Optimization with R

Guy Yollin

Seattle R Users

Personal Introduction

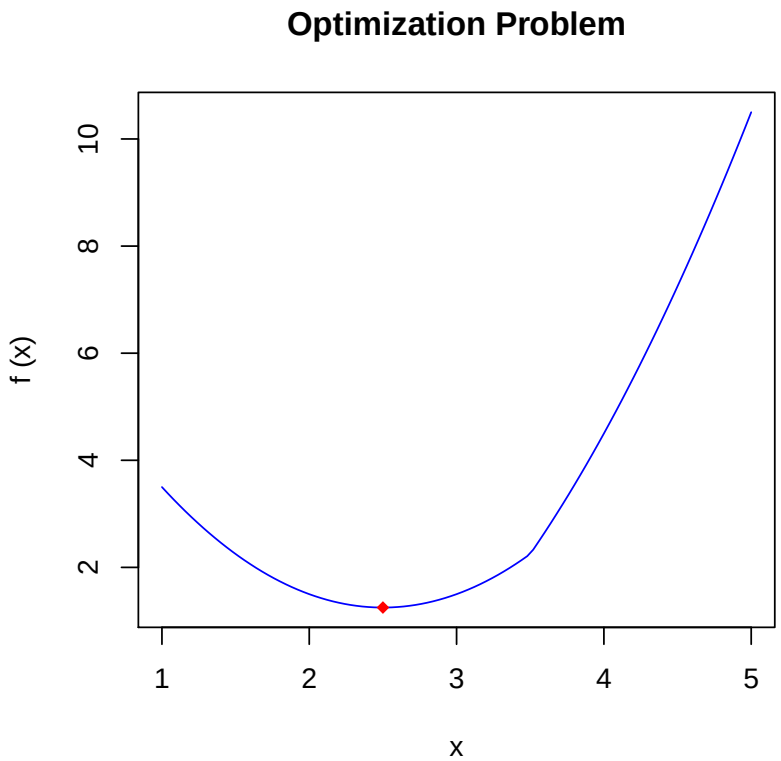
- Guy Yollin - `http://www.linkedin.com/in/guyyollin`
- Professional Experience
  - Rotella Capital Management, Quantitative Research Analyst
  - Insightful Corporation, Director of Financial Engineering
  - J.E. Moody, LLC, Financial Engineer
  - Oregon Graduate Institute (OHSU), Adjunct Instructor
  - Electro Scientific Industries, Director of Engineering, Vision Products Division
- Education
  - Oregon Graduate Institute, Masters in Computational Finance
  - Drexel University, Bachelors in Electrical Engineering
- Using R since 1999

Outline

Outline

Overview of Optimization

- The generic optimization problem:
  - Given a function $f(x)$
  - Find the value x^*
  - Such that $f(x^*)$ obtains a *maximum (or minimum)* value
  - Subject to other constraints on x



Overview of Optimization

- Traditional optimization applications:
  - Financial and investments
  - Manufacturing and industrial
  - Distribution and networks
- Applications in computational statistics:
  - Model fitting
  - Parameter estimation
  - Maximizing likelihood

Optimization Functions in R

R directly supports a number of powerful optimization capabilities

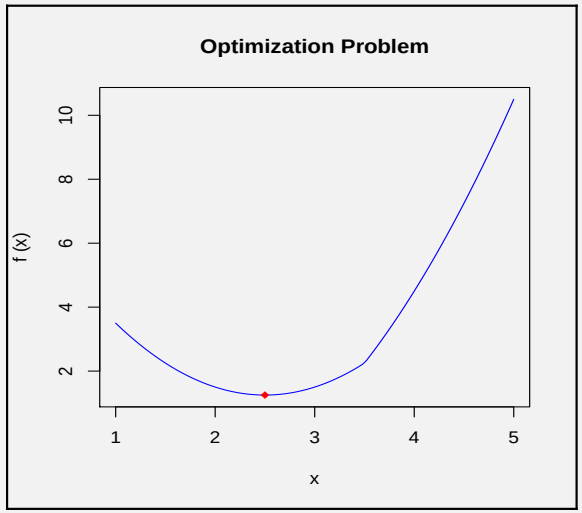
optimize	Single variable optimization over an interval
optim	General-purpose optimization based on Nelder-Mead, quasi-Newton and conjugate-gradient algorithms
nlminb	Unconstrained and constrained optimization using PORT routines
Rglpk	R interface to the GNU Linear Programming Kit for solving LPs and MILPs
solve.QP	Quadratic programming (QP) solver
DEoptim	Performs evolutionary global optimization via the differential evolution algorithm

optimize()

Single variable optimization: $f(x) = |x - 3.5| + (x - 2)^2$

R Code:

```
> args(optimize)
function (f, interval, ..., lower = min(interval), upper = max(interval),
    maximum = FALSE, tol = .Machine$double.eps^0.25)
NULL
> f <- function(x) {
     abs(x - 3.5) + (x - 2)^2
 }
> op <- optimize(f = f, interval = c(1, 5))
> op
$minimum
[1] 2.5

$objective
[1] 1.25
```



Outline

The optim function

The optim function is a general purpose multi-variate optimizer

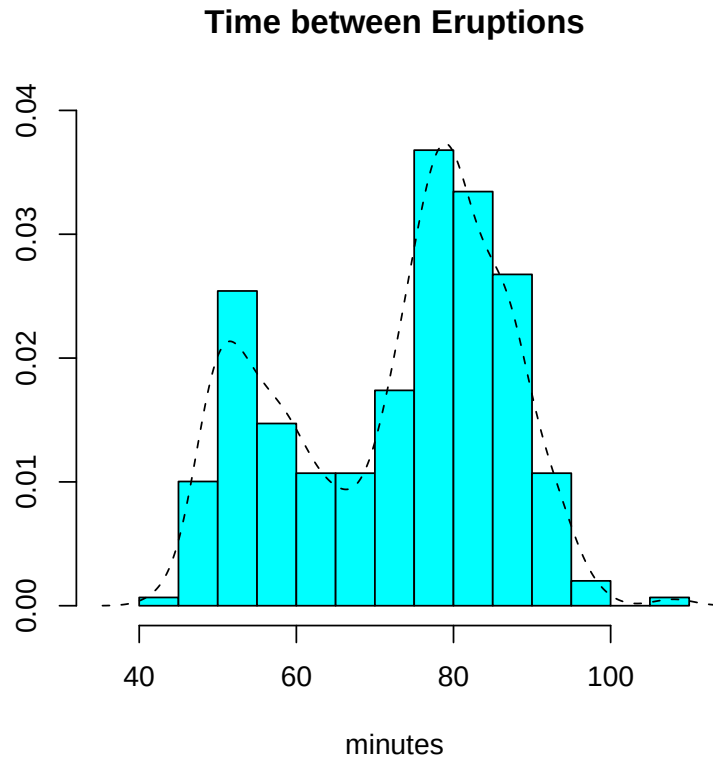
R Code: optim **arguments**

```
> args(optim)
function (par, fn, gr = NULL, ..., method = c("Nelder-Mead",
    "BFGS", "CG", "L-BFGS-B", "SANN"), lower = -Inf, upper = Inf,
    control = list(), hessian = FALSE)
NULL
```

`fn`	objective function that takes a vector of parameters (required)
`par`	initial value of the parameters (required)
`gr`	the gradient function (optional)
`method`	optimization method to be used (optional)
`lower`	box constraints of parameters (optional)
`upper`	box constraints of parameters (optional)

Fitting a Mixture Model

- From section 16.3 of Venables & Ripley 4th Edition
- Time between Old Faithful geyser eruptions
- Objective function is the log-likelihood of a mixture of 2 normal distributions



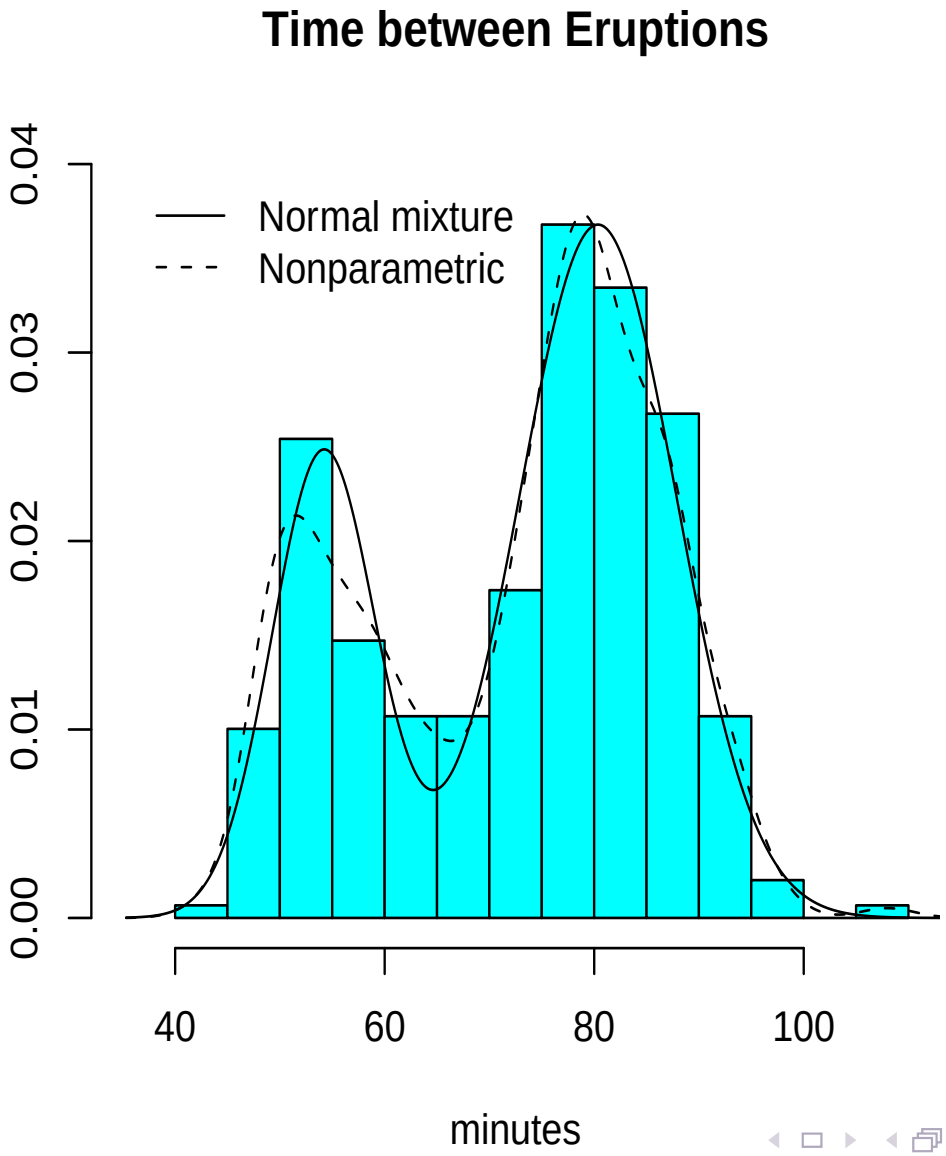
$$L(\pi, \mu_1, \sigma_1, \mu_2, \sigma_2) = \sum_{i=1}^{n} log[\frac{\pi}{\sigma_1}\phi(\frac{y_i - \mu_1}{\sigma_1}) + \frac{1-\pi}{\sigma_2}\phi(\frac{y_i - \mu_2}{\sigma_2})]$$

