Beyond Particular Problem Instances: How to Create Meaningful and Generalizable Results

Thomas Bartz-Beielstein

Cologne University of Applied Sciences

spotseven.org

First Workshop on Applied Meta-Modeling

November 16, 2012

Questions

Q-1: How to generate test problems?

Q-2: How to generalize results?

Agenda

Motivation
- Problem Classes and Instances
- SASP
- MASP and SAMP

How to Generate Problem Instances
- Natural Problem Classes

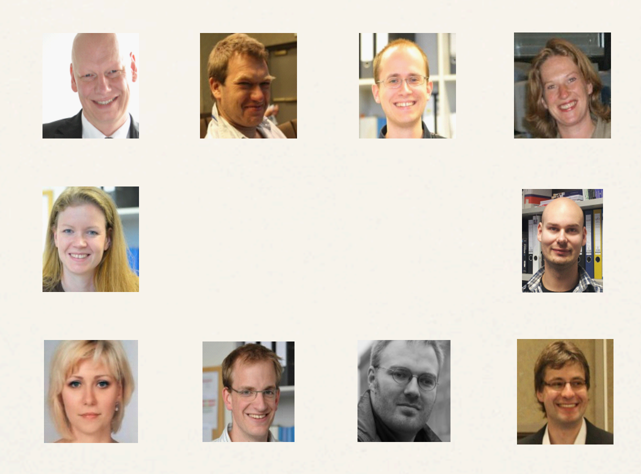
Algorithm

Case Study: SAMP

Summary

Outlook
- MAMP

The SPOTSeven Team



- www.spotseven.org

Benchmarking: General Rules

- Validity
- Reproducibility
- Comparability
- Commons rules:
  - Use statistics
  - Documentation
  - Comparisons
- On-going discussion

Computational Intelligence: State-of-the-Art Methoden und Benchmarkprobleme

Frank Hoffmann[1], **Ralf Mikut**[2], **Andreas Kroll**[3], **Markus Reischl**[2], **Oliver Nelles**[4], **Horst Schulte**[5], **Torsten Bertram**[1]

[1]Technische Universität Dortmund, Lehrstuhl für Regelungssystemtechnik
E-Mail: {frank.hoffmann}{torsten.bertram}@tu-dortmund.de
[2]Karlsruher Institut für Technologie, Institut für Angewandte Informatik
E-Mail: {ralf.mikut}{markus.reischl}@kit.edu
[3]Universität Kassel, FB Maschinenbau, FG Mess- und Regelungstechnik
E-Mail: andreas.kroll@mrt.uni-kassel.de
[4]Universität Siegen, Mess- und Regelungstechnik - Mechatronik Department Maschinenbau
E-Mail: oliver.nelles@uni-siegen.de
[5] HTW Berlin, FB Ingenieurwissenschaften I, FG Regelungstechnik und Systemdynamik
E-Mail: schulte@htw-berlin.de

Zusammenfassung: Dieser Beitrag gibt einen Überblick über den Stand der Technik in der Computational Intelligence für Methoden zur Klassifikation, zum Text Mining, zur nichtlinearen Regression, nichtlinearen Systemidentifikation und Regelung. Im Fokus steht eine systematische, wissenschaftlichen Ansprüchen genügende Vorgehensweise bei der vergleichenden Bewertung und Analyse alternativer Ansätze. Die einzelnen Abschnitte geben praktikable Hinweise auf vorhandene, möglichst frei verfügbare Implementierungen, Benchmarkdatensätze und -probleme als Hilfestellung für den Methodenvergleich zukünftiger Publikationen innerhalb des CI-Workshops.

