UNIVERSITÄT BIELEFELD
Faculty of Business Administration and Economics

# Initialization of Numerical Optimization in R

## Advertising the {ino} Package

Lennart Oelschläger   Marius Ötting   Dietmar Bauer

Bielefeld University, Department of Empirical Methods

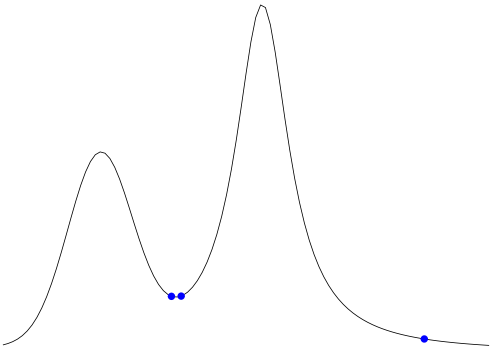
7 September 2023

# Outline

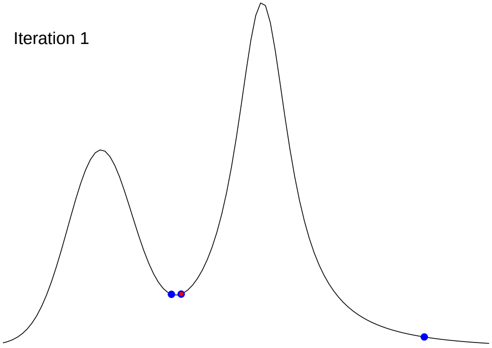
1. What does initialization of numerical optimization mean?
2. Why should we as statisticians care?
3. We implemented the {ino} toolbox in R
4. Three package demonstrations
5. Takeaways and outlook

# Outline

Numerical optimization of $f$ means iteratively adjusting $x$ to find the *globally* best $f(x)$.
And most algorithms need to start at some user-defined initial value $x_0$.



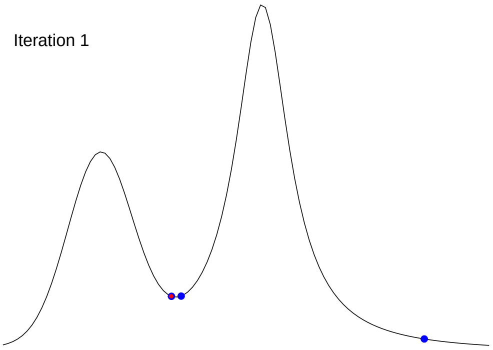
Numerical optimization of $f$ means iteratively adjusting $x$ to find the *globally* best $f(x)$.
And most algorithms need to start at some user-defined initial value $x_0$.




Depending on $x_0$, the optimization

- works fine 🙂
-
-

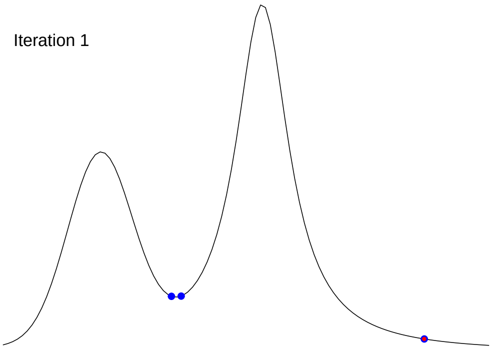
Numerical optimization of $f$ means iteratively adjusting $x$ to find the *globally* best $f(x)$.
And most algorithms need to start at some user-defined initial value $x_0$.




Depending on $x_0$, the optimization

- works fine
- does not work
-

Numerical optimization of $f$ means iteratively adjusting $x$ to find the *globally* best $f(x)$.
And most algorithms need to start at some user-defined initial value $x_0$.

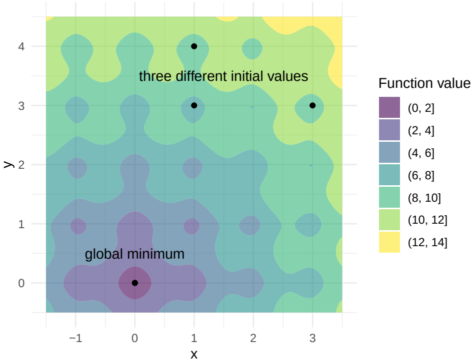



Depending on $x_0$, the optimization

- works fine 🙂
- does not work ☹️
- takes an eternity 😴

The Ackley function is a challenging test for optimizers:

$$f(x,y) = -20\exp\left(-0.2\sqrt{(x^2+y^2)/2}\right) - \exp\left(\frac{\cos(2\pi x)+\cos(2\pi y)}{2}\right) + 20 + \exp(1)$$



# Outline

# Maximum likelihood estimation

We seek to globally optimize log-likelihood functions:

