The Revised Sequential Parameter Optimization Toolbox

Thomas Bartz-Beielstein, Martin Zaefferer, Jörg Stork and Sebastian Krey

TH Köln University of Applied Sciences

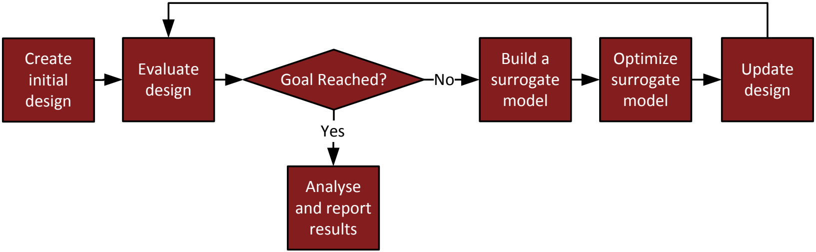
5th July 2017

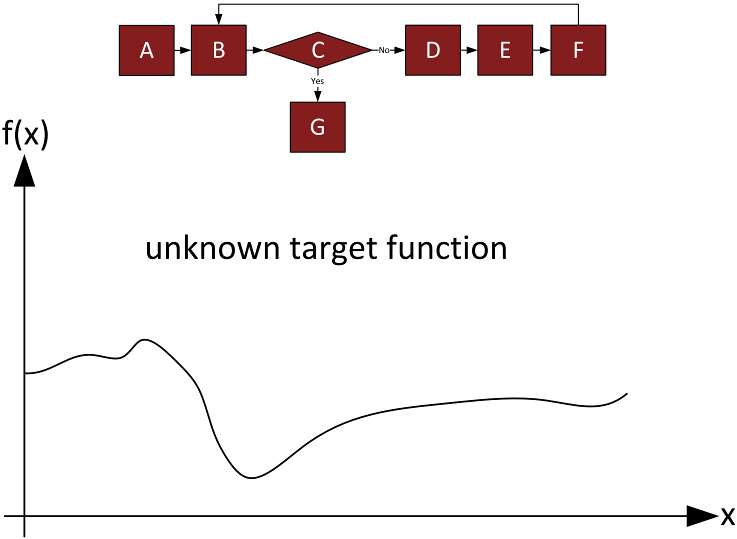
Technology
Arts Sciences
TH Köln

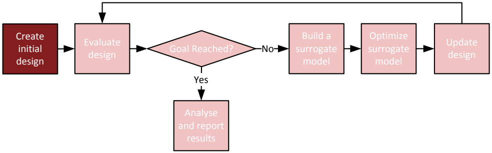
Sequential Parameter Optimization: Overview

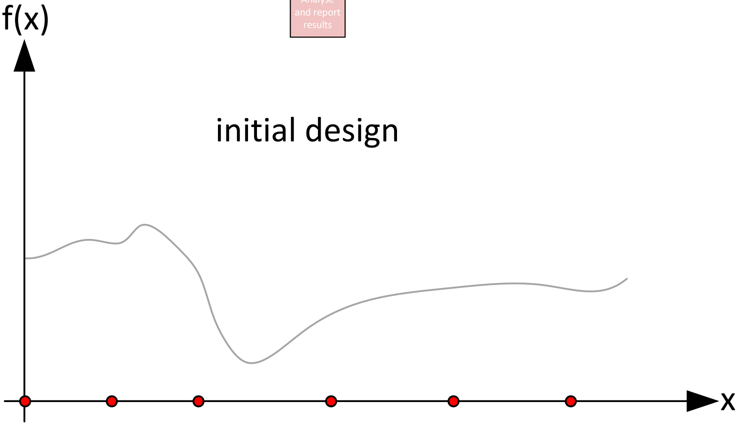
- Developed: Bartz-Beielstein et al. (2005)
- Core purpose:
  - Derive understanding of problem, parameters
  - Reduce load of costly target functions
  - Statistically sound comparisons
- Combines approaches from different fields
  - Design of Experiment
  - Statistics
  - Optimization algorithms
- Areas of application
  - Algorithm tuning
  - Engineering design
  - And many more (Bartz-Beielstein, 2010)
- R-package maintained by SPOTSeven research group