1 Einführung

Die Methodik und Vorgehensweise bei der Bewertung, dem Vergleich und systematischen Analyse neuartiger Methoden der Mustererkennung und Funktionsapproximation hat auf vergangenen Computational Intelligence Workshops zu Kritik und Diskussionen geführt. In einigen Beiträgen fehlte

Benchmarking: Features

- Difficult to solve using simple methods such as hill climbers
- Nonlinear, non separable, non symmetric
- Scalable with respect to
  - problem dimensionality
  - evaluation time
- Tunable by a small number of user parameters

See,e.g, [4]

Benchmarking: Current Situation

- Authors report parameter values which seem to work reasonably well
- Each algorithm will be run for some number, say ten, on each problem. Statistics are reported, e.g., mean, standard deviation
- One expert compares his new algorithm with establishes approaches. Subjective (unfair?) comparison
- Many experts compare their algorithms on several, standardized data. Objective (fair) comparison
- Use accepted data bases, e.g., UCI
- Divide data into train, validation, and test data
- What is the problem of this approach?

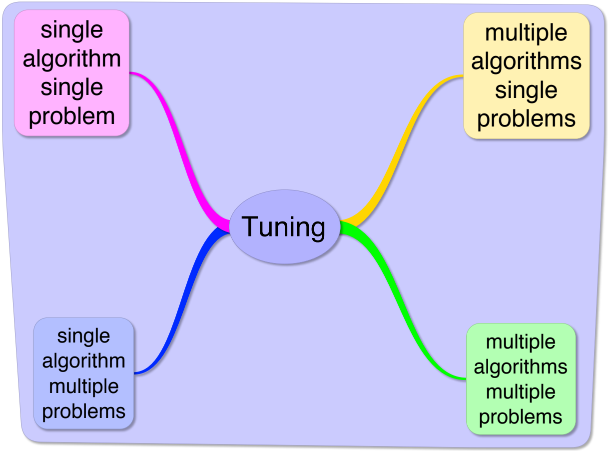
Benchmarking: Open Questions

- Algorithms are trained for this specific set of benchmark functions
  - Who defines this set of functions?
  - Fixed set of test data?
- In practice, I do not need an algorithm which performs good on a set of test problems (which was developed by some experts)
- Really wanted:
  - An algorithm, which performs very good on my set of real-word test problems
  - Not only demonstrating
  - Understanding!
- Let's have a short look at the problem

A Taxonomy of Algorithm and Problem Designs

- Classify parameters
- Parameters may be *qualitative*, like for the presence or not of an recombination operator or *numerical*, like for parameters that assume real values
- Our interest: understanding the contribution of these components
- Statistically speaking: parameters are called *factors*
- The interest is in the effects of the specific *levels* chosen for these factors

Problems and Algorithms



- How to perform comparisons?
- Adequate statistics and models?

SASP: Algorithm and Problem Designs

- Basic design: assess the performance of an *optimization algorithm* on a single problem instance π
- Randomized optimization algorithms $\Rightarrow$ performance Y on one instance is a random variable
- Experiment: On an instance π algorithm is run r times $\Rightarrow$ collect sample data $Y_1, \ldots, Y_r$ (independent, identically distributed)
- One instance π, run the algorithm r times $\Rightarrow$ r replicates of the performance measure Y, denoted by $Y_1, \ldots, Y_r$
- Samples are conditionally on the sampled instance and given the random nature of the algorithm, independent and identically distributed (i.i.d.), i.e.,

$$p(y_1, \ldots, y_r | \pi) = \prod_{j=1}^{r} p(y_j | \pi). \tag{1}$$

MASP and SAMP: Algorithm and Problem Designs

- MASP
  - Several optimization algorithms are compared on one fixed problem instance π
  - Experiment: collect sample data $Y_1, \ldots, Y_R$ (independent, identically distributed)
  - Goal: comparison of algorithms on one (real-world) problem instance π
  - No generalization
- SAMP
  - Generalization!
  - Goal: Drawing conclusions about a certain *class* or *population* of instances Π
  - This is Q-1: How to generate a population of problem instances?

Test Problem Generators

- Artificial
- Natural

- Three fundamental steps for generating natural problem instances, namely
    - Describing the real-world system and its data
    - Feature extraction
    - Instance generation

Example: Test Problem Generators

- Describing the real-world system and its data
- Classic Box and Jenkins airline data [2]
- Monthly totals of international airline passengers, 1949 to 1960
- `> str(AirPassengers)`

```
Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119
```

Example: Test Problem Generators

- Feature extraction based on methods from time-series analysis
- Multiplicative Holt-Winters (HW) prediction function (for time series with period length p) is

$$\hat{Y}_{t+h} = (a_t + hb_t)s_{t-p+1+(h-1) \mod p},$$

where a_t, b_t and s_t are given by

$$\begin{aligned} a_t &= \alpha(Y_t/s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \\ b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(Y_t/a_t) + (1-\gamma)s_{t-p} \end{aligned}$$

- The optimal values of α, β and γ are determined by minimizing the squared one-step prediction error

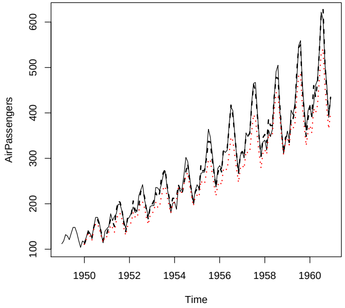
Example: Test Problem Generators

- Instance generation
- HW parameters α, β, and γ are estimated from original time-series data Y_t
- To generate new problem instances, these parameters can be slightly modified
- Based on these modified values, the model is re-fitted
- Extract the new time series. Here, we plot the original data, the Holt-Winters predictions and the modified time series.

Example: Test Problem Generators

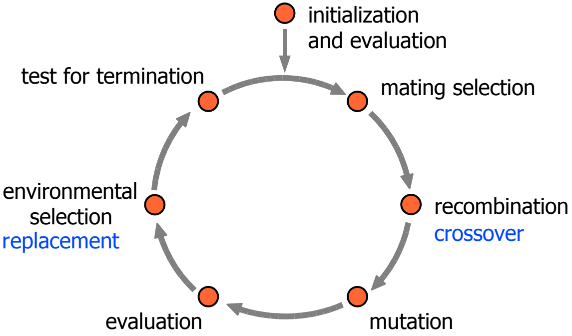
```
> generateHW <- function(a,b,c){
+ ## Estimation
+   m <- HoltWinters(AirPassengers, seasonal = "mult")
+ ## Extraction
+   alpha0<-m$alpha
+   beta0<-m$beta
+   gamma0<-m$gamma
+ ## Modification
+   alpha1 <- alpha0*a
+   beta1 <- beta0*b
+   gamma1 <- gamma0*c
+ ## Re-estimation
+   m1 <- HoltWinters(AirPassengers, alpha=alpha1
+   , beta = beta1, gamma = gamma1)
+ ## Instance generation
+   plot(AirPassengers)
+   lines(fitted(m)[,1], col = 1, lty=2, lw=2)
+   lines(fitted(m1)[,1], lty = 3, lw =2, col = 2)
+ }
> generateHW(a=.05,b=.025,c=.5)
```

Example: Test Problem Generators



- HW problem instance generator: *solid line:* real data, *dotted line:* predictions from the Holt-Winters model, *fine dotted red line:* modified predictions

