.1 Galaxy data example (Median)

Example: we want to know the median m of the galaxies data.

Show the data. Run the R-sample to show 'density estimates' as well as 'rug' plot.

Q. best bet is the **sample median**: 20.8335

Q What's the big question after that?

How accurate/reliable is the estimator? What's the variability?

Large sample theory: sample median $\sim N(m, 1/4nf(m)^2)$

What's the problem with this approach? The density is unknown. Can try density estimator but it's unreliable.

Try bootstrap!

```
library(MASS)
data(galaxies)
gal <- galaxies/1000
median(gal)

plot(x = c(0, 40), y = c(0, 0.3), type = "n", bty = "1",
      xlab = "velocity of galaxy (1000km/s)", ylab = "density")
rug(gal)
lines(density(gal, width = 3.25, n = 200), lty = 1)
lines(density(gal, width = 2.56, n = 200), lty = 3)

set.seed(101)
m <- 1000
res <- numeric(m)
for(i in 1:m) res[i] <- median(sample(gal, replace=T))
```

```
mean(res - median(gal))
sqrt(var(res))

truehist(res, h=0.1)
lines(density(res, width="SJ-dpi", n=256))
```