Modeling of Vaccinations & Infectious Diseases

Calibration I – Optimization

1st GENID Summer School on Infectious Disease Modeling

Dr. Stefan Scholz

14.09.2022

Use of infectious disease models

- The spread of infectious diseases is usually very dynamic ⇒ Population effects of intervention are hard to capture with studies
- Modeling is an established tool for assessing the effectiveness of intervention (even for questions around reimbursement)
- Different understanding of *modeling*:
  - (Mechanistic) simulations vs. (statistic) models
  - Prognosis- vs. scenario models
- Infectious disease models are often a statistical models with structural (mechanistic) components to answer counter-factual (what-if) questions

Use of infectious disease models

- The spread of infectious diseases is usually very dynamic $\Rightarrow$ Population effects of intervention are hard to capture with studies
- Modeling is an established tool for assessing the effectiveness of intervention (even for questions around reimbursement)
- Different understanding of *modeling*:
  - (Mechanistic) simulations vs. (statistic) models
  - Prognosis- vs. scenario models
- Infectious disease models are often a statistical models with structural (mechanistic) components to answer counter-factual (what-if) questions

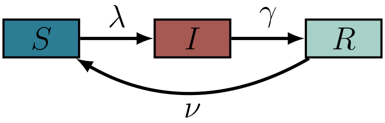
Use of infectious disease models

- The spread of infectious diseases is usually very dynamic $\Rightarrow$ Population effects of intervention are hard to capture with studies
- Modeling is an established tool for assessing the effectiveness of intervention (even for questions around reimbursement)
- Different understanding of *modeling*:
  - (Mechanistic) simulations vs. (statistic) models
  - Prognosis- vs. scenario models
- Infectious disease models are often a statistical models with structural (mechanistic) components to answer counter-factual (what-if) questions

Use of infectious disease models

- The spread of infectious diseases is usually very dynamic $\Rightarrow$ Population effects of intervention are hard to capture with studies
- Modeling is an established tool for assessing the effectiveness of intervention (even for questions around reimbursement)
- Different understanding of *modeling*:
  - (Mechanistic) simulations vs. (statistic) models
  - Prognosis- vs. scenario models
- Infectious disease models are often a statistical models with structural (mechanistic) components to answer counter-factual (what-if) questions

Mechanistic Model



- Model structure is given by a-priori knowledge about the system
- Goal: Estimating the effect of changes of input parameter values on the model outcome / overall system

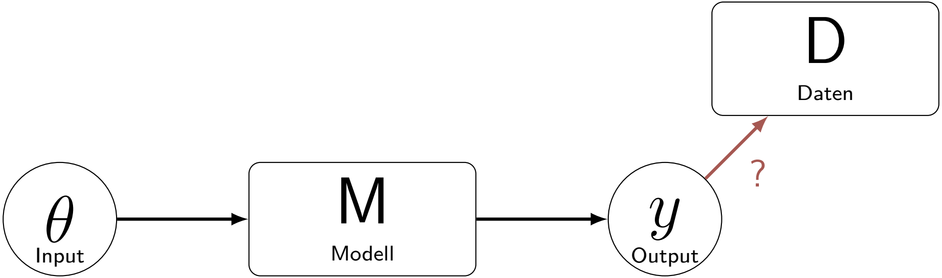
Statistical Model

$$y = \alpha + \beta X$$

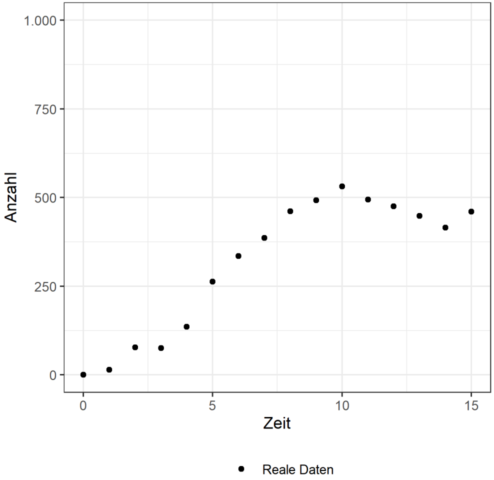
- Flexible and/or agnostic "model structure"
- Goal: Find associations between variables in data

- Due to incomplete evidence, purely mechanistic models (simulations) will not reproduce the observed dynamic of the transmission in the population
- The combination of mechanistic and statistic model allows,
  - finding missing parameter values
  - interpolation of missing data points (or prediction of future data points)
  - To test hypothesis about associations between input parameters of the model
  - extrapolate counter-factual (what-if) scenarios

How can we make a model reproduce observed data using calibration / model fitting?

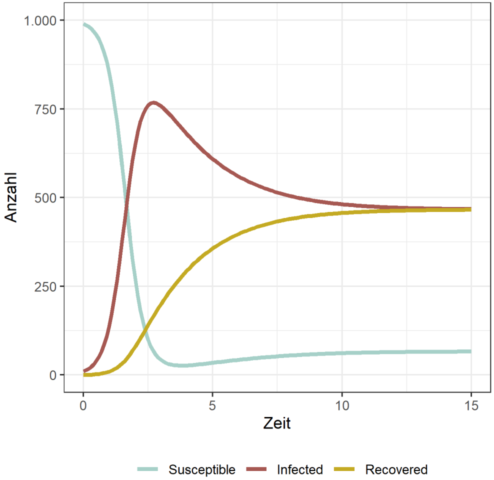


Procedure of model calibration



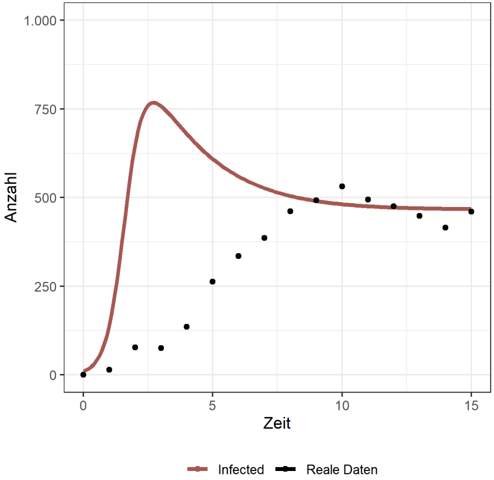
1. Define a **calibration target** (i.e., what data should the model reproduce)
2. Define **"free" parameters** and perform an initial model run (using *best guess* values from the literature)
3. Compare real, observed data with the model output via a **Goodness-of-Fit** (GoF) function
4. **Change** the values of the "free" parameters $\Rightarrow$ go to step 2