Evolution Strategy



Evolution Strategy

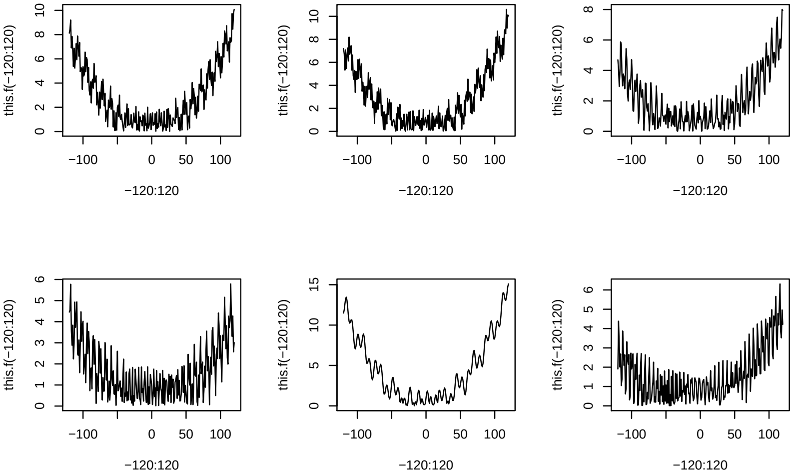
Parameter	Symbol	Name	Range	Value
mue	μ	Number of parent individuals	$\mathbb{N}$	5
nu	$\nu = \lambda/\mu$	Offspring-parent ratio	$\mathtt{R}_+$	2
sigmaInit	$\sigma_i^{(0)}$	Initial standard deviations	$\mathtt{R}_+$	1
nSigma	n_σ	Number of standard deviations. d denotes the problem dimension	$\{1, d\}$	1
	c_τ	Multiplier for mutation	$\mathtt{R}_+$	1
tau0			$\mathtt{R}_+$	0
tau			$\mathtt{R}_+$	1
rho	ρ	Mixing number	$\{1, \mu\}$	2
sel	κ	Maximum age	$\mathtt{R}_+$	1
sreco	r_σ	Recombination: strategy vars	$\{1, 2, 3, 4\}$	3
oreco	r_x	Recombination: object vars	$\{1, 2, 3, 4\}$	2
mutation		Mutation	$\{1, 2\}$	2

SAMP: Fixed Algorithm and Randomized Problem Designs

- SAMP-1: Algorithm and Problem Instances
- SAMP-2: Validation of the Model Assumptions
- SAMP-3: Building the Model and ANOVA
- SAMP-4: Hypothesis Testing
- SAMP-5: Confidence Intervals and Prediction

SAMP-1: Problem Instances

- Nine problem instances, which were randomly drawn from an infinite number of instances: `fSeed`



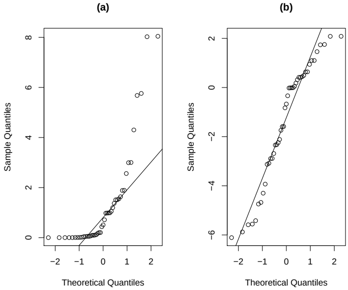
SAMP-1: Algorithm and Problem Instances

- ES, run $r = 5$ times on a set of randomly generated problem instances

```
'data.frame': 45 obs. of  5 variables:
 $ y      : num  0.2036 0.0557 0.0979 0.7142 4.3018 ...
 $ mut    : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
 $ fSeed  : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 2 2 2 2 2 ...
 $ algSeed: Factor w/ 5 levels "1","2","3","4",..: 1 2 3 4 5 1 2 3 4 5 ...
 $ yLog   : num  -1.592 -2.887 -2.324 -0.337 1.459 ...
```

SAMP-2 Validation of the Model Assumptions

- Quantile plots (QQ plots) to validate normality assumptions



SAMP-3 Building the Model and ANOVA

- Linear statistical model

$$Y_{ij} = \mu + \tau_i + \varepsilon_{ij} \begin{cases} i = 1, \ldots, q \\ j = 1, \ldots, r, \end{cases} \tag{2}$$

  where μ is an overall mean and ε_{ij} is a random error term for replication j on instance i
- Note, in contrast to the fixed-effects model, τ_i is a random variable representing the effect of instance i
- The stochastic behavior of the response variable originates from both the instance and the algorithm
- This is reflected in (2), where both τ_i and ϵ_{ij} are random variables
- The model (2) is the so-called *random-effects model*, cf. [6, p. 512] or [3, p. 229].

