

# Package ‘impala’

June 19, 2026

**Type** Package

**Title** Bayesian Model Calibration

**Version** 0.1.0

**Description** Package provides tools for modular Bayesian model calibration. these tools allow for posterior exploration with sampling methods including tempering and adaptive Markov Chain Monte Carlo (MCMC).

Allows for pooled calibration or hierarchal calibration of parameters.

For more information see Francom et al., 2025 <[DOI:10.1137/24M1644092](https://doi.org/10.1137/24M1644092)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** cli, methods, progress, einsum,

**Suggests** fdastrvf, BASS, BayesPPR, mvBayes, coda, bayesplot, gridExtra,  
hexbin, knitr, rmarkdown

**Config/roxygen2/version** 8.0.0

**URL** <https://github.com/sandialabs/rImpala>

**BugReports** <https://github.com/sandialabs/rImpala/issues>

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** J. Derek Tucker [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-8844-2169>>),  
Sandia National Laboratories [cph, fnd]

**Maintainer** J. Derek Tucker <jdtuck@sandia.gov>

**Repository** CRAN

**Date/Publication** 2026-06-19 12:00:36 UTC

## Contents

addVecExperiments . . . . .	2
calibPool . . . . .	3
CalibSetup . . . . .	4

cf_bounds . . . . .	5
evalm . . . . .	6
ModelBassPca_func . . . . .	6
ModelmvBayes . . . . .	7
ModelmvBayes_elastic . . . . .	7
ModelmvBayes_elastic_GP . . . . .	8
ModelmvBayes_GP . . . . .	9
setMCMC . . . . .	9
setTemperatureLadder . . . . .	10
tran_unif . . . . .	11

## Index 12

---

addVecExperiments	<i>Add vector experiments</i>
-------------------	-------------------------------

---

### Description

This method adds vector experiments to calibration object

### Usage

```
addVecExperiments(
  obj,
  yobs,
  model,
  sd_est,
  s2_df,
  s2_ind,
  meas_error_cor = NULL,
  theta_ind = NULL,
  D = NULL,
  discrep_tau = 1
)
```

### Arguments

obj	‘CalibSetup‘ Object
yobs	a vector of the experiment or observation
model	emulator (currently expecting a object of class ‘ModelBassPca_func‘ or ‘ModelmvBayes‘)
sd_est	estimate of standard deviation
s2_df	degrees of freedom of inverse gamma prior
s2_ind	indices of function
meas_error_cor	measurement error correlation (default: ‘NULL‘)
theta_ind	indices of theta (default: ‘NULL‘)
D	discrepancy basis (matrix of columns of basis, default: ‘NaN‘)
discrep_tau	discrepancy sampling tau

**Value**

An object of class ‘CalibSetup‘

---

calibPool

*Pool Calibration*

---

**Description**

This function runs pooled Bayesian Model Calibration with adaptive MCMC, tempering, and decorrelation steps. input theta will be normalized to 0-1 and sampled from uniform priors

**Usage**

```
calibPool(setup)
```

**Arguments**

setup                    an object of class ‘CalibSetup‘

**Value**

a list with the following elements - theta: mcmc samples of variables - s2: mcmc samples of error variance - count: number of counts of acceptance - count\_s2: number of counts of acceptance on error variance - count\_decor: number of times decorrelation occurred - cov\_theta\_cand: final theta covariance - cov\_ls2\_cand: final error covariance - pred\_curr: current emulator predictions - discrep\_vars: discrepancy coefficients - llik: log likelihood - theta\_native: mcmc samples of variables in native scale