Procedure of model calibration



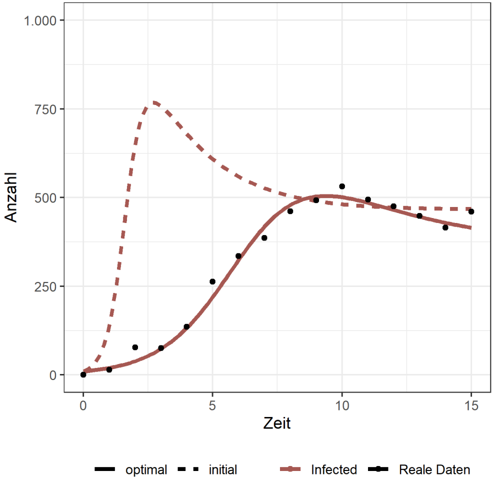
1. Define a **calibration target** (i.e., what data should the model reproduce)
2. Define **"free" parameters** and perform an initial model run (using *best guess* values from the literature)
3. Compare real, observed data with the model output via a **Goodness-of-Fit** (GoF) function
4. **Change** the values of the "free" parameters $\Rightarrow$ go to step 2

Procedure of model calibration



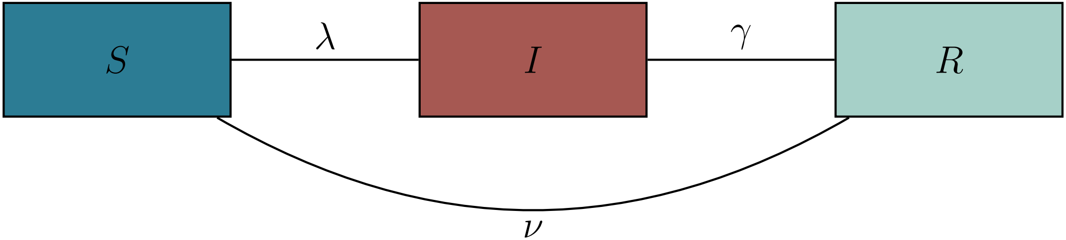
1. Define a **calibration target** (i.e., what data should the model reproduce)
2. Define **"free" parameters** and perform an initial model run (using *best guess* values from the literature)
3. Compare real, observed data with the model output via a **Goodness-of-Fit** (GoF) function
4. **Change** the values of the "free" parameters ⇒ go to step 2

Procedure of model calibration



1. Define a **calibration target** (i.e., what data should the model reproduce)
2. Define **"free" parameters** and perform an initial model run (using *best guess* values from the literature)
3. Compare real, observed data with the model output via a **Goodness-of-Fit** (GoF) function
4. **Change** the values of the "free" parameters $\Rightarrow$ go to step 2

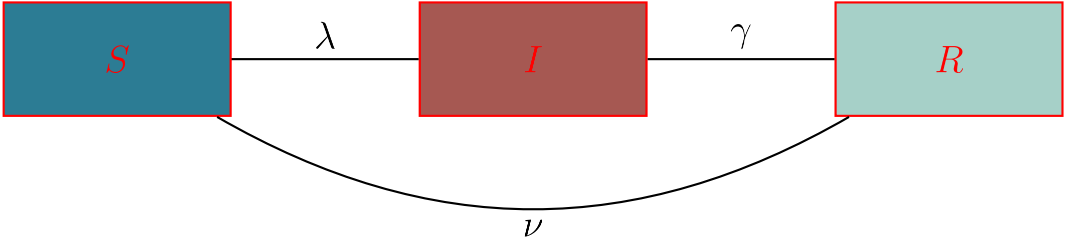
- Usually, infectious disease models are fitted to **prevalence** or **incidence** of a certain disease outcome
- Outcomes usually consist of
  - Infections
  - Symptomatic infections
  - Cases using the health care system (e.g. outpatient visit, hospitalizations)
  - Deaths
- **Prevalence** corresponds to the number of persons in a *State* at time t
- **Incidence** corresponds to the *Flow* into a state over time period $t_0 - -t_1$

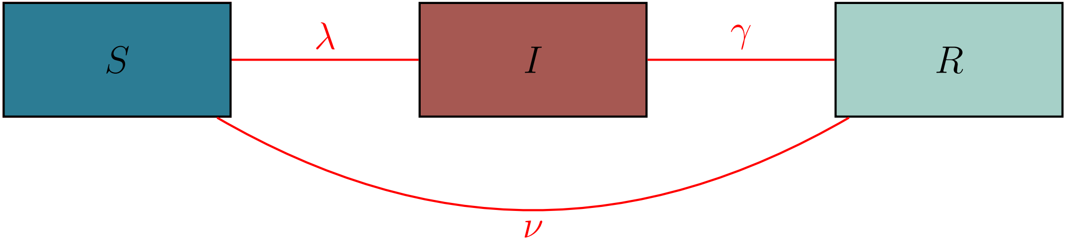


S
λ
I
γ
R
ν

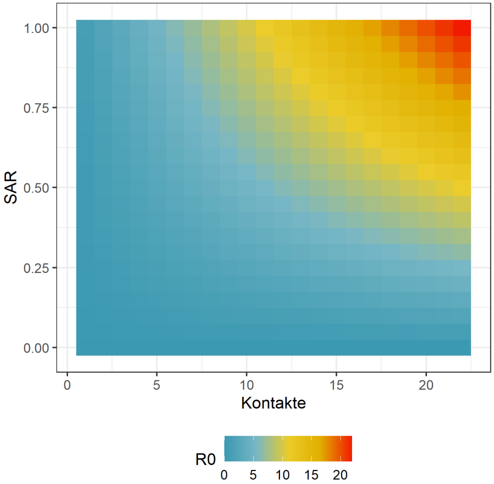
1. Kalibrierungsziele

Procedure of model calibration




S
λ
I
γ
R
ν

- Basically, one can set as many parameters as free parameters as you like (even all of them) (*states* as well as *flows*)
- Infectious disease model usually calibrate the *secondary attack rate* (SAR) or the reproduction number R_0 directly, as these are often unknown and have a great impact on the transmission dynamics
- When choosing free parameters, the problem of **identifiability** may arise

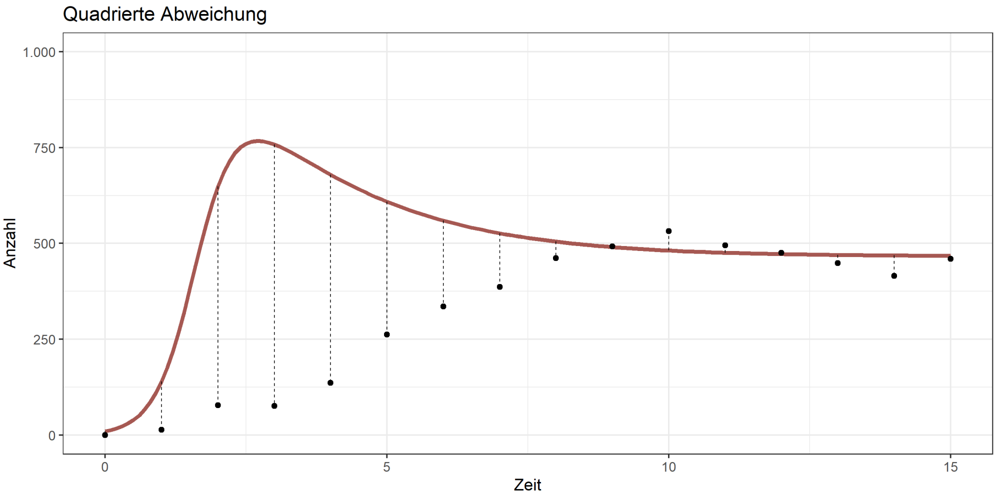


- No unique solutions can be found, if parameters are strongly correlated or if one free parameter can be represented as a linear-combination of other free parameters
- For example, a certain value of R_0 can either be achieved by a high number of contacts and a lower SAR or via a low number of contacts and a high SAR