SAMP-3: The classical ANOVA

- Similar to classical ANOVA: variability in the observations can be partitioned into a component that measures the variation between treatments and a component that measures the variation within treatments
- Based on ANOVA identity $\mathrm{SS}_{\mathrm{total}} = \mathrm{SS}_{\mathrm{treat}} + \mathrm{SS}_{\mathrm{err}}$, we define

$$\mathrm{MS}_{\mathrm{treat}} = \frac{\mathrm{SS}_{\mathrm{treat}}}{q-1} = \frac{r\sum_{i=1}^{q}(\bar{Y}_{i.} - \bar{Y}_{..})^2}{q-1},$$

$$\mathrm{MS}_{\mathrm{err}} = \frac{\mathrm{SS}_{\mathrm{err}}}{q(r-1)} = \frac{\sum_{i=1}^{q}\sum_{j=1}^{r}(Y_{ij} - \bar{Y}_{i.})^2}{q(r-1)}$$

- It can be shown [6] that

$$E(\mathrm{MS}_{\mathrm{treat}}) = \sigma^2 + r\sigma_\tau^2 \quad \text{and} \quad E(\mathrm{MS}_{\mathrm{err}}) = \sigma^2, \tag{3}$$

- Estimators of variance components

$$\hat{\sigma}^2 = \mathrm{MS}_{\mathrm{err}} \qquad \text{and} \qquad \hat{\sigma_\tau^2} = \frac{\mathrm{MS}_{\mathrm{treat}} - \mathrm{MS}_{\mathrm{err}}}{r} \tag{4}$$

SAMP-3: The classical ANOVA

Table : ANOVA table for a one-factor fixed and random effects models

Source of Variation	Sum of Squares	Degrees of freedom	Mean Square	EMS Fixed	EMS Random
Treatment	SS_{treat}	$q-1$	MS_{treat}	$\sigma^2 + r\frac{\sum_{i=1}^q \tau_i^2}{q-1}$	$\sigma^2 + r\sigma_\tau^2$
Error	SS_{err}	$q(r-1)$	MS_{err}	σ^2	σ^2
Total	SS_{total}	$qr-1$			

- Expected mean squares differ

SAMP-3: ANOVA Calculations in R (1/2)

- Extract mean squared values MSA (treatment) and MSE (error) from ANOVA model
- Calculate estimators of variance components from (4): $\hat{\sigma}^2$ as the mean squared error and the second component $\hat{\sigma}_\tau^2$

```
> samp.aov <- aov(yLog ~fSeed, data=samp.df)
> (M1 <- anova(samp.aov))

Analysis of Variance Table

Response: yLog
          Df  Sum Sq Mean Sq F value Pr(>F)
fSeed      8  48.832  6.1040  1.0707 0.4048
Residuals 36 205.230  5.7008

> (MSA <- M1[1,3])

[1] 6.10401

> (MSE <- M1[2,3])

[1] 5.700838

> r <-length(unique(samp.df$algSeed)); q <- nlevels(samp.df$fSeed)
> (var.A <- (MSA - MSE)/(r))

[1] 0.0806345

> (var.E <- MSE)

[1] 5.700838
```

SAMP-3: ANOVA Calculations in R (2/2)

- Finally, the mean μ from (2) can extracted

```
> coef(samp.aov)[1]
```

```
(Intercept)
  -1.136131
```

- The p value in the ANOVA table is calculated as

```
> 1-pf(MSA/MSE,q-1,q*(r-1))
```

```
[1] 0.4047883
```

- Store ANOVA MSA for later:

```
> MSA.anova <- MSA
```

SAMP-3: ANOVA Problems?

- In some cases, the standard ANOVA, which was used in our example, produces a negative estimate of a variance component
- This can be seen in (4): If $\mathrm{MS}_{\mathrm{err}} > \mathrm{MS}_{\mathrm{treat}}$, negative values occur
- By definition, variance components are positive
- Methods, which always yield positive variance components have been developed: restricted maximum likelihood estimators (REML)
- The ANOVA method of variance component estimation, which is a method of moments procedure, and REML estimation may lead to different results