Optimization Example

R Code: optimizing with `optim`

```
> # objective function
> mix.obj <- function(p, x)
 {
 e <- p[1]/p[3] * dnorm( (x - p[2])/p[3] ) +
        (1-p[1])/p[5] * dnorm( (x - p[4])/p[5] )
   if(any(e <= 0)) Inf else -sum(log(e))
 }
> # define initial values
> (p0 <- c(p = mean(waiting < 70), u1 = 50, s1 = 5, u2 = 80, s2 = 5))
        p        u1        s1        u2        s2
 0.361204 50.000000  5.000000 80.000000  5.000000
> # optimize
> mix.nl0 <- optim(p0, mix.obj, x = waiting)
> mix.nl0$par
         p         u1         s1         u2         s2
 0.3073520 54.1964325  4.9483442 80.3601824  7.5110485
> mix.nl0$convergence
[1] 0
```

The Mixture Model Fit



Outline

Linear Programming

Linear programming (LP) problems have a linear objective function and linear constraints

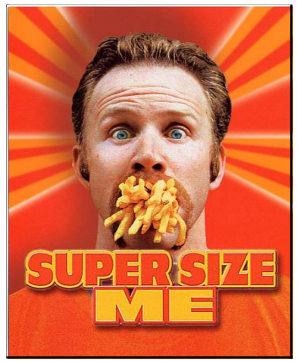
$$\begin{aligned} min_x \quad & c^T x \\ & Ax \geq b \\ & x \geq 0 \end{aligned}$$

where

- x is a vector of variables to be opitmized (length n)
- c is a vector of objective coefficients (length n)
- A is a constraints matrix (dimension m x n)
- b is a vector of constraints bounds (length m)

Menu Planner

- Minimize the cost of a day's worth of meals at Mickey D's
- Subject to
  - nutritional constraints
  - variety constraints (max 4 of any item)
  - integer constraints (no fractional Big Macs)



Food	Cost	Cals	Carbs	Protein	VitaA	VitaB	Calc	Iron
Quarter Pounder	1.84	510	34	28	15	6	30	20
McLean Delux	2.19	370	35	24	15	10	20	20
Big Mac	1.84	500	42	25	6	2	25	20
FiletOFish	1.44	370	38	14	2	0	15	10
McGrilled Chicken	2.29	400	42	31	8	15	15	8
Small Fries	0.77	220	26	3	0	15	0	2
Sausage McMuffin	1.29	345	27	15	4	0	20	15
Lowfat Milk	0.60	110	12	9	10	4	30	0
Orange Juice	0.72	80	20	1	2	120	2	2
Minimum		2000	350	55	100	100	100	100
Maximum		3500	375					

The Rglpk package

R Code: the Rglpk_solve_LP

```
> library(Rglpk)
Using the GLPK callable library version 4.42
> args(Rglpk_solve_LP)
function (obj, mat, dir, rhs, types = NULL, max = FALSE, bounds = NULL,
    verbose = FALSE)
NULL
```

obj	vector of objective coefficients
mat	constraints matrix
dir	gt, gte, lt, lte, eq for each constraint
rhs	vector of constraints bounds
type	type of variables (continuous, integer, binary)
bounds	box constraints on variables
max	minimization/maximization problem

Preparing to call the solver

R Code: Objects in the diet problem

```
> fmat
                  Cals Carbs Protein VitaA VitaB Calc Iron
Quarter Pounder    510    34      28    15     6   30   20
McLean Delux       370    35      24    15    10   20   20
Big Mac            500    42      25     6     2   25   20
FiletOFish         370    38      14     2     0   15   10
McGrilled Chicken  400    42      31     8    15   15    8
Small Fries        220    26       3     0    15    0    2
Sausage McMuffin   345    27      15     4     0   20   15
Lowfat Milk        110    12       9    10     4   30    0
Orange Juice        80    20       1     2   120    2    2
> Cost
  Quarter Pounder       McLean Delux            Big Mac
             1.84               2.19               1.84
       FiletOFish McGrilled Chicken        Small Fries
             1.44               2.29               0.77
 Sausage McMuffin        Lowfat Milk       Orange Juice
             1.29               0.60               0.72
```

Preparing to call the solver

R Code: Objects in the diet problem

```
> Amat <- rbind(t(fmat),t(fmat))
> minAmt
   Cals   Carbs Protein   VitaA   VitaB    Calc    Iron
   2000     350      55     100     100     100     100
> maxAmt
   Cals   Carbs Protein   VitaA   VitaB    Calc    Iron
   3500     375     Inf     Inf     Inf     Inf     Inf
> dir <- c(rep(">=",numCons),rep("<=",numCons))
> bounds <- list(lower = list(ind = 1:numVars, val = rep(0,numVars)),
                 upper = list(ind = 1:numVars, val = rep(4,numVars)))
> bounds$upper
$ind
[1] 1 2 3 4 5 6 7 8 9

$val
[1] 4 4 4 4 4 4 4 4 4
```

Calling the LP Solver

R Code: call to `Rglpk_solve_LP`

```
> sol <- Rglpk_solve_LP(obj=Cost, mat=Amat, dir=dir, rhs=c(minAmt,maxAmt),
     types = rep("I",numVars), max = FALSE, bounds=bounds, verbose = F)
> (x <- sol$solution)
[1] 2 1 1 0 0 2 1 4 4
> t(fmat) %*% x
        [,1]
Cals    3435
Carbs    352
Protein  166
VitaA    103
VitaB    550
Calc     253
Iron     107
> Cost %*% x
      [,1]
[1,] 15.82
```

Outline

Quadratic Programming

Quadratic programming (QP) problems have a quadratic objective function and linear constraints

$$\begin{aligned} min_x \quad & c^T x + \frac{1}{2} x^T Q x \\ s.t. \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

where

- x is a vector of variables to be optimized (length n)
- Q is a symmetric matrix (dimension m x n)
- c is a vector of objective coefficients (length n)
- A is a constraints matrix (dimension m x n)
- b is a vector of constraints bounds (length m)

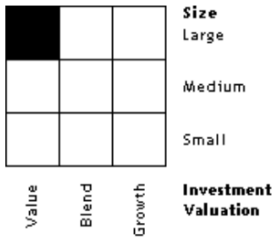
Style Analysis

Vanguard Windsor Investor
Po Box 2600V37Valley Forge PA 19482
Map
Phone: 800-662-7447

Fund Overview	
Category:	Large Value
Fund Family:	Vanguard
Net Assets:	12.25B
Year-to-Date Return:	1.99%
Yield:	1.54%
Morningstar Rating:	★★★
Fund Inception Date:	1958-10-23

Morningstar Style Box

Large Value
[View Category Definition]



View Top Large Value Funds
About the Morningstar Style Box

Style Analysis is a technique that attempts to determine the fundamental drivers of a mutual fund's returns

Sharpe's return-based style analysis:

$$min_w \quad var(R_{MF} - w_1 R_{B_1} - \ldots w_n R_{B_n})$$

$$s.t. \quad \sum w_i = 1$$

$$0 \leq w_i \leq 1$$

$$var(R_{MF} - w_1 R_{B_1} - \ldots w_n R_{B_n}) = 2(\frac{w^T V w}{2} - c^T w) + var(R_{MF})$$

The quadprog package

R Code: the solve.QP function

```
> library(quadprog)
> args(solve.QP)
function (Dmat, dvec, Amat, bvec, meq = 0, factorized = FALSE)
NULL
```

Dmat	matrix in quadratic part of objective function
dvec	vector in linear part of objective function
Amat	constraints matrix
bvec	vector of constraints bounds
meq	first *meq* constraints are equality constraints

Quadratic Programming

R Code: setup to call solve.QP

```
> head(z)
              WINDSOR  LARGEVALUE  LARGEGROWTH SMALLVALUE SMALLGROWTH
2005-10-03 -0.1593626 -0.09535756 -0.005995983  0.3403397   0.4733873
2005-10-04 -0.7202912 -1.27326280 -0.762412906 -1.0561886  -0.9461192
2005-10-05 -1.3748700 -1.66080920 -1.475015837 -2.8568633  -2.9082448
2005-10-06 -0.2445986 -0.51217665 -0.485592013 -0.5641660  -1.1559082
2005-10-07  0.4073325  0.41993619  0.330131430  0.7721558   0.7480504
2005-10-10 -0.8163311 -0.88589613 -0.564549912 -0.9743418  -1.0118028
> (Dmat = var(z[,-1]))
            LARGEVALUE LARGEGROWTH SMALLVALUE SMALLGROWTH
LARGEVALUE    2.959438    2.433880   3.276065    2.951333
LARGEGROWTH   2.433880    2.191768   2.760910    2.613387
SMALLVALUE    3.276065    2.760910   4.249303    3.773710
SMALLGROWTH   2.951333    2.613387   3.773710    3.559450
> (dvec = var(z[,1], z[,-1]))
     LARGEVALUE LARGEGROWTH SMALLVALUE SMALLGROWTH
[1,]   2.878281    2.437179   3.214248    2.948217
```

Quadratic Programming

R Code: setup to call solve.QP

```
> (Amat <- t(rbind(rep(1,N), diag(N), -diag(N))))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    1    1    0    0    0   -1    0    0    0
[2,]    1    0    1    0    0    0   -1    0    0
[3,]    1    0    0    1    0    0    0   -1    0
[4,]    1    0    0    0    1    0    0    0   -1
> # weights sum to 1, weights >=0, weights <= 0
> b0 <- c(1, rep(0,N),rep(-1,N))
> #
> optimal <- solve.QP(Dmat, dvec, Amat, bvec = b0, meq = 1)
> W <- matrix(round(100*optimal$solution),nrow=2,byrow=T)
> dimnames(W) <- list(c("large","small"),c("value","growth"))
> W
      value growth
large    69     27
small     0      4
```

