Nonlinear Modeling: Parameter Estimation

Introduction to Statistical Modelling

Prof. Joris Vankerschaver

Outline

Learning outcomes

You should be able to

- Determine the parameters of a nonlinear model via minimization (using R)
- Understand the principles behind various minimization algorithms, as well as their advantages and disadvantages
- Be able to assess the fit of a model

Example: building a stock-recruitment model

M. merluccius: stock-recruitment model

European hake (*M. merluccius*)

- Deep water fish
- Important for European fisheries
- Similar to 명태 in Korea

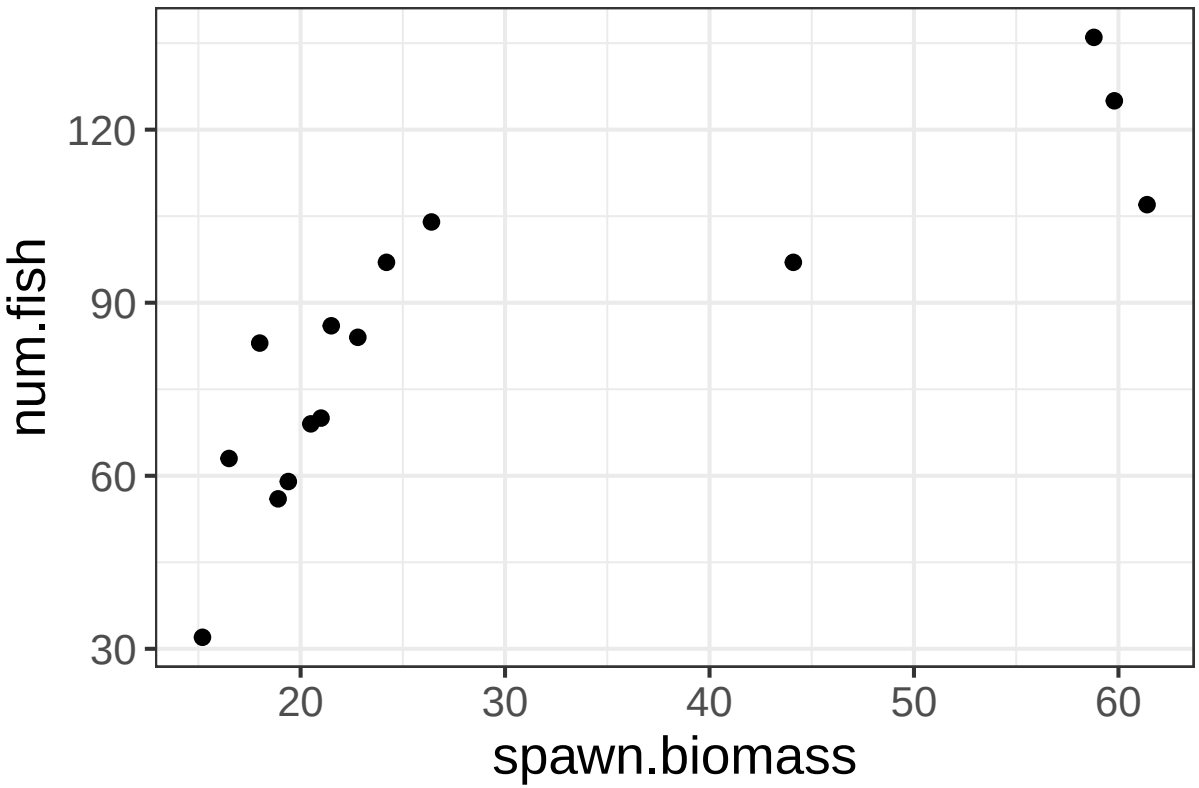
Stock-recruitment model

Model of **number of adult fish** (recruitment) as a function of **spawning biomass** (fish that can reproduce).

M.merluccius: Dataset

15 observations, 3 features:

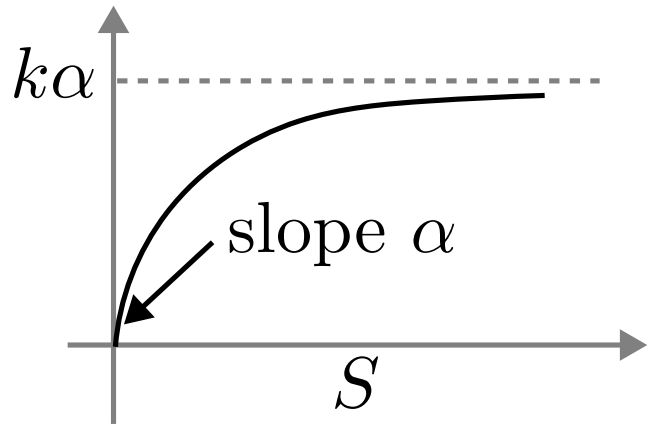
- `spawn.biomass`: spawning (stock) biomass
- `num.fish`: number of fish (recruitment)
- `year`: not used



M. merluccius: Beverton-Holt model

Beverton-Holt model (1956):

$$f(S;\alpha,k) = \frac{\alpha S}{1 + S/k}$$



Parameters:

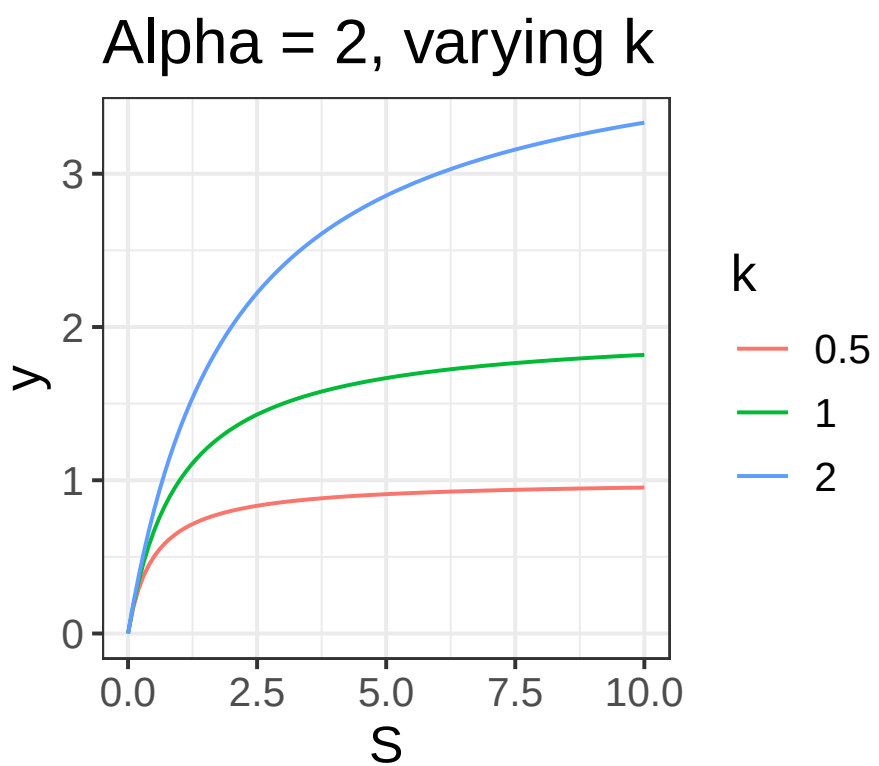
- α: initial growth rate (for $S = 0$)

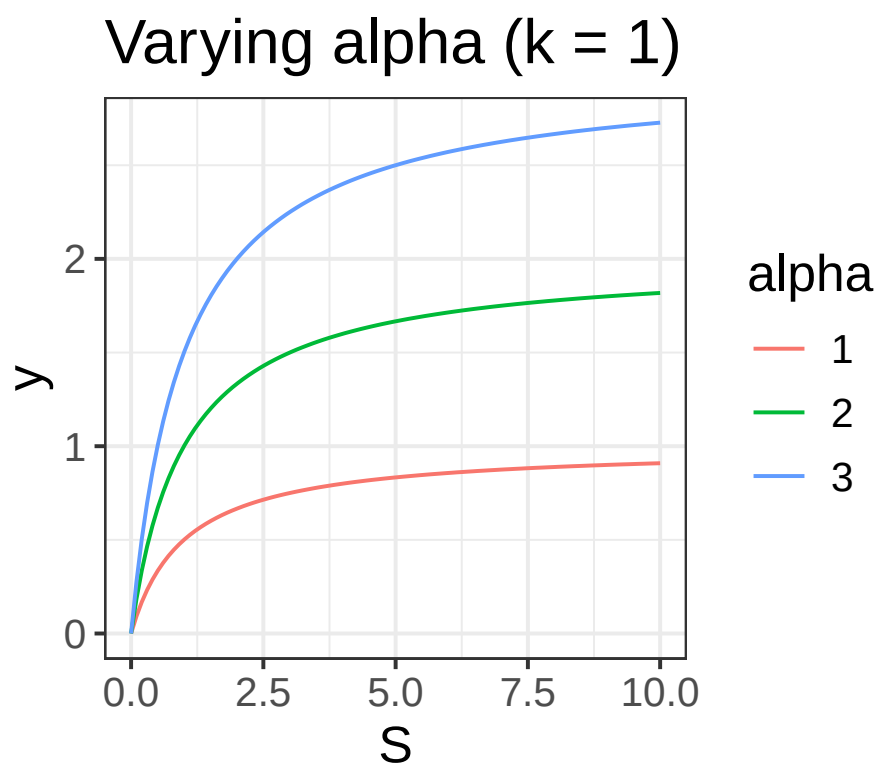
$$\alpha = f'(0;\alpha,k)$$

- k: related to behavior for large S

$$k\alpha = \lim_{S\to+\infty} f(S;\alpha,k)$$

Beverton-Holt: Effect of varying α and k





Goals

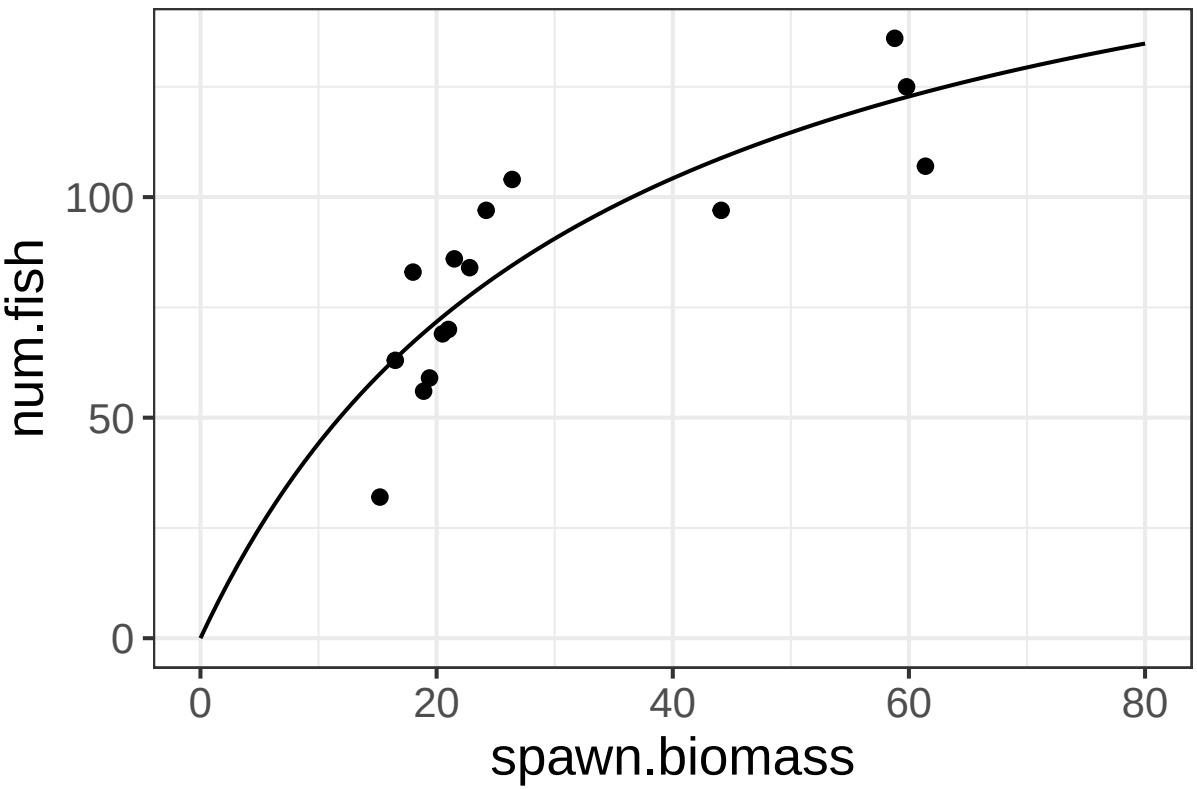
- **Parameter estimation**: Find values $\hat{\alpha}$ and $\hat{k}$ that best fit data.
- **Uncertainty quantification**: Provide a measure of uncertainty for parameter values (confidence interval)
- **Sensitivity analysis**: Understand how model changes if parameters are varied

Parameter estimation

What is parameter estimation?