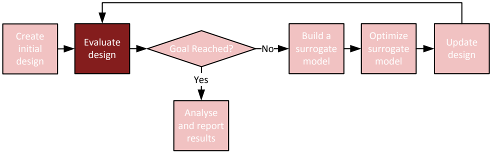
Sequential Parameter Optimization: Concept

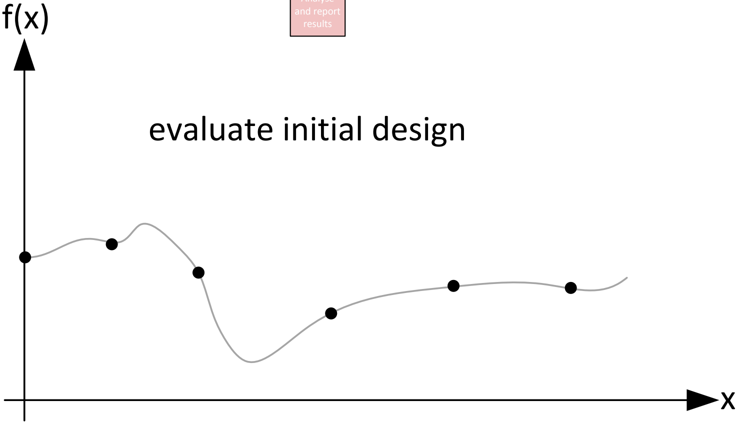


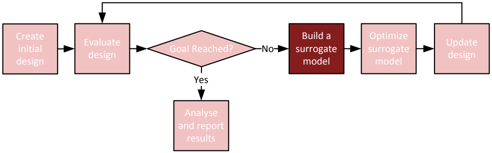

A
B
C
No
Yes
D
E
F
G
f(x)
unknown target function
x

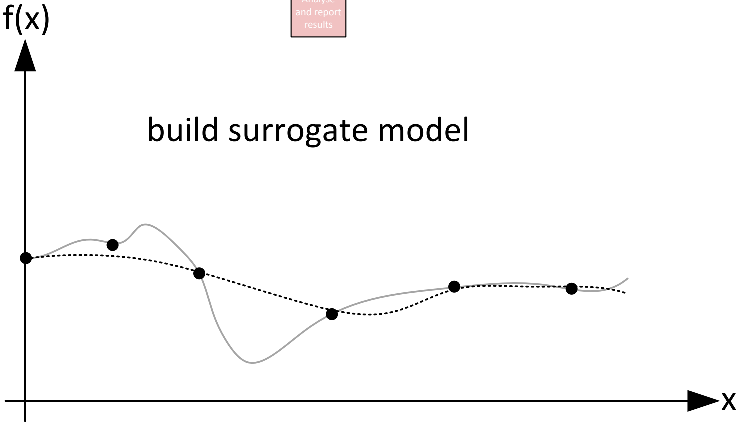

Create initial design
Evaluate design
Goal Reached?
No
Yes
Build a surrogate model
Optimize surrogate model
Update design
Analyse and report results

