Class Notes

Statistical Computing & Machine Learning

Class 9

Review

Likelihood.

- Choose the "best" of a set of competing models. Often the set is parameterized by quantities such as slope m, intercept b, rate , mean, standard deviation, ...
- "Figure of merit" for each model is the probability of the data given the model.
- Often, models have a deterministic part (e.g. $mx + b$) and a random part ϵ.
- Part of the model is our choice for the distribution of ϵ.
- Given that distribution, and treating each error ϵ_i as random, to calculate the likelihood we find the probability of each ϵ_i and multiply them together.
- For practical reasons (both algebraic and computational) we work with the log-likelihood.

Programming Basics: Conditionals

- a function: `ifelse(condition, yes_value, no_value)` Carries out the test for each element in the vector.
- a special form: `if (condition) {statements} else {statements}`

  Both of these can be nested.

`ifelse()` examples

```
library(ISLR)
data(Default) # loan balances and defaults.
```

Determine annual payment amount for student loans. E.g.

- If it's a student, no payment, otherwise $100
- If it's a student, no payment. Otherwise 10% of the balance.
- If the balance is less than 10 times the income, 10% of the balance, otherwise 5% of income.
- For those in default, nothing. For the others, any of the above three schemes.

if . . . else . . . examples

1. Calculate the median of a set of values.
   - if an odd number in the set, then the middle value
   - if an even number in the set, the mean of the two values in the middle.
2. Write a function that takes the case-by-case maximum of each of two input vectors.

- Unless the vectors are either numeric or character, and of the same type, throw an error.
- Add an argument `handle_na` which, if `TRUE` replaces `NA` with `-Inf` for the purposes of the comparison.
- Add an argument `na_rid=`
  - if "either", throw away the cases where either of the values is `NA`
  - if "both", throw away cases where both are `NA` and otherwise handle `NA` as `-Inf`
  - if "neither", keep all cases.

Some problems

Simple

Write functions that return, case by case,

- the maximum of two vectors.
- three vectors
- four vectors
- five vectors
- Write a supervisory function that does it for 1 to 5 vectors. Use `...`

```
max_in_parallel <- function(...) {
  Vecs <- list(...)
}
```

- Write a supervisory function that will handle more than 5 vectors.

Blood testing

It is known that 5% of the members of a population have disease X, which can be discovered by a blood test (that is assumed to perfectly identify both diseased and nondiseased populations). Suppose that N people are to be tested, and the cost of the test is nontrivial. The testing can be done in two ways:

a. Everyone can be tested separately; or
b. the blood samples of k people are pooled to be analyzed.

> Assume that N = nk with n being an integer. If the test is negative, all the people tested are healthy (that is, just this one test is needed). If the test result is positive, each of the k people must be tested separately (that is, a total of k + 1 tests are needed for that group).

i. For fixed k what is the expected number of tests needed in (B)?
ii. Find the k that will minimize the expected number of tests in (B).
iii. Using the k that minimizes the number of tests, on average how many tests does (B) save in comparison with (A)? Be sure to check your answer using an empirical simulation.

```
ntests <- function(p = 0.05, npools = 500, pool_size = 10, nsims=1000) {
  # generate the number of infected people in each pool
  infected_in_pool <- rbinom(npools, p = p, size = pool_size)
  # if one or more in a pool is infected, pool_size+1 tests,
  # otherwise 1 test
  tests_in_each_pool <- ifelse(infected_in_pool > 0, pool_size + 1, 1)
  # total across all pools
  sum(tests_in_each_pool)
}
```

Can we do this recursively to get more savings?

```
people <- runif(100000) < .05
ntests <- function(population) {
  if ( (! any(population)) || length(population) == 1) {
    # we're done!
    total_tests <- 1
  } else {
    # Split into two groups and test again
    split_point <- round(length(population)/2)
    group1 <- population[1:split_point]
    group2 <- population[(split_point + 1) : length(population)]
    total_tests <-
      ntests(group1) +
      ntests(group2)
  }

  total_tests
}
```

How many tests needed for a population of 10,000 with a prevalence of 1%?

```
do(10) * ntests(runif(10000) < 0.01)
```

```
##    ntests
## 1     643
## 2     715
## 3     690
## 4     657
## 5     715
## 6     762
## 7     616
## 8     621
## 9     700
## 10    601
```

The (hyper)-volume of the hypersphere.

A framework for the volumes: $C_n r^n$.

- $n = 1$ — the line segment of length $2r$. Volume is $2r^1$ so $C_n = 2$.
- $n = 2$ — the circle of radius r: volume is πr^2, so $C_2 = \pi$
- $n = 3$ — the sphere of radius r: volume is $\frac{4}{3}\pi r^3$, so $C_3 = \frac{4}{3}\pi$

TASK: Find $C_4, C_5, \ldots, C_8$.
Programming approach:

a. Write the logic. Give explicit names to all quantities that you might want to change later.

```
dim <- 3  # might vary
npts <- 1000  # might vary
pts <- matrix(runif(dim * npts, min=0, max=1), ncol=dim)
dists <- rowSums( pts^2 )
(2^dim) * sum(dists <= 1) / length(dists)
```

```
## [1] 4.136
```

b. Make the "might vary" quantities the arguments to a function that encapsulates the rest of the logic.

```
sphere_volume <- function(dim=3, npts=1000000) {
  pts <- matrix(runif(dim * npts, min=0, max=1), ncol=dim)
  dists <- rowSums( pts^2 )

  (2^dim) * sum(dists <= 1) / length(dists)
}
```

```
sapply(1:20, FUN = sphere_volume)
```

```
##  [1] 2.000000 3.140576 4.190248 4.941552
##  [5] 5.267168 5.152768 4.702720 4.010752
##  [9] 3.292160 2.549760 1.902592 1.466368
## [13] 0.876544 0.524288 0.360448 0.196608
## [17] 0.131072 0.262144 0.000000 0.000000
```

Why does the volume go to zero as the dimension increases?
Theoretical formula: $V_n(R) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)} R^n$

```
sapply(1:20, FUN=function(n) (pi^(n/2) / gamma(n/2 + 1)) )
```

```
##  [1] 2.00000000 3.14159265 4.18879020
##  [4] 4.93480220 5.26378901 5.16771278
##  [7] 4.72476597 4.05871213 3.29850890
## [10] 2.55016404 1.88410388 1.33526277
## [13] 0.91062875 0.59926453 0.38144328
## [16] 0.23533063 0.14098111 0.08214589
## [19] 0.04662160 0.02580689
```

Find the surface area, $D_n r^{n-1}$.

- $D_1 = 0$
- $D_2 = 2\pi$
- $D_3 = 4\pi$
- Find $D_4, D_5, \ldots, D_7$
- Two ways:

1. Take the derivative wrt r of the volume.
2. Find the fraction of points in a narrow shell between distance 0.99 and 1.01 from the origin.

In-class programming activity

Explanation of draw poker

- cards: ranks and suits
- hands: royal flush, straight-flush,

Draw poker link