Psychology 350: An introduction to R for Psychological Research
Week 3 Multidimensional analysis

William Revelle
Northwestern University
Evanston, Illinois USA

https://personality-project.org/courses/350

April, 2020

Outline

Fitting functions as way of testing theory

1. Closed form versus open form
2. Much of classical statistics is "closed form". That is, it is a matter of solving some equations using basic algebra.
   - Examples include the
   - t.test
   - F. test
   - basic linear regression
3. However, other functions are "open form" and have to be estimated using the process of iteration.
4. Although computers are useful for solving closed form equations, they are particularly useful for solving open forms.

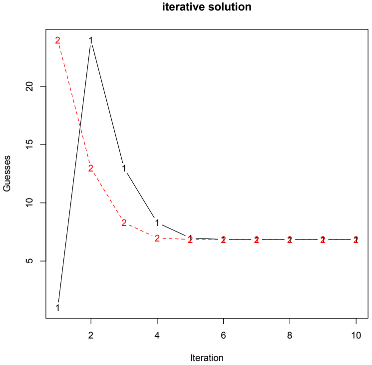
The basic concept of fitting

- Given some fit statistic, try to find the optimal values for that fit
  - Consider the case of the square root of 47
    - Guess X
    - find $47/Guess$
    - try a new $Guess = 47/Guess$
    - do it again
- The next example is a baby function to do this
- It is the concept, not the programming that is important

Iterative fitting to find a square root

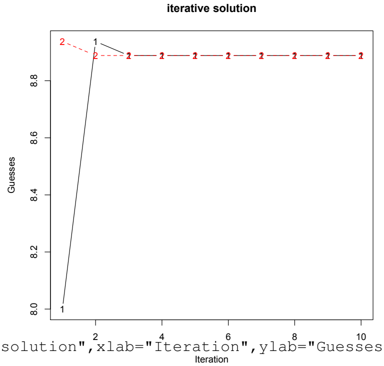
```
iterative.sqrt <- function(X,guess) {
if(missing(guess)) guess <- 1  #a dumb guess
iterate <- function(X,guess) {
    low <- guess
    high <- X/guess
    guess <-((high+low)/2)
    guess}
Iter <- matrix(NA,ncol=2,nrow=10)
for (i in 1:10) {

Iter[i,1] <- guess
Iter[i,2] <- guess <- iterate(X,guess)
}
Iter}   #this returns the value
```

```
> X <- 47
> iter <- iterative.sqrt(47)
> iter
           [,1]      [,2]
 [1,]  1.000000 24.000000
 [2,] 24.000000 12.979167
 [3,] 12.979167  8.300177
 [4,]  8.300177  6.981353
 [5,]  6.981353  6.856786
 [6,]  6.856786  6.855655
 [7,]  6.855655  6.855655
 [8,]  6.855655  6.855655
 [9,]  6.855655  6.855655
[10,]  6.855655  6.855655
```



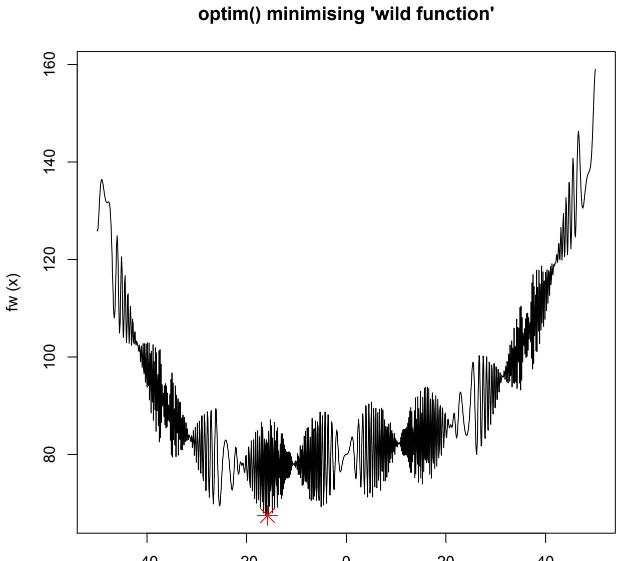
Iterative fitting to find a square root – a better guess

```
> X <- 79
> iter <- iterative.sqrt(X,8)
> iter
          [,1]     [,2]
 [1,] 8.000000 8.937500
 [2,] 8.937500 8.888330
 [3,] 8.888330 8.888194
 [4,] 8.888194 8.888194
 [5,] 8.888194 8.888194
 [6,] 8.888194 8.888194
 [7,] 8.888194 8.888194
 [8,] 8.888194 8.888194
 [9,] 8.888194 8.888194
[10,] 8.888194 8.888194
```




```
> matplot(iter,typ="b",main="iterative solution",xlab="Iteration",ylab="Guesses")
```