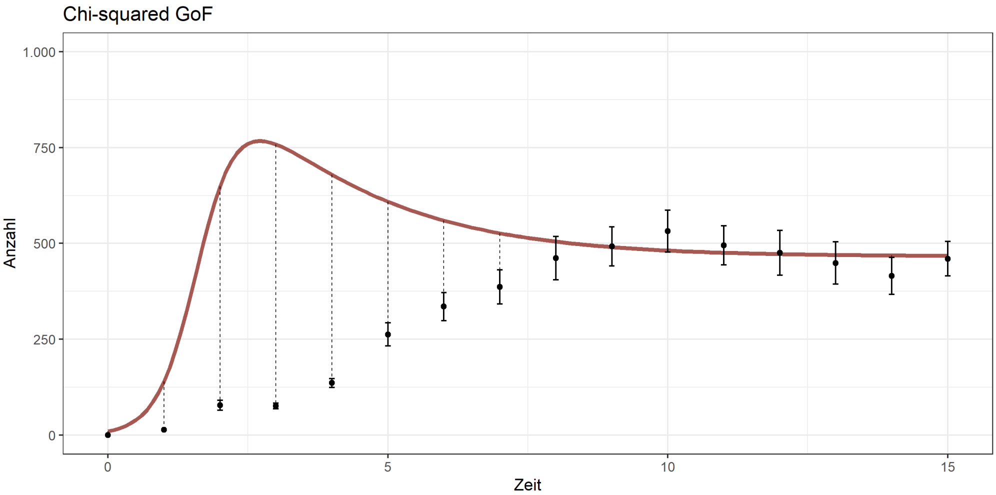
- How can you measure the goodness-of-fit of a model (i.e., how well the model output corresponds to observations)?
  - **Squared Error:** $Q(\theta) = \sum_{i=1}^{n}(f(x_i, \theta) - x_i)^2$
  - **Chi-Squared:** $\chi(\theta) = \sum_{i=1}^{n}(\frac{f(x_i,\theta)-x_i}{\sigma_i})^2$
  - **likelihood:** $\mathcal{L}(\theta|x) = P_\theta(X = x)$ (discrete variable) bzw. $\mathcal{L}(\theta|x) = f_\theta(x)$ (continuous variable)
- Squared Error (also *Mean Squared Error*; MSE or *Root Mean Squared Error*; RMSE) should be minimized
- Likelihood (or rather log-likelihood $\ell(\theta|x)$) should be maximized

- How can you measure the goodness-of-fit of a model (i.e., how well the model output corresponds to observations)?
  - **Squared Error:** $Q(\theta) = \sum_{i=1}^{n} (f(x_i, \theta) - x_i)^2$
  - **Chi-Squared:** $\chi(\theta) = \sum_{i=1}^{n} (\frac{f(x_i,\theta)-x_i)}{\sigma_i})^2$
  - **likelihood:** $\mathcal{L}(\theta|x) = P_\theta(X = x)$ (discrete variable) bzw. $\mathcal{L}(\theta|x) = f_\theta(x)$ (continuous variable)
- Squared Error (also *Mean Squared Error*; MSE or *Root Mean Squared Error*; RMSE) should be minimized
- Likelihood (or rather log-likelihood $\ell(\theta|x)$) should be maximized

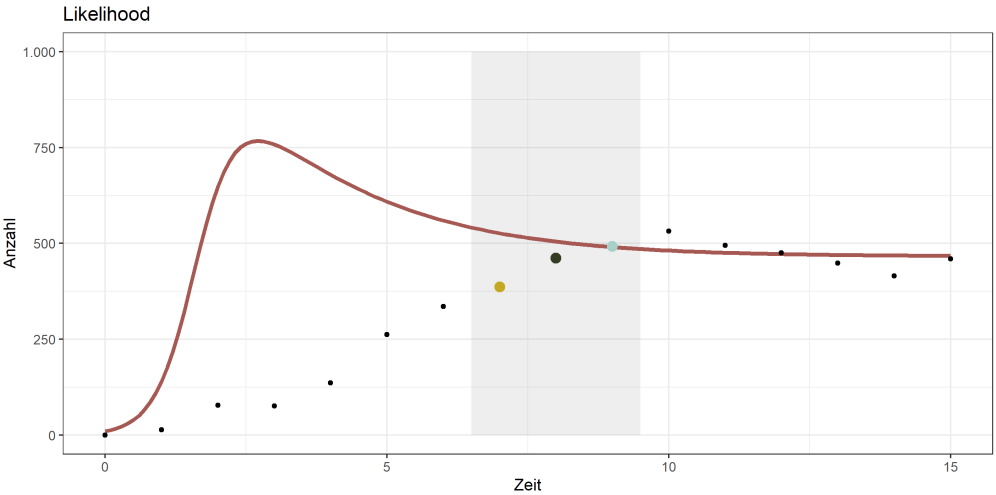
- How can you measure the goodness-of-fit of a model (i.e., how well the model output corresponds to observations)?
  - **Squared Error:** $Q(\theta) = \sum_{i=1}^{n} (f(x_i, \theta) - x_i)^2$
  - **Chi-Squared:** $\chi(\theta) = \sum_{i=1}^{n} (\frac{f(x_i,\theta) - x_i}{\sigma_i})^2$
  - **likelihood:** $\mathcal{L}(\theta|x) = P_\theta(X = x)$ (discrete variable) bzw. $\mathcal{L}(\theta|x) = f_\theta(x)$ (continuous variable)
- Squared Error (also *Mean Squared Error*; MSE or *Root Mean Squared Error*; RMSE) should be minimized
- Likelihood (or rather log-likelihood $\ell(\theta|x)$) should be maximized