SAMP-3: Restricted Maximum Likelihood

- Based on same data: fit the random-effects model (2) using function `Rlmer` from `R` package `Rlmefour` [1]:

```
> library(lme4)
> samp.lmer <- lmer(yLog~ 1 +(1|fSeed),data=samp.df)
> print(samp.lmer, digits = 4, corr = FALSE)
```

```
Linear mixed model fit by REML
Formula: yLog ~ 1 + (1 | fSeed)
   Data: samp.df
   AIC   BIC logLik deviance REMLdev
 211.8 217.2 -102.9    205.6   205.8
Random effects:
 Groups   Name        Variance   Std.Dev.
 fSeed    (Intercept) 2.6192e-11 5.1179e-06
 Residual             5.7741e+00 2.4029e+00
Number of obs: 45, groups: fSeed, 9

Fixed effects:
            Estimate Std. Error t value
(Intercept)  -1.3528     0.3582  -3.776
```

SAMP-4 Hypothesis Testing

- Testing hypotheses about individual treatments (instances) is useless, because problem instances π_i samples from some larger population of instances Π
- We test hypotheses about the variance component σ_τ^2, i.e., the null hypothesis

$$H_0 : \sigma_\tau^2 = 0 \qquad \text{is tested versus the alternative} \qquad H_1 : \sigma_\tau^2 > 0. \tag{5}$$

- Under H_0, all treatments are identical, i.e., $r\sigma_\tau^2$ is very small
- Conclude from (3): $E(\mathrm{MS_{treat}}) = \sigma^2 + r\sigma_\tau^2$ and $E(\mathrm{MS_{err}}) = \sigma^2$ are similar
- Under the alternative, variability exists between treatments.
- Standard analysis shows: $\mathrm{SS_{err}}/\sigma^2$ is distributed as chi-square with $q(r-1)$ degrees of freedom. Under H_0, the ratio

$$F_0 = \frac{\frac{\mathrm{SS_{treat}}}{q-1}}{\frac{\mathrm{SS_{err}}}{q(r-1)}} = \frac{\mathrm{MS_{treat}}}{\mathrm{MS_{err}}} \sim F_{q-1,q(r-1)}$$

- Requirements for testing hypotheses in (2): $\tau_1, \ldots, \tau_q$ are i.i.d. $\mathcal{N}(0, \sigma_\tau^2)$, ε_{ij}, $i = 1, \ldots, q$, $j = 1, \ldots, r$, are i.i.d. $\mathcal{N}(0, \sigma^2)$, and all τ_i and ε_{ij} are independent of each other

SAMP-4 Hypothesis Testing and Decision Rules

- Considerations lead decision rule to reject H_0 at the significance level α if

$$f_0 > F(1-\alpha; q-1, q(r-1)), \tag{6}$$

  where f_0 is the realization of F_0 from the observed data
- Intuitive motivation for the form of statistic F_0 can be obtained from the expected mean squares:
  - Under H_0 both $\mathrm{MS}_{\mathrm{treat}}$ and $\mathrm{MS}_{\mathrm{err}}$ estimate σ^2 in an unbiased way, and F_0 can be expected to be close to one
  - On the other hand, large values of F_0 give evidence against H_0

SAMP-4 Hypothesis Testing and Decision Rules in R

- Based on (3), we can determine the F statistic and the p values:

```
> VC <- VarCorr(samp.lmer)
> (sigma.tau <- as.numeric(attr(VC$fSeed,"stddev")))

[1] 5.117856e-06

> (sigma <- as.numeric(attr(VC,"sc")))

[1] 2.402944

> q <- nlevels(samp.df$fSeed); r <- length(unique(samp.df$algSeed))
> (MSA <- sigma^2+r*sigma.tau^2)

[1] 5.774142

> (MSE <- sigma^2)

[1] 5.774142
```

  Determine p value based on (6):

```
> 1-pf(MSA/MSE,q-1,q*(r-1))

[1] 0.4529257
```

- Since p value is large, the null hypothesis $H_0 : \sigma_\tau^2 = 0$ from (5) can not be rejected, i.e., this indicates that there is no instance effect
- A similar conclusion was obtained from the ANOVA method of variance component estimation