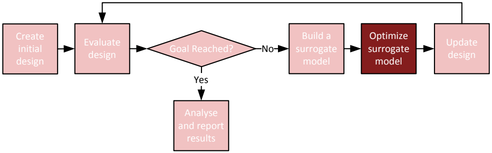

f(x)
initial design
x

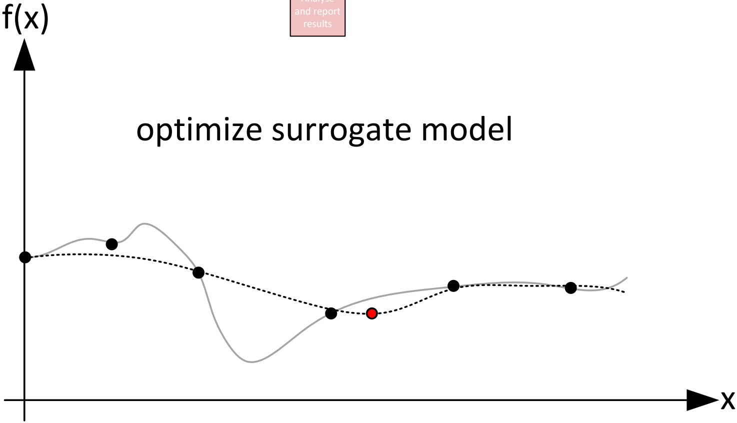

Create initial design
Evaluate design
Goal Reached?
No
Yes
Build a surrogate model
Optimize surrogate model
Update design
Analyse and report results

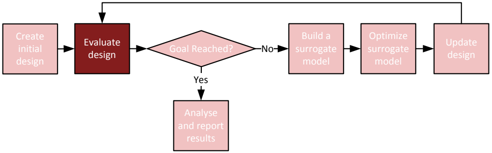

f(x)
evaluate initial design
x

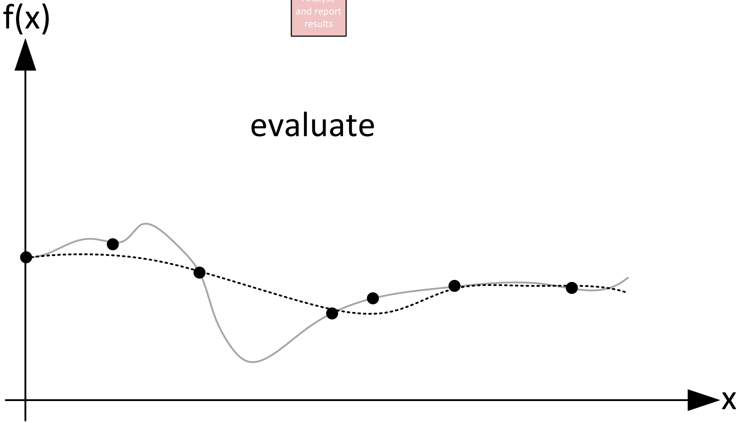

Create initial design
Evaluate design
Goal Reached?
No
Build a surrogate model
Optimize surrogate model
Update design
Yes
Analyse and report results

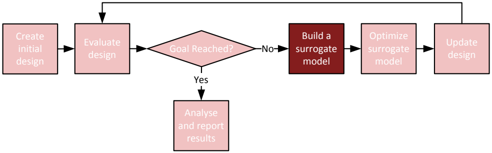

f(x)
build surrogate model
x

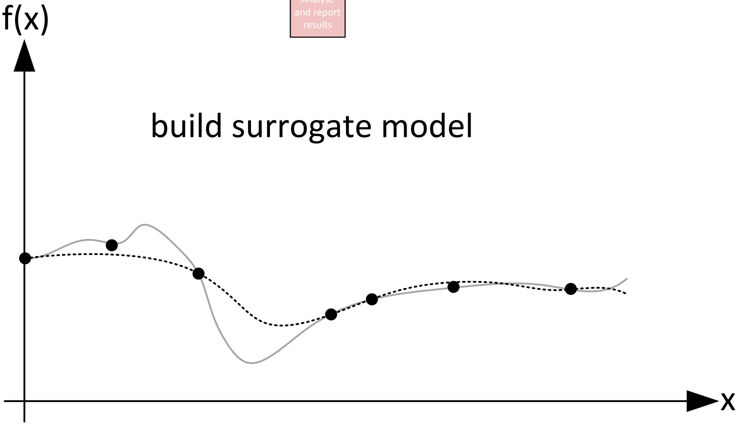

Create initial design
Evaluate design
Goal Reached?
No
Build a surrogate model
Optimize surrogate model
Update design
Yes
Analyse and report results