Determining the **optimal values for the parameters** using the experimental data, assuming that the model is known.

Example: For *M. merluccius*, we will see that $\hat{\alpha} = 5.75$, $\hat{k} = 33.16$.



Specifying a model

Assume that we are **given** a nonlinear model

$$y = f(x; \theta) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is normally distributed noise.

- x: inputs, predictors, features (e.g. `spawn.biomass`)
- y: outcome, depent variable (e.g. `num.fish`)
- θ: (vector of) parameters (e.g. $\theta = (\alpha, k)$)

We will not talk about **building** a model (see one of your many other courses)

The objective function

Given a dataset $(x_1, y_1), \ldots, (x_N, y_N)$, we want to quantify how well the model fits the data.

Objective function: measures difference (squared) between predictions $f(x_i; \theta)$ and actual values y_i:

$$J(\theta) = \sum_{i=1}^{N} (y_i - f(x_i; \theta))^2$$

Minimizing the objective function

Goal: Find the parameter value(s) $\hat{\theta}$ so that $J(\theta)$ is minimal:

$$\hat{\theta} = \text{argmin}_\theta \, J(\theta).$$

Problems:

- Depending on $f(x; \theta)$ this can be very difficult
- There may be multiple (local) minima
- Almost always needs to be done numerically

Example: linear regression

In linear regression, $f(x;\theta) = \alpha + \beta x$, so that

$$J(\alpha, \beta) = \sum_{i=1}^{N} (y_i - \alpha - \beta x_i)^2.$$

Minimizing $J(\alpha, \beta)$ can be done by setting the partial derivatives equal to zero and gives the usual formulas:

$$\hat{\beta} = R\frac{s_y}{s_x}, \quad \hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}.$$

In general, **no closed-form formula exists** for the optimal parameters.

Before parameter estimation: select parameters

More parameters = more work and less certainty:

- Solver may not converge
- Wider uncertainty estimates for parameters and outputs
- Correlations between parameters can make it impossible to find parameters

Consider selecting subset of parameters to estimate:

- Fix parameters at experimental values
- Omit least sensitive parameters

Before parameter estimation: select initial values

Numerical optimization algorithm requires a good **starting guess** for the parameters. When choice is bad:

- Algorithm will converge slowly (take many iterations)
- Optimization will fail altogether

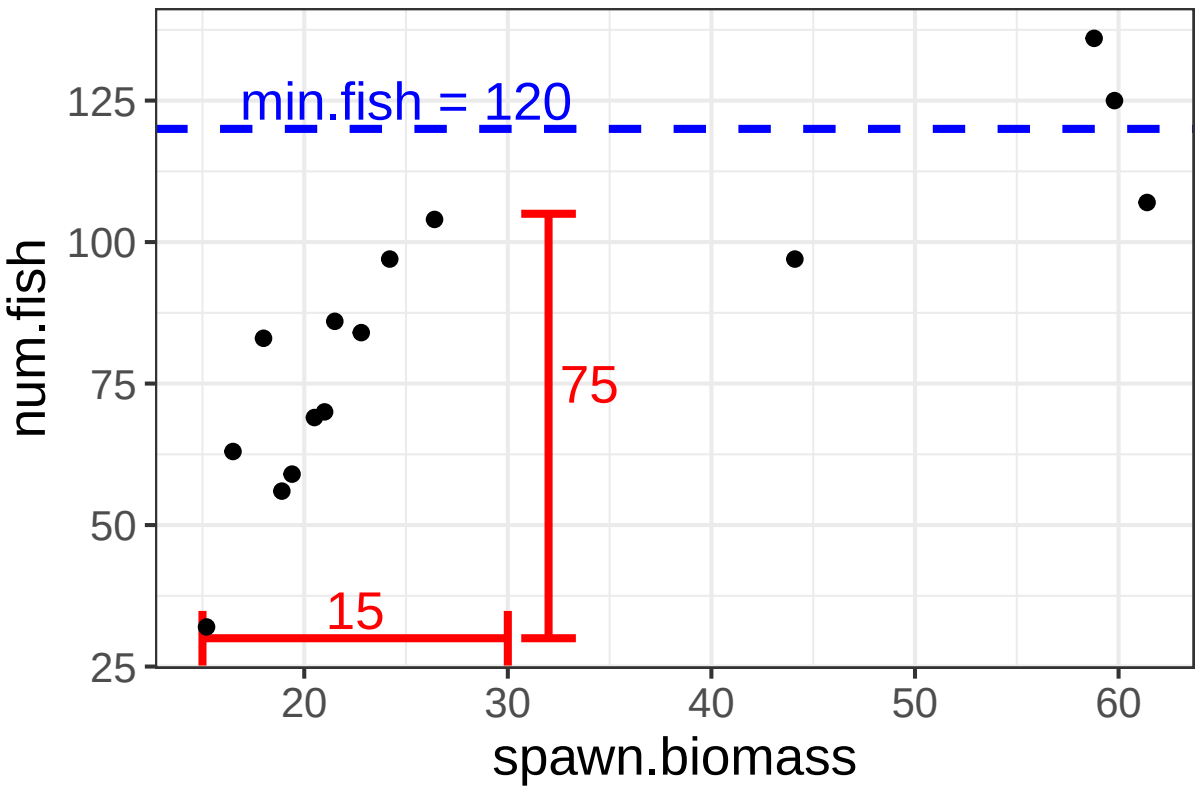
How to find initial guess:

- Determine from model properties (growth rate, asymptotes)
- Use (known) experimental values
- Use trial and error (select from grid of values)

Doesn't need to be overly precise, a rough estimate is usually sufficient.

Initial values for M. merluccius

- Slope: $\alpha_0 = \frac{75}{15} = 6$
- Horizontal asymptote: $k_0\alpha_0 = 120$, so $k_0 = 20$.



Later, we will see that the initial guesses are close to the optimal parameters $\hat{\alpha} = 5.75$, $\hat{k} = 33.16$.

Preparation: Determining boundaries for parameters

Some parameters come with bounds, for example:

- Kinetic rate: $k > 0$
- Probability: $0 \leq p \leq 1$

Two ways of accounting for parameter bounds:

- Adding penalty terms to the objective function
- Transforming the parameter so it becomes unconstrained

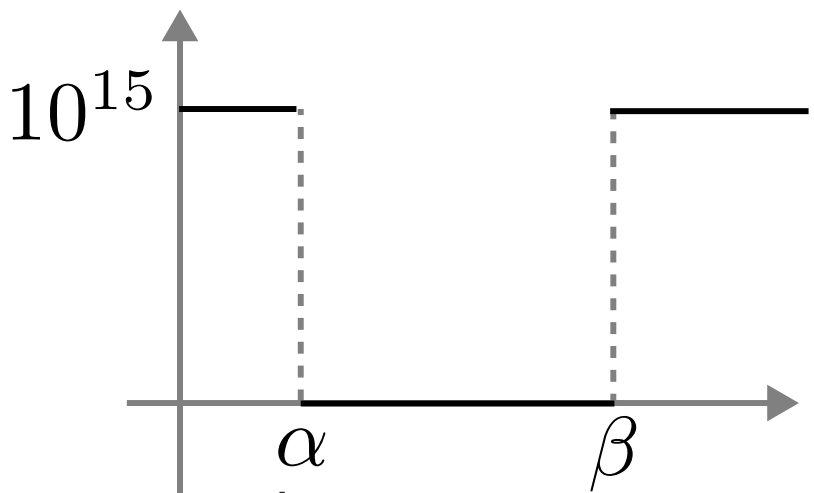
Adding penalty terms

Suppose we want $\alpha \leq \theta \leq \beta$. Add **penalty term** to objective function:

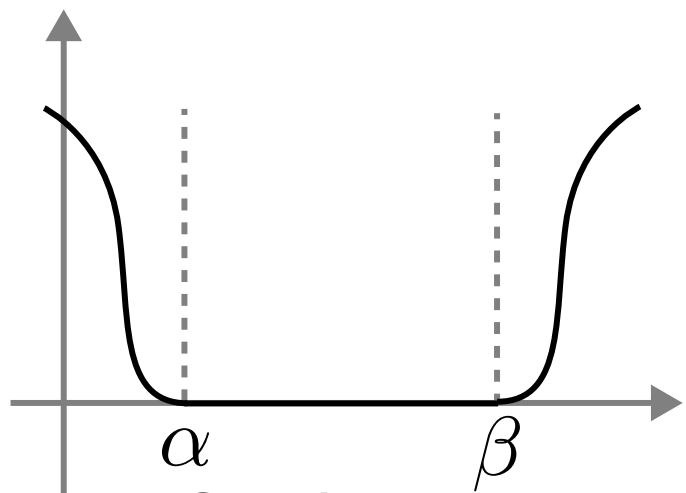
$$J_{\text{constrained}}(\theta) = J_{\text{unconstrained}}(\theta) + J_{\text{penalty}}(\theta)$$

where $J_{\text{penalty}}(\theta)$ is

- Roughly zero between α and β
- Very large for $\theta < \alpha$ or $\theta > \beta$.



discontinuous
(step function)



Continuous
(smooth)

Transformation parameters

Transform constrained problem into equivalent **unconstrained** problem.

Some examples:

- If $\theta > 0$: write $\theta = \exp \varphi$ or $\theta = \varphi^2$
- If $-1 < \theta < 1$: write $\theta = \tanh \varphi$

In either case, φ is unconstrained (can range from $-\infty$ to $+\infty$). Now substitute this transformation into the objective function, and optimize in terms of φ.

Preparation: Dealing with non-identifiability

In some cases, parameters cannot be determined uniquely. For example, exponential model with parameters A, B, C:

$$y = A \exp(Bx + C) = (Ae^C)e^{Bx}$$

Only B and the combination Ae^C can be determined.

- **Structural** identifiability: all parameters can be uniquely determined, given perfect data.
- **Practical** identifiability: same, but from finite, noisy data.

Preparation: Dealing with non-identifiability

Minimization of objective function will **fail** if some parameters are not identifiable. Workarounds:

- Add penalty term to J to privilege certain parameter values
- Rewrite J so all parameters are identifiable

Example: put $k = Ae^C$ and write exponential model as

$$y = k \exp(Bx).$$

Both k and B are identifiable.

Minimizing the objective function

General approach

Recall that we are trying to find θ so that

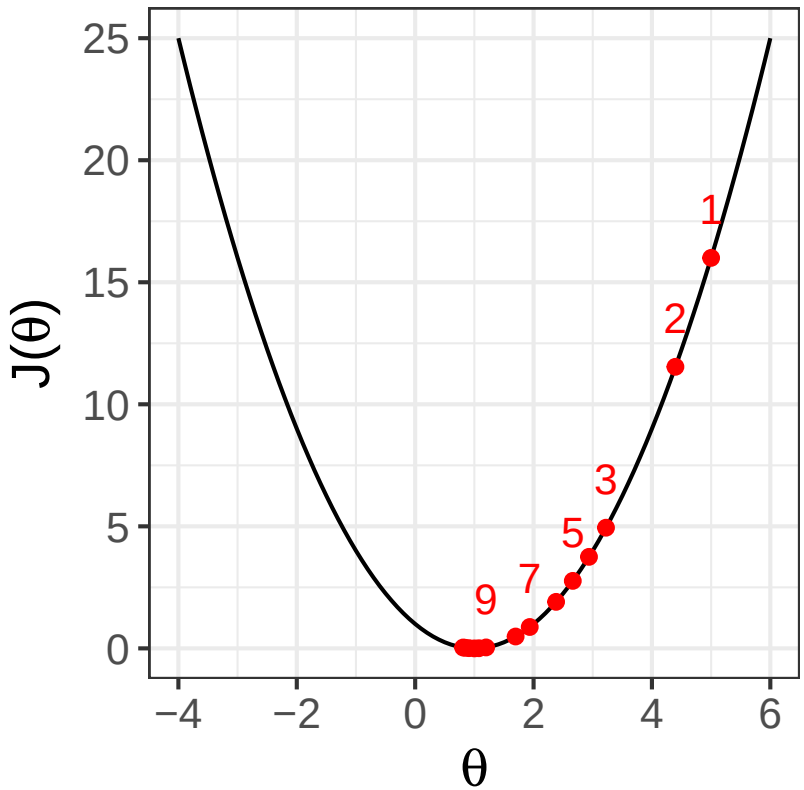
$$J(\theta) = \sum_{i=1}^{N} (y_i - f(x_i; \theta))^2$$

is minimized.