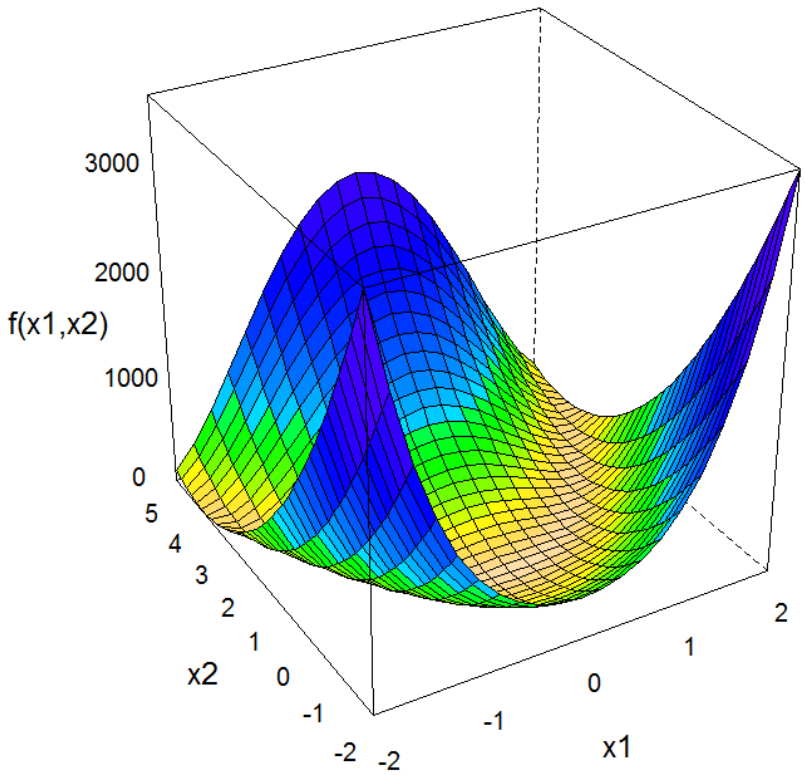
Outline

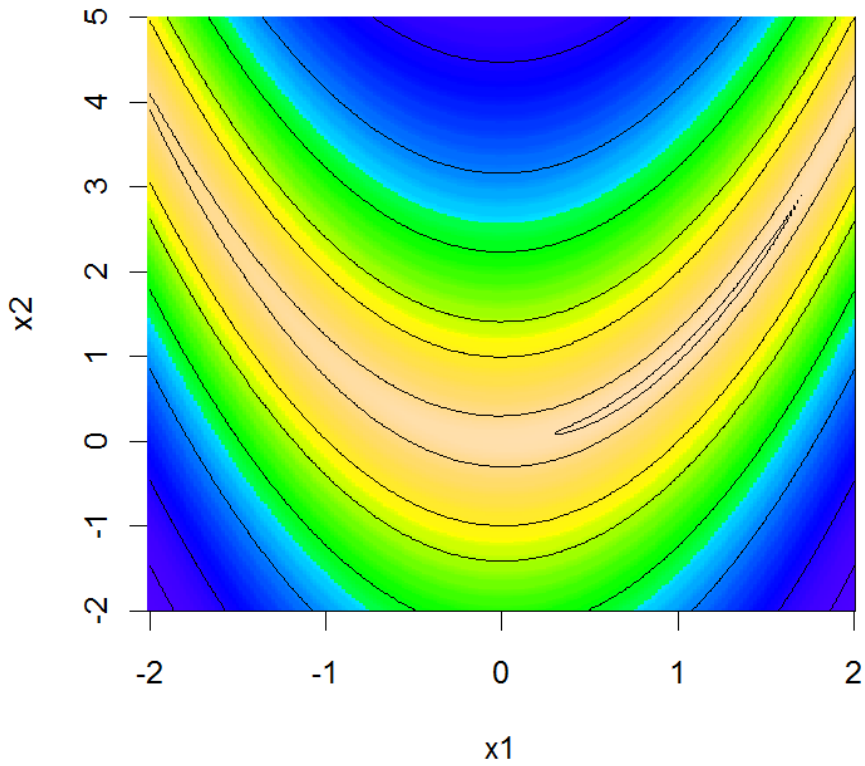
Differential Evolution Algorithm

- Differential Evolution (*DE*) is a very *simple* and yet very *powerful* population based stochastic function minimizer
- Ideal for global optimization of multidimensional, nonlinear, multimodal, highly-constrained functions (i.e. really hard problems)
- Developed in mid-1990 by Berkeley researchers Ken Price and Rainer Storn
- Implemented in R in the package `DEoptim`