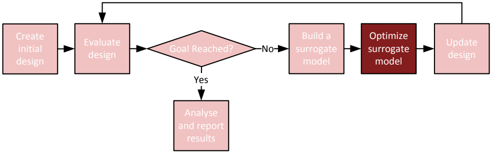

f(x)
optimize surrogate model
x

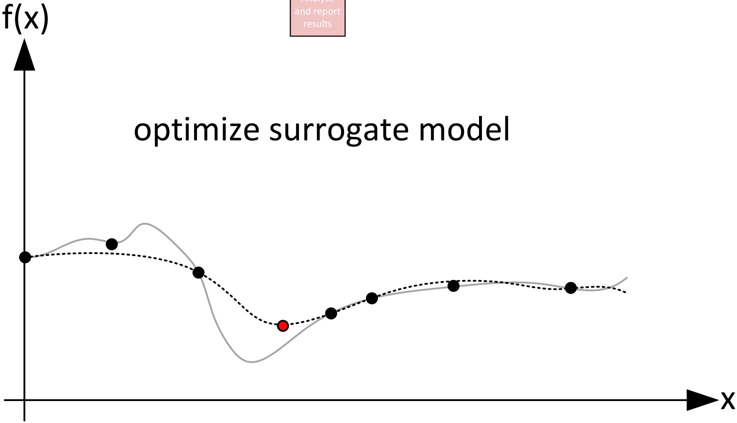

Create initial design
Evaluate design
Goal Reached?
No
Yes
Build a surrogate model
Optimize surrogate model
Update design
Analyse and report results


f(x)
evaluate
x

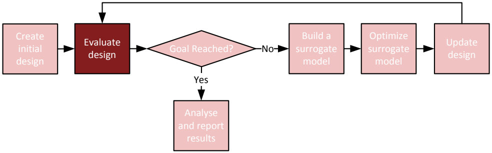

Create initial design
Evaluate design
Goal Reached?
No
Yes
Build a surrogate model
Optimize surrogate model
Update design
Analyse and report results

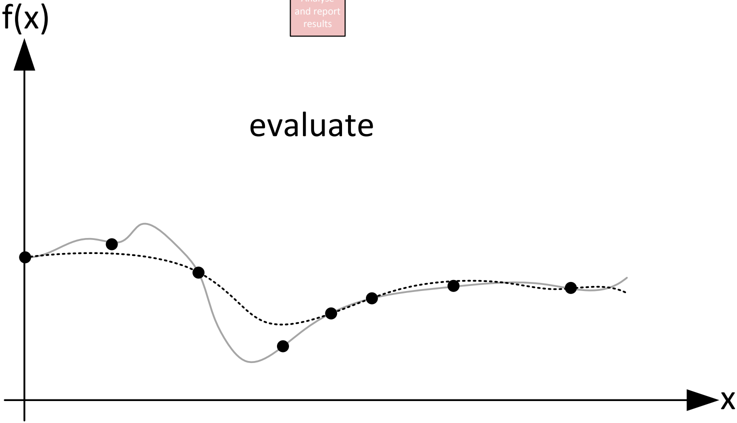

f(x)
build surrogate model
x

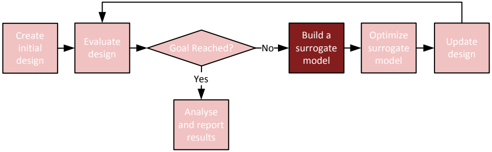




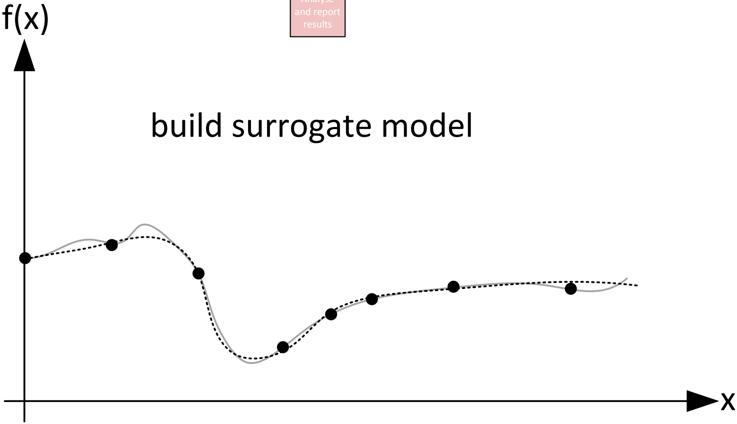
optimize surrogate model


Create initial design
Evaluate design
Goal Reached?
No
Yes
Build a surrogate model
Optimize surrogate model
Update design
Analyse and report results