- For **linear** model: direct, one-step solution
- For **nonlinear** model: iterative algorithm. Typically:
  1. Start with initial guess for $\hat{\theta}$
  2. Slightly change $\hat{\theta}$ and compute $J(\hat{\theta})$
  3. Repeat if $\hat{\theta}$ not good enough

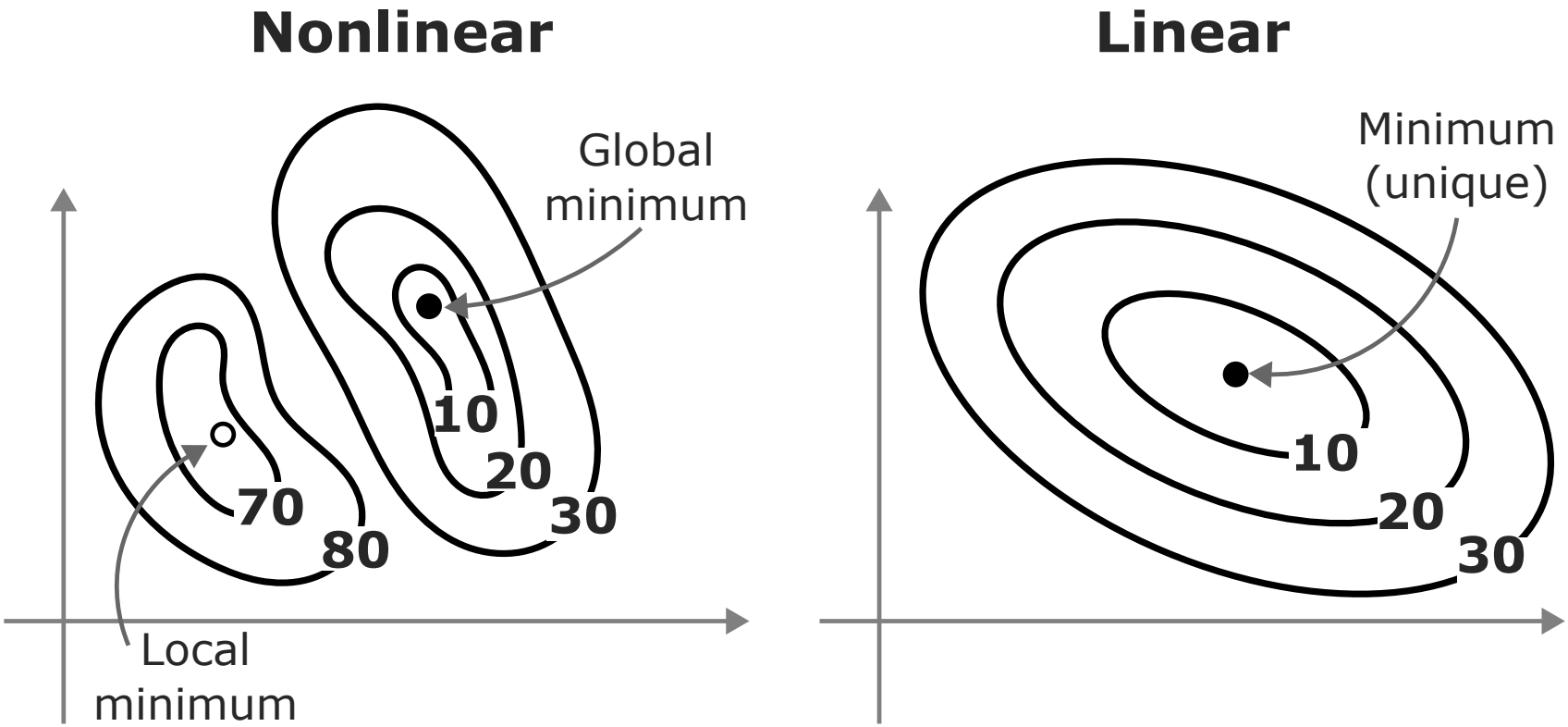
Very simple minimization algorithm: hill descender

```
# Initial guess
theta <- 5.0

for (i in 1:100) {
  # Add random noise to theta
  theta_new <-
    theta + 0.5 * rnorm(1)

  # Accept if objective is lower
  if (J(theta_new) < J(theta)) {
    theta <- theta_new
  }
}
```



Caveat: local and global minima

- Linear problems: unique minium
- Nonlinear problems: (typically) several local minima



Most minimization algorithms only guarantee **convergence to a local minimum**.

Minimization algorithms

Gradient-based minimization algorithms

Two main classes of minimization algorithms:

1. **Gradient-based methods**
2. Gradient-free methods

Gradient-based methods:

- Are typically faster
- Require the objective function to be differentiable
- Can fail to converge

Examples:

- Steepest descent
- Newton
- Gauss-Newton
- Levenberg-Marquardt

Method of steepest descent

You want to go down the mountain into the valley as efficiently as possible.

The fog prevents you from seeing more than a few meters in every direction.

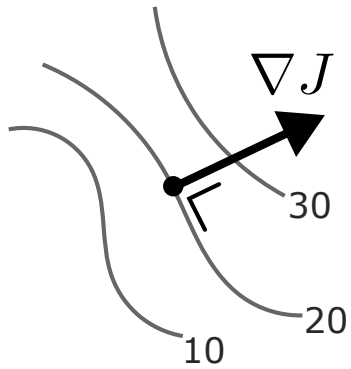
How do you proceed?

💡 Walk in the direction of **steepest descent**

Direction of steepest descent

Gradient:

- Perpendicular to level sets of J
- Direction of steepest ascent



To **decrease** $J(\theta)$, take a small step in direction of negative gradient:

$$
\begin{aligned}
s_k &= -\nabla J(\theta^k) \\
&= -\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1}|_{\theta^k} \\ \frac{\partial J(\theta)}{\partial \theta_2}|_{\theta^k} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n}|_{\theta^k} \end{bmatrix}.
\end{aligned}
$$

Method of steepest descent: Algorithm

Algorithm:

- Compute gradient $\nabla J(\theta^k)$ at current value θ^k.
- Follow negative gradient to update θ^k:

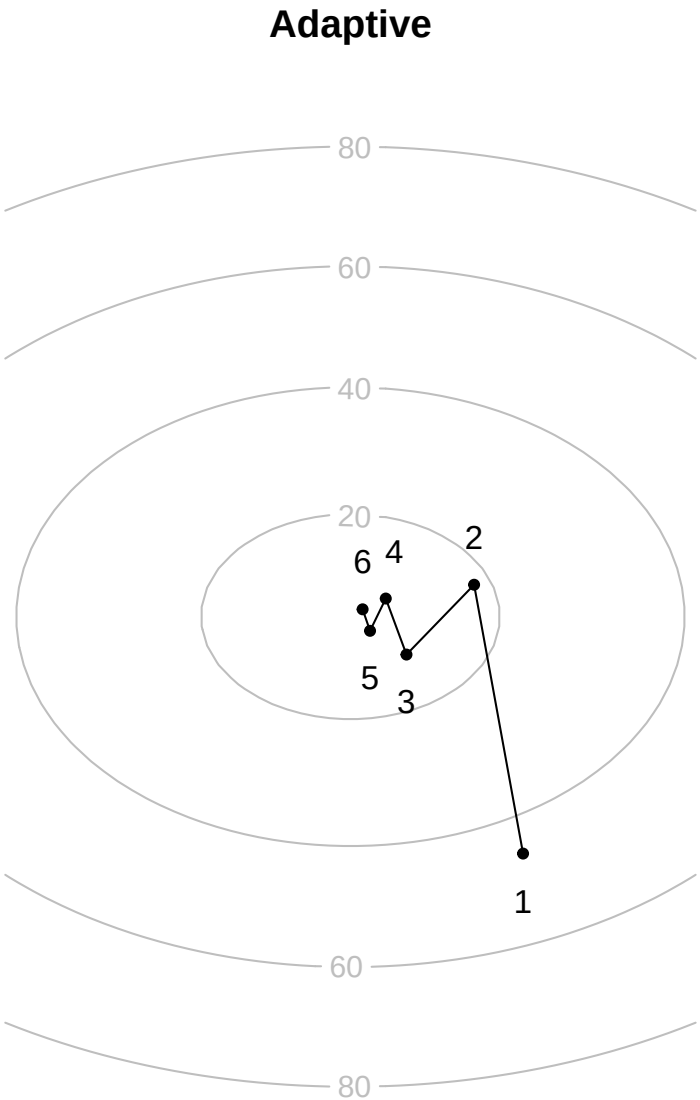
$$\theta^{k+1} = \theta^k - \alpha_k \nabla J(\theta^k),$$

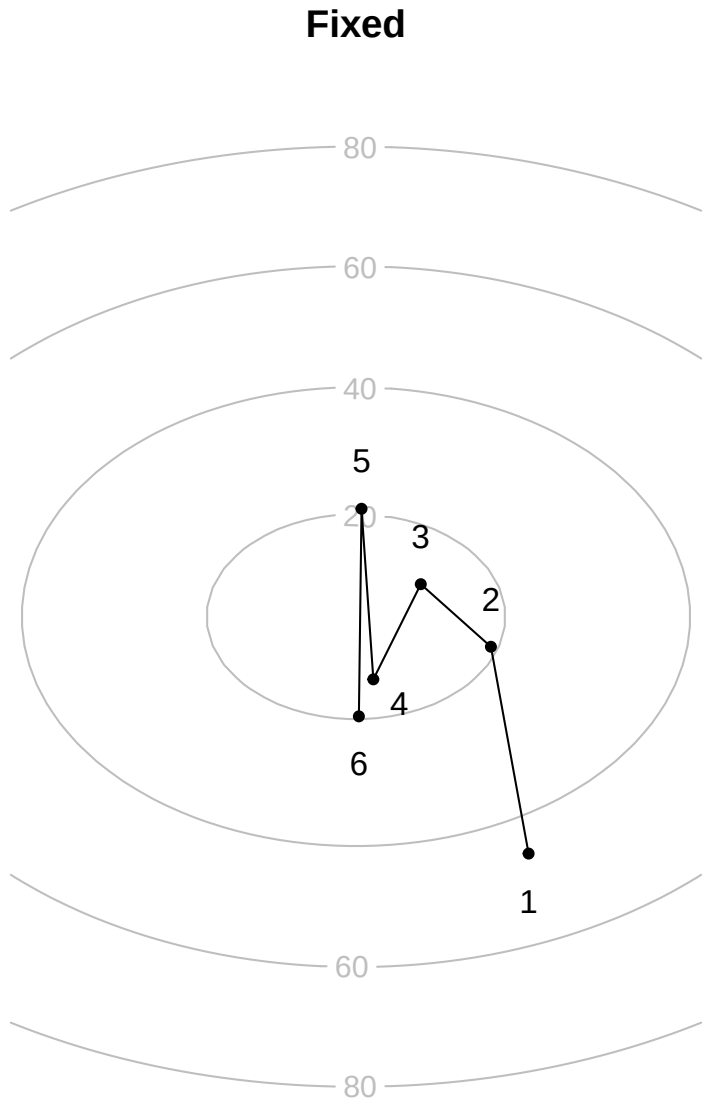
  with α_k the step size.
- Repeat until convergence

Step size α_k can be

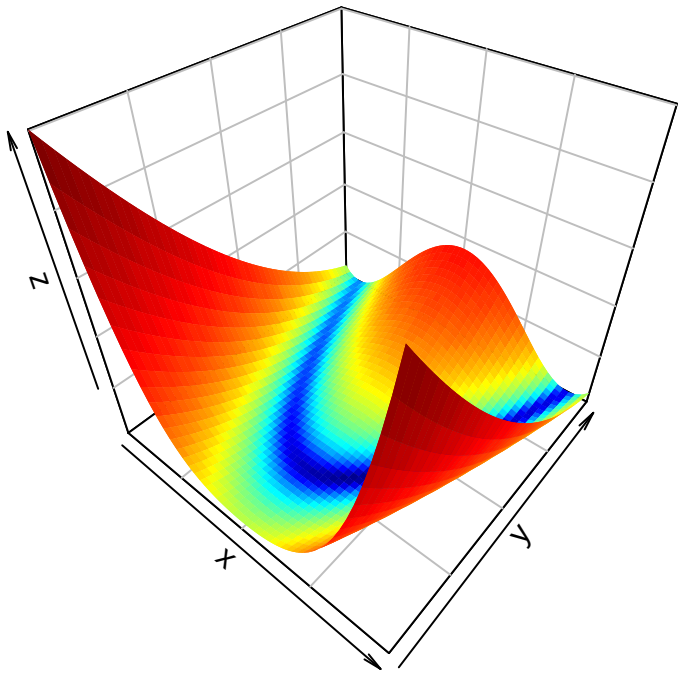
- Fixed: $\alpha_k = \alpha$ for a small fixed α (e.g. $\alpha = 0.01$).
- Adaptive: determine the best α_k at each step.