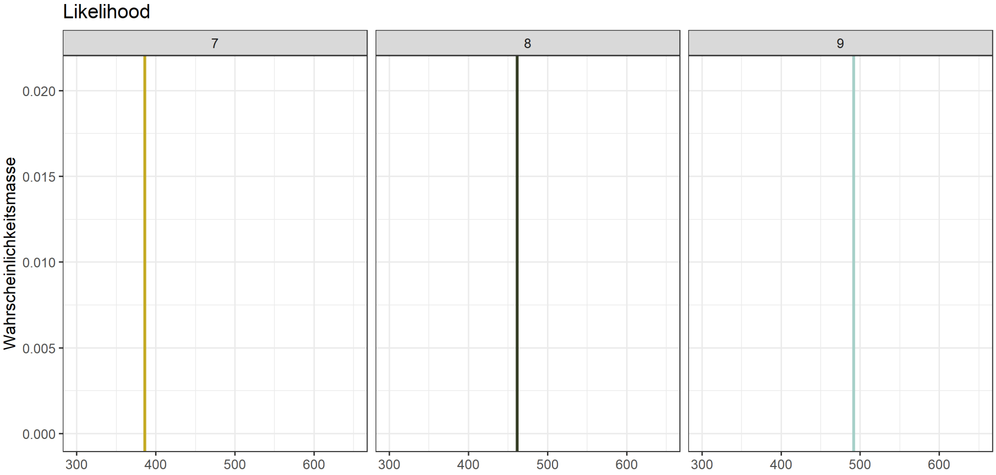
Quadrierte Abweichung
1.000
750
500
250
0
Anzahl
0
5
10
15
Zeit



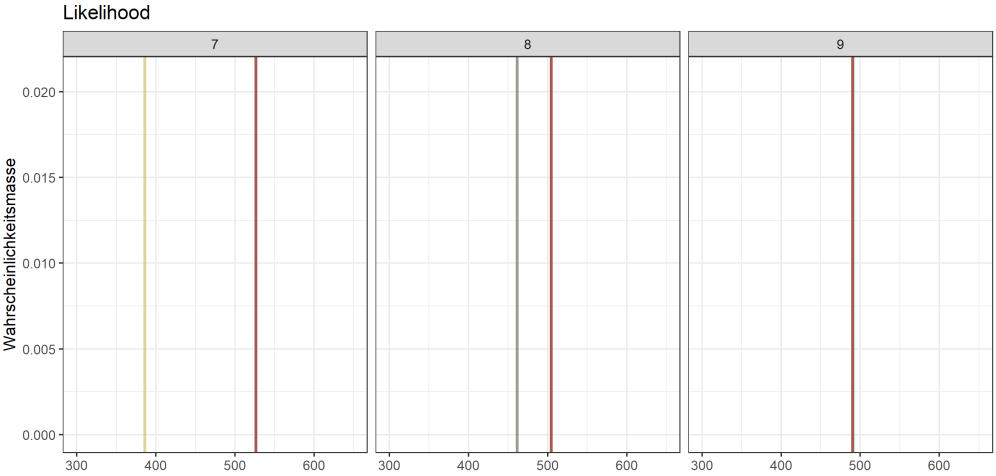
Chi-squared GoF
Anzahl
1.000
750
500
250
0
0
5
10
15
Zeit



Likelihood
1.000
750
500
250
0
Anzahl
0
5
10
15
Zeit

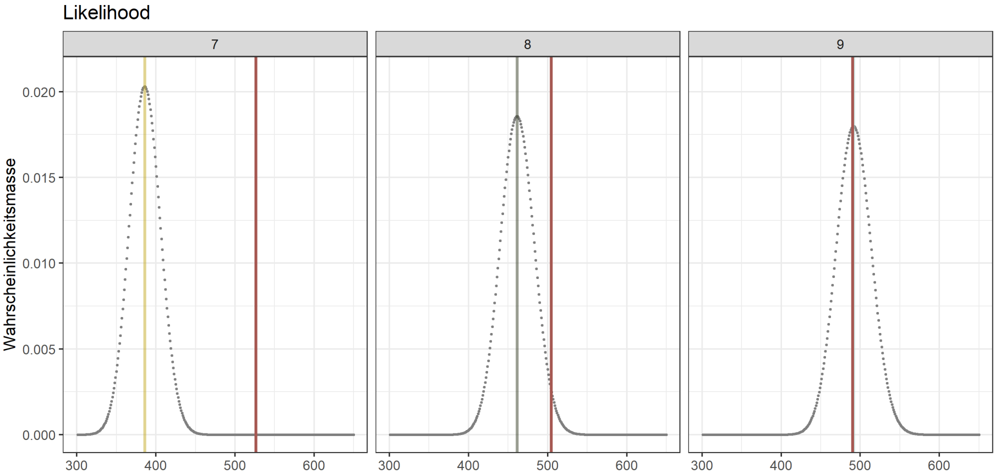


Likelihood
7
8
9
Wahrscheinlichkeitsmasse
0.020
0.015
0.010
0.005
0.000
300
400
500
600



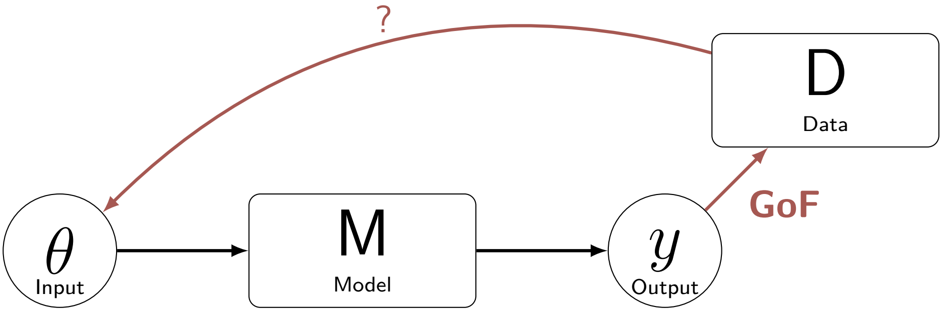
Likelihood
7
8
9
Wahrscheinlichkeitsmasse
0.020
0.015
0.010
0.005
0.000
300
400
500
600

3. Goodness-of-Fit
Procedure of model calibration

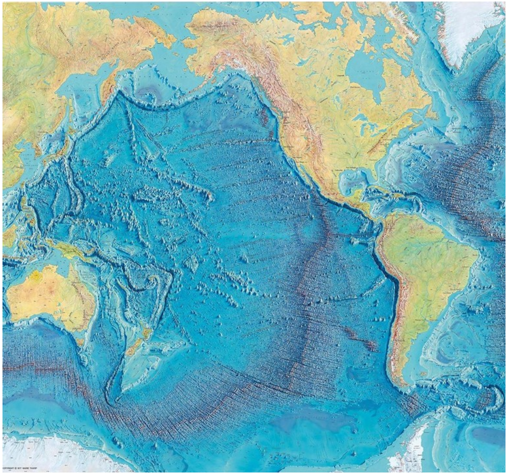


- *log-likelihood* is the most common GoF-measure
- The calibration process will be slow or difficult, if the initial values of the free parameters are far away from the optimal values
- The calibration target usually dictates the choice of the probability distribution
  - Poisson- or negative Binomial-distribution for count data
  - Gamma-distribution for variables with support $x \in \mathbb{R}^+$
  - Beta- or Binomial-distribution for $x \in [0, 1]$
- Combinations of calibration targes or rather their GoFs can be combined

Goal of the **adaption** step is finding a value for θ leading to the optimal GoF

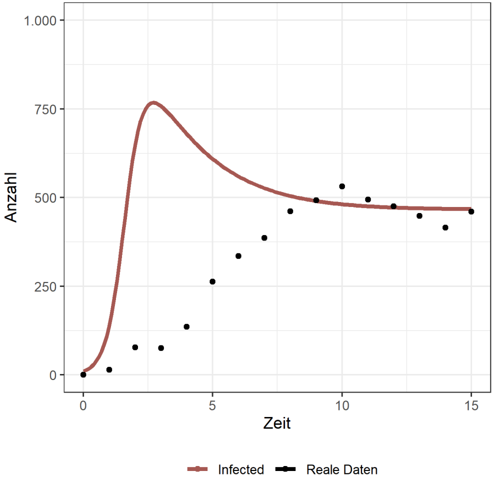


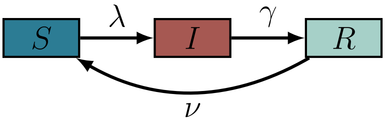
How do we find the deepest point of the ocean?



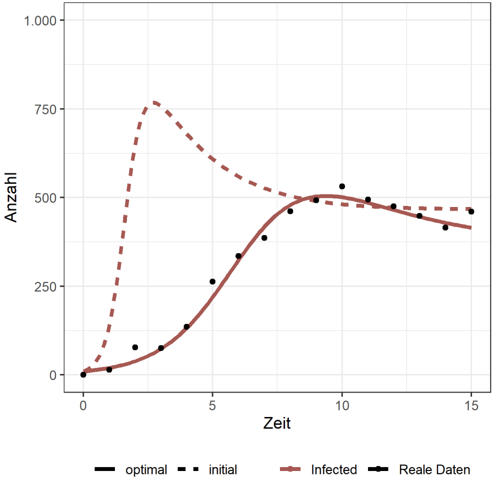
- How do we find latitude and longitude of the deepest point of the ocean?
- We do not know the map of the ocean
- With each dive, we can only explore one geographic position on the ocean floor
- Dives are very expensive, so
  - we want to find the deepest point as fast as possible or with the least amount of dives
  - there is a maximum number of dives that can be performed
- Our echosounder can only measure the depth in meters (not cm or mm)

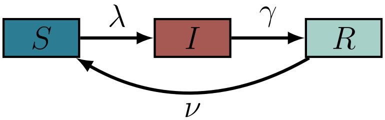
Example	**Models**
Depth	GoF
Latitude & Longitude	free parameters θ
Dive	Model run
Costs of dive	*run-time* of the model
precision of echosounder	acceptance criteria



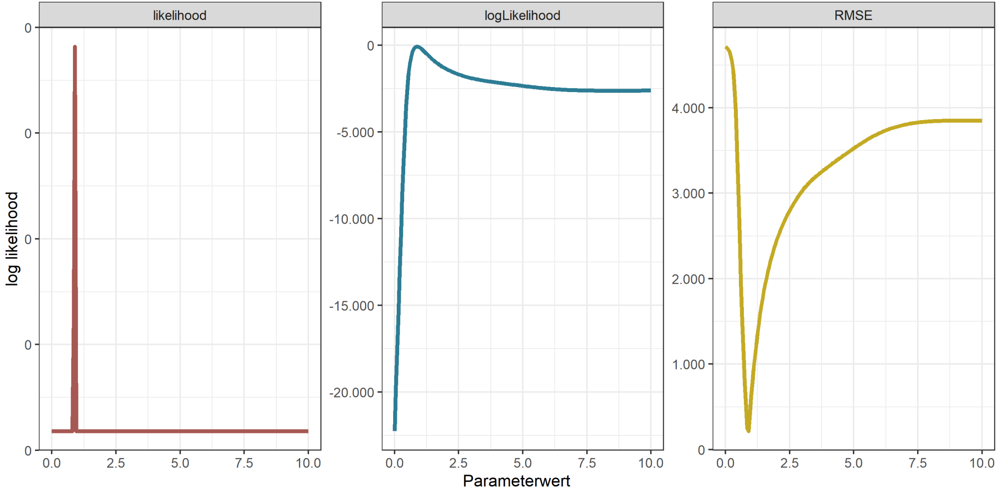


- $\gamma = \dfrac{1}{5}$ and $\nu = \dfrac{1}{5}$
- Starting population $S = 990$, $I = 10$ and $R = 0$
- Free parameter λ
- How to we arrive at the true (optimal) $\lambda = 0.9$ starting from $\lambda = 3$?





- $\gamma = \dfrac{1}{5}$ and $\nu = \dfrac{1}{5}$
- Starting population $S = 990$, $I = 10$ and $R = 0$
- Free parameter λ
- How to we arrive at the true (optimal) $\lambda = 0.9$ starting from $\lambda = 3$?


likelihood
logLikelihood
RMSE
log likelihood
0
0
0
0
0
0.0
2.5
5.0
7.5
10.0
0
-5.000
-10.000
-15.000
-20.000
Parameterwert
4.000
3.000
2.000
1.000
0

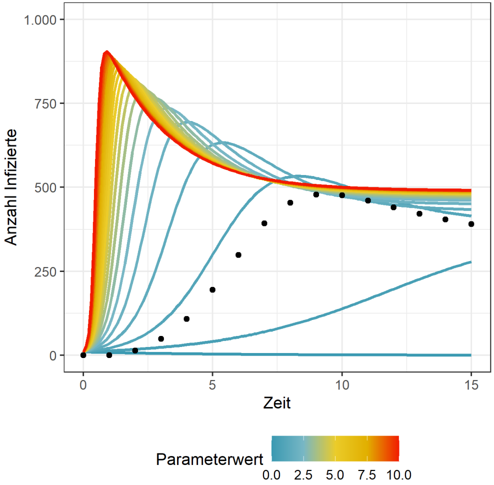
Overview algorithms

Adaption step

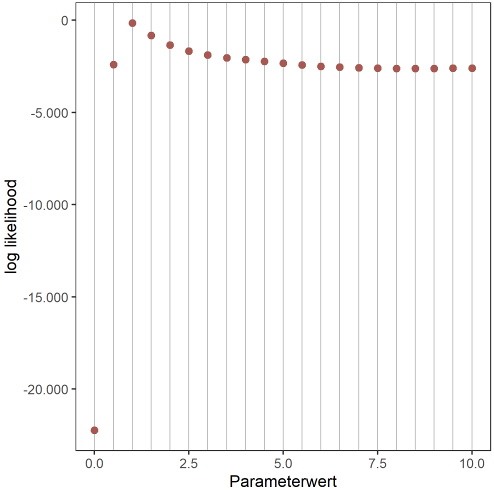
- There are many methods and algorithms available for adaption step
- Even for the same algorithm, countless implementations with slight variations can be found
- The following slides try to show the basic ideas of three different classes of **optimization** algorithms