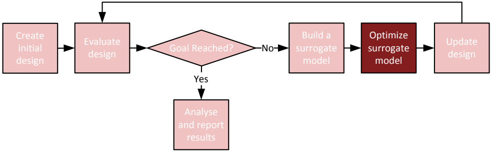

f(x)
evaluate
x



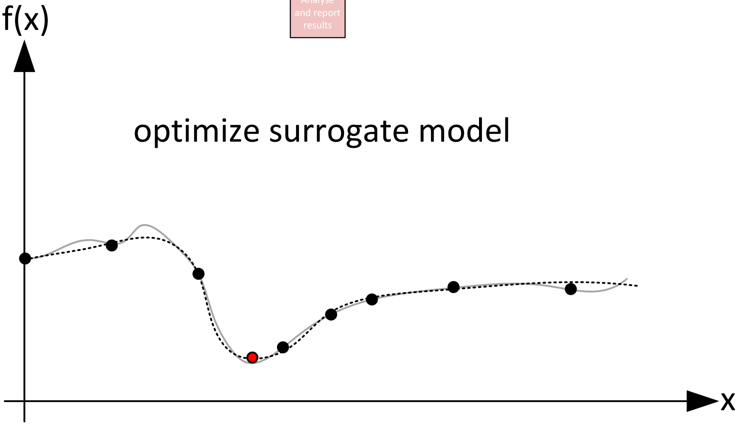
Create initial design
Evaluate design
Goal Reached?
No
Build a surrogate model
Optimize surrogate model
Update design
Yes
Analyse and report results



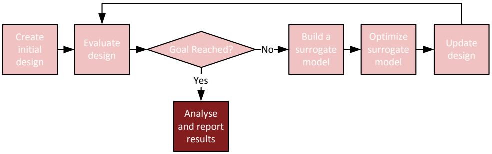
f(x)
build surrogate model
x



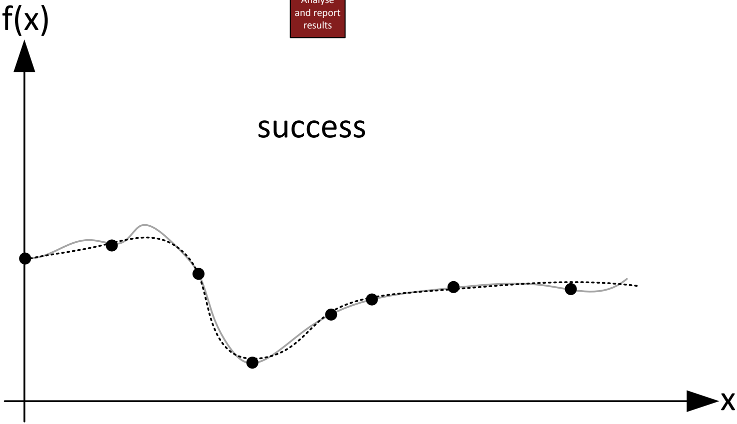
Create initial design
Evaluate design
Goal Reached?
No
Yes
Build a surrogate model
Optimize surrogate model
Update design
Analyse and report results



f(x)
optimize surrogate model
x


Create initial design
Evaluate design
Goal Reached?
No
Yes
Build a surrogate model
Optimize surrogate model
Update design
Analyse and report results

success


f(x)
x

Aims of the revised SPOT package

- High prediction quality
- Stable numerics
- Fast
- Modular structure for good extensibility
- Standardized objects and user interfaces
- Easy comprehensible code
- Good usability

What is new?