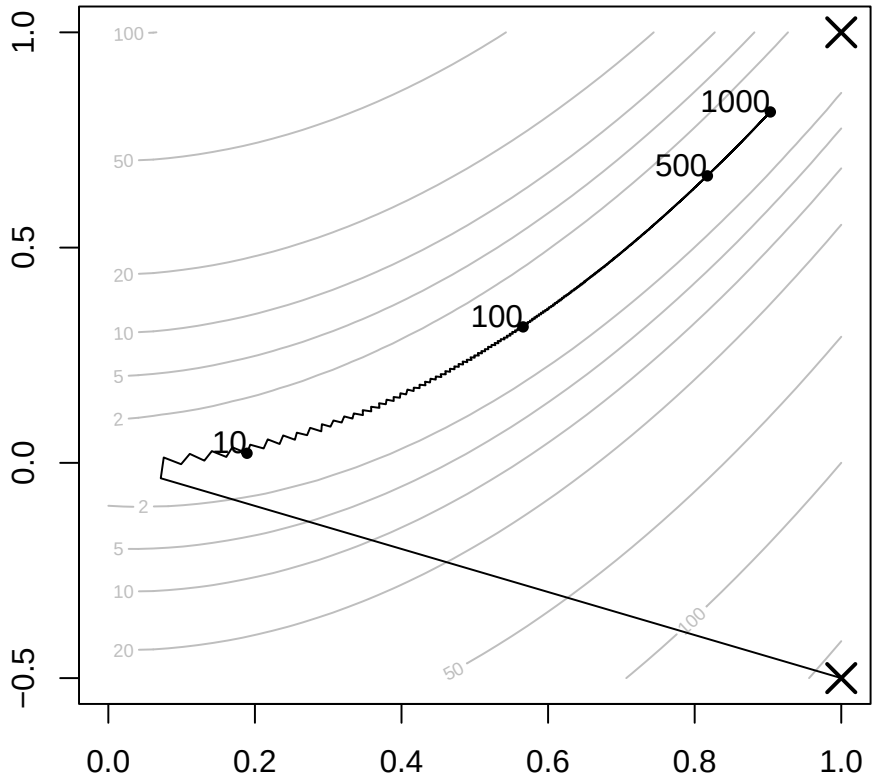
Method of steepest descent: variable step size





Method of steepest descent: disadvantages

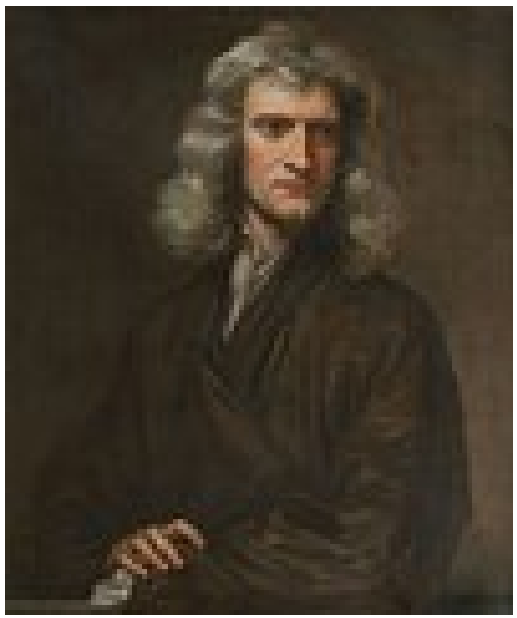




- Convergence can be slow (e.g for minimum hidden inside narrow “valley”)
- Steepest descent path will zigzag towards minimum, making little progress at each iteration.

Method of Newton: 1D case

Find a minimum of $J(\theta)$ by solving $J'(\theta) = 0$.



- For a starting point θ_k, look for a search direction s_k such that $J'(\theta_k + s_k) \approx 0$.
- Taylor: $J'(\theta_k + s_k)$ is approximately

$$J'(\theta_k + s_k) \approx J'(\theta_k) + s_k J''(\theta_k).$$

- Search direction:

$$s_k = -\frac{J'(\theta_k)}{J''(\theta_k)}$$

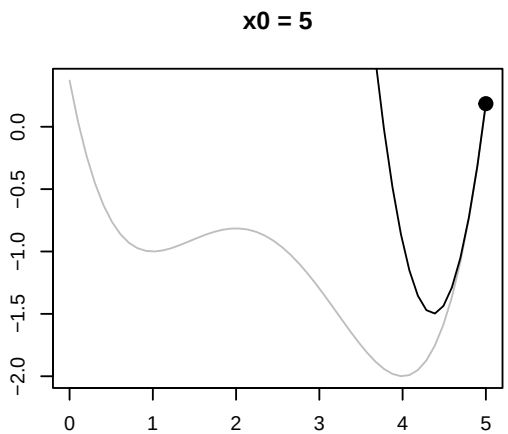
Uses information from **first** and **second** derivatives.

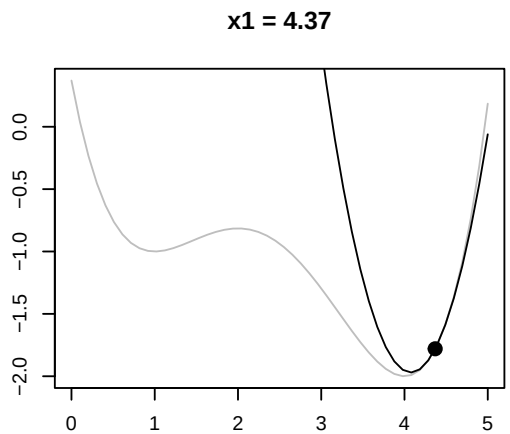
Method of Newton: properties

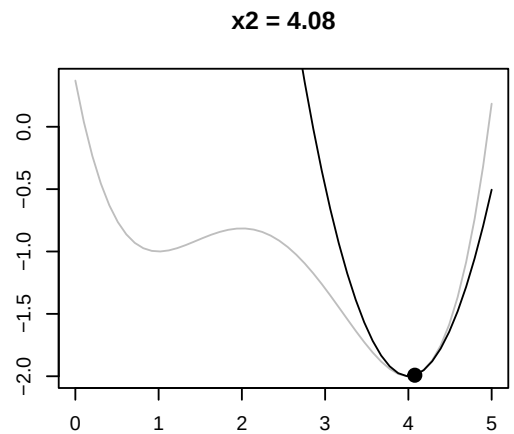
For a quadratic function $J(x) = Ax^2 + Bx + C$, Newton's method finds the minimum in **one step**.

Geometric interpretation:

- Approximate $J(x)$ around x_k by best-fitting parabola.
- Jump to bottom of parabola to find x_{k+1}.
- Repeat!







Method of Newton: higher dimensions

Search direction uses gradient and **Hessian**

$$s_k = -\left[H(\theta^k)\right]^{-1} \nabla J(\theta^k)$$

where

$$H(\theta^k) = \nabla^2 J(\theta^k) = \begin{bmatrix} \frac{\partial^2 J(\theta)}{\partial \theta_1^2}\big|_{\theta^k} & \frac{\partial^2 J(\theta)}{\partial \theta_1 \partial \theta_2}\big|_{\theta^k} & \cdots & \frac{\partial^2 J(\theta)}{\partial \theta_1 \partial \theta_n}\big|_{\theta^k} \\ \frac{\partial^2 J(\theta)}{\partial \theta_2 \partial \theta_1}\big|_{\theta^k} & \frac{\partial^2 J(\theta)}{\partial \theta_2^2}\big|_{\theta^k} & \cdots & \frac{\partial^2 J(\theta)}{\partial \theta_2 \partial \theta_n}\big|_{\theta^k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\theta)}{\partial \theta_n \partial \theta_1}\big|_{\theta^k} & \frac{\partial^2 J(\theta)}{\partial \theta_n \partial \theta_2}\big|_{\theta^k} & \cdots & \frac{\partial^2 J(\theta)}{\partial \theta_n^2}\big|_{\theta^k} \end{bmatrix}$$

- In practice, not necessary to invert $H(\theta)$
- Still requires $\mathcal{O}(D^2)$ computation at each step (expensive)

Method of Newton: advantages and disadvantages

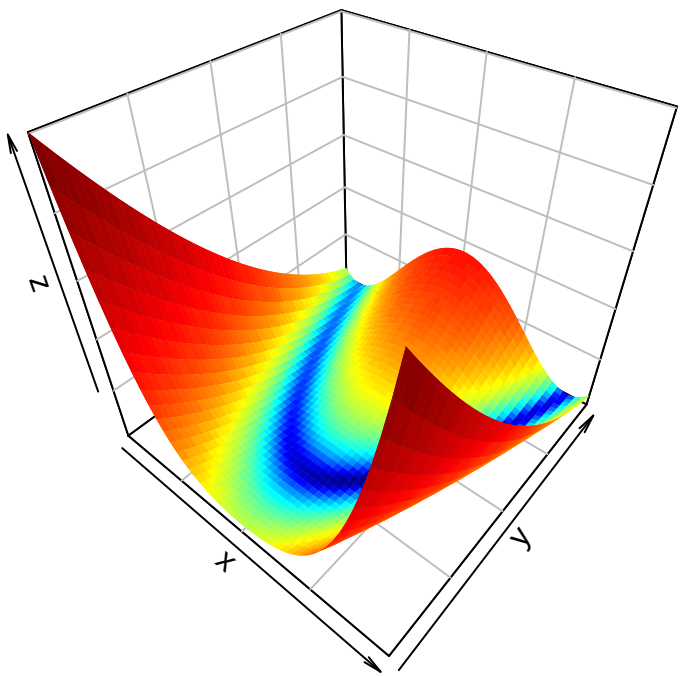
Advantages:

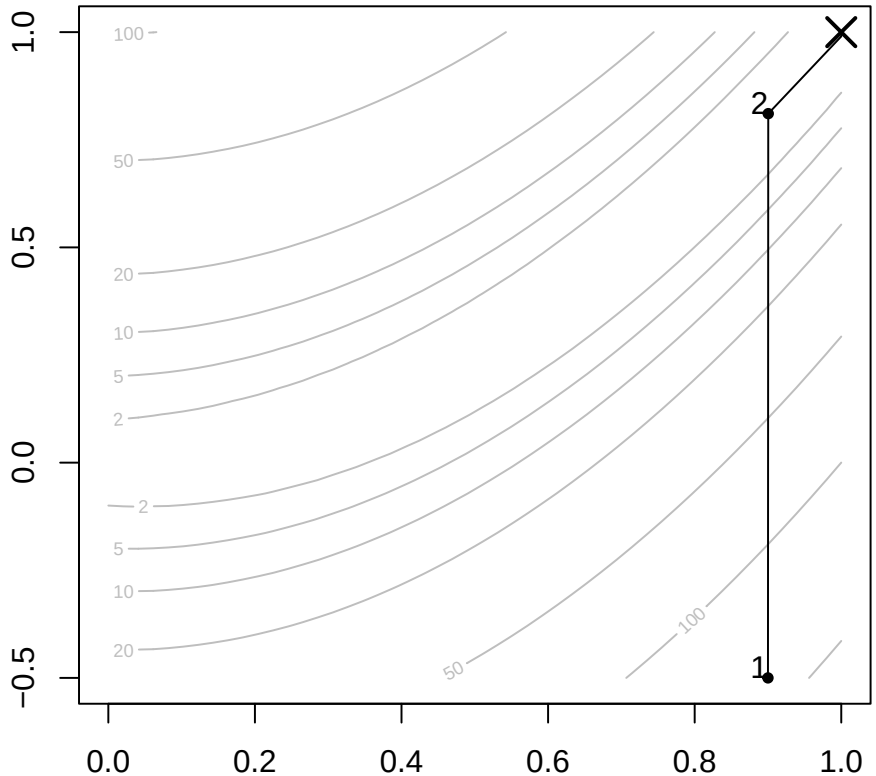
- Less iterations needed
- Choice direction more efficient: descent and curvature

Disadvantages:

- More sensitive to local extrema
- First **and** second order differentials
- Step size $\alpha = 1$. If initial vector too far from minimum, method will often not converge to minimum.

