The cppad_mixed Capture Example and Speed Test

Bradley M. Bell

Applied Physics Laboratory,
Health Metrics and Evaluation,
University of Washington,
`bradbell@uw.edu`

ADMB and TMB Developer's Workshop
June, 2016

Contents

Contents

Contents

Contents

Contents

Introduction

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB+16].

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB+16].
- Can be Back-end for TMB in ADMB, Stan, python, ...

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB$^+$16].
- Can be Back-end for TMB in ADMB, Stan, python, ...
- Starting point and proposal sampling for MCMC methods.

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB+16].
- Can be Back-end for TMB in ADMB, Stan, python, ...
- Starting point and proposal sampling for MCMC methods.
- Separate fixed and random likelihoods.

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB+16].
- Can be Back-end for TMB in ADMB, Stan, python, ...
- Starting point and proposal sampling for MCMC methods.
- Separate fixed and random likelihoods.
- Gaussian or Laplace priors (Lasso) on fixed effects.

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB+16].
- Can be Back-end for TMB in ADMB, Stan, python, ...
- Starting point and proposal sampling for MCMC methods.
- Separate fixed and random likelihoods.
- Gaussian or Laplace priors (Lasso) on fixed effects.
- Constraints on the fixed effects, on random effects.

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB+16].
- Can be Back-end for TMB in ADMB, Stan, python, ...
- Starting point and proposal sampling for MCMC methods.
- Separate fixed and random likelihoods.
- Gaussian or Laplace priors (Lasso) on fixed effects.
- Constraints on the fixed effects, on random effects.
- No user defined C++ template functions.

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB^{+}16].
- Can be Back-end for TMB in ADMB, Stan, python, ...
- Starting point and proposal sampling for MCMC methods.
- Separate fixed and random likelihoods.
- Gaussian or Laplace priors (Lasso) on fixed effects.
- Constraints on the fixed effects, on random effects.
- No user defined C++ template functions.
- CppAD add-on, traps CppAD errors, recovers from errors.

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB+16].
- Can be Back-end for TMB in ADMB, Stan, python, ...
- Starting point and proposal sampling for MCMC methods.
- Separate fixed and random likelihoods.
- Gaussian or Laplace priors (Lasso) on fixed effects.
- Constraints on the fixed effects, on random effects.
- No user defined C++ template functions.
- CppAD add-on, traps CppAD errors, recovers from errors.
- Easy debugging and asserts in complex modeling situations.

CppAD Mixed

C++ Laplace approximation for mixed effect models [Bel16].

- Implements the method in TMB paper [KNB+16].
- Can be Back-end for TMB in ADMB, Stan, python, ...
- Starting point and proposal sampling for MCMC methods.
- Separate fixed and random likelihoods.
- Gaussian or Laplace priors (Lasso) on fixed effects.
- Constraints on the fixed effects, on random effects.
- No user defined C++ template functions.
- CppAD add-on, traps CppAD errors, recovers from errors.
- Easy debugging and asserts in complex modeling situations.

Install

Looking for package managers for particular systems:

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:
  - bin/install_eigen.sh

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:
  - bin/install_eigen.sh
  - bin/install_ipopt.sh

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:
  - bin/install_eigen.sh
  - bin/install_ipopt.sh
  - bin/install_suitesparse.sh

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:
    - bin/install_eigen.sh
    - bin/install_ipopt.sh
    - bin/install_suitesparse.sh
    - bin/install_cppad.sh

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:
  - bin/install_eigen.sh
  - bin/install_ipopt.sh
  - bin/install_suitesparse.sh
  - bin/install_cppad.sh
- bin/run_cmake.sh: edit the run.

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:
  - bin/install_eigen.sh
  - bin/install_ipopt.sh
  - bin/install_suitesparse.sh
  - bin/install_cppad.sh
- bin/run_cmake.sh: edit the run.
- make check, make speed, make install

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:
  - bin/install_eigen.sh
  - bin/install_ipopt.sh
  - bin/install_suitesparse.sh
  - bin/install_cppad.sh
- bin/run_cmake.sh: edit the run.
- make check, make speed, make install
- bin/debian_install.sh: puts it all together for a Debian system

Install

Looking for package managers for particular systems:

- bash shell and web get program wget.
- Compilers: C++ and Fortran.
- gsl: The GNU scientific library.
- cmake: cross platform configuration and installer.
- package-config: information about previous installs.
- Set the install prefix and choose debug / relase for:
  - bin/install_eigen.sh
  - bin/install_ipopt.sh
  - bin/install_suitesparse.sh
  - bin/install_cppad.sh
- bin/run_cmake.sh: edit the run.
- make check, make speed, make install
- bin/debian_install.sh: puts it all together for a Debian system

capture_xam

This is a `cppad_mixed` example; see [Roy04].

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.
- Reports correctness test results.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.
- Reports correctness test results.
- Reports memory usage results.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.
- Reports correctness test results.
- Reports memory usage results.
- Separate control of data set size.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.
- Reports correctness test results.
- Reports memory usage results.
- Separate control of data set size.
- Separate control of number of random effects.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.
- Reports correctness test results.
- Reports memory usage results.
- Separate control of data set size.
- Separate control of number of random effects.
- Separate control of computational load.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.
- Reports correctness test results.
- Reports memory usage results.
- Separate control of data set size.
- Separate control of number of random effects.
- Separate control of computational load.
- Compare Newton and quasi-Newton methods.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.
- Reports correctness test results.
- Reports memory usage results.
- Separate control of data set size.
- Separate control of number of random effects.
- Separate control of computational load.
- Compare Newton and quasi-Newton methods.
- Interesting numerical AD method used to avoid overflow.

capture_xam

This is a `cppad_mixed` example; see [Roy04].

- Tests both optimization and posterior sampling.
- Reports timing test results.
- Reports correctness test results.
- Reports memory usage results.
- Separate control of data set size.
- Separate control of number of random effects.
- Separate control of computational load.
- Compare Newton and quasi-Newton methods.
- Interesting numerical AD method used to avoid overflow.
- Optional constraints.

Data Model

Data Given Population Size and Capture Probability

- $y_{i,t}$ is the number of captures at location i and time t.

Data Given Population Size and Capture Probability

- $y_{i,t}$ is the number of captures at location i and time t.
- N_i is the size of the population at location i.

Data Given Population Size and Capture Probability

- $y_{i,t}$ is the number of captures at location i and time t.
- N_i is the size of the population at location i.
- q_t is the probability of capture at time t.

Data Given Population Size and Capture Probability

- $y_{i,t}$ is the number of captures at location i and time t.
- N_i is the size of the population at location i.
- q_t is the probability of capture at time t.
- The conditional probability of $y_{i,t}$ given N_i and q_t is

$$\mathbf{p}(y_{i,t}|N_i, q_t) = \left(\begin{array}{c} N_i \\ y_{i,t} \end{array} \right) q_t^{y(i,t)} (1 - q_t)^{y(i,t)}$$

Population Size Given Fixed Effects

- θ_0 is the mean of N_i .

Population Size Given Fixed Effects

- θ_0 is the mean of N_i .
- θ is the vector of fixed effects.

Population Size Given Fixed Effects

- θ_0 is the mean of N_i .
- θ is the vector of fixed effects.
- We use a Poisson distribution to model the probability of N_i given θ

$$\mathbf{p}(N_i|\theta) = \theta_0^{N(i)} \frac{\exp[-\theta_0]}{N_i!}$$

Capture Probability Given Fixed and Random Effects

- θ_1 is the mean of the logit of q_t.

Capture Probability Given Fixed and Random Effects

- θ_1 is the mean of the logit of q_t.
- u_t is the random effect at time t.

Capture Probability Given Fixed and Random Effects

- θ_1 is the mean of the logit of q_t.
- u_t is the random effect at time t.
- u is the vector of random effects.

Capture Probability Given Fixed and Random Effects

- θ_1 is the mean of the logit of q_t.
- u_t is the random effect at time t.
- u is the vector of random effects.
- The logit of q_t is

$$\log[q_t/(1-q_t)] = u_t + \theta_1$$

Capture Probability Given Fixed and Random Effects

- θ_1 is the mean of the logit of q_t.
- u_t is the random effect at time t.
- u is the vector of random effects.
- The logit of q_t is

$$\log[q_t/(1-q_t)] = u_t + \theta_1$$

- The probability of capture at time t given θ and u is

$$q_t(\theta, u) = \mathbf{p}(q_t|\theta, u) = [1 + \exp(-u_t - \theta_1)]^{-1}$$

Data Given Fixed and Random Effects

- y_i vector of data at location i and all times.

Data Given Fixed and Random Effects

- y_i vector of data at location i and all times.
- T is the number of times.

Data Given Fixed and Random Effects

- y_i vector of data at location i and all times.
- T is the number of times.
- The probability of y_i given N_i, θ, and u is

$$\mathbf{p}(y_i|N_i, \theta, u) = \prod_{t=0}^{T-1} \left(\begin{array}{c} N(i) \\ y_{i,t} \end{array} \right) q_t(\theta, u)^{y(i,t)} (1 - q_t(\theta, u))^{y(i,t)}$$

Data Given Fixed and Random Effects

- y_i vector of data at location i and all times.
- T is the number of times.
- The probability of y_i given N_i, θ, and u is

$$\mathbf{p}(y_i|N_i,\theta,u) = \prod_{t=0}^{T-1} \binom{N(i)}{y_{i,t}} q_t(\theta,u)^{y(i,t)} (1 - q_t(\theta,u))^{y(i,t)}$$

- K is maximum population in truncation of infinite summation.

Data Given Fixed and Random Effects

- y_i vector of data at location i and all times.
- T is the number of times.
- The probability of y_i given N_i, θ, and u is

$$\mathbf{p}(y_i|N_i,\theta,u) = \prod_{t=0}^{T-1} \left(\begin{array}{c} N(i) \\ y_{i,t} \end{array} \right) q_t(\theta,u)^{y(i,t)} \left(1 - q_t(\theta,u)\right)^{y(i,t)}$$

- K is maximum population in truncation of infinite summation.
- Probability of y_i given θ, u, is modeled using the sum

$$\mathbf{p}(y_i|\theta,u) = \sum_{k=0}^{K} \mathbf{p}(y_i|N_i = k,\theta,u)\mathbf{p}(N_i = k|\theta)$$

Data Given Fixed and Random Effects

- y_i vector of data at location i and all times.
- T is the number of times.
- The probability of y_i given N_i, θ, and u is

$$\mathbf{p}(y_i|N_i,\theta,u) = \prod_{t=0}^{T-1} \left(\begin{array}{c} N(i) \\ y_{i,t} \end{array} \right) q_t(\theta,u)^{y(i,t)} \left(1 - q_t(\theta,u)\right)^{y(i,t)}$$

- K is maximum population in truncation of infinite summation.
- Probability of y_i given θ, u, is modeled using the sum

$$\mathbf{p}(y_i|\theta,u) = \sum_{k=0}^{K} \mathbf{p}(y_i|N_i = k,\theta,u)\mathbf{p}(N_i = k|\theta)$$

Data Given Fixed and Random Effects

- y vector of data at all locations and all times.

Data Given Fixed and Random Effects

- y vector of data at all locations and all times.
- R is the number of locations.

Data Given Fixed and Random Effects

- y vector of data at all locations and all times.
- R is the number of locations.
- The probability of y given θ, and u is

$$\mathbf{p}(y|\theta, u) = \prod_{i=0}^{R-1} \mathbf{p}(y_i|\theta, u) =$$

Data Given Fixed and Random Effects

- y vector of data at all locations and all times.
- R is the number of locations.
- The probability of y given θ, and u is

$$\mathbf{p}(y|\theta, u) = \prod_{i=0}^{R-1} \mathbf{p}(y_i|\theta, u) =$$

$$\prod_{i=0}^{R-1} \left[\sum_{k=0}^{K-1} \theta_0^k \frac{\exp[-\theta_0]}{k!} \prod_{t=0}^{T-1} \begin{pmatrix} k \\ y_{i,t} \end{pmatrix} q_t(\theta, u)^{y(i,t)} \left(1 - q_t(\theta, u)\right)^{y(i,t)} \right]$$

Program Input

Random Likelihood Function

- θ_2 is the standard deviation of the logit of q_t.

Random Likelihood Function

- θ_2 is the standard deviation of the logit of q_t.
- We use a normal distribution for u given θ,

$$\mathbf{p}(u|\theta) = \prod_{t=0}^{T-1} \frac{1}{\theta_2\sqrt{2\pi}} \exp\left[-\frac{1}{2}\frac{u_t^2}{\theta_2^2}\right]$$

Random Likelihood Function

- θ_2 is the standard deviation of the logit of q_t.
- We use a normal distribution for u given θ,

$$\mathbf{p}(u|\theta) = \prod_{t=0}^{T-1} \frac{1}{\theta_2 \sqrt{2\pi}} \exp \left[-\frac{1}{2} \frac{u_t^2}{\theta_2^2} \right]$$

- The user defined random likelihood for this example is

$$f(\theta, u) = -\log[\mathbf{p}(y|\theta, u)\mathbf{p}(u|\theta)]$$

Other User Defined Functions

- There is no prior for the fixed effects; i.e., $\mathbf{p}(\theta)$.

Other User Defined Functions

- There is no prior for the fixed effects; i.e., $\mathbf{p}(\theta)$.
- There is no data that only depends on fixed effects; i.e., $\mathbf{p}(z|\theta)$.

Other User Defined Functions

- There is no prior for the fixed effects; i.e., $\mathbf{p}(\theta)$.
- There is no data that only depends on fixed effects; i.e., $\mathbf{p}(z|\theta)$.
- There is no fixed constraint functions; i.e, $c(\theta)$.

Other User Defined Functions

- There is no prior for the fixed effects; i.e., $\mathbf{p}(\theta)$.
- There is no data that only depends on fixed effects; i.e., $\mathbf{p}(z|\theta)$.
- There is no fixed constraint functions; i.e, $c(\theta)$.
- $$\hat{u}(\theta) = \text{argmin } f(\theta, u) \text{ w.r.t. } u$$

Other User Defined Functions

- There is no prior for the fixed effects; i.e., $\mathbf{p}(\theta)$.
- There is no data that only depends on fixed effects; i.e., $\mathbf{p}(z|\theta)$.
- There is no fixed constraint functions; i.e, $c(\theta)$.
- $$\hat{u}(\theta) = \mathrm{argmin} \; f(\theta, u) \text{ w.r.t. } u$$
- A is the random constraint matrix.
- The random constraint function is

$$0 = A\hat{u}(\theta) = \hat{u}_1(\theta) + \cdots + \hat{u}_{T-1}(\theta)$$

Command Line Arguments

- random_seed   non-negative integer 0 , match previous

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10
max_population	positive integer , K	50

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10
max_population	positive integer , K	50
mean_population	positive real , θ_0	5.0

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10
max_population	positive integer , K	50
mean_population	positive real , θ_0	5.0
mean_logit_probability	real , θ_1	-0.5

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10
max_population	positive integer , K	50
mean_population	positive real , θ_0	5.0
mean_logit_probability	real , θ_1	-0.5
std_logit_probability	positive real , θ_2	0.5

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10
max_population	positive integer , K	50
mean_population	positive real , θ_0	5.0
mean_logit_probability	real , θ_1	-0.5
std_logit_probability	positive real , θ_2	0.5
quasi_fixed	true or false	true

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10
max_population	positive integer , K	50
mean_population	positive real , θ_0	5.0
mean_logit_probability	real , θ_1	-0.5
std_logit_probability	positive real , θ_2	0.5
quasi_fixed	true or false	true
random_constraint	true or false	false , true

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10
max_population	positive integer , K	50
mean_population	positive real , θ_0	5.0
mean_logit_probability	real , θ_1	-0.5
std_logit_probability	positive real , θ_2	0.5
quasi_fixed	true or false	true
random_constraint	true or false	false , true
trace_optimize_fixed	true or false	false

Command Line Arguments

random_seed	non-negative integer	0 , match previous
number_fixed_samples	positive integer	1000
number_locations	positive integer , R	50
number_times	positive integer , T	10
max_population	positive integer , K	50
mean_population	positive real , θ_0	5.0
mean_logit_probability	real , θ_1	-0.5
std_logit_probability	positive real , θ_2	0.5
quasi_fixed	true or false	true
random_constraint	true or false	false , true
trace_optimize_fixed	true or false	false

Program Output

Seed, Memory, Timing Results

actual_seed	1466520673 , 1466520673

Seed, Memory, Timing Results

actual_seed	1466520673 , 1466520673
initialize_bytes	27,728,078 , 27,728,078

Seed, Memory, Timing Results

actual_seed	1466520673 , 1466520673
initialize_bytes	27,728,078 , 27,728,078
initialize_seconds	0.244 , 0.259

Seed, Memory, Timing Results

actual_seed	1466520673 , 1466520673
initialize_bytes	27,728,078 , 27,728,078
initialize_seconds	0.244 , 0.259
optimize_fixed_seconds	5.49 , 2.96

Seed, Memory, Timing Results

actual_seed	1466520673 , 1466520673
initialize_bytes	27,728,078 , 27,728,078
initialize_seconds	0.244 , 0.259
optimize_fixed_seconds	5.49 , 2.96
optimize_random_seconds	0.077 , 0.101

Seed, Memory, Timing Results

actual_seed	1466520673 , 1466520673
initialize_bytes	27,728,078 , 27,728,078
initialize_seconds	0.244 , 0.259
optimize_fixed_seconds	5.49 , 2.96
optimize_random_seconds	0.077 , 0.101
information_mat_seconds	1.24 , 1.21

Seed, Memory, Timing Results

actual_seed	1466520673 , 1466520673
initialize_bytes	27,728,078 , 27,728,078
initialize_seconds	0.244 , 0.259
optimize_fixed_seconds	5.49 , 2.96
optimize_random_seconds	0.077 , 0.101
information_mat_seconds	1.24 , 1.21
sample_fixed_seconds	0.024 , 0.041

Seed, Memory, Timing Results

actual_seed	1466520673 , 1466520673
initialize_bytes	27,728,078 , 27,728,078
initialize_seconds	0.244 , 0.259
optimize_fixed_seconds	5.49 , 2.96
optimize_random_seconds	0.077 , 0.101
information_mat_seconds	1.24 , 1.21
sample_fixed_seconds	0.024 , 0.041

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35

Estimate Results

▶ sum_random_effects	-0.0698 , 1.33227e-15
▶ mean_population_estimate	5.84 , 5.35
▶ mean_logit_probability_estimate	-0.607 , -0.494

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35
mean_logit_probability_estimate	-0.607 , -0.494
std_logit_probability_estimate	0.583 , 0.667

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35
mean_logit_probability_estimate	-0.607 , -0.494
std_logit_probability_estimate	0.583 , 0.667
mean_population_std	0.579 , 0.492

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35
mean_logit_probability_estimate	-0.607 , -0.494
std_logit_probability_estimate	0.583 , 0.667
mean_population_std	0.579 , 0.492
mean_logit_probability_std	0.23 , 0.092

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35
mean_logit_probability_estimate	-0.607 , -0.494
std_logit_probability_estimate	0.583 , 0.667
mean_population_std	0.579 , 0.492
mean_logit_probability_std	0.23 , 0.092
std_logit_probability_std	0.142 , 0.161

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35
mean_logit_probability_estimate	-0.607 , -0.494
std_logit_probability_estimate	0.583 , 0.667
mean_population_std	0.579 , 0.492
mean_logit_probability_std	0.23 , 0.092
std_logit_probability_std	0.142 , 0.161
mean_population_ratio	1.45 , 0.702

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35
mean_logit_probability_estimate	-0.607 , -0.494
std_logit_probability_estimate	0.583 , 0.667
mean_population_std	0.579 , 0.492
mean_logit_probability_std	0.23 , 0.092
std_logit_probability_std	0.142 , 0.161
mean_population_ratio	1.45 , 0.702
mean_logit_probability_ratio	-0.464 , 0.07