- No text files for configuration and data exchange anymore
- Everything implemented in `R`
- Object-oriented data structures as input and output for the individual functions
- Consistent with core `R` functionality
- Standardized and modular structure of the functions form a harmonized and easy understandable user interface
- Kriging with categorical inputs
- Stacking of different models for better prediction performance Bartz-Beielstein and Zaefferer (2017)

Create initial design

```
designLHD(x = NULL, lower, upper, control = list())
```

- Arguments
  - x: optional matrix of fixed user defined design points
  - lower/upper: vectors with boundaries for the design variables
  - control: list with the following controls:
  - size: number of design points
  - retries: number of retries during design creation
  - types: vector with the data type for each design parameter
  - replicates: integer for replications of each design point
- Returns matrix with design points (rows) for each variable (columns)

Model building

Different models can be chosen

- Linear models
- Kriging / Gaussian process regression
- Random Forest
- ...

```
buildKriging(x, y, control = list())
```

- Arguments

  x: design matrix (sample locations)

  y: vector of observations at `x`

  control: list with the options for the model building procedure

- Returns an object of class `kriging`, basically a list, with the options and found parameters for the model which has to be passed to the predictor function

Optimization

```
optimLBFGSB(x = NULL, fun, lower, upper, control = list(), ...)
```

- Wrapper function for `optim` with `method = "L-BFGS-B"`
- Arguments

  x: optional matrix of data-points, only first row used as start-point

  fun: objective function, which receives a matrix `x` and returns observations `y`

  lower/upper: boundary of the search space

  control: list of control parameters, passed to `optim`

  funEvals: number of function evaluations allowed

  ...: passed to `fun`

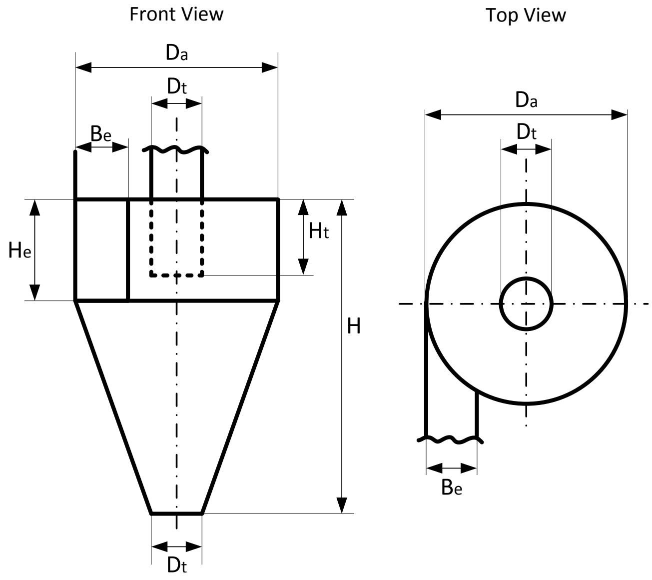
- Returns list with best solution (`xbest, ybest`), number of function evaluations (`count`) and messages from the optimizer

Why SPOT instead of package . . .

A lot of packages provide methods for model based optimization, Kriging, etc.
For example `mlrMBO`, `diceKriging`, `diceOptim`, `mleGP`, ...

- easy usage
- own Kriging implementation for stable numerics (based on Matlab code from Forrester et al. (2008))
- fast
- good and easy extensibility
- well proven methods for good results in real world problems

Cyclone optimization



Cyclone optimization

```
funCyclone(c(1260,2500)) #[1] 1626.194527   -0.886269
## create vectorized target funcion for the first objective only
tfunvecF1 <-function(x){apply(x,1,funCyclone)[2,]}
fixed <- matrix(c(1260,2500,1000,2000),2,2,byrow=TRUE)
lower <- c(1000,2000)
upper <- c(2000,3000)
## optimize with spot
res <- spot(x = designLHD(x = fixed, lower = lower, upper = upper, co
            fun = tfunvecF1,
            lower = lower,
            upper = upper,
            control = list(modelControl = list(target="ei"),
            model = buildKriging,
            optimizer = optimLBFGSB,
            plots=TRUE))
## best found solution ...
res$xbest #[1,] 2000 2861.775
## ... and its objective function value
res$ybest #[1,] -0.95085
```

Cyclone optimization

A more complex cyclone optimization, building a stacking ensemble of models from lab experiments, CFD simulations and analytical models can be found in Bartz-Beielstein et al. (2016).