The optim function is one way to fit complex models



Iterative fitting has several steps

- Specifying a model and loss function
  - Ideally supplying a first derivative to the loss function
  - Can be done empirically
- Some way to evaluate the quality of the fit
- Minimization of function, but then how good is this minimum

Multivariate analysis

- Many procedures use this concept of iterative fitting
- Some, such as principal components are "closed form" and can just be solved directly
- Others, such as "factor analysis" need to find solutions by successive approximations
- The issue of factor or component rotation is an iterative procedure.
- Principal Components and Factor analysis are all the result of some basic matrix equations to approximate a matrix
- Conceptually, we are just taking the square root of a correlation matrix:
- $R \approx CC'$ or $R \approx FF' + U2$
- For any given correlation matrix, R, can we find a C or an F matrix which approximates it?

Consider the following matrix

What would be its "square root"? That is to say, what simpler matrix, times itself is equal to R?

```
R
     V1   V2   V3   V4
V1 1.00 0.56 0.48 0.40
V2 0.56 1.00 0.42 0.35
V3 0.48 0.42 1.00 0.30
V4 0.40 0.35 0.30 1.00
```

We create this matrix using the `sim.congeneric` function

R code

```
"sim.congeneric" <-  function{
 (loads = c(0.8, 0.7, 0.6, 0.5),...)  {
..
n <- length(loads)
 loading <- matrix(loads, nrow = n)
error <- diag(1, nrow = n)
diag(error) <- sqrt(1 - loading^2)

 model <- pattern %*% t(pattern)

...
 result <- model
 return(result)
 }
```

1. Give some default values
2. Create a correlation matrix using matrix algebra
3. Return the value

A congeneric matrix is one with just one factor

R code

```
R <- sim.congeneric()
R
 f1 <- fa(R)
```

```
> R
     V1   V2   V3   V4
V1 1.00 0.56 0.48 0.40
V2 0.56 1.00 0.42 0.35
V3 0.48 0.42 1.00 0.30
V4 0.40 0.35 0.30 1.00
> f1 <- fa(R)
> f1
Factor Analysis using method =  minres
Call: fa(r = R)
Standardized loadings (pattern matrix) based upon correlation matrix
   MR1   h2   u2 com
V1 0.8 0.64 0.36   1
V2 0.7 0.49 0.51   1
V3 0.6 0.36 0.64   1
V4 0.5 0.25 0.75   1

                MR1
SS loadings    1.74
Proportion Var 0.43
```

But does this really fit?

Think about the residuals (Data - model)

R code

```
R
round(f1$model,2)
residuals(f1)
```

```
> R
     V1   V2   V3   V4
V1 1.00 0.56 0.48 0.40
V2 0.56 1.00 0.42 0.35
V3 0.48 0.42 1.00 0.30
V4 0.40 0.35 0.30 1.00
> round(f1$model,2)
     V1   V2   V3   V4
V1 0.64 0.56 0.48 0.40
V2 0.56 0.49 0.42 0.35
V3 0.48 0.42 0.36 0.30
V4 0.40 0.35 0.30 0.25
> residuals(f1)
   V1   V2   V3   V4
V1 0.36
V2 0.00 0.51
V3 0.00 0.00 0.64
V4 0.00 0.00 0.00 0.75
```

Hence we add the fudge factor U2 = diagonal of residuals and say $R = FF' + U2$