- Probit model

$$f(\boldsymbol{\theta} = (\beta, \Sigma) \mid X, y) = \sum_{n,j} 1(y_n = j) \log \Phi_{0,\Delta_j \Sigma \Delta_j'}(-\Delta_j X_n \beta)$$

- Hidden Markov model

$$f(\boldsymbol{\theta} = (\Gamma, \mu, \sigma, \delta) \mid X, N) = \log \delta P(x_1 \mid \mu, \sigma, N) \Gamma P(x_2 \mid \mu, \sigma, N) \cdots \Gamma P(x_T \mid \mu, \sigma, N) 1'$$

- Gaussian mixture model (two classes)

$$f(\boldsymbol{\theta} = (\pi, \mu, \sigma) \mid X) = \sum_{n} \log \left( \pi \phi_{\mu_1, \sigma_1^2}(x_n) + (1 - \pi) \phi_{\mu_2, \sigma_2^2}(x_n) \right)$$

# Maximum likelihood estimation

We seek to globally optimize log-likelihood functions:

- Probit model

$$f(\boldsymbol{\theta} = (\beta, \Sigma) \mid X, y) = \sum_{n,j} 1(y_n = j) \log \Phi_{0,\Delta_j \Sigma \Delta_j'}(-\Delta_j X_n \beta)$$

- Hidden Markov model

$$f(\boldsymbol{\theta} = (\Gamma, \mu, \sigma, \delta) \mid X, N) = \log \delta P(x_1 \mid \mu, \sigma, N) \Gamma P(x_2 \mid \mu, \sigma, N) \cdots \Gamma P(x_T \mid \mu, \sigma, N) 1'$$

- Gaussian mixture model (two classes)

$$f(\boldsymbol{\theta} = (\pi, \mu, \sigma) \mid X) = \sum_n \log \left( \pi \phi_{\mu_1, \sigma_1^2}(x_n) + (1 - \pi) \phi_{\mu_2, \sigma_2^2}(x_n) \right)$$

They depend on observed data, which yields ideas for initialization strategies.

# Outline

# The {ino} package

An R package to compare

1. any optimizer for any real-valued function
2. different initialization strategies

Designed to be user-friendly

1. only a single R6 object
2. every user input is validated
3. detailed function documentation and vignettes

And some more convenience, like parallel optimization, progress bar, standardized outputs of optimizers, time limit for long optimizations, trace of optimization path, plots, etc.

```r
ackley <- TestFunctions::TF_ackley
ackley(c(0, 0))
```

```
## [1] 4.440892e-16
```

```r
ackley <- TestFunctions::TF_ackley
ackley(c(0, 0))
```

```
## [1] 4.440892e-16
```

```r
Nop$new(f = ackley, npar = 2)$
  set_optimizer(optimizer_nlm())$
  initialize_fixed(list(c(1, 4), c(1, 3), c(3, 3)))$
  optimize()$
  summary(c("value", "parameter", "initial"), digits = 2)
```

```
##   value  parameter initial
## 1  2.58 0.95, 0.00    1, 4
## 2  0.00       0, 0    1, 3
## 3  6.56 1.97, 1.97    3, 3
```

# Outline

# Speed up probit likelihood maximization

First package demonstration: fitting a probit model.

# Speed up probit likelihood maximization

First package demonstration: fitting a probit model. The probit model

- connects covariates $X_n$ to discrete choices $y_n$
- via latent utilities $U_n$ that are defined as
  1. a linear function $V_n = X_n\beta$
  2. plus a Gaussian error $\varepsilon_n$

$$U_n = V_n + \varepsilon_n$$
$$V_n = X_n\beta$$
$$\varepsilon_n \sim N(0, \Sigma)$$
$$y_n = \arg\max U_n$$

# Speed up probit likelihood maximization

First package demonstration: fitting a probit model. The probit model

- connects covariates $X_n$ to discrete choices $y_n$
- via latent utilities $U_n$ that are defined as
  1. a linear function $V_n = X_n\beta$
  2. plus a Gaussian error $\varepsilon_n$

$$U_n = V_n + \varepsilon_n$$
$$V_n = X_n\beta$$
$$\varepsilon_n \sim N(0, \Sigma)$$
$$y_n = \arg\max U_n$$

The probit likelihood sums multiple Gaussian CDFs.

$\Rightarrow$ Optimization is time-consuming.

$\Rightarrow$ Good initial values can save time in comparison to random initialization.