Method of Newton: convergence





- Very fast convergence for Rosenbrock function (3 iterations)
- In general: **quadratic convergence**

Many advanced gradient-based methods exist

- Broyden-Fletcher-Goldfarb-Shanno (BFGS): approximation of Hessian
- Levenberg-Marquardt: very popular, combines
  - Steepest descent: robust but slow
  - Method of Newton: fast, but often not convergent
- Powell/Brent: search along set of directions

Optimization in R

Use `optim(par, fn)`, where

- `par`: initial guess
- `fn`: the function to optimize
- `method`: "Nelder-Mead" (default), "BFGS", "Brent", ⋯

Worked-out example: M. merluccius

1. Define the objective function:

```
J <- function(theta, x, y) {
  resid <- y - beverton_holt(x, theta)
  return(sum(resid^2))
}
```

2. Specify the initial parameters:

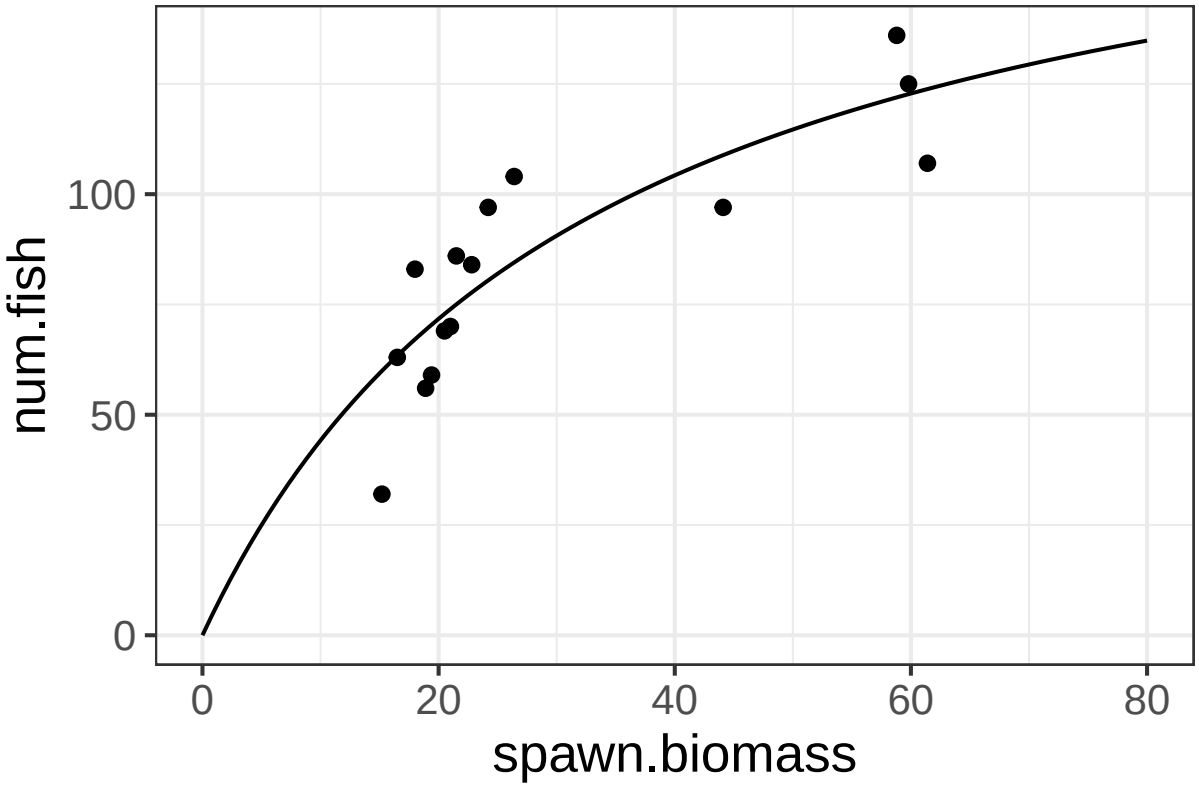
```
theta0 <- c(6, 20)
```

❸ Run the optimizer

```
fit <- optim(theta0, J,
             method = "BFGS",
             x = M.merluccius$spawn.biomass,
             y = M.merluccius$num.fish)
fit
```

```
$par
[1]  5.751024 33.157154

$value
[1] 2809.001

$counts
function gradient
      45       15

$convergence
[1] 0

$message
NULL
```

4. Evaluate the fit



More sophisticated ways to look at the fit will come later.

Gradient-free minimization algorithms

Two main classes of minimization algorithms:

1. Gradient-based methods
2. **Gradient-free methods**

Gradient-free methods:

- Are typically slower
- Can work even if the objective function is not differentiable
- Are more robust

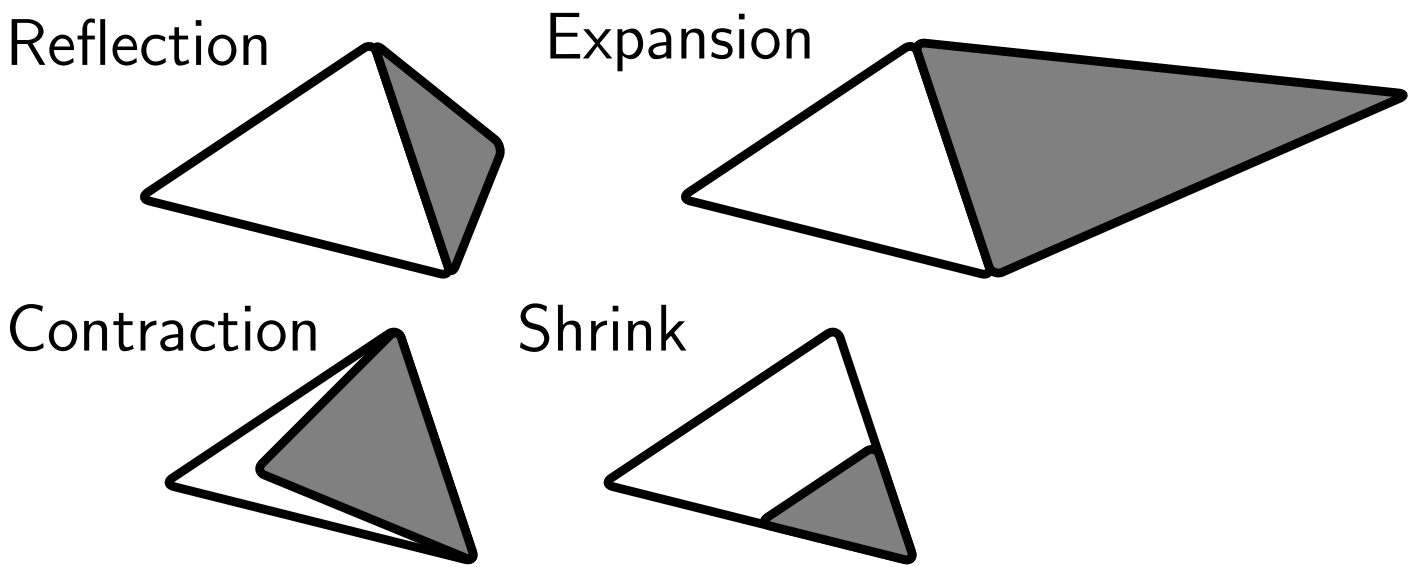
Examples:

- Direction set (Powell, Brent)
- Simplex
- Global minimisation

Simplex algorithm (Nelder-Mead 1965)

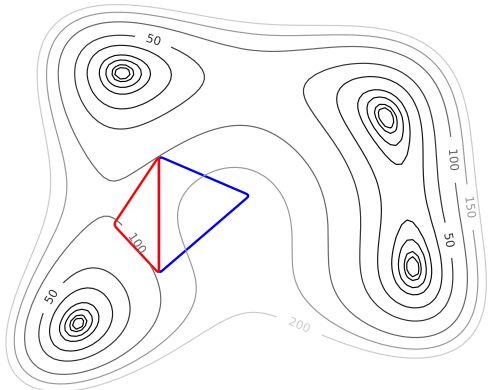
Basic idea: Capture optimal value inside simplex (triangle, pyramid, ...)

- Start with random simplex.
- Adjust worst corner of simplex by using different "actions".
- Repeat until convergence.

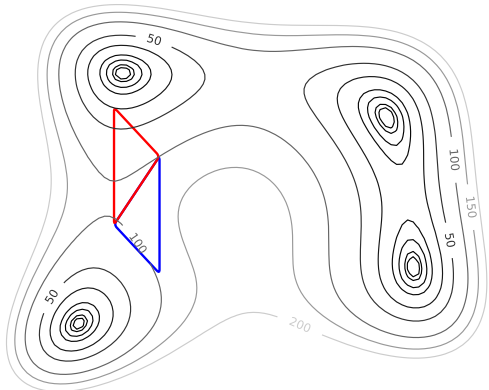


Simplex algorithm

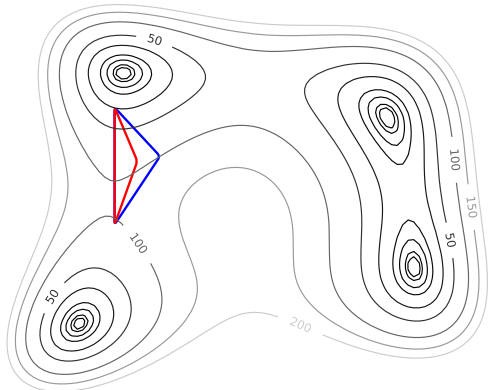
1. Outside contraction



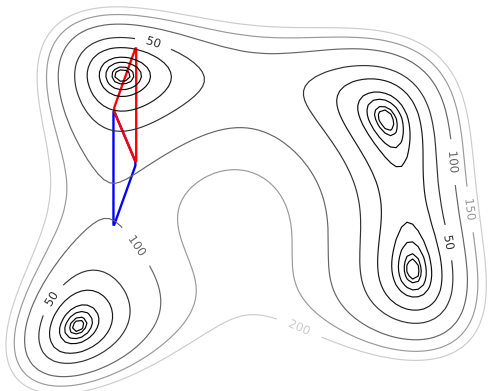
2. Expansion



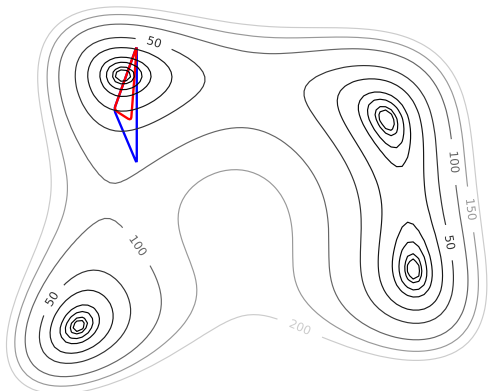
3. Inside contraction



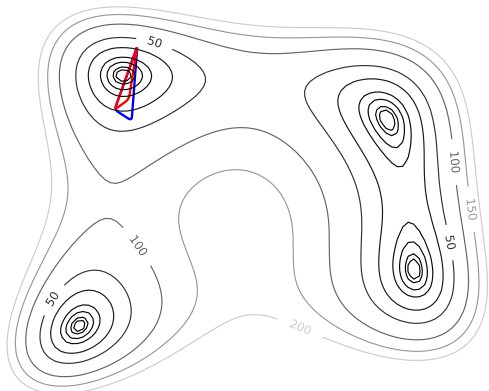
4. Reflection



5. Inside contraction



6. Inside contraction



Simplex algorithm: advantages and disadvantages

- Does not require gradient, Hessian, $\cdots$ information
- Robust: often finds a minimum where other optimizers cannot.
- Can find a rough approximation of a minimum in just a few updates$\cdots$
- $\cdots$ but may take a long time to converge completely.