The *grid search* method **systematically** tries different values for λ

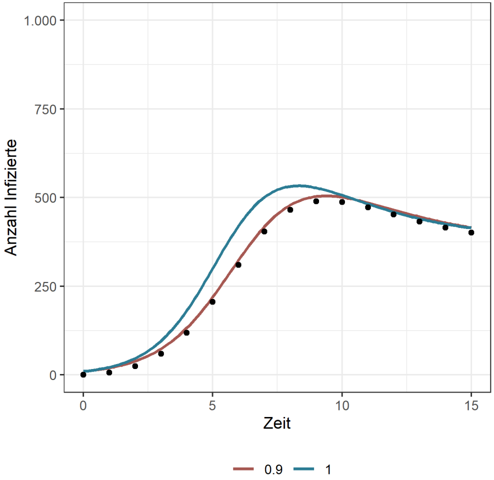
1. Define limits for the values of each free parameter (e.g., $p_1 \in [0, 100]$ and $p_2 \in [0, 1]$)
2. Define an interval for each of the value ranges defined in step 1 (e.g., 10 for p_1 and 0.1 for p_2)
3. Evaluate / run the model for all combinations of the previously defined values (e.g. $\{(0; 0), (0; 0.1), \ldots, (10; 0), (10; 0.1), \ldots, (100; 1)\}$)



- Variation of parameter λ between 0 and 10 using a step of 0.5
- Calculate the GoF-measure for each model run
- Choos the model with the best (optimal) GoF-measure



- Variation of parameter λ between 0 and 10 using a step of 0.5
- Calculate the GoF-measure for each model run
- Choos the model with the best (optimal) GoF-measure



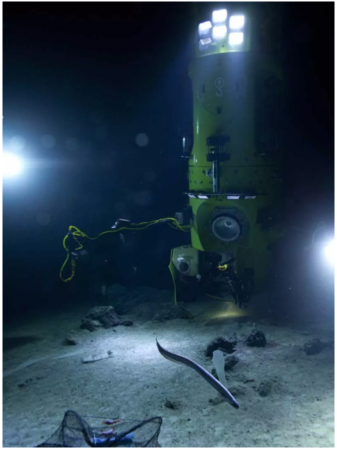
- Variation of parameter λ between 0 and 10 using a step of 0.5
- Calculate the GoF-measure for each model run
- Choos the model with the best (optimal) GoF-measure

Pros

- Easy to implement
- Parallelization easily possible
- Highly reproducible

Cons

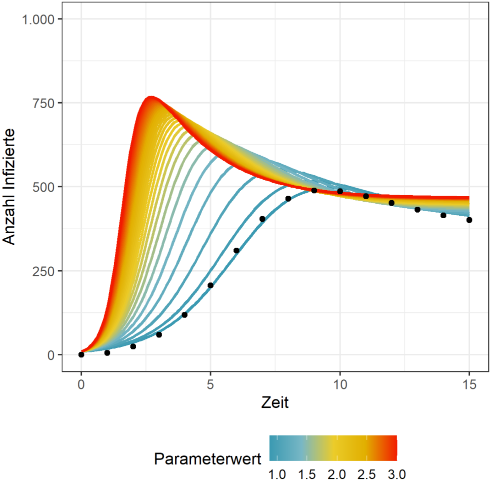
- Unclear, if the best value is within the defined value range
- Unclear, if better value might be between intervals
- *curse of dimensionality*: Using interval with 100 values and 5 Parameters
  $\Rightarrow 100^5 = 10,000,000,000$ model runs (*run-time* of 0.1 seconds leads to 31.8 years of computation time)



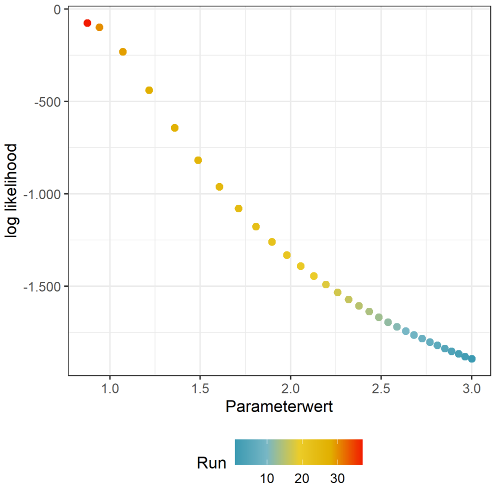
- We measure the depth with each dive
- With a little more effort (costs) we can retrieve more information on the ocean floor:
  - How strongly and in which direction does the floor descent? (1. derivaite)
  - Is it getting more steaper or more shallower? (2. derivative)
- Gradient-based algorithms use this additional information, but come with additional costs

Gradient Descent algorithm

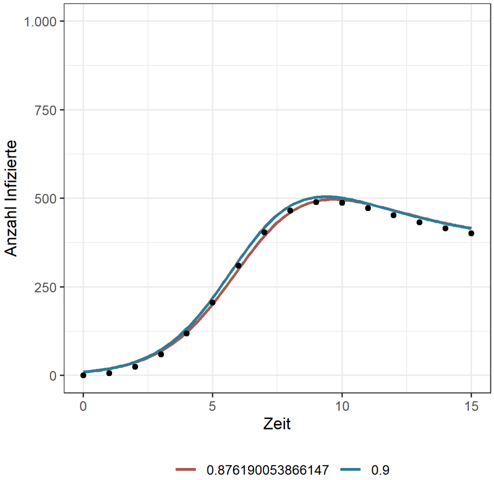
1. We first measure at an (random) initial point (x_0) of the free parameters and calculate the GoF ($F(x)$)
2. Calculate the 1. derivative at intial point $x_0 \Rightarrow$ Calculate direction of steepest descent (Jacobian $\nabla F(x)$)
3. We follow the direction of the steepest descent for a distance γ to point x_1
4. Repeating steps 2 and 3 ($x_{n+1} = x_n - \gamma_n \nabla F(x_n)$) ideally leads to the optimal values x^* of the free parameters
5. The search stops, as soon as we hit a maximum number of model runs, or as soon as the GoF of x_{n+1} does not improve by a certain increment compared to the GoF of x_n



- Begin gradient descent at initial parameter value 3
- Step size γ is set to $1/20,000$
- Maximum of 100 iterations
- The algorithm converges regularly after 37 Iterationen, i.e., it found the optimal value



- Begin gradient descent at initial parameter value 3
- Step size γ is set to $1/20,000$
- Maximum of 100 iterations
- The algorithm converges regularly after 37 Iterationen, i.e., it found the optimal value