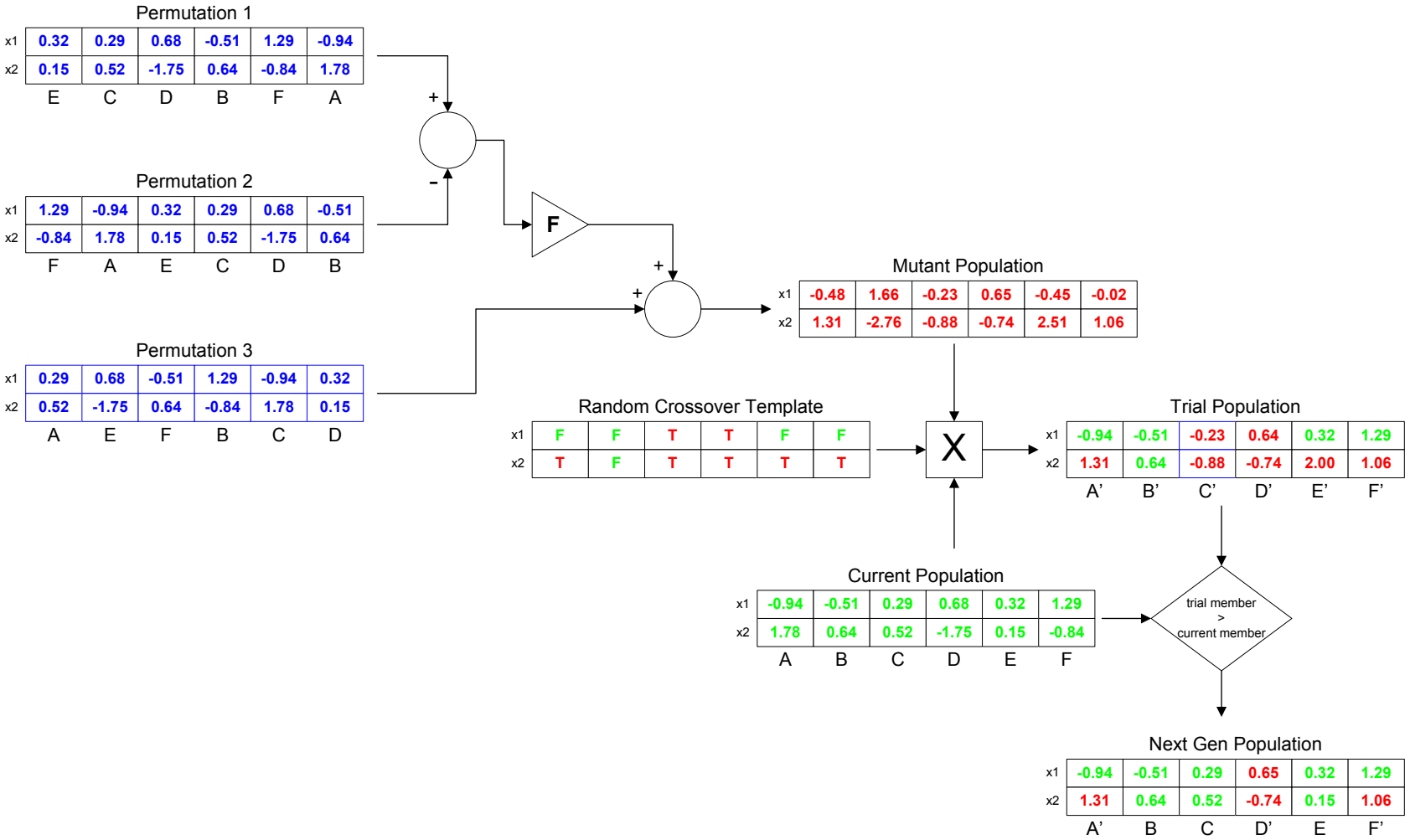
Rosenbrock Banana Function

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

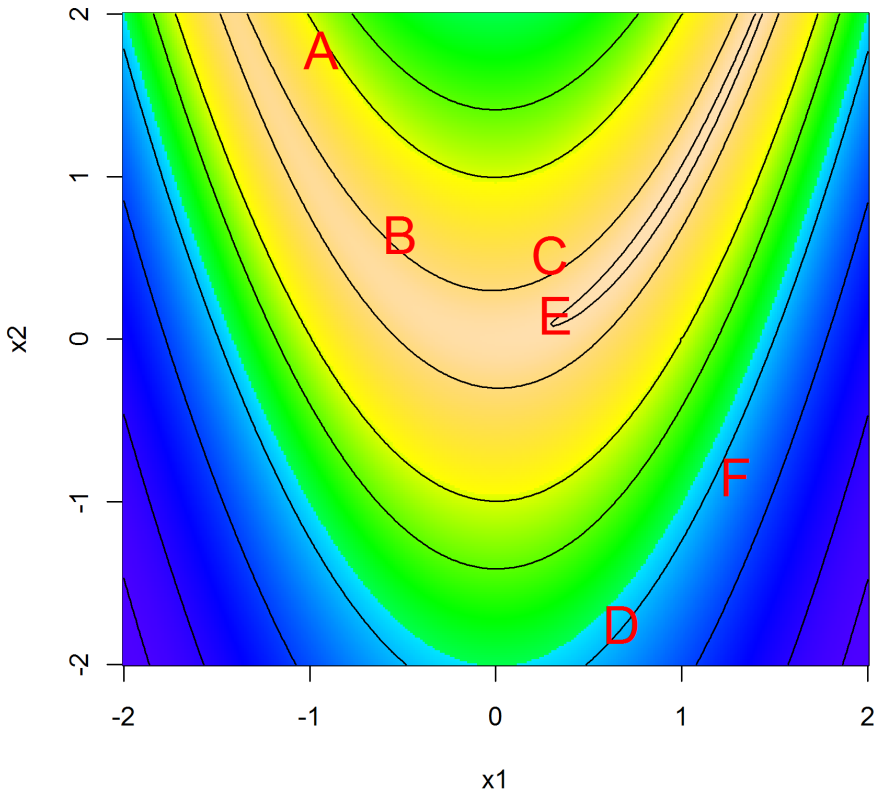


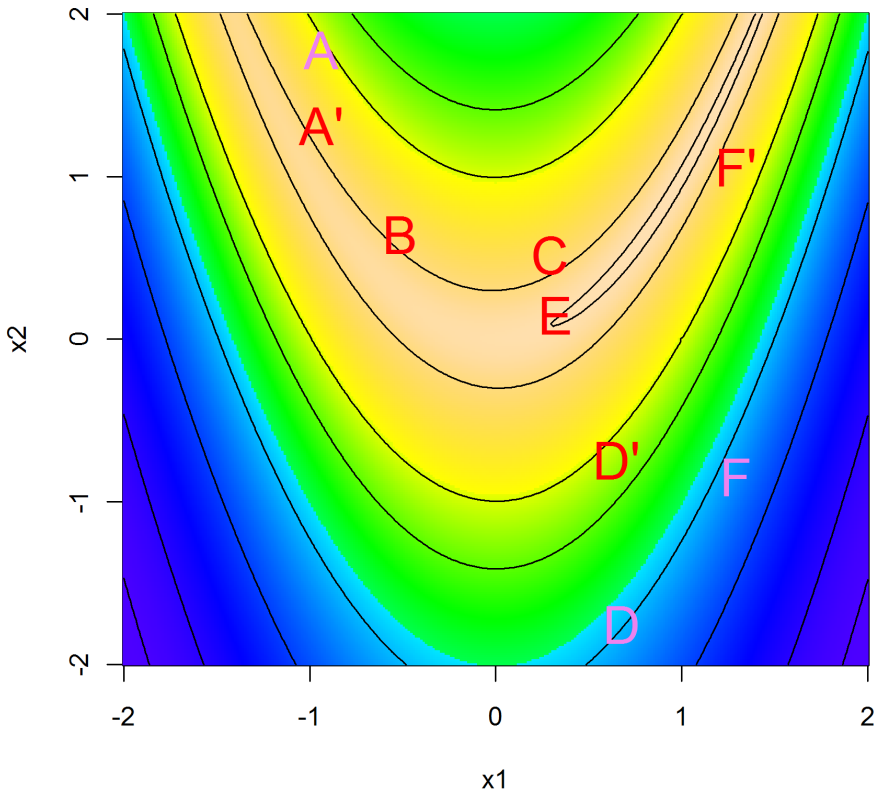


Differential Evolution Algorithm



Differential Evolution Rosenbrock Example

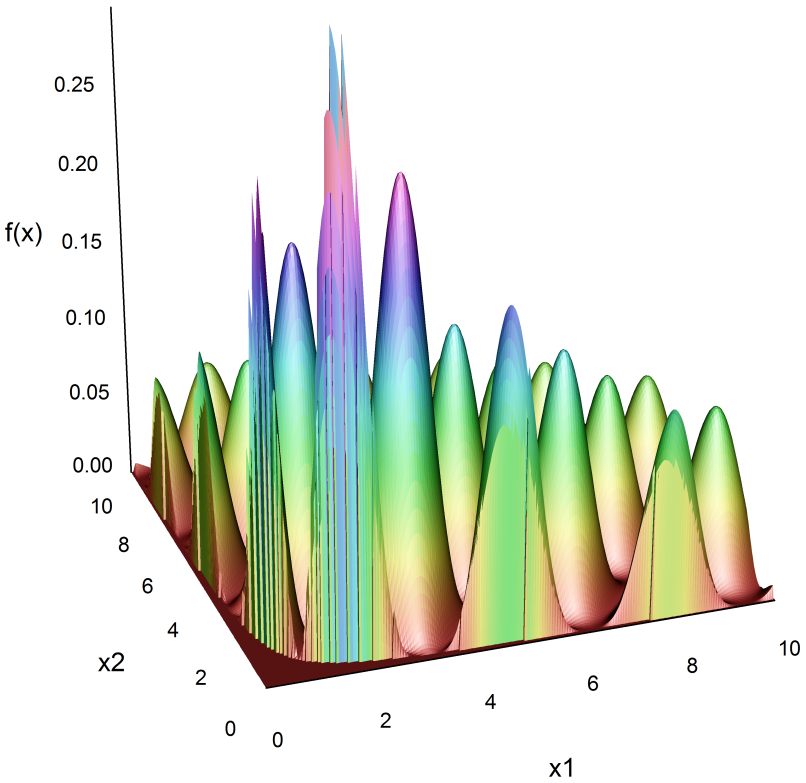


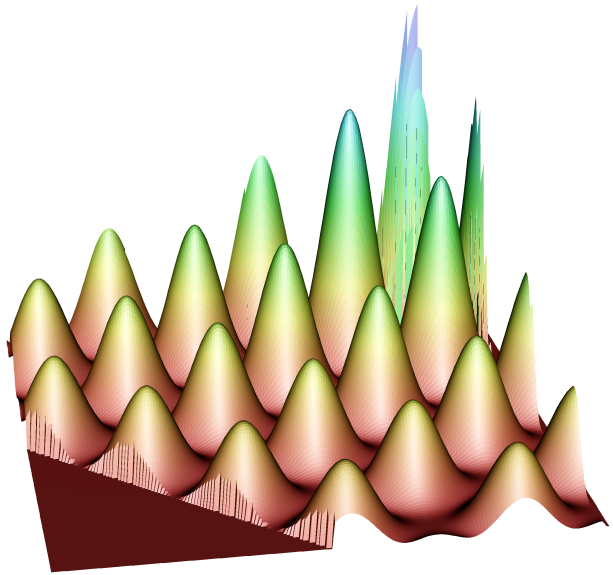


G2 Function

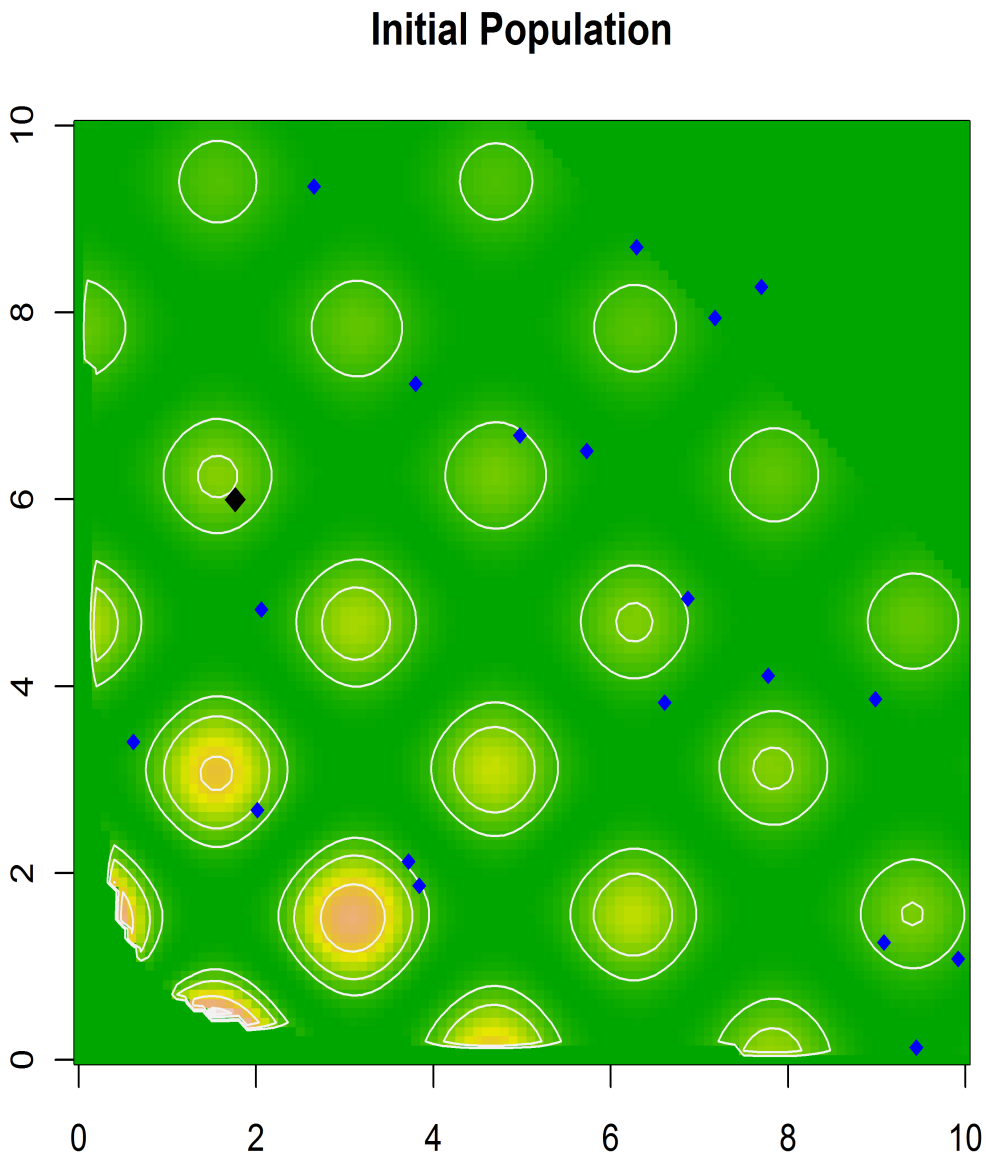
$$f(x_1, x_2) = \left| \frac{\cos(x_1)^4 + \cos(x_2)^2 - 2\cos(x_1)^2 \cos(x_2)^2}{\sqrt{x_1^2 + 2x_2^2}} \right|$$

where: $0 \leq x_1 \leq 10$, $0 \leq x_2 \leq 10$, $x_1 x_2 \geq 0.75$, $x_1 + x_2 \leq 15$