# Speed up probit likelihood maximization

First package demonstration: fitting a probit model. The probit model

- connects covariates $X_n$ to discrete choices $y_n$
- via latent utilities $U_n$ that are defined as
  1. a linear function $V_n = X_n\beta$
  2. plus a Gaussian error $\varepsilon_n$

$$
\begin{aligned}
U_n &= V_n + \varepsilon_n \\
V_n &= X_n\beta \\
\varepsilon_n &\sim N(0, \Sigma) \\
y_n &= \arg\max U_n
\end{aligned}
$$

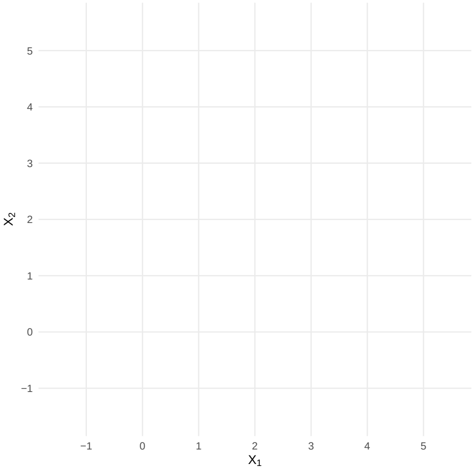
The probit likelihood sums multiple Gaussian CDFs.

$\Rightarrow$ Optimization is time-consuming.

$\Rightarrow$ Good initial values can save time in comparison to random initialization.

We developed an initialization strategy that can be tested via {`ino`}.

# Speed up probit likelihood maximization

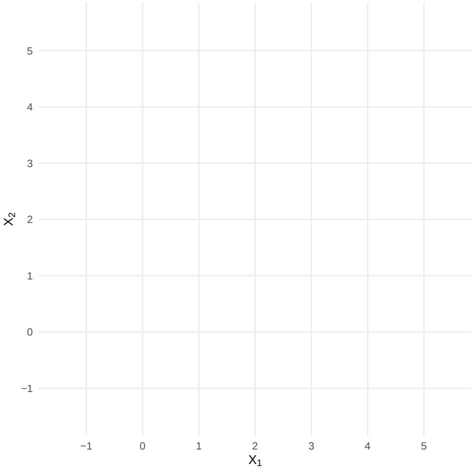



Assume

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

# Speed up probit likelihood maximization




Assume

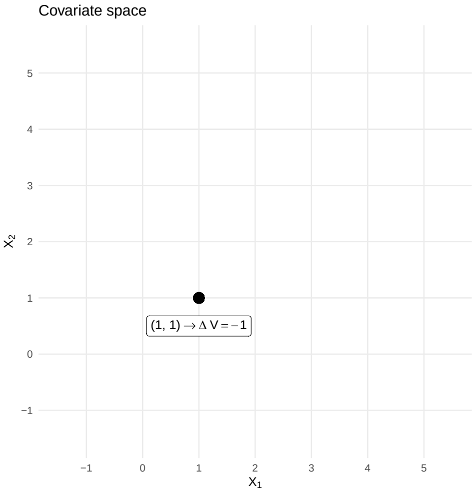
$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

Let

$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

# Speed up probit likelihood maximization




Assume

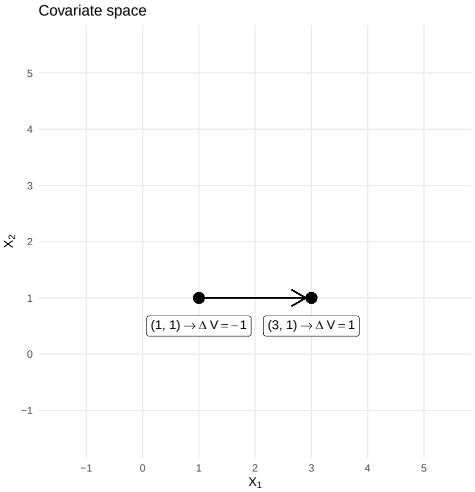
$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

Let

$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

# Speed up probit likelihood maximization




Assume

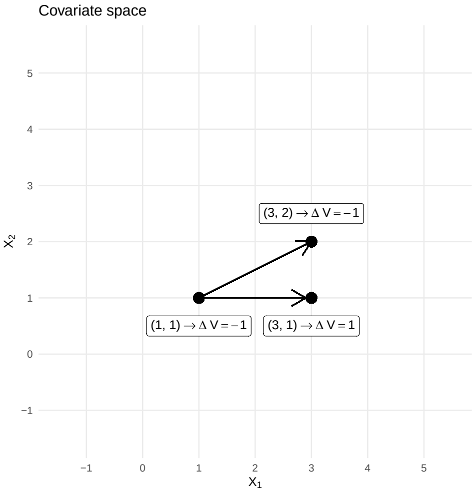
$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

Let

$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

# Speed up probit likelihood maximization




Assume

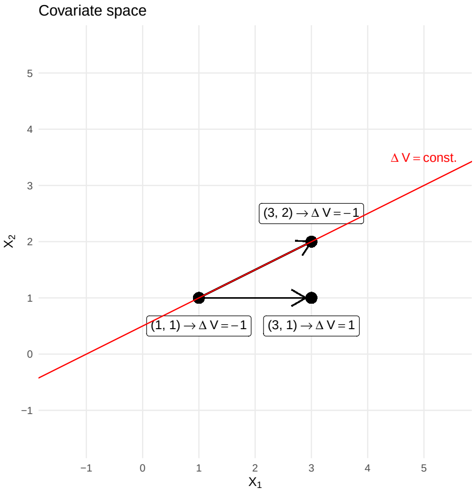
$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