SAMP-5 Confidence Intervals and Prediction

- Unbiased estimator of the overall mean μ is

$$\sum_{i=1}^{q} \sum_{j=1}^{r} \frac{y_{ij}}{qr}$$

- Its estimated standard error is given by $\mathrm{se}(\hat{\mu}) = \sqrt{\mathrm{MStreat}/qr}$ and

$$\frac{\bar{Y}_{..} - \mu}{\sqrt{\mathrm{MStreat}/qr}} \sim t(q-1)$$

- Hence, [3, p. 232] show that confidence limits for μ can be derived as

$$\bar{y}_{..} \pm t(1-\alpha/2; q-1)\sqrt{\mathrm{MStreat}/qr} \tag{7}$$

SAMP-5 Confidence Intervals and Prediction in R (MLE)

- Prediction of the algorithm's performance on a new instance
- Based on (7), the 95% confidence interval can be calculated as follows.

```
> s <- sqrt(MSA/(q*r))
> Y.. <- mean(samp.df$yLog)
> qsr <- qt(1-0.025,r)
> c( exp(Y.. - qsr * s), exp(Y.. + qsr * s))
```

```
[1] 0.1029441 0.6492394
```

- Since we performed the analysis on log data, the exp() function was applied to the final result.
- Hence, 95% confidence interval for μ is $[0.10; 0.65]$.

SAMP-5 Confidence Intervals and Prediction in R (ANOVA)

- Using the ANOVA results from above, we obtain the following confidence interval for the performance of the ES:

```
> s <- sqrt(MSA.anova/(q*r))
> Y.. <- mean(samp.df$yLog)
> qsr <- qt(1-0.025,5)
> c( exp(Y.. - qsr * s), exp(Y.. + qsr * s))
```

```
[1] 0.1003084 0.6662989
```

Summary

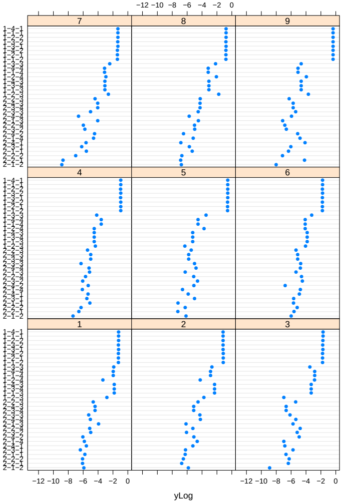
Q-1: How to generate test problems?

- Randomization!

Q-2: How to generalize results?

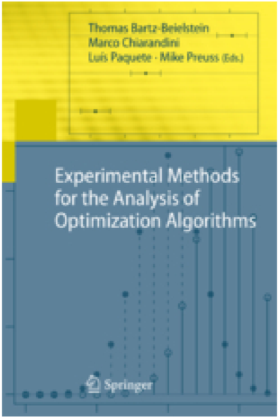
- Randomization!

Outlook



- MAMP

Suggested Reading



- http://www.spotseven.org

- Experimental Methods for the Analysis of Optimization Algorithms
- See also Kleijnen [5], Saltelli et al.

Acknowledgments

- This work has been supported by the Federal Ministry of Education and Research (BMBF) under the grants MCIOP (FKZ 17N0311) and CIMO (FKZ 17002X11)

[1] Douglas M. Bates.
lme4: Mixed-effects modeling with R.
2010.

[2] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel.
Time Series Analysis, Forecasting and Control.
Holden-Day, 1976.

[3] Marco Chiarandini and Yuri Goegebeur.
Mixed models for the analysis of optimization algorithms.
In Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 225–264. Springer, Germany, 2010.
Preliminary version available as *Tech. Rep.* DMF-2009-07-001 at the The Danish Mathematical Society.

[4] M. Gallagher and B. Yuan.
A general-purpose tunable landscape generator.
IEEE transactions on evolutionary computation, 10(5):590–603, 2006.

[5] J. P. C. Kleijnen.
Design and analysis of simulation experiments.

Springer, New York NY, 2008.

[6] D. C. Montgomery.
Design and Analysis of Experiments.
Wiley, New York NY, 5th edition, 2001.