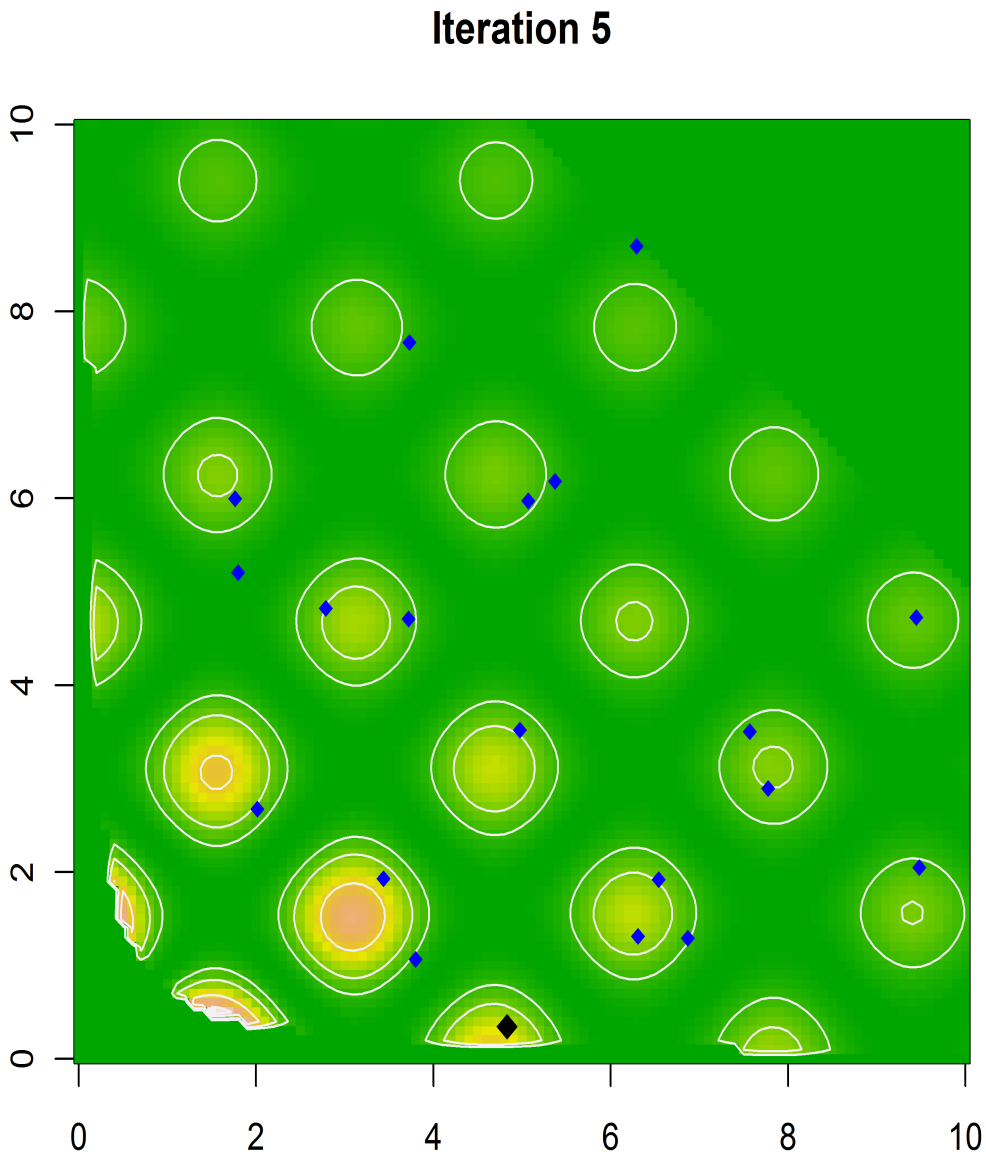




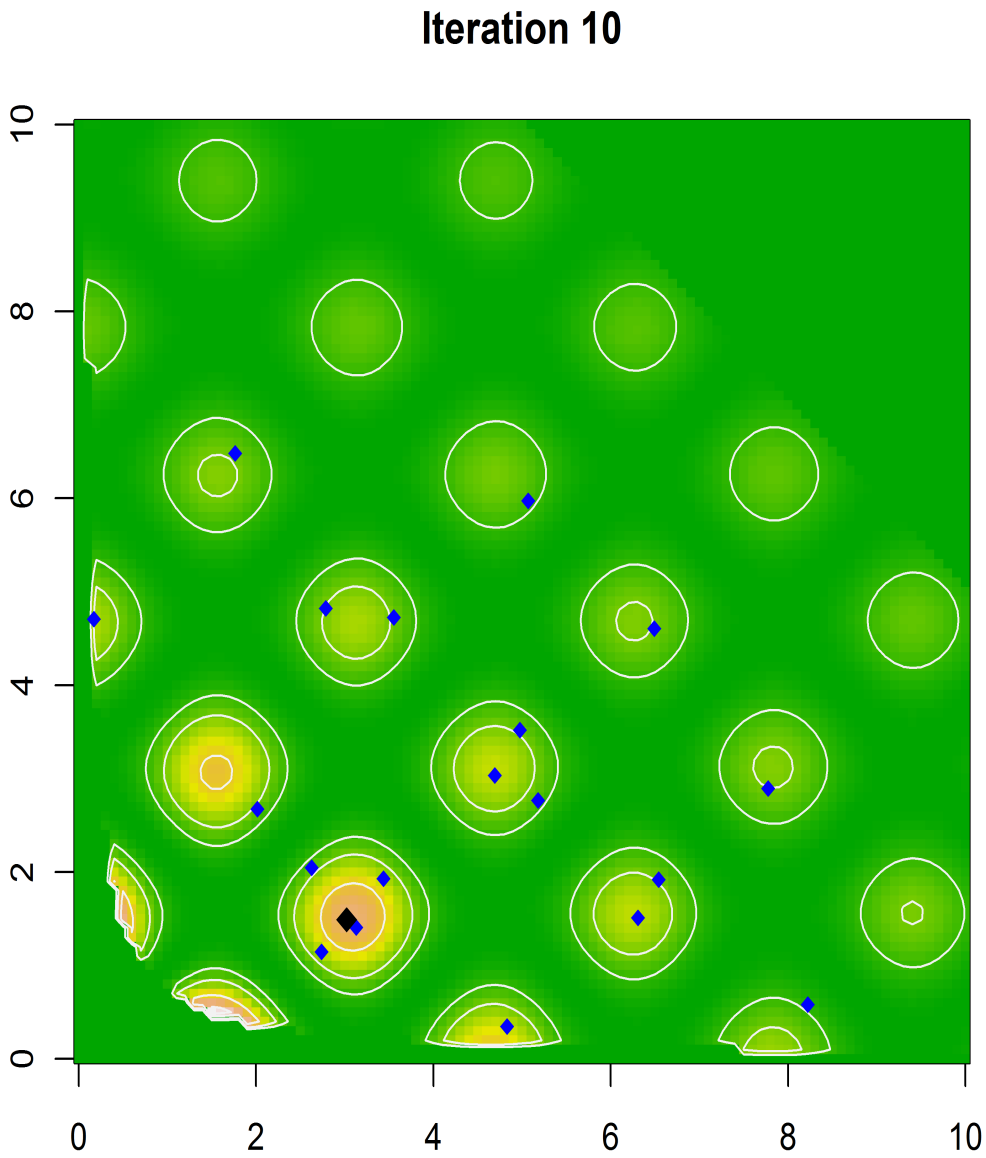
Finding G2 Maximum with Differential Evolution



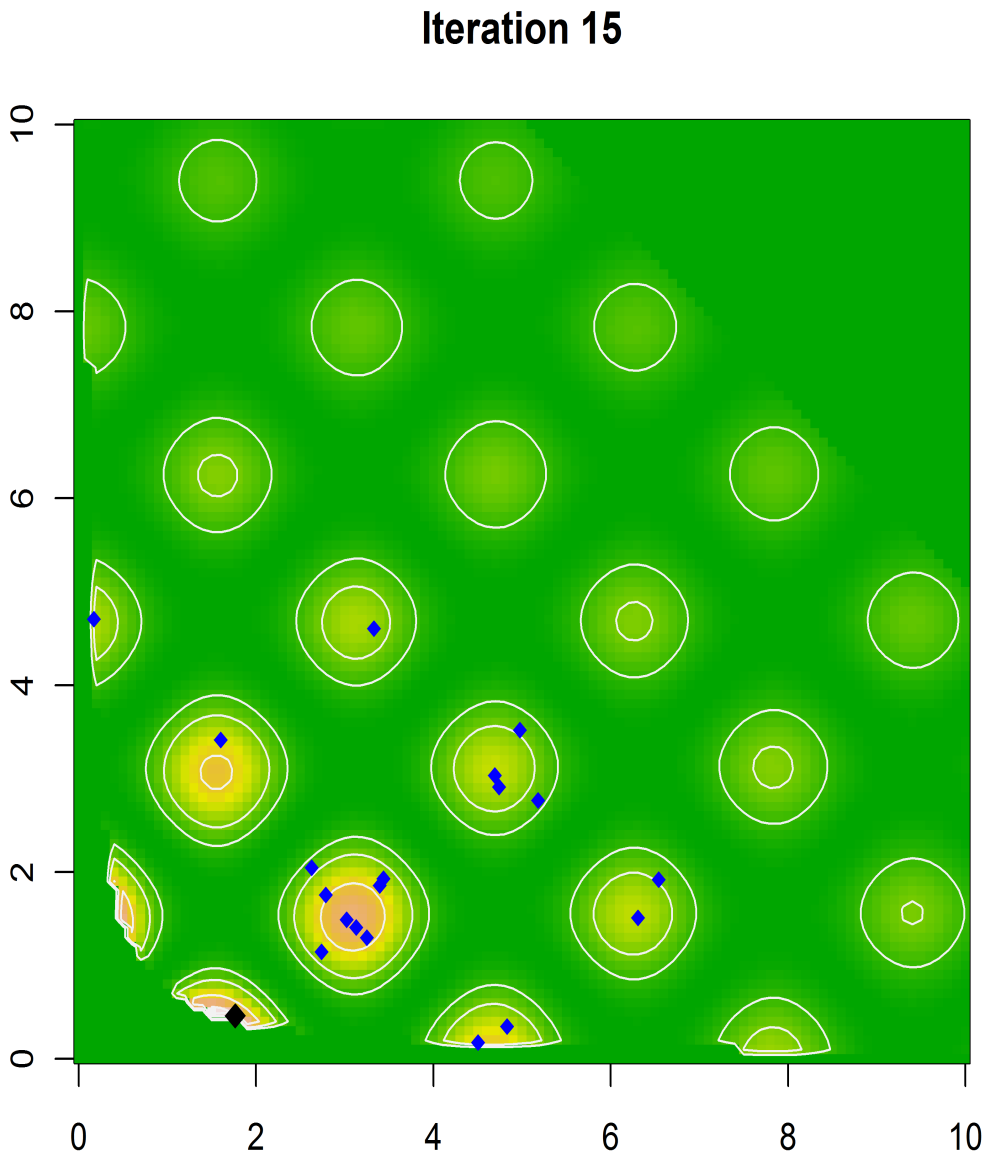
Finding G2 Maximum with Differential Evolution