**Examples**

```
library(impala)
library(BASS)
library(mvBayes)
library(fdasrvf)
library(bayesplot)

# generate functions
f <- function(x) {
  dnorm(seq(0, 1, length.out = 99),
        sin(2 * pi * x[1] ^ 2) / 4 - x[1] / 10 + 0.5,
        0.05) * x[2]
}

n = 100
nt = 99
p = 3
x_train = matrix(runif(n * p), n)
e = rnorm(n * 99)
y_train = matrix(0, n, nt)
```

```

for (i in 1:n) {
  y_train[i, ] = f(x_train[i, ])
}

# generate obs -----
x_true = c(0.1028, 0.5930)
ftilde_obs = f(x_true)
gam_obs = seq(0, 1, length.out = nt)
vv_obs = gam_to_v(gam_obs)

tt = seq(0, 1, length.out = nt)
out = multiple_align_functions(t(y_train), tt, ftilde_obs, 0.01)
gam_train = out$warping_functions
vv_train = gam_to_v(gam_train)
ftilde_train = out$fn
qtilde_train = out$qn

# fit emulator -----
emu_ftilde = mvBayes(bass, x_train, t(ftilde_train), nBasis=2)

emu_vv = mvBayes(bass, x_train, t(vv_train), nBasis=2)

# impala -----
input_names = c("theta0", "theta1", "theta2")
bounds = list()
bounds[['theta0']] = c(0, 1)
bounds[['theta1']] = c(0, 1)
bounds[['theta2']] = c(0, 1)

setup = CalibSetup(bounds, cf_bounds)

model_ftilde = ModelmvBayes(emu_ftilde, input_names)
model_vv = ModelmvBayes(emu_vv, input_names)

setup = addVecExperiments(setup, t(ftilde_obs), model_ftilde, 0.01, 20, rep(1, nt))
setup = addVecExperiments(setup, t(vv_obs), model_vv, 0.01, 20, rep(1, nt))
setup = setTemperatureLadder(setup, 1.05 ^ (0:2))
setup = setMCMC(setup, 1000, 500, 1, 10)
out_cal = calibPool(setup)

```

---

CalibSetup

*Pooled Bayesian Model Calibration*


---

### Description

This function setups up calibration object

**Usage**

```
CalibSetup(bounds, constraint_func)
```

**Arguments**

`bounds` a list with fields of variable names with values of two dimensional arrays where first element is lower bound and second element is upper bound.

`constraint_func` function handle to constraint function on variables. Defaults to function on checking bounds

**Value**

An object of class 'CalibSetup' which is a list with the following components:

- 'nexp': number of experiments

---

cf_bounds	<i>Compare to bounds</i>
-----------	--------------------------

---

**Description**

This function compares variables to bounds

**Usage**

```
cf_bounds(x, bounds)
```

**Arguments**

`x` list of of parameters

`bounds` list of bounds for each parameter that is a two parameter vector with high and low

**Value**

a vector of 'TRUE' or 'FALSE' if the values are within the bounds

---

evalm	<i>evalm constructor</i>
-------	--------------------------

---

**Description**

Default constructor for evalm class

**Usage**

```
evalm(obj, ...)
```

**Arguments**

obj	evalm object
...	additional arguments passed to method

**Value**

An object of class 'evalm'

---

ModelBassPca_func	<i>PCA Based Model Emulator using BASS</i>
-------------------	--

---

**Description**

This function setups up emulator object

**Usage**

```
ModelBassPca_func(bmod, input_names, exp_ind = NULL, s2 = "MH")
```

**Arguments**

bmod	a object of the type 'bassBasis'
input_names	cell array of strings of input variable names
exp_ind	experiment indices (default: NULL)
s2	how to sample error variance (default: 'MH')

**Value**

An object of class 'ModelBassPca\_func'

---

ModelmvBayes	<i>mvBayes Emulator for Functional Outputs (can use different BASS/BPPR type emulators)</i>
--------------	---

---

**Description**

ModelmvBayes Handles larger-dimensional functional responses (e.g., on large spatial fields) using various inversion tricks. We require any other covariance e.g., from discrepancy, measurement error, and basis truncation error) to be diagonal. This function setups up emulator object.