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35
mean_logit_probability_estimate	-0.607 , -0.494
std_logit_probability_estimate	0.583 , 0.667
mean_population_std	0.579 , 0.492
mean_logit_probability_std	0.23 , 0.092
std_logit_probability_std	0.142 , 0.161
mean_population_ratio	1.45 , 0.702
mean_logit_probability_ratio	-0.464 , 0.07
std_logit_probability_ratio	0.584 , 1.04

Estimate Results

sum_random_effects	-0.0698 , 1.33227e-15
mean_population_estimate	5.84 , 5.35
mean_logit_probability_estimate	-0.607 , -0.494
std_logit_probability_estimate	0.583 , 0.667
mean_population_std	0.579 , 0.492
mean_logit_probability_std	0.23 , 0.092
std_logit_probability_std	0.142 , 0.161
mean_population_ratio	1.45 , 0.702
mean_logit_probability_ratio	-0.464 , 0.07
std_logit_probability_ratio	0.584 , 1.04

Averages for Twenty Runs Each

- optimize_fixed_seconds_avg 5.59 , 2.71

Averages for Twenty Runs Each

▶ optimize_fixed_seconds_avg	5.59 , 2.71
▶ mean_population_error_avg	0.41 , 0.37

Averages for Twenty Runs Each

optimize_fixed_seconds_avg	5.59 , 2.71
mean_population_error_avg	0.41 , 0.37
mean_population_std_avg	0.52 , 0.49

Averages for Twenty Runs Each

optimize_fixed_seconds_avg	5.59 , 2.71
mean_population_error_avg	0.41 , 0.37
mean_population_std_avg	0.52 , 0.49
mean_logit_probability_error_avg	0.18 , 0.10

Averages for Twenty Runs Each

optimize_fixed_seconds_avg	5.59 , 2.71
mean_population_error_avg	0.41 , 0.37
mean_population_std_avg	0.52 , 0.49
mean_logit_probability_error_avg	0.18 , 0.10
mean_logit_probability_std_avg	0.21 , 0.09

Averages for Twenty Runs Each

optimize_fixed_seconds_avg	5.59 , 2.71
mean_population_error_avg	0.41 , 0.37
mean_population_std_avg	0.52 , 0.49
mean_logit_probability_error_avg	0.18 , 0.10
mean_logit_probability_std_avg	0.21 , 0.09
std_logit_probability_error_avg	0.13 , 0.13

Averages for Twenty Runs Each

optimize_fixed_seconds_avg	5.59 , 2.71
mean_population_error_avg	0.41 , 0.37
mean_population_std_avg	0.52 , 0.49
mean_logit_probability_error_avg	0.18 , 0.10
mean_logit_probability_std_avg	0.21 , 0.09
std_logit_probability_error_avg	0.13 , 0.13
std_logit_probability_std_avg	0.12 , 0.12

Averages for Twenty Runs Each

optimize_fixed_seconds_avg	5.59 , 2.71
mean_population_error_avg	0.41 , 0.37
mean_population_std_avg	0.52 , 0.49
mean_logit_probability_error_avg	0.18 , 0.10
mean_logit_probability_std_avg	0.21 , 0.09
std_logit_probability_error_avg	0.13 , 0.13
std_logit_probability_std_avg	0.12 , 0.12

Bibliography

[Bel16] Bradley M. Bell.
cppad_mixed: Laplace Approximation of Mixed Effects Models.
IHME: Institute for Health Metrics and Evaluation, University of Washington, http://moby.ihme.washington.edu/bradbell/cppad_mixed/, 2016.

[KNB$^+$16] Kasper Kristensen, Anders Nielsen, Casper W. Berg, Hans Skaug, and Bradley M. Bell.
TMB: Automatic differentiation and Laplace approximation.
Journal of Statistical Software, 70:1–21, 2016.

[Roy04] Andrew J. Royle.
N-mixture models for estimating population size from spatially replicated counts.
Biometrics, 60:108–115, 2004.