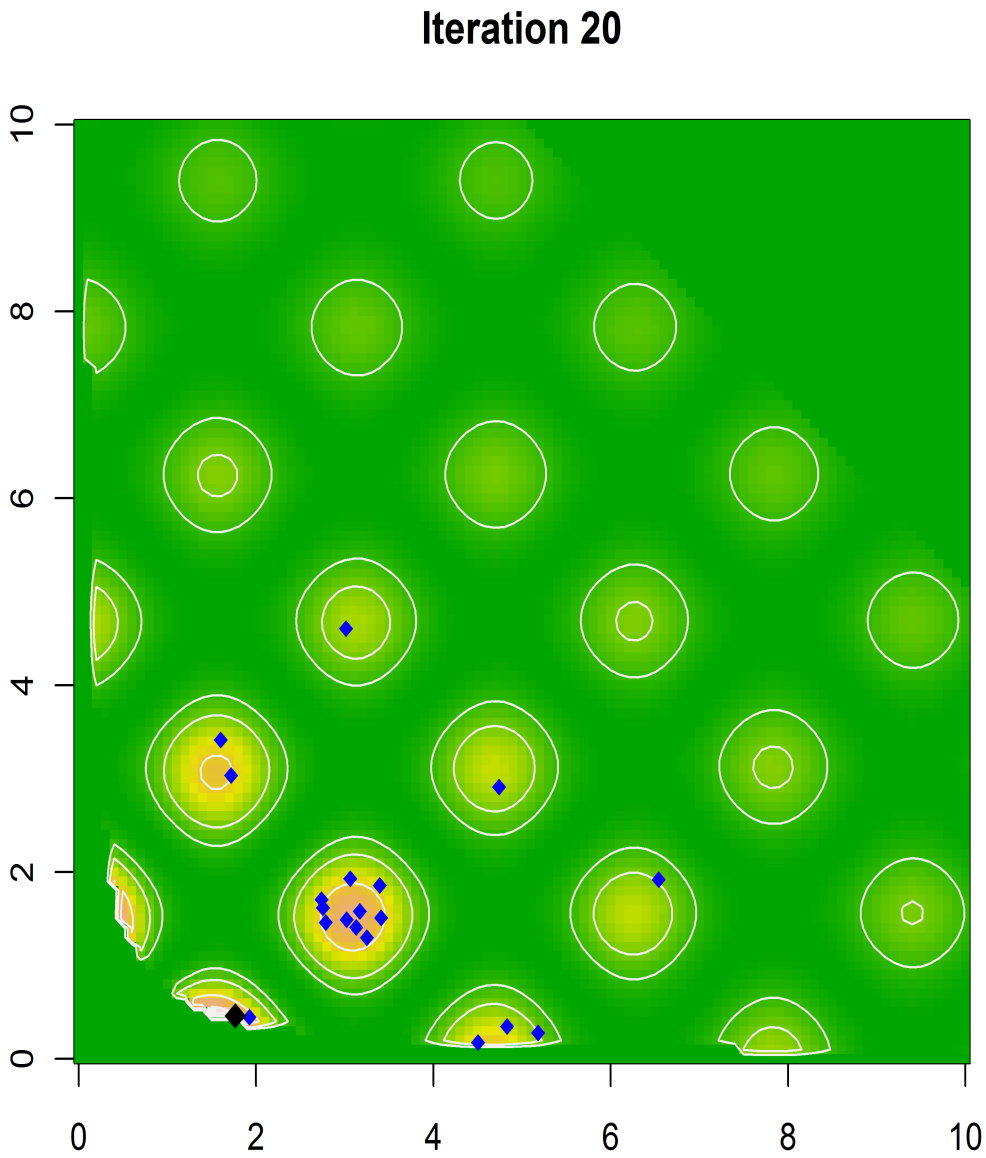
Finding G2 Maximum with Differential Evolution



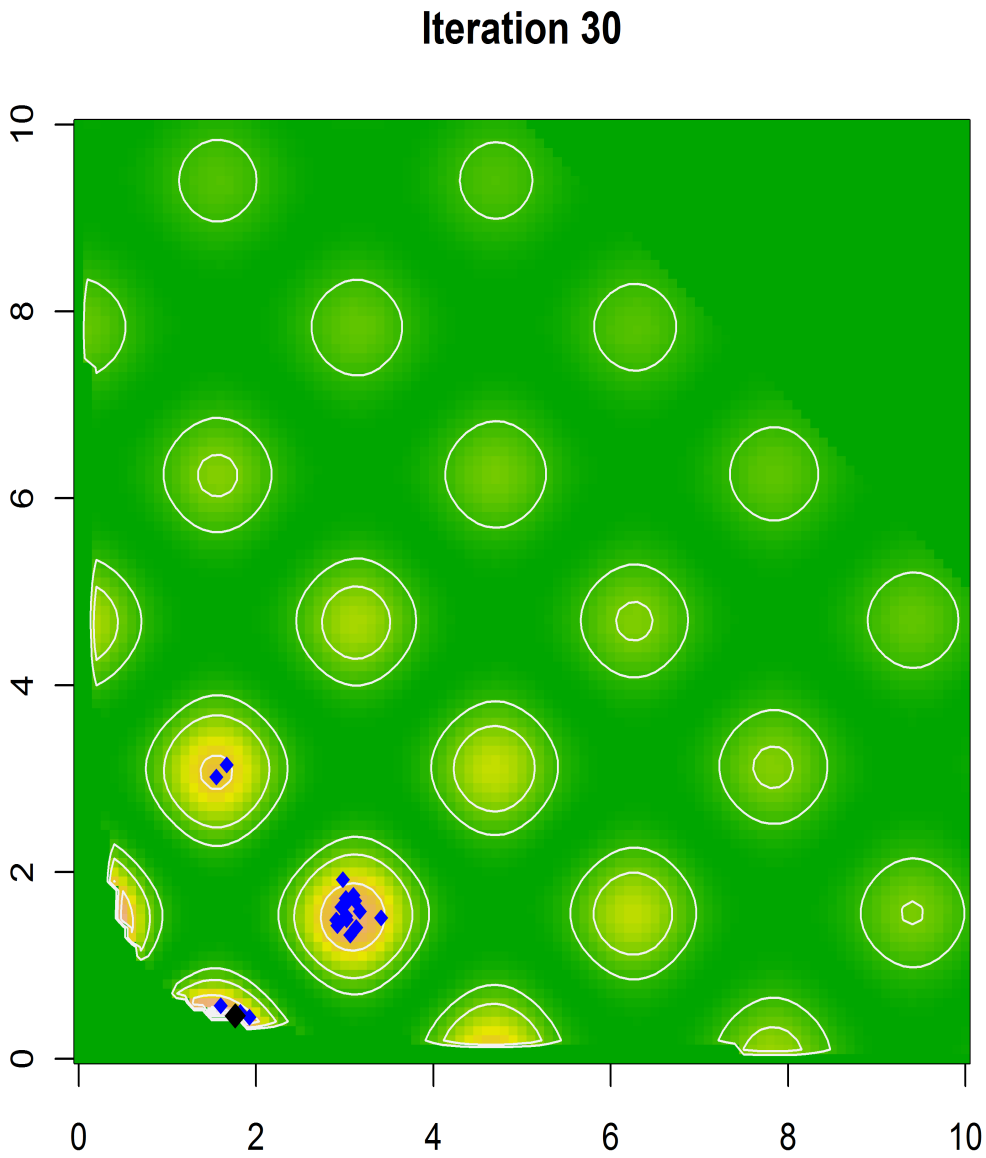
Finding G2 Maximum with Differential Evolution



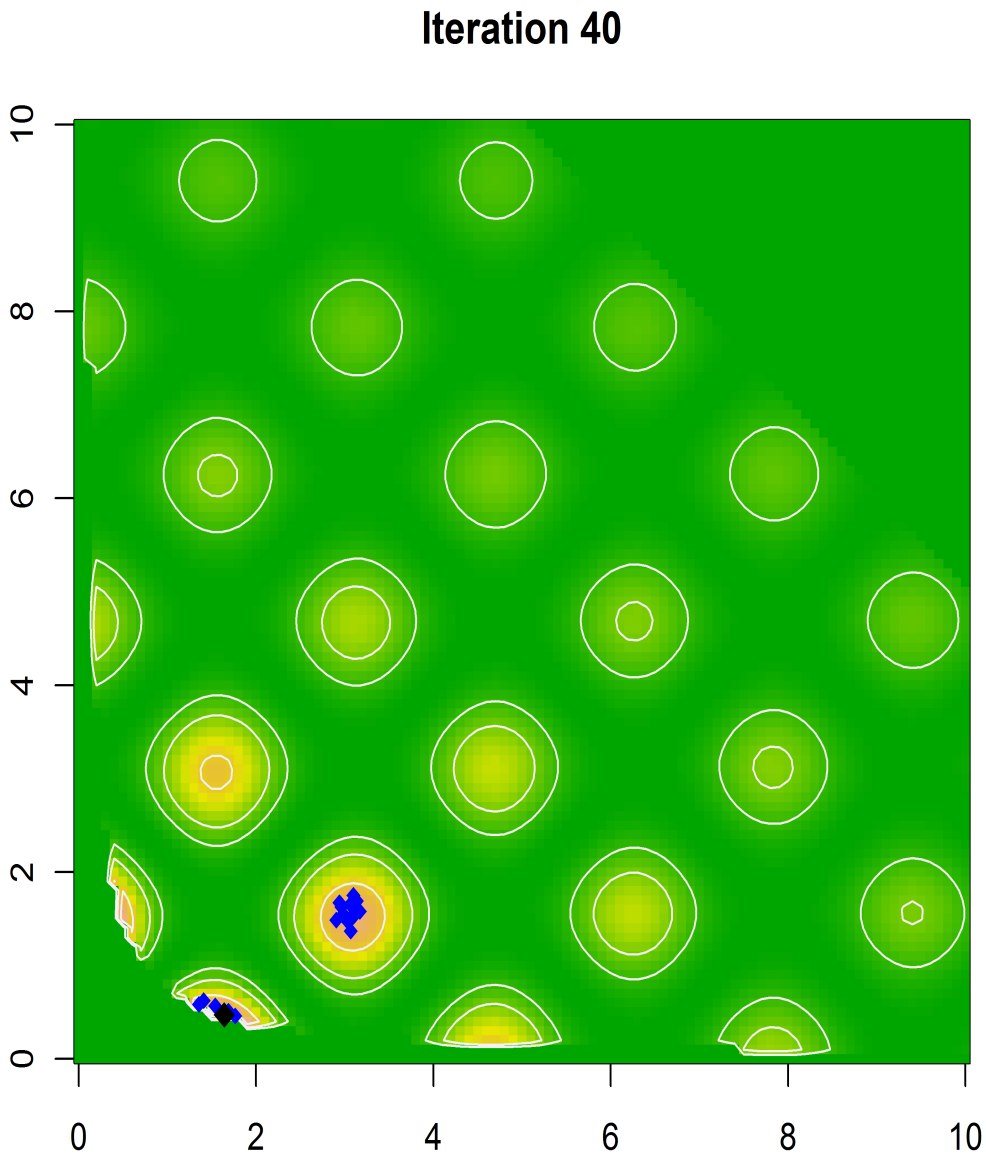
Finding G2 Maximum with Differential Evolution