- Begin gradient descent at initial parameter value 3
- Step size γ is set to $1/20,000$
- Maximum of 100 iterations
- The algorithm converges regularly after 37 Iterationen, i.e., it found the optimal value

Pros

- Very efficient, i.e., finds the optimal value with few model runs
- Very fast, if the 1. derivative of $F(x)$ (Jacobian) is known or can be analytically derived

Cons

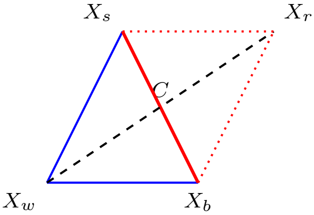
- GoF-function needs to be differenciable (problematic for non-continuous GoF-functions)
- Approximation of the Jacobian needs additional model runs
- Might get stuck in local minima
- Step size of $gamma$ too large $\rightarrow$ Zick-Zack-ing over the optimal values
- Step size of $gamma$ too small $\rightarrow$ algorithm becomes in-efficient

- Most algorithm are searching for parameter values in $\mathbb{R}$
- This might lead to unwanted effects, e.g., if SAR becomes negative $\Rightarrow$ persons vanish from the model population
- Transforming parameters can solve this problem:
  - $\mathbb{R} \to \mathbb{R}^+$: e^x (log-link)
  - $\mathbb{R} \to (0, 1)$: $\frac{e^x}{1+e^x}$ (logit-link)

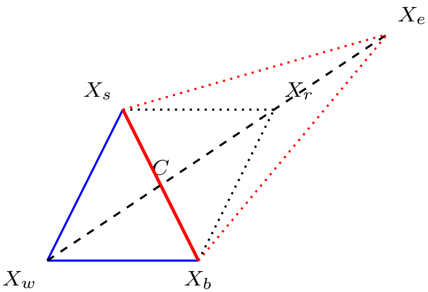
- Extensions of the *gradient descent* algorithm are mostly about determining the step size γ, ideally making large steps with small descent and small steps for great descent
- E.g., Newton-method of BFGS are using the Hessian **H** (2. derivative) instead of the Jacobian **J** (1. derivative)
- This increases the requirenments on the GoF-function (at leat 2 times differenciable, ideally with onyl one minimum)
- If it is possible to approximate **H**, the negative inverse $-\mathbf{H}^{-1}$ can be used to approximate the covariance matrix of the free parameters. I.e, this makes it possible to calculate the variance or confidence intervals of the estimates.

Nelder-Mead algorithm

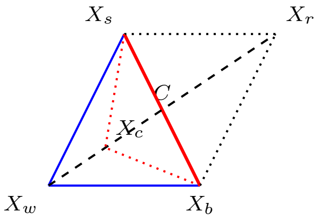
1. for n free parameters calculate the GoF for a simplex of $n + 1$ points $x_1, \ldots, x_{n+1}$
2. Sort the points by their GoF-function $f(x)$
3. Calculate the centroids C between all points, besides the point with the worst GoF (x_w)
4. Perform one of **4** possible calculations steps or transformations
5. Stop the search after maximum of iterations, reaching a specific size of the simples or no relevant improvement of the GoF



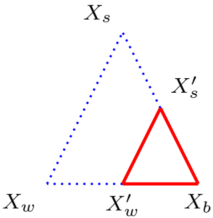
1. **Reflection**: Calculate $x_r = C + \alpha(C - x_w)$, if $f(x_w) < f(x_r) \leq f(x_b)$ replace x_w by x_r
2. **Expansion**: If $f(x_r) > f(x_b)$, calculate additional point $x_e = C + \gamma(x_r - C)$. Replace x_w by x_r or x_e, whichone is better
3. **Contraction**: If $f(x_r) < f(x_s)$, calculate $x_c = C + \beta(x_w - C)$. If $f(x_c) > f(x_w)$, replace x_w by x_c
4. **Shrink Contraction**: If none of the above, reject all points besides x_b and calculate new coordinates for all other points via $x_j = x_b + \delta(x_j - x_b)$



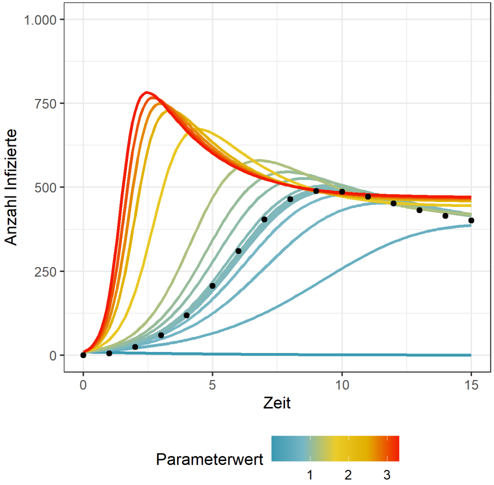
1. **Reflection**: Calculate $x_r = C + \alpha(C - x_w)$, if $f(x_w) < f(x_r) \leq f(x_b)$ replace x_w by x_r
2. **Expansion**: If $f(x_r) > f(x_b)$, calculate additional point $x_e = C + \gamma(x_r - C)$. Replace x_w by x_r or x_e, whichone is better
3. **Contraction**: If $f(x_r) < f(x_s)$, calculate $x_c = C + \beta(x_w - C)$. If $f(x_c) > f(x_w)$, replace x_w by x_c
4. **Shrink Contraction**: If none of the above, reject all points besides x_b and calculate new coordinates for all other points via $x_j = x_b + \delta(x_j - x_b)$



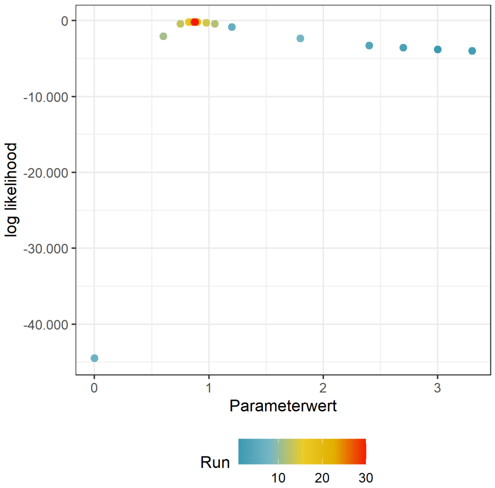
1. **Reflection**: Calculate $x_r = C + \alpha(C - x_w)$, if $f(x_w) < f(x_r) \leq f(x_b)$ replace x_w by x_r
2. **Expansion**: If $f(x_r) > f(x_b)$, calculate additional point $x_e = C + \gamma(x_r - C)$. Replace x_w by x_r or x_e, whichone is better
3. **Contraction**: If $f(x_r) < f(x_s)$, calculate $x_c = C + \beta(x_w - C)$. If $f(x_c) > f(x_w)$, replace x_w by x_c
4. **Shrink Contraction**: If none of the above, reject all points besides x_b and calculate new coordinates for all other points via $x_j = x_b + \delta(x_j - x_b)$



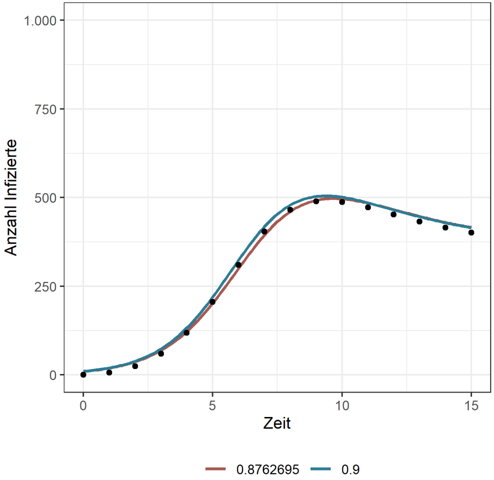
1. **Reflection**: Calculate $x_r = C + \alpha(C - x_w)$, if $f(x_w) < f(x_r) \leq f(x_b)$ replace x_w by x_r
2. **Expansion**: If $f(x_r) > f(x_b)$, calculate additional point $x_e = C + \gamma(x_r - C)$. Replace x_w by x_r or x_e, whichone is better
3. **Contraction**: If $f(x_r) < f(x_s)$, calculate $x_c = C + \beta(x_w - C)$. If $f(x_c) > f(x_w)$, replace x_w by x_c
4. **Shrink Contraction**: If none of the above, reject all points besides x_b and calculate new coordinates for all other points via $x_j = x_b + \delta(x_j - x_b)$



- Start Nelder-Mead for inital parameter value 3
- Using the following values for the control parameters of the algorihtm:
  - *Reflection* α: 1
  - *Expansion* γ: 2
  - *Contraction* β: 0.5
  - *Shrink Contraction* δ: 0.5
- Maximum of 100 iterations possible
- Algorithm converges successfully after 30 iterations, i.e., finding the optimal value



- Start Nelder-Mead for inital parameter value 3
- Using the following values for the control parameters of the algorihtm:
  - *Reflection* α: 1
  - *Expansion* γ: 2
  - *Contraction* β: 0.5
  - *Shrink Contraction* δ: 0.5
- Maximum of 100 iterations possible
- Algorithm converges successfully after 30 iterations, i.e., finding the optimal value



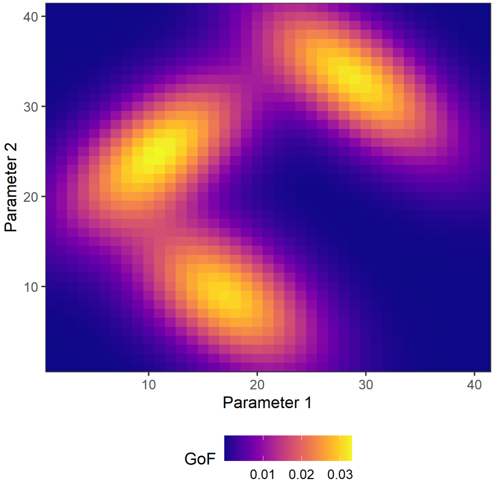
- Start Nelder-Mead for inital parameter value 3
- Using the following values for the control parameters of the algorihtm:
  - *Reflection* α: 1
  - *Expansion* γ: 2
  - *Contraction* β: 0.5
  - *Shrink Contraction* δ: 0.5
- Maximum of 100 iterations possible
- Algorithm converges successfully after 30 iterations, i.e., finding the optimal value

Pros

- Can find the optimal parameter values with few model runs
- It is not necessary to calculate or approximate $\mathbb{J}$ or $\mathbb{H}$ and less problematic for non-continuous GoF functions

Cons

- Selecting the initial simplex may lead to local search resultung in local minimum
- Unlucky choice of control parameters α, β, γ and δ may lead to long run times
- Not suitable for many free parameters $(N > 20)$
- Several model evaluations (model runs) per iteration of the algorithm
- Results for very shallow GoF functions might depend on the acceptance criterium



- Gradient-based algorithms are very efficient, but prone to find local minima and are problematic for "unsmooth" GoF-functions
- **J** and **H** are usually not known or cannot be derived analytically for (complex) differential equation models (or agent-based models) and need to be approximated (costs!)
- Acceptance criteria for IDM usually need to be set more forgiving than for other problems

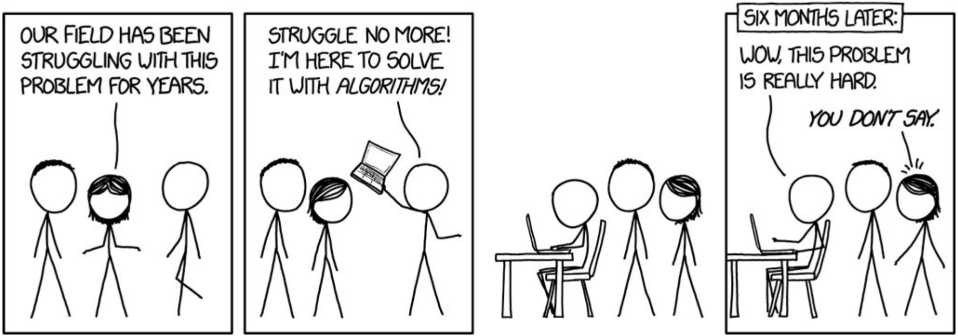
- In contrast to regression models, optimization algorithms for infectious diseae models rather take hours or days than seconds
- Keeping the cost of each model run low is absolutely essential. I.e., having fast code and remove unnecessary parts.
- A run-time of 1 second translates to 28 hours for 100.000 iterations
- Only few (mostly inefficient) algorithms *grid-search* can be parallelized, as step $n+1$ depends on the result of step n

- Keep the number of free parameters to a minimum to avoid *overfitting* (especially for scenario models)
- Start with few parameters and try to identify from the results where additional parameters might yield gains in the GoF, e.g., age-specific rates
- It maybe useful to use "meta"-parameters, for example for age- or time-dependencies
- Instead of having one free parameter per age group or time step, work with linear equations (e.g., $\beta_0 + \beta_1 * AGE$) or *splines*
- But watch for artefacts in your extrapolations!

Fazit

- Calibration is the second most time-intensive part of modeling (after searching for data input)
- All algorithms require certain assumptions, to be effienct at ”clever guessing” the right parameter values
- Violating these assumptions might lead to false results or inefficient calibration
- The choice of the free parameters, the calibration targets and the choice and setup of the opimization algorithm should be in line

Thank you very much for your attention!



Dr. Stefan Scholz

stefan.scholz@uk-halle.de

@StefanScholz85