Example: M. mercullius

```
fit <- optim(theta0, J,
             method = "Nelder-Mead",
             x = M.merluccius$spawn.biomass,
             y = M.merluccius$num.fish)
fit$par
```

```
[1]  5.751348 33.153782
```

```
fit$count
```

```
function gradient
      73       NA
```

Compared to BFGS:

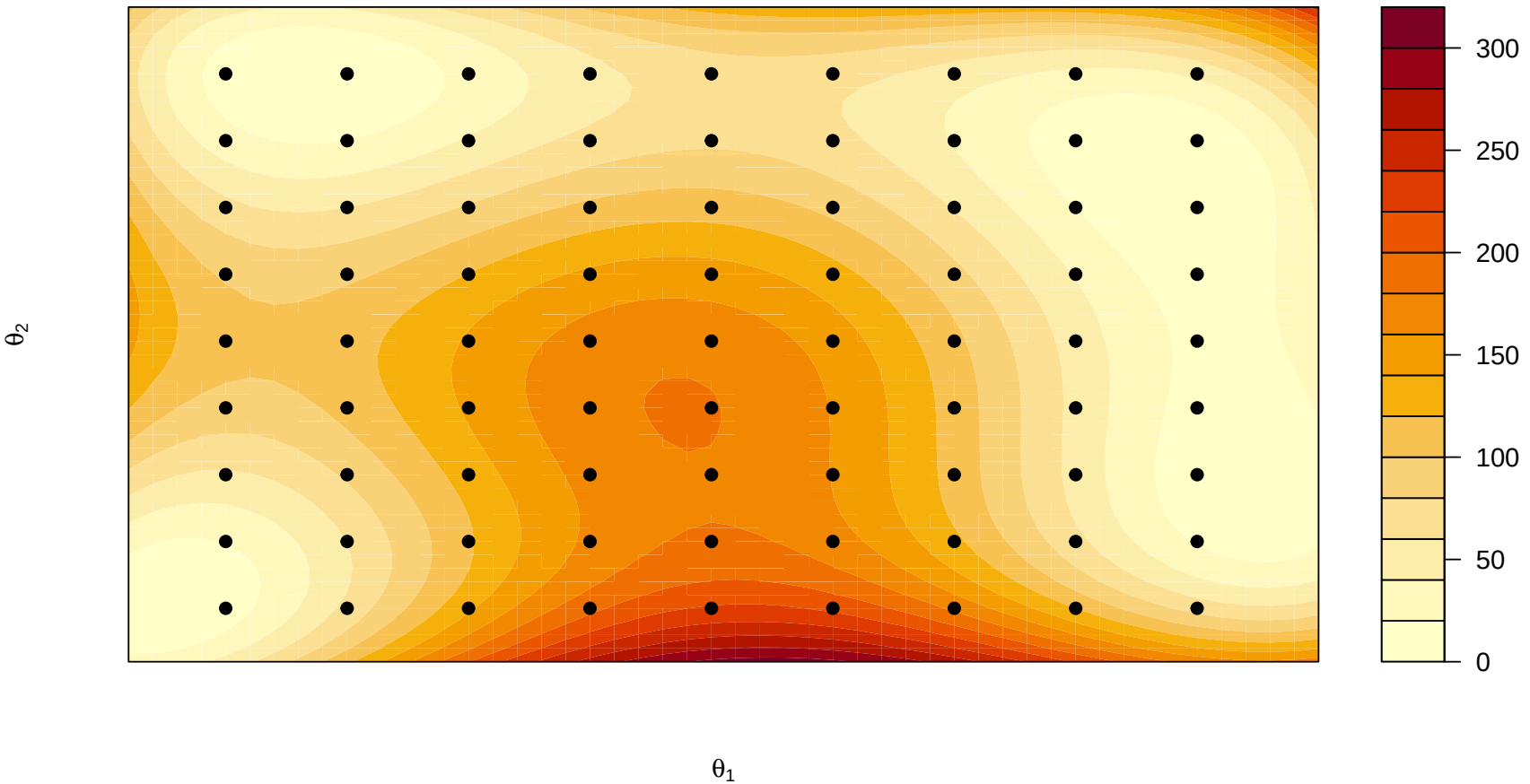
- Almost same parameter values
- More function evaluations, no gradient evaluations

Global minimization

- Disadvantage local techniques: local minima can never be completely excluded
- Global techniques insensitive to this problem
- Disadvantage: needs a lot of evaluations of J
- Types:
  - Gridding
  - Random methods

Global minimisation: Gridding

- Evaluate J for a grid of parameter values θ
- Select minimum among grid values



Global minimisation: Gridding

The finer the grid:

- the more likely to find the optimum,
- BUT the more calculations needed

Iterative:

- Start with a coarse-grained grid
- Refine parameter domain and repeat

Brute force, inefficient

Global minimisation: Random methods

Evaluate J for random parameter sets

- Choose PDF for each parameter
- Random sampling; Latin hypercube sampling

Retain

- Optimal set (with J_{min})
- Some sets below certain critical value (J_{crit})

Examples:

- Genetic algorithms
- Shuffled complex evolution
- Ant colony optimization
- Particle swarm optimization
- Simulated annealing
- ...

Assessing the quality of a fit

Residuals

Model:

$$y = f(x; \theta) + \epsilon$$

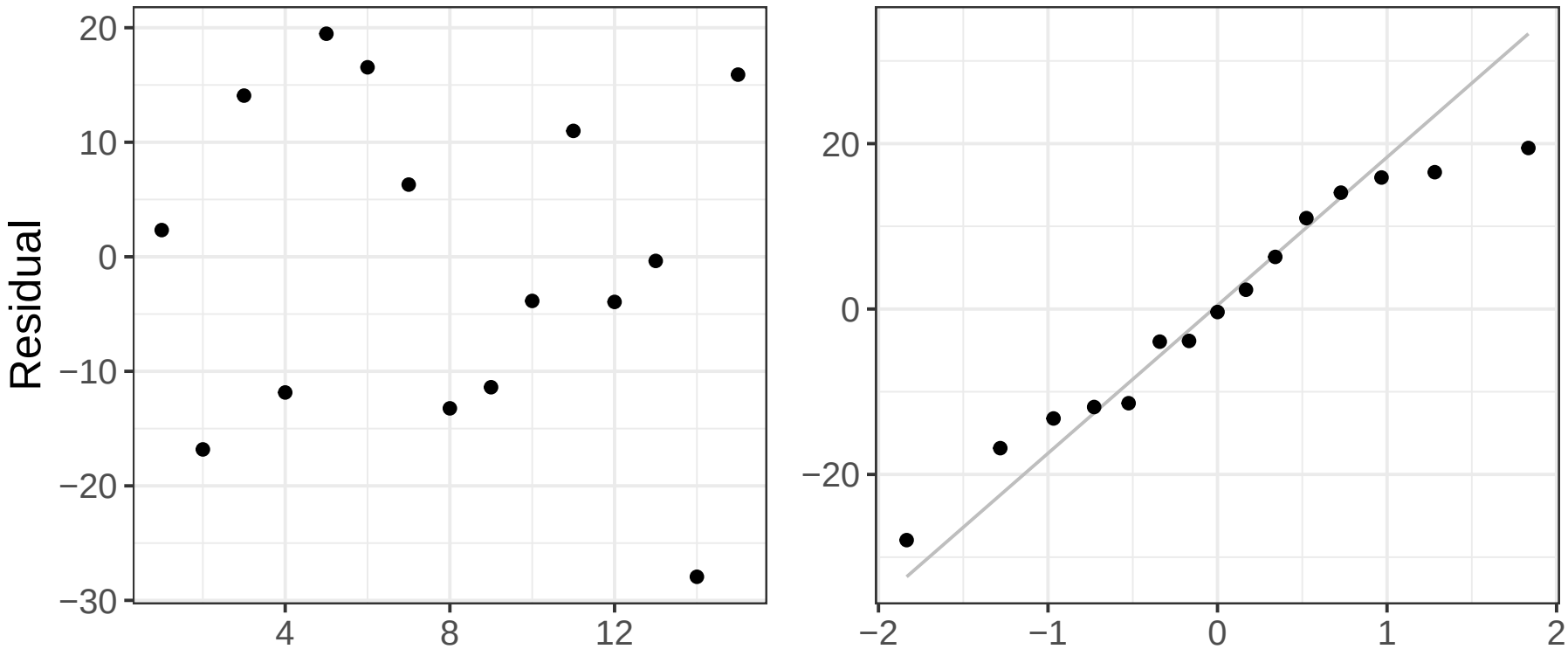
where ϵ is normally distributed.

If the model is well-fit, the residuals $e_i = y_i - f(x_i; \theta)$ should be

- Independent
- Normally distributed with mean 0 and constant variance.

Can be checked with QQ-plot of residuals

Example: M. mercullius



No pattern in residuals + normality: model appears well-fit.

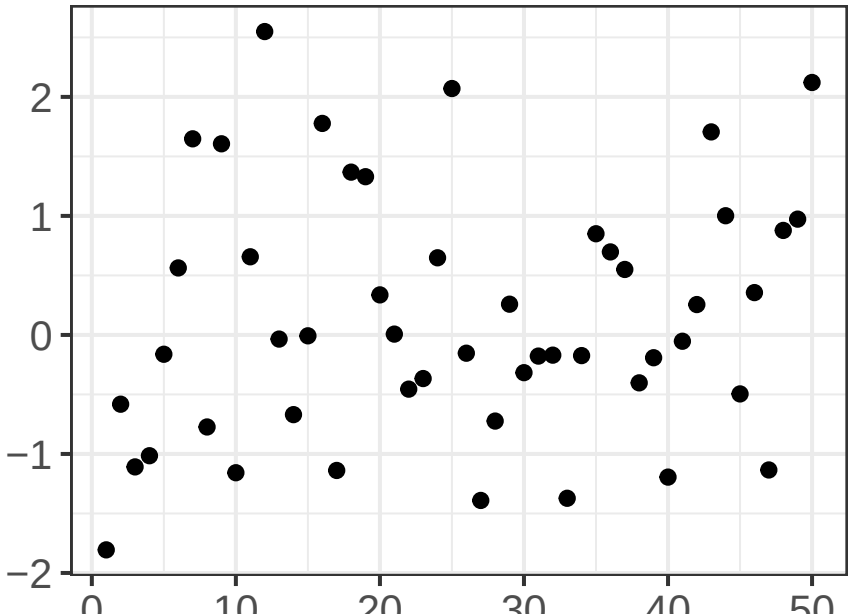
Correlations in time series (Optional)

Residuals: correlation and independence

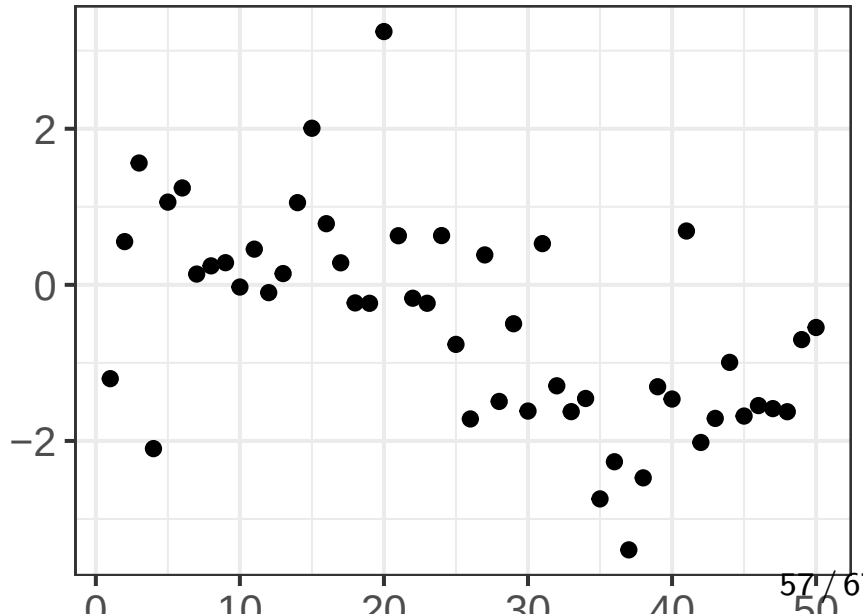
- We often assume that residuals are independent. But this is not always the case, especially in **time series**.
- Correlations in residuals are often a sign that something is missing from model fit.