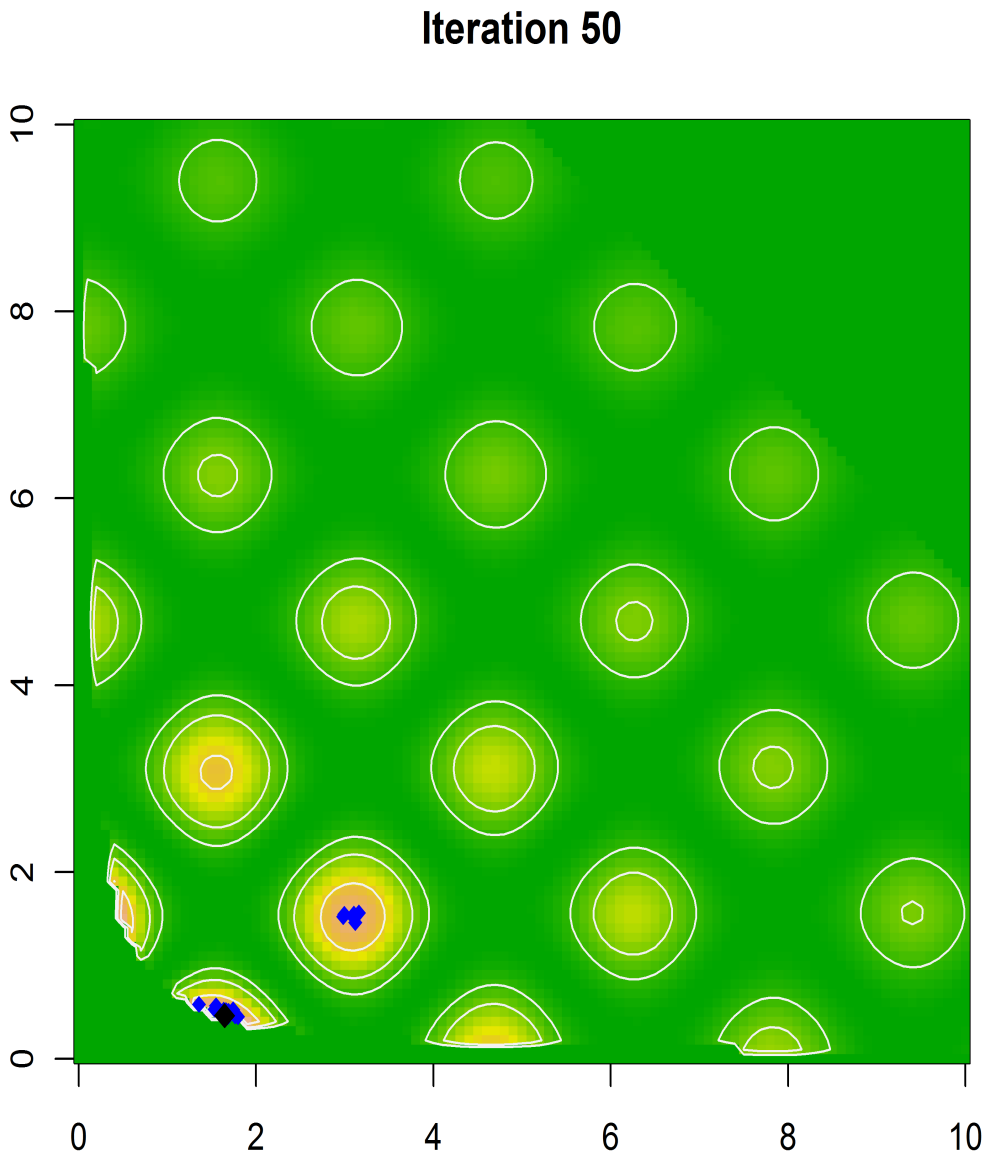
Finding G2 Maximum with Differential Evolution



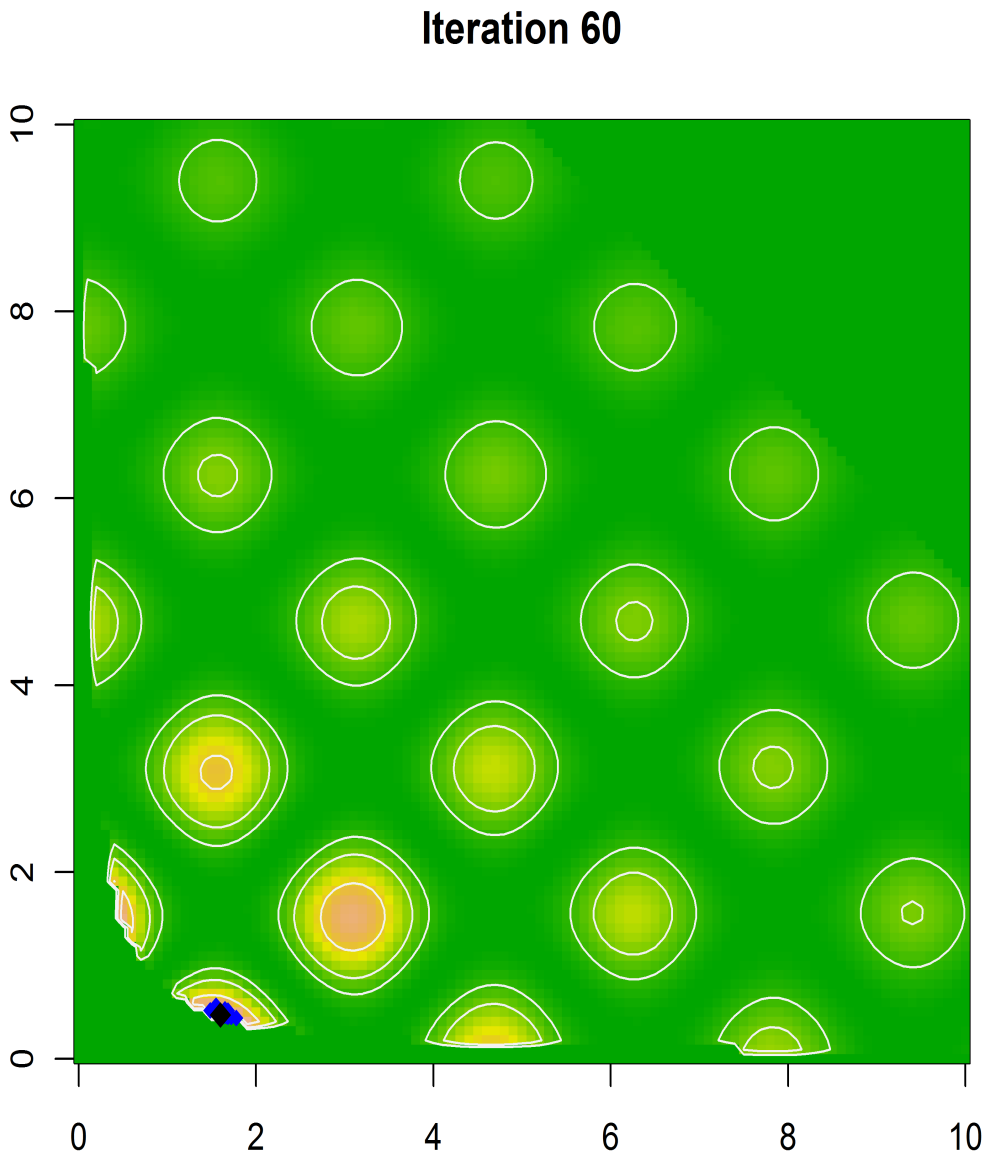
Finding G2 Maximum with Differential Evolution



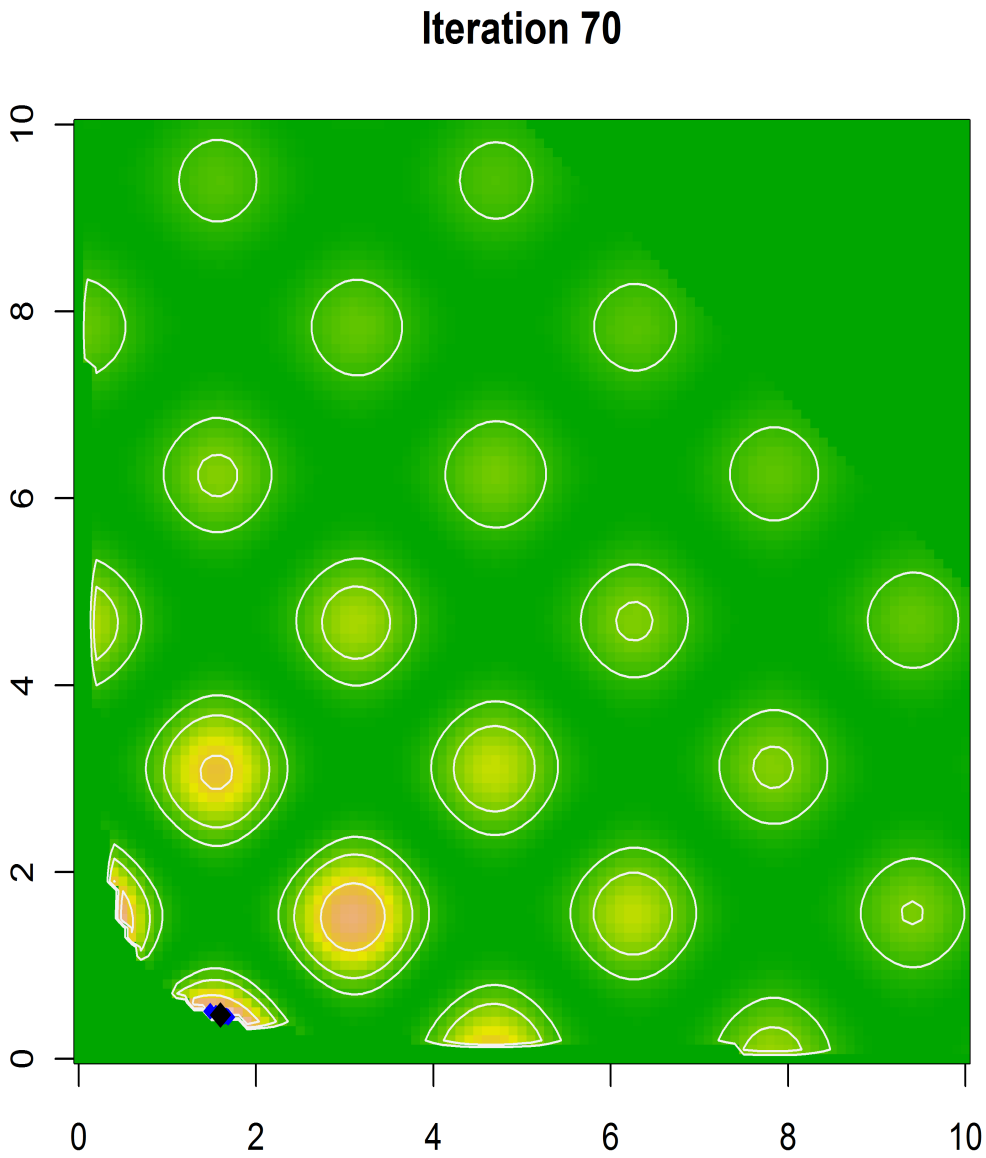
Finding G2 Maximum with Differential Evolution