Let

$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

# Speed up probit likelihood maximization




Assume

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

Let

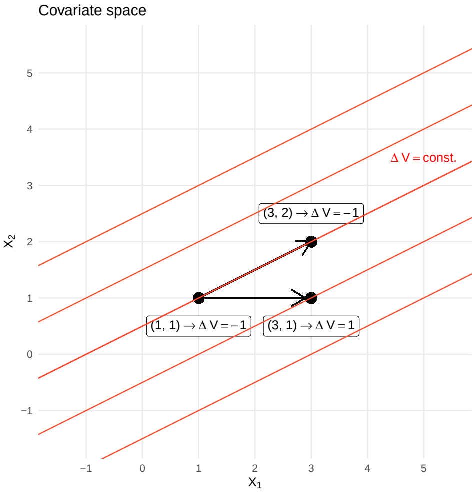
$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

We find the direction

$$\overrightarrow{\begin{pmatrix} 1 \\ 0.5 \end{pmatrix}} = \overrightarrow{\begin{pmatrix} 1/\beta_1 \\ 1/\beta_2 \end{pmatrix}}$$

in which $\Delta V = V_1 - V_2 = \text{const.}$

# Speed up probit likelihood maximization




Assume

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

Let

$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

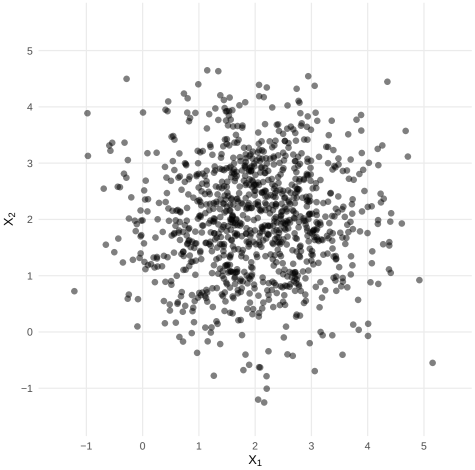
We find the direction

$$\overrightarrow{\begin{pmatrix} 1 \\ 0.5 \end{pmatrix}} = \overrightarrow{\begin{pmatrix} 1/\beta_1 \\ 1/\beta_2 \end{pmatrix}}$$

in which $\Delta V = V_1 - V_2 = \text{const.}$

# Speed up probit likelihood maximization

Simulated data with N = 1000




Assume

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

Let

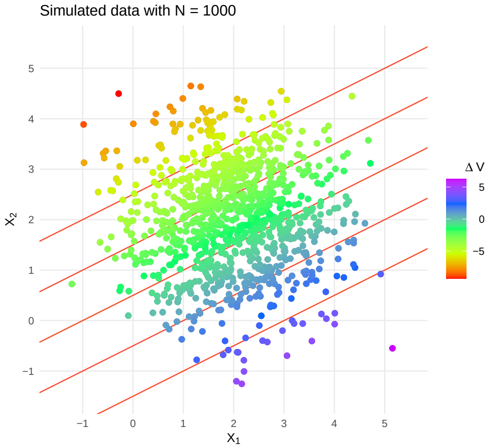
$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

We find the direction

$$\overrightarrow{\begin{pmatrix} 1 \\ 0.5 \end{pmatrix}} = \overrightarrow{\begin{pmatrix} 1/\beta_1 \\ 1/\beta_2 \end{pmatrix}}$$

in which $\Delta V = V_1 - V_2 = \text{const.}$

# Speed up probit likelihood maximization




Assume

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

(such a definition is typical for alternative-varying $X$ with alternative-specific $\beta$, e.g., travel time)

Let

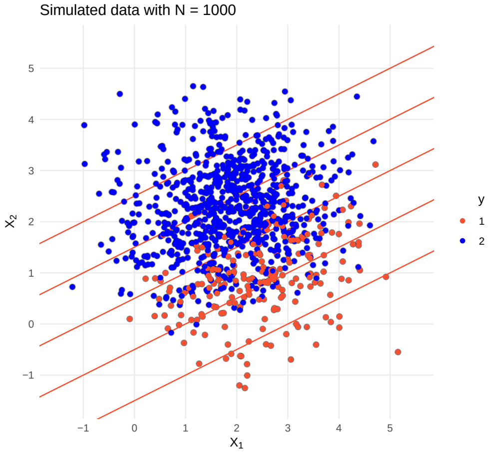
$$\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

We find the direction

$$\overrightarrow{\begin{pmatrix} 1 \\ 0.5 \end{pmatrix}} = \overrightarrow{\begin{pmatrix} 1/\beta_1 \\ 1/\beta_2 \end{pmatrix}}$$

in which $\Delta V = V_1 - V_2 = \text{const.}$

# Speed up probit likelihood maximization

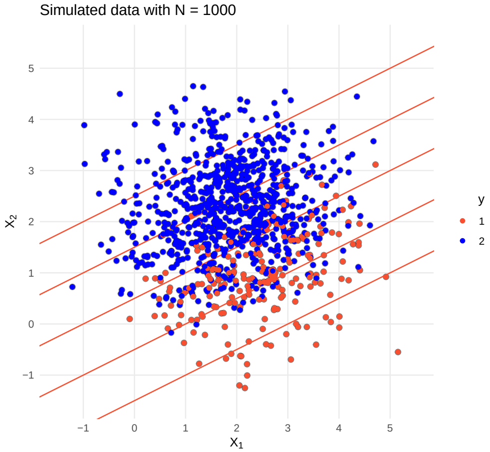



We do not observe $\Delta V$

(depends on the unknown $\beta$)

but we observe the choices $y$.

# Speed up probit likelihood maximization




We do not observe $\Delta V$

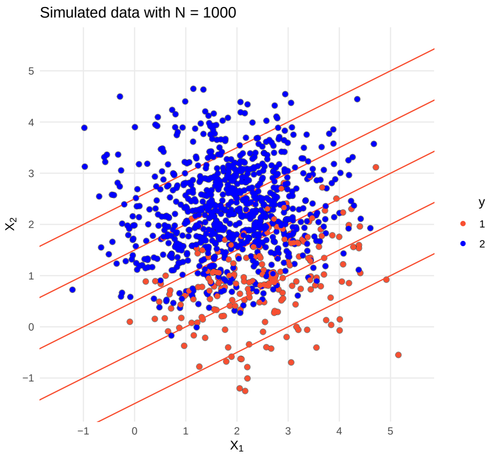
(depends on the unknown $\beta$)

but we observe the choices $y$.

They are disturbed by the error-term $\varepsilon$.

(here I used $\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$)

# Speed up probit likelihood maximization




We do not observe $\Delta V$

(depends on the unknown $\beta$)

but we observe the choices $y$.

They are disturbed by the error-term $\varepsilon$.

(here I used $\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$)

But we can still identify

$$\overrightarrow{\begin{pmatrix} 1/\beta_1 \\ 1/\beta_2 \end{pmatrix}}$$

as the kernel of $\text{Cor}(X, y)$.

(constant choice probability in this direction)

# Speed up probit likelihood maximization

This strategy can easily be tested with {ino}:

```
beta <- c(1, -1, 2)
Sigma <- matrix(c(2, 1, 0, 1, 2, 1, 0, 1, 2), 3)
J <- nrow(Sigma)
P <- length(beta)

Nop_probit <- Nop$new(f = logLik_probit, npar = 5, fix_beta1 = 1)$
  set_optimizer(optimizer_nlm())

for (i in 1:100) {

  probit_data <- sim_mnp(
    N = 200, J = J, P = P, beta = beta, Sigma = Sigma,
    X = function(n) diag(stats::rnorm(P))
  )
  direction <- find_direction(probit_data, fix_beta1 = 1)

  Nop_probit$
    argument("set", data = probit_data)$
    initialize_random()$
    optimize(optimization_label = "random", which_direction = "max")$
    initialize_fixed(at = c(direction, stats::rnorm(J * (J - 1) / 2)))$
    optimize(optimization_label = "strategy", which_direction = "max")
}
```

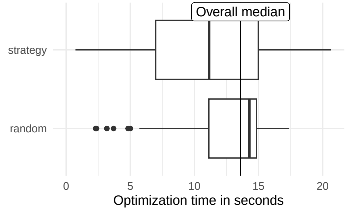
</> For the implementations of `logLik_probit` and `sim_mnp` see the {ino} probit application vignette. The implementation of `find_direction` can be provided on request.

# Speed up probit likelihood maximization

This strategy can easily be tested with {ino}:

```
beta <- c(1, -1, 2)
Sigma <- matrix(c(2, 1, 0, 1, 2, 1, 0, 1, 2), 3)
J <- nrow(Sigma)
P <- length(beta)

Nop_probit <- Nop$new(f = logLik_probit, npar = 5, fix_beta1 = 1)$
  set_optimizer(optimizer_nlm())

for (i in 1:100) {

  probit_data <- sim_mnp(
    N = 200, J = J, P = P, beta = beta, Sigma = Sigma,
    X = function(n) diag(stats::rnorm(P))
  )
  direction <- find_direction(probit_data, fix_beta1 = 1)

  Nop_probit$
    argument("set", data = probit_data)$
    initialize_random()$
    optimize(optimization_label = "random", which_direction = "max")$
    initialize_fixed(at = c(direction, stats::rnorm(J * (J - 1) / 2)))$
    optimize(optimization_label = "strategy", which_direction = "max")
}
```

```
Nop_probit$plot(group_by = ".optimization_label")
```



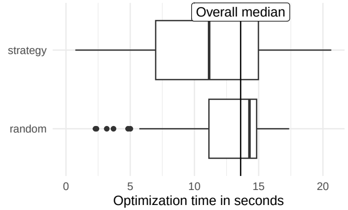

</> For the implementations of `logLik_probit` and `sim_mnp` see the {ino} probit application vignette. The implementation of `find_direction` can be provided on request.

# Speed up probit likelihood maximization

This strategy can easily be tested with {ino}:

```
beta <- c(1, -1, 2)
Sigma <- matrix(c(2, 1, 0, 1, 2, 1, 0, 1, 2), 3)
J <- nrow(Sigma)
P <- length(beta)

Nop_probit <- Nop$new(f = logLik_probit, npar = 5, fix_beta1 = 1)$
  set_optimizer(optimizer_nlm())

for (i in 1:100) {

  probit_data <- sim_mnp(
    N = 200, J = J, P = P, beta = beta, Sigma = Sigma,
    X = function(n) diag(stats::rnorm(P))
  )
  direction <- find_direction(probit_data, fix_beta1 = 1)

  Nop_probit$
    argument("set", data = probit_data)$
    initialize_random()$
    optimize(optimization_label = "random", which_direction = "max")$
    initialize_fixed(at = c(direction, stats::rnorm(J * (J - 1) / 2)))$
    optimize(optimization_label = "strategy", which_direction = "max")
}
```

```
Nop_probit$plot(group_by = ".optimization_label")
```



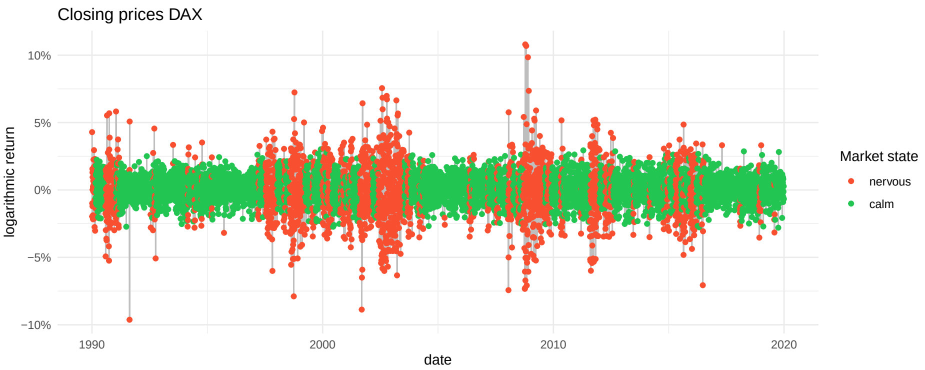

Conclusion: computation time reduced by 1/3

(will be more pronounced for more covariates / observations)

</> For the implementations of `logLik_probit` and `sim_mnp` see the {ino} probit application vignette. The implementation of `find_direction` can be provided on request.

# HMM for financial time series

Second package demonstration: fitting a two-state HMM.



# HMM for financial time series

HMMs are known to have multiple local optima, let's see how many we can find here:

```
Nop_hmm <- Nop$new(f = logLik_hmm, npar = 6, data = dax$logreturn, N = 2)$
  set_optimizer(optimizer_nlm())$
  initialize_random(runs = 100, seed = 1)$
  optimize(optimization_label = "random", which_direction = "max", reset_initial = FALSE)
```

</> For the implementation of `logLik_hmm` see the {ino} HMM application vignette.

# HMM for financial time series

HMMs are known to have multiple local optima, let's see how many we can find here:

```
Nop_hmm <- Nop$new(f = logLik_hmm, npar = 6, data = dax$logreturn, N = 2)$
  set_optimizer(optimizer_nlm())$
  initialize_random(runs = 100, seed = 1)$
  optimize(optimization_label = "random", which_direction = "max", reset_initial = FALSE)
```

Get an overview of the optima:

```
Nop_hmm$optima(digits = 0)
```

```
##   value frequency
## 1 22446        67
## 2 21372        32
## 3 20755         1
```

</> For the implementation of `logLik_hmm` see the {ino} HMM application vignette.

# HMM for financial time series

HMMs are known to have multiple local optima, let's see how many we can find here:

```
Nop_hmm <- Nop$new(f = logLik_hmm, npar = 6, data = dax$logreturn, N = 2)$
  set_optimizer(optimizer_nlm())$
  initialize_random(runs = 100, seed = 1)$
  optimize(optimization_label = "random", which_direction = "max", reset_initial = FALSE)
```

Get an overview of the optima:

```
Nop_hmm$optima(digits = 0)
```

```
##   value frequency
## 1 22446        67
## 2 21372        32
## 3 20755         1
```

Get the best parameter:

```
Nop_hmm$best("parameter", which_direction = "max",
             digits = 2)
```

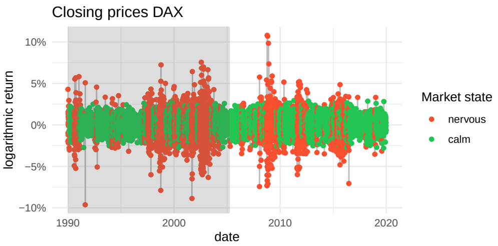
```
## [1] -4.52 -3.70  0.00  0.00 -3.86 -4.72
## attr(,".run_id")
## [1] 64
## attr(,".optimizer_label")
## [1] "stats::nlm"
```

</> For the implementation of `logLik_hmm` see the {ino} HMM application vignette.

# HMM for financial time series

A simple idea to reduce computation time:

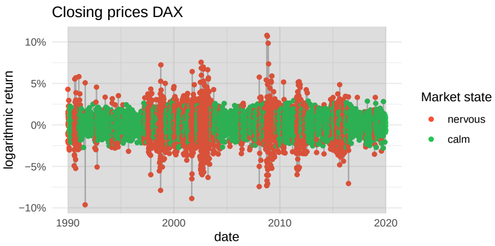
1. fit the HMM to the first 50% data



# HMM for financial time series

A simple idea to reduce computation time:

1. fit the HMM to the first 50% data
2. use the estimates as initial values for the estimation on the full data set




Will adding computation time from 1. and 2. be faster than full data estimation directly?

# HMM for financial time series

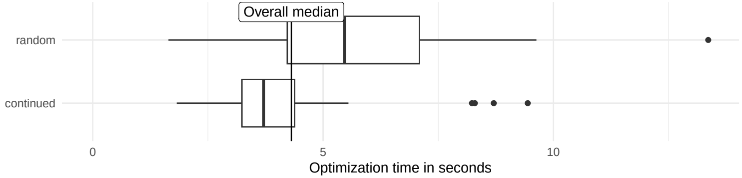
This idea can easily be tested with {ino}:

```
Nop_hmm$
  argument("subset", name = "data", how = "first", proportion = 0.5)$
  optimize(which_direction = "max")$
  argument("reset", name = "data")$
  initialize_continue()$
  optimize(optimization_label = "continued", which_direction = "max")$
  plot(group_by = ".optimization_label")
```

# HMM for financial time series

This idea can easily be tested with {ino}:

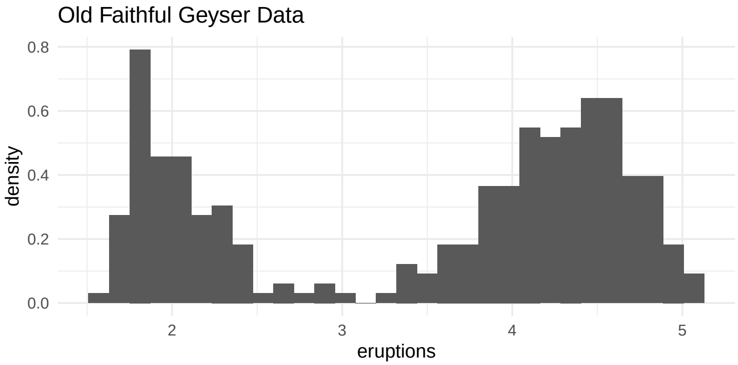
```
Nop_hmm$
  argument("subset", name = "data", how = "first", proportion = 0.5)$
  optimize(which_direction = "max")$
  argument("reset", name = "data")$
  initialize_continue()$
  optimize(optimization_label = "continued", which_direction = "max")$
  plot(group_by = ".optimization_label")
```




Conclusion: reduces computation time a bit (will probably be more pronounced for more complex models)

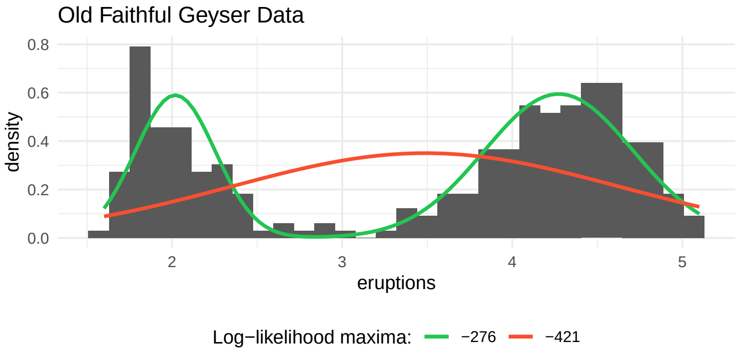
# The EM-algorithm for mixture models

Third and final package demonstration: fitting a two-class Gaussian mixture model.



# The EM-algorithm for mixture models

Third and final package demonstration: fitting a two-class Gaussian mixture model.




Depending on the initial value, either a sensible or a one-class solution is estimated.

# The EM-algorithm for mixture models

Two popular approaches:

1. Gradient-based optimization
2. EM-algorithm

Which one is faster and will reach -276 more often than -421?

</> For the implementations of `logLik_mixture` and `optimizer_em` see the {ino} introductory vignette.

# The EM-algorithm for mixture models

Two popular approaches:

1. Gradient-based optimization
2. EM-algorithm

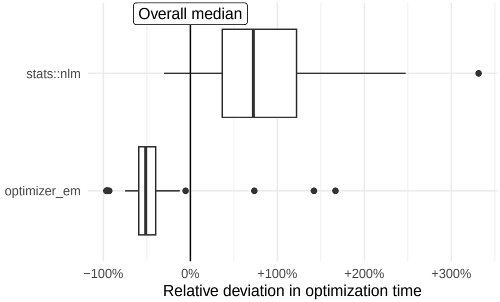
Which one is faster and will reach -276 more often than -421? Easy comparison with {ino}:

```
Nop_mixture <- Nop$new(
    f = logLik_mixture, npar = 5, data = faithful$eruptions
  )$
  set_optimizer(optimizer_nlm())$
  set_optimizer(optimizer_em)$
  initialize_random(sampler = function() rnorm(n = 5), runs = 100, seed = 1)$
  optimize(which_direction = "max")
```

</> For the implementations of logLik_mixture and optimizer_em see the {ino} introductory vignette.

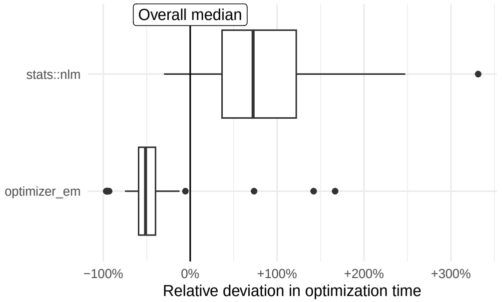
# The EM-algorithm for mixture models

```
Nop_mixture$plot(
  group_by = ".optimizer_label",
  relative = TRUE
)
```



# The EM-algorithm for mixture models

```r
Nop_mixture$plot(
  group_by = ".optimizer_label",
  relative = TRUE
)
```

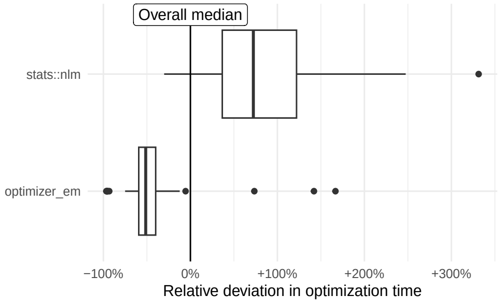



```r
Nop_mixture$optima(
  group_by = ".optimizer_label", digits = 0
)
```

```
## $optimizer_em
##   value frequency
## 1  -276        79
## 2  <NA>        12
## 3  -421         9
##
## $`stats::nlm`
##   value frequency
## 1  -421        79
## 2  -276        20
## 3  -340         1
```

# The EM-algorithm for mixture models

```r
Nop_mixture$plot(
  group_by = ".optimizer_label",
  relative = TRUE
)
```




```r
Nop_mixture$optima(
  group_by = ".optimizer_label", digits = 0
)
```

```
## $optimizer_em
##   value frequency
## 1  -276        79
## 2  <NA>        12
## 3  -421         9
##
## $`stats::nlm`
##   value frequency
## 1  -421        79
## 2  -276        20
## 3  -340         1
```

Conclusion: EM is superior (at least in this example)

# Outline

UNIVERSITÄT BIELEFELD
Faculty of Business Administration and Economics

# Takeaways and outlook

- Clever initialization in numerical likelihood optimization likely
  1. reduces computation time and
  2. increases the probability of reaching the global optimum
- {ino} simplifies comparing initialization strategies and optimization methods
- Test it for your problem and share the outcomes with us!
- Next step: crafting a package manual for the RJournal

# Takeaways and outlook

- Clever initialization in numerical likelihood optimization likely
  1. reduces computation time and
  2. increases the probability of reaching the global optimum
- {ino} simplifies comparing initialization strategies and optimization methods
- Test it for your problem and share the outcomes with us!
- Next step: crafting a package manual for the RJournal

Thanks for your attention! Do you have any questions or comments? 🙂

lennart.oelschlaeger@uni-bielefeld.de
loelschlaeger.de/talks