How can we detect patterns, correlations, ⋯ in residuals?

Random residuals



Correlated residuals



Autocorrelation: how are residuals related?

Autocorrelation with lag τ answers the following questions:

- To what extent does a residual depend on a previous residual?
- Is there correlation between residuals in time?

$$r_\varepsilon(\tau) = \frac{1}{r_\varepsilon(0)} \sum_{k=1}^{N-\tau} \frac{\varepsilon(t_k) \cdot \varepsilon(t_k + \tau)}{N - \tau}$$

where $r_\varepsilon(0) = \sum_{k=1}^{N} \frac{\varepsilon^2(t_k)}{N}$

Detecting significant autocorrelations

If data is uncorrelated, then autocorrelation is normally distributed:

$$r_\varepsilon(\tau) \sim \mathcal{N}\left(0, \frac{1}{N}\right).$$

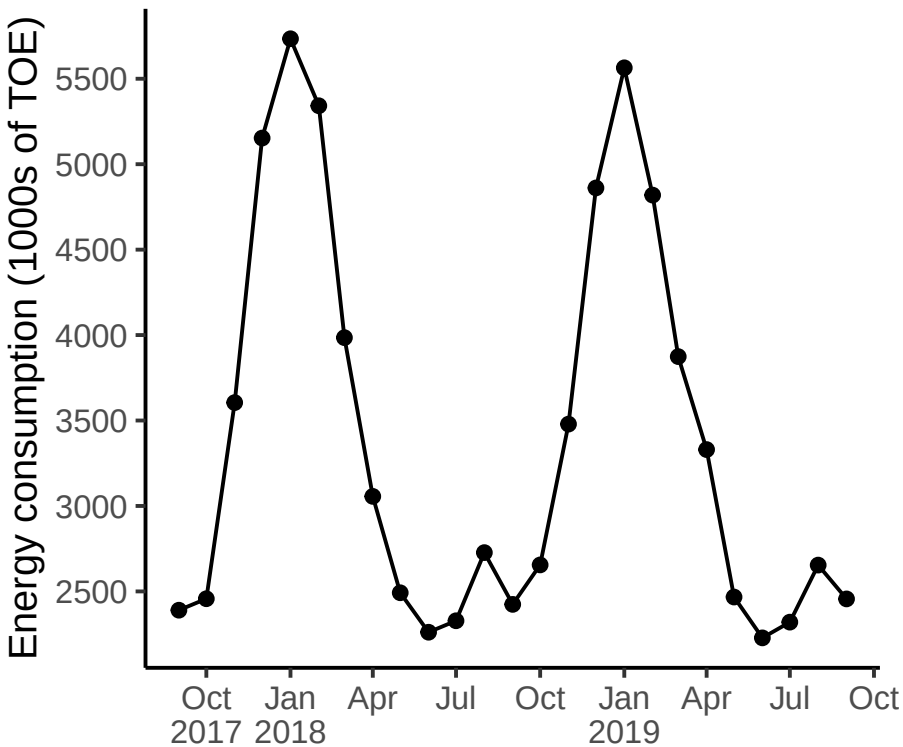
Can be used to detect "abnormally high" correlations:

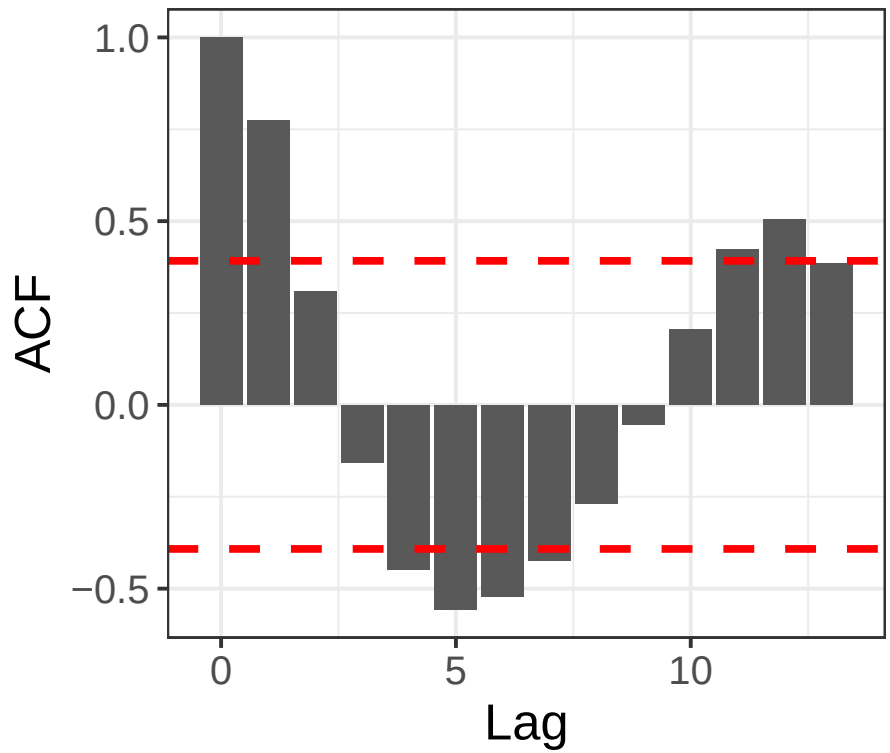
- Only about 5% of values outside range $\pm 1.96/\sqrt{N}$.
- If more, sign that data is correlated.

Example: Energy consumption in Korea (2017-19)

Autocorrelation uncovers repeating patterns in signal:

- Highly correlated over 12-month basis
- Anticorrelated over 6-month basis





Source: Korea Energy Economics Institute.

How to deal with correlations in residuals?

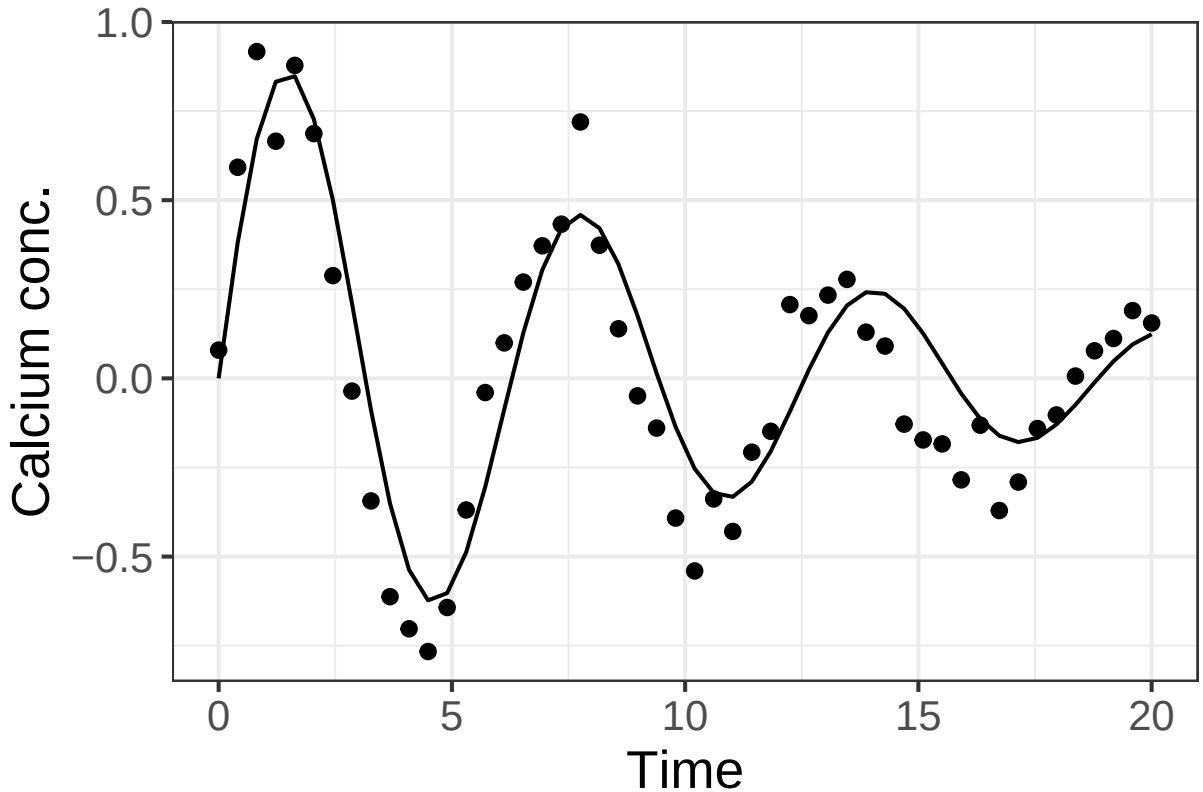
- **Make model bigger**: next slides
- Subsample data to reduce strength of correlations: not recommended
- Use modelling technique that does not need uncorrelated residuals (e.g. autoregressive models): outside scope of this course

Example: Calcium flows (simulated data)

Over the course of exercise, calcium ions flow in and out of the muscle cells. On biological grounds, model calcium concentration as exponentially damped sine:

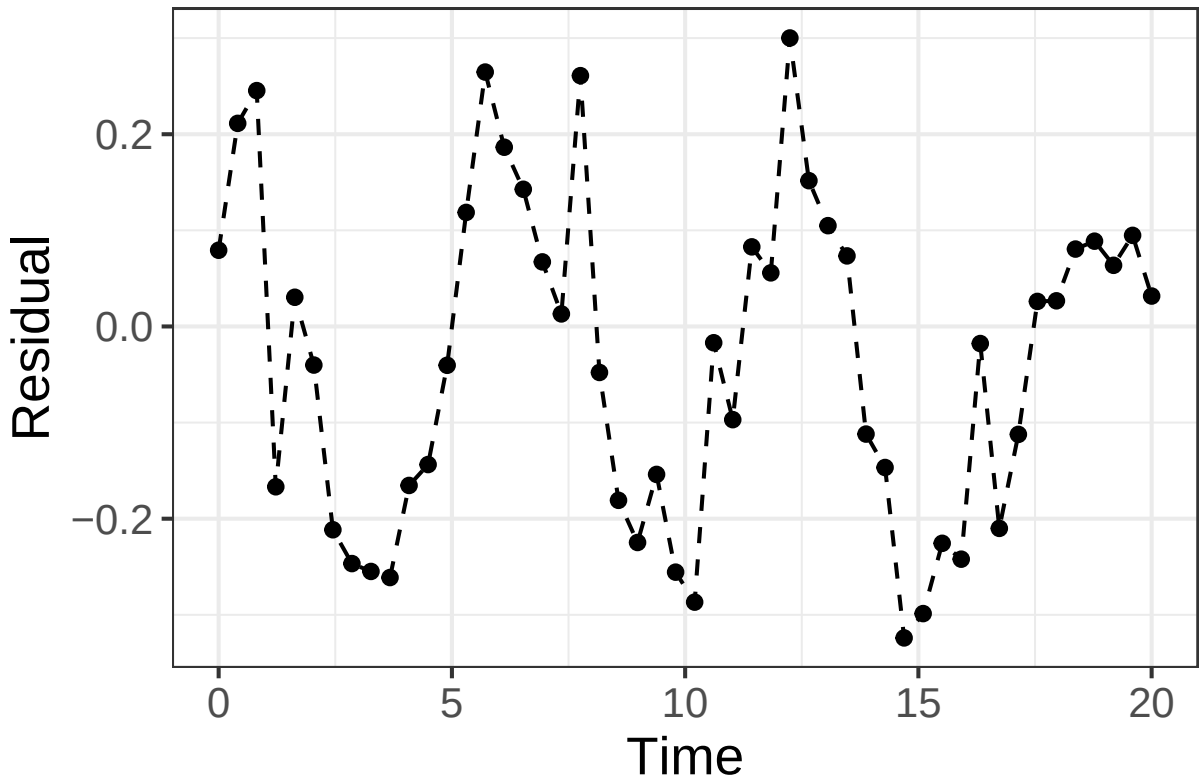
$$C(t) = \exp(-At)\sin(t)$$

Data and model fit:



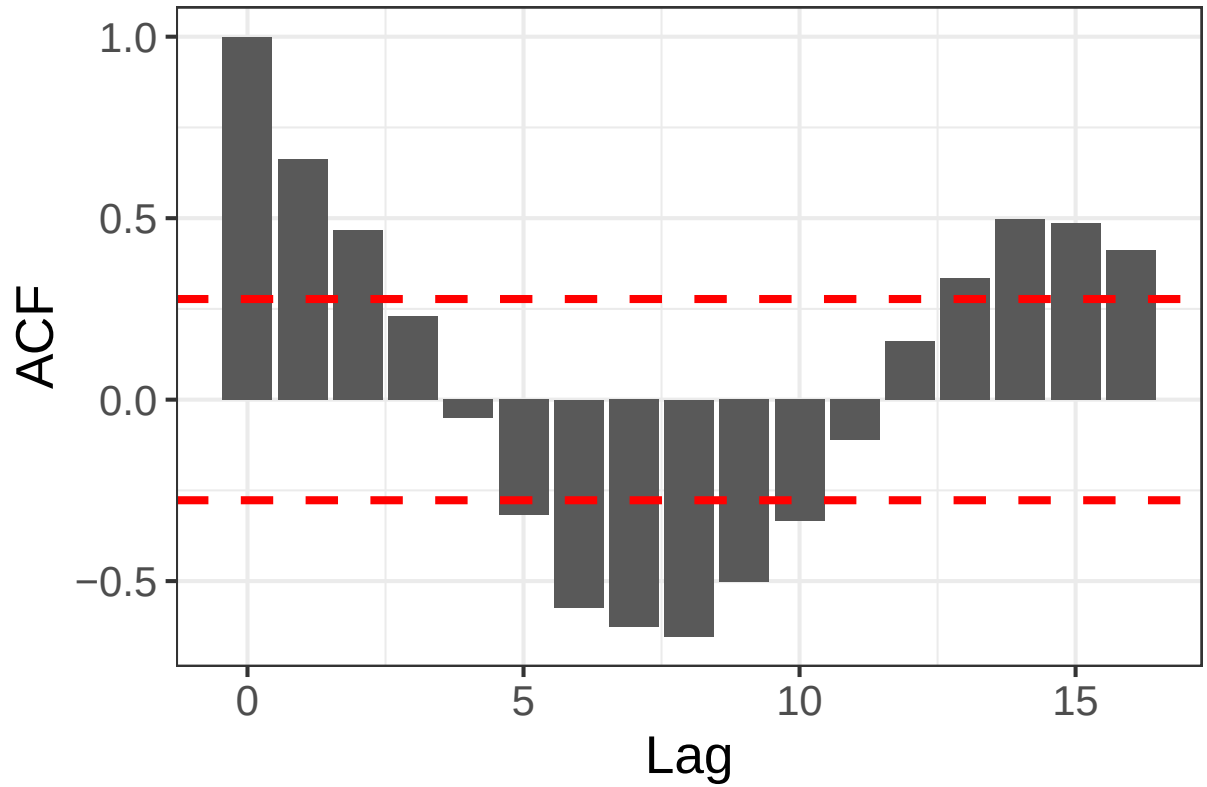
Residual plot

Model fit is good, but not perfect. Clear **repeating pattern** in the residuals.



Autocorrelation plot

Lack of model fit, repeating pattern in the residuals can also be seen from the autocorrelation plot.

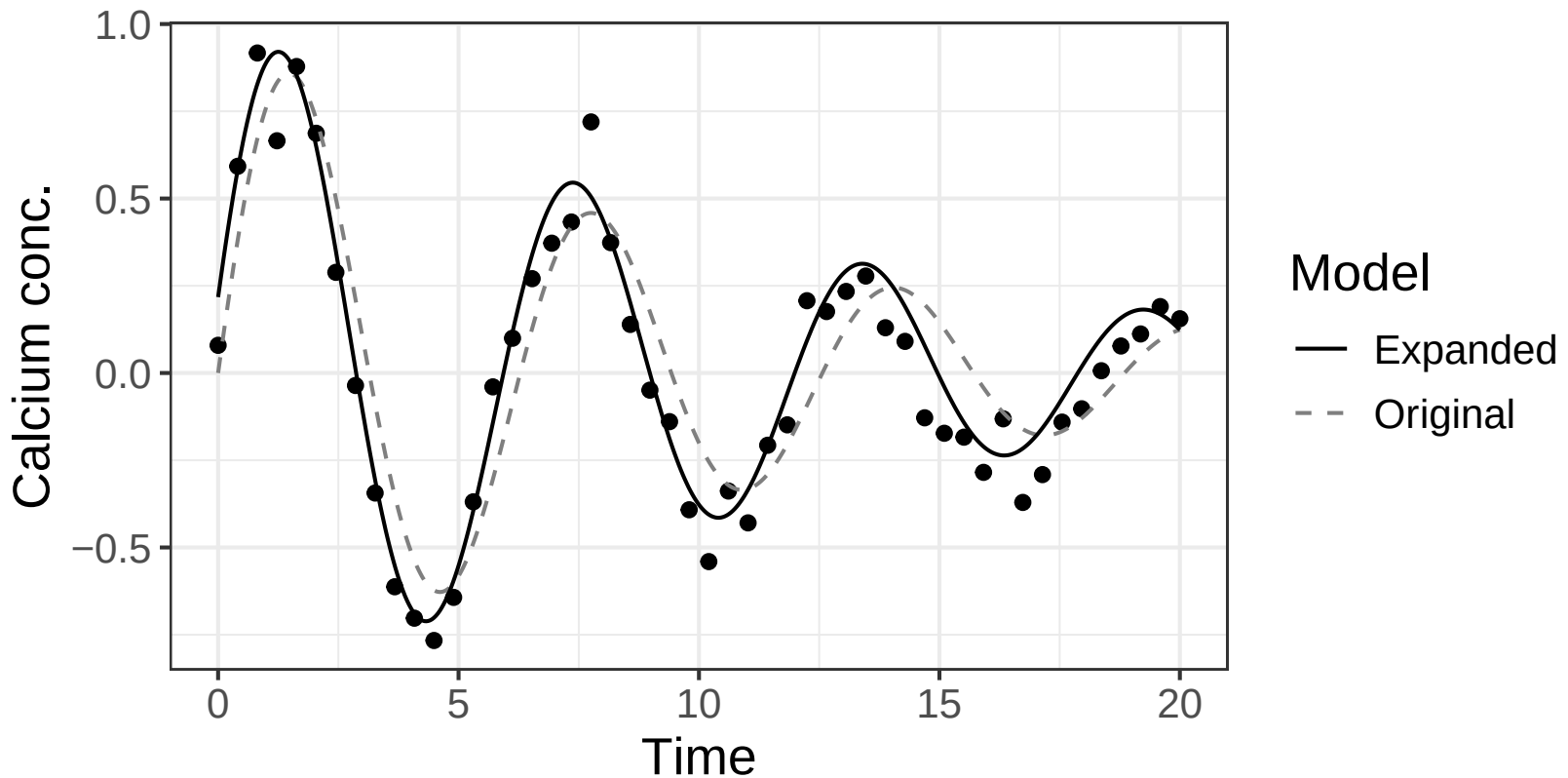


- Red lines: thresholds $1.96/\sqrt{50} = 0.227$.
- 13 out of 17 autocorrelations (76%) exceed threshold

Expanding the model

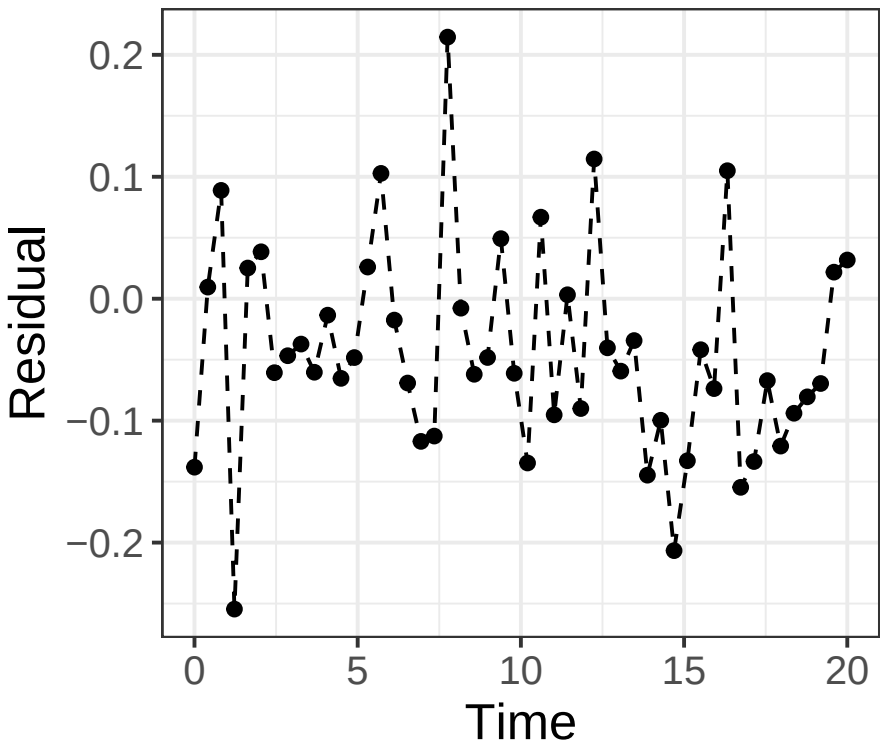
Pattern in residuals is a clear sign that **something is missing** in our modelling approach. Given the periodic oscillations, propose

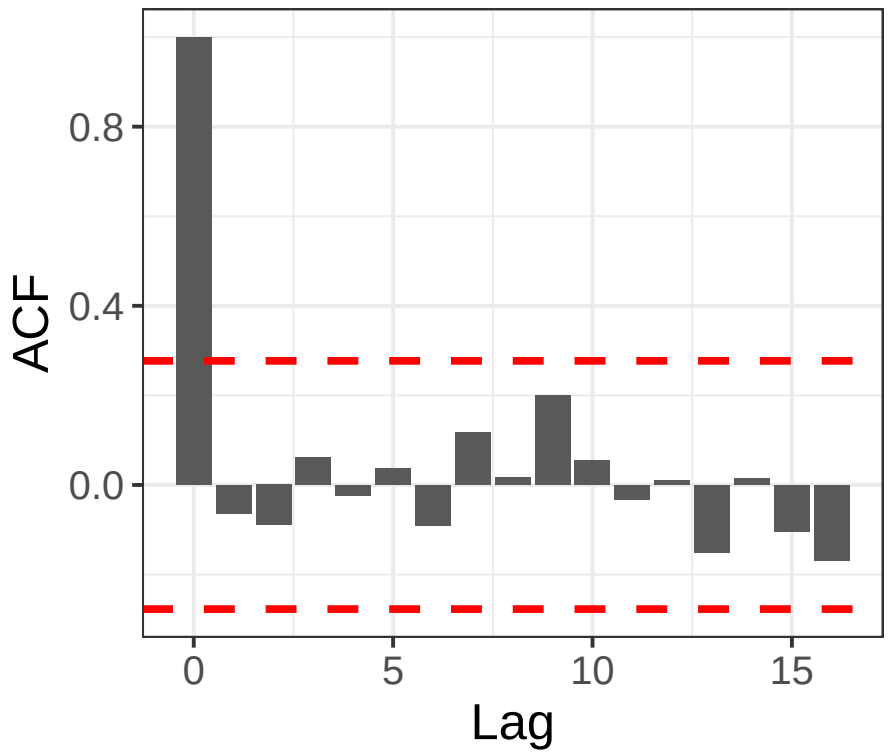
$$C(t) = \exp(-At)\sin(t) + B\cos(\omega t).$$



Residual and autocorrelation plot

No residual pattern visible in residuals. The model is well fit.





Residual QQ-plots