Finding G2 Maximum with Differential Evolution



Finding G2 Maximum with Differential Evolution



The DEoptim function

R Code: `DEoptim` **arguments**

```
> args(DEoptim)
function (fn, lower, upper, control = DEoptim.control(), ...)
NULL
```

`fn`	objective function to be optimized
`lower`	lower bound on parameters
`upper`	upper bound on parameters
`control`	list of control parameters

The DEoptim.control function

R Code: `DEoptim.control` **arguments**

```
> args(DEoptim.control)
function (VTR = -Inf, strategy = 2, bs = FALSE, NP = 50, itermax = 200,
    CR = 0.5, F = 0.8, trace = TRUE, initialpop = NULL, storepopfrom = itermax +
        1, storepopfreq = 1, checkWinner = FALSE, avWinner = TRUE,
    p = 0.2)
NULL
```

`NP`	number of population member
`itermax`	max number of iterations (population generations)
`strategy`	differential evolution strategy
`F`	step size for scaling difference
`CR`	crossover probability
`VTR`	value-to-reach

The DEoptim function

R Code: call to DEoptim

```
> G2 = function( x ) {
   if( x[1] >=0 & x[1] <=10 & x[2] >=0 & x[2] <= 10 &
       x[1] * x[2] >= 0.75 & x[1] + x[2] <= 15 ) {
     s <- cos(x[1])^4+cos(x[2])^4
     p <- 2*cos(x[1])^2*cos(x[2])^2
     r <- sqrt(x[1]^2+2*x[2]^2)
     f <- -abs((s-p)/r)
   } else {
     f <- 0
   }
   return(f)
 }
> lower = c(0,0)
> upper = c(10,10)
> res = DEoptim(G2, lower, upper,
   control = list(NP = 20, itermax = 100, strategy=1, trace=F))
```

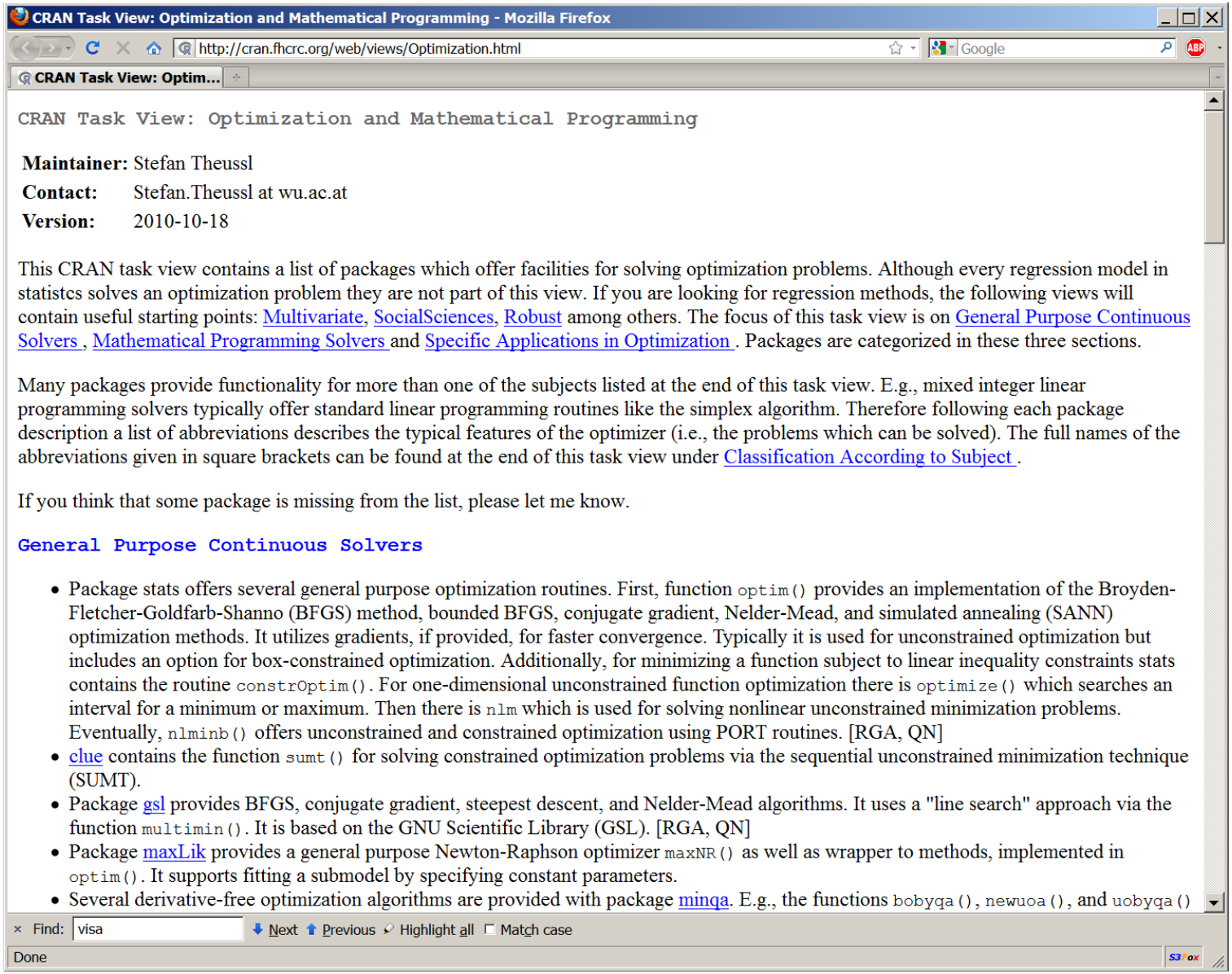
The DEoptim function

R Code: results of DEoptim

```
> res$optim
$bestmem
     par1      par2
1.6013695 0.4683657

$bestval
[1] -0.3649653

$nfeval
[1] 2020

$iter
[1] 100
```

Outline

Optimization Task View



Other optimization activities to keep an eye on

Project	Description	Location	Contributors
optimizer	optimX, numerical derivatives	R-Forge	Nash, Mullen, Gilbert
Rmetrics2AMPL	R interface to AMPL environment	rmetrics.org	Wuertz, Chalabi
COIN-OR	open source software for OR	coin-or.org	COIN-OR Foundation
RnuOPT	R interface to the NuOPT optimizer	msi.co.jp	Yamashita, Tanabe

Outline

General Optimization References


W. N. Venables and B. D. Ripley
Modern Applied Statistics with S, 4th Edition.
Springer, 2004.


O. Jones and R. Maillardet
Introduction to Scientific Programming and Simulation Using R.
Chapman and Hall/CRC, 2009.


W. J. Braun and D. J. Murdoch
A First Course in Statistical Programming with R.
Cambridge University Press/CRC, 2008.

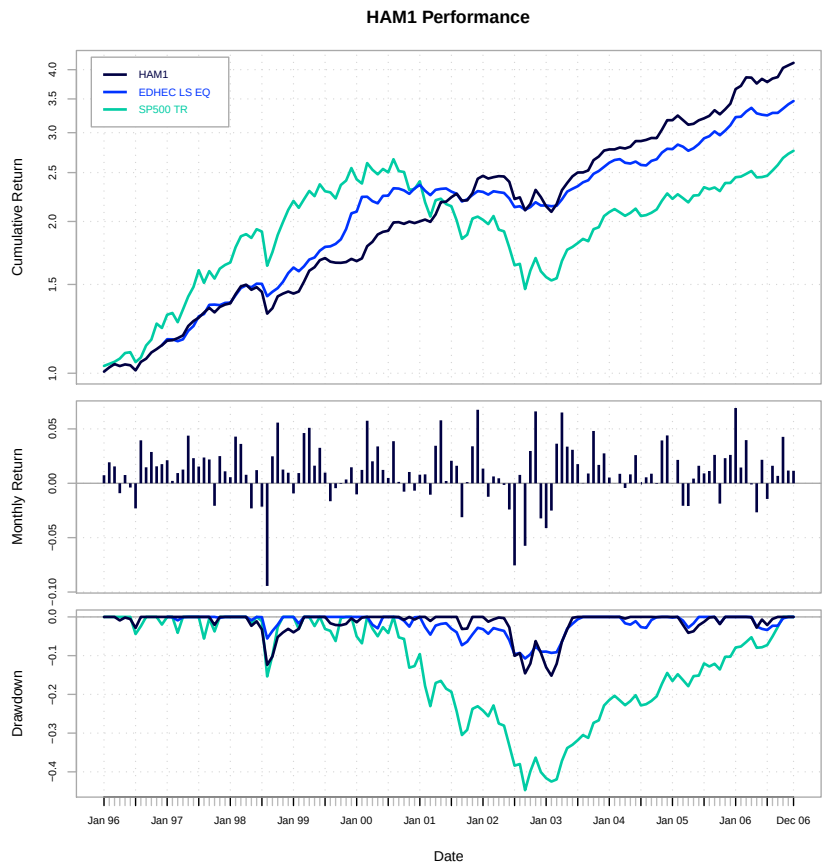
Differential Evolution References

K. M. Mullen and D. Ardia
DEoptim: An R Package for Global Optimization by Differential Evolution.
2009.

K. Price and R. Storn
Differential Evolution: A Practical Approach to Global Optimization.
Springer, 2005.

University of Washington Online Certificate in Computational Finance

- Survey computational finance methods and techniques through the development of R software
  - Statistical analysis of asset returns
  - Financial time series modeling
  - Factor models
  - Portfolio optimization



Plot from the `PerformanceAnalytics` package

Conclusion

Thank You for Your Time!