The necessary datasets and the source code for this optimization is available here: `http://www.gm.fh-koeln.de/~bartz/Bart16e.d/`

Stacking example

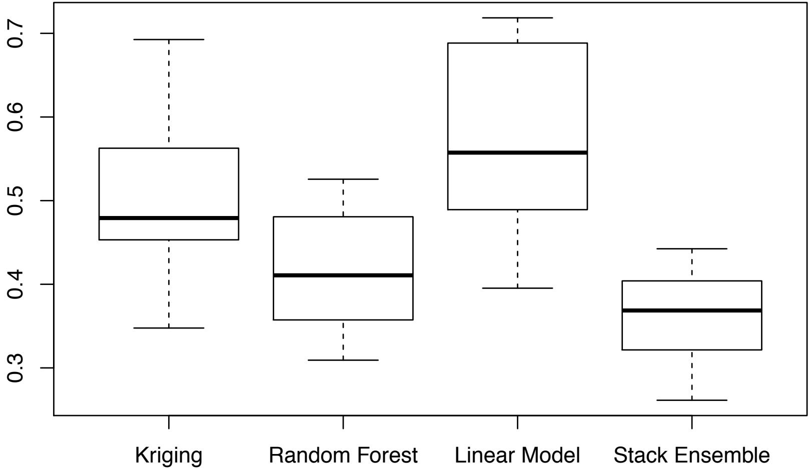
```
require(SPOT); require(CEGO)

train <- dataGasSensor[dataGasSensor[,11]==1,1:10]
test <- dataGasSensor[dataGasSensor[,11]==2,1:10]

  #define an optimizer:
optimizer <- function(x,fun,lower,upper,control,...){
  CEGO::optimInterface(x, fun, lower, upper,
    control=list(method="NLOPT_GN_DIRECT_L", funEvals=10,
                 reltol=1e-6, restarts=2), ...)
}

fitStack <- buildEnsembleStack(
  data.matrix(train[,c("Y","X7","Sensor","Batch")]),
  data.matrix(train$X1),
  control=list(modelL0Control=list(list(), list(),
               list(algTheta=optimizer,reinterpolate=FALSE)
    )
  )
)
predtest <- predict(fitStack,
  data.matrix(test[,c("Y","X7","Sensor","Batch")]))$y
mse <- mean(abs(predtest - data.matrix(test$X1))^2) # [1] 0.2627715
```

Stacking example



Summary and Outlook

- `SPOT 2` provides a good base for real world optimization problems
- Interfaces and object structures are stable and allow easy extensions
- Reporting functions are still missing (current work in progress)

Summary and Outlook

- `SPOT 2` provides a good base for real world optimization problems
- Interfaces and object structures are stable and allow easy extensions
- Reporting functions are still missing (current work in progress)

Thank you for your attention!

References

Bartz-Beielstein, T. (2010). Sequential parameter optimization—an annotated bibliography. CIOP Technical Report 04/10, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Cologne University of Applied Science, Faculty of Computer Science and Engineering Science.

Bartz-Beielstein, T., Lasarczyk, C., and Preuß, M. (2005). Sequential parameter optimization. In McKay, B. et al., editors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1, pages 773–780, Piscataway NJ. IEEE Press.

Bartz-Beielstein, T., Stenzel, H., Zaefferer, M., Breiderhoff, B., Pham, Q. C., Gusew, D., Mengi, A., Kabacali, B., Tünte, J., Büscher, L., Wüstlich, S., and Friesen, T. (2016). Optimization of the cyclone separator geometry via multimodel simulation.

Bartz-Beielstein, T. and Zaefferer, M. (2017). Model-based methods for continuous and discrete global optimization. *Applied Soft Computing*, 55:154 – 167.

Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering Design via Surrogate Modelling*. Wiley.