**Usage**

```
ModelmvBayes(bmod, input_names, exp_ind = NULL, s2 = "MH", h = FALSE)
```

**Arguments**

bmod	a object of the type 'mvBayes'
input_names	cell array of strings of input variable names
exp_ind	experiment indices (default: NULL)
s2	how to sample error variance (default: 'MH')
h	h representation of warping function (default: FALSE)

**Value**

An object of class 'ModelmvBayes'

---

ModelmvBayes_elastic	<i>mvBayes elastic Emulator for Functional Outputs (can use different BASS/BPPR type emulators)</i>
----------------------	---

---

**Description**

ModelmvBayes\_elastic Handles larger-dimensional functional responses (e.g., on large spatial fields) using various inversion tricks. We require any other covariance e.g., from discrepancy, measurement error, and basis truncation error) to be diagonal. Contains a change on the likelihood

**Usage**

```
ModelmvBayes_elastic(
  bmod,
  bmod_warp,
  input_names,
  exp_ind = NULL,
  s2 = "MH",
  h = FALSE
)
```

**Arguments**

bmod	a object of the type 'mvBayes' of aligned functions
bmod_warp	a object of the type 'mvBayes' of warping functions
input_names	cell array of strings of input variable names
exp_ind	experiment indices (default: NULL)
s2	how to sample error variance (default: 'MH')
h	h representation of warping function (default: FALSE)

**Details**

This function setups up emulator object.

**Value**

An object of class 'ModelmvBayes\_elastic'

---

ModelmvBayes\_elastic\_GP

*mvBayes elastic Emulator for Functional Outputs (can use different BASS/BPPR type emulators)*

---

**Description**

mvBayes\_elastic Handles larger-dimensional functional responses (e.g., on large spatial fields) using various inversion tricks. We require any other covariance e.g., from discrepancy, measurement error, and basis truncation error) to be diagonal. Contains a change on the likelihood and built for a GP emulator. This function setups up emulator object.

**Usage**

```
ModelmvBayes_elastic_GP(
  bmod,
  bmod_warp,
  input_names,
  exp_ind = NULL,
  s2 = "MH",
  h = FALSE
)
```

**Arguments**

bmod	a object of the type 'mvBayes' of aligned functions
bmod_warp	a object of the type 'mvBayes' of warping functions
input_names	cell array of strings of input variable names
exp_ind	experiment indices (default: NULL)
s2	how to sample error variance (default: 'MH')
h	h representation of warping function (default: FALSE)

**Value**

An object of class ‘ModelmvBayes\_elastic\_GP’

---

ModelmvBayes_GP	<i>mvBayes Emulator for Functional Outputs (GP)</i>
-----------------	---

---

**Description**

ModelmvBayes\_GP Handles larger-dimensional functional responses (e.g., on large spatial fields) using various inversion tricks. We require any other covariance e.g., from discrepancy, measurement error, and basis truncation error) to be diagonal. Built for a GP emulator.

**Usage**

```
ModelmvBayes_GP(bmod, input_names, exp_ind = NULL, s2 = "MH", h = FALSE)
```

**Arguments**

bmod	a object of the type ‘mvBayes’
input_names	cell array of strings of input variable names
exp_ind	experiment indices (default: NULL)
s2	how to sample error variance (default: ‘MH’)
h	h representation of warping function (default: FALSE)

**Details**

This function setups up emulator object.

**Value**

An object of class ‘ModelmvBayes\_GP’

---

setMCMC	<i>Set MCMC Parameters</i>
---------	----------------------------

---

**Description**

This function setups up MCMC parameters for adaptive MCMC, also includes tempering and decorrelation steps

**Usage**

```

setMCMC(
  obj,
  nmcmc,
  nburn = 0,
  thin = 1,
  decor = 100,
  start_var_theta = 1e-08,
  start_tau_theta = 0,
  start_var_ls2 = 1e-05,
  start_tau_ls2 = 0,
  start_adapt_iter = 300
)

```

**Arguments**

obj	‘calibPool‘ object
nmcmc	number of mcmc iterations
nburn	number of mcmc burn in iterations (default: 0)
thin	number of samples to thin (default: 1)
decor	number of mcmc iterations before decorrelation step (default: 100)
start_var_theta	start variance of theta proposal (default: 1e-8)
start_tau_theta	start tau of theta (default: 0)
start_var_ls2	start variance of sigma proposal (default: 1e-5)
start_tau_ls2	start tau of sigma (default: 0)
start_adapt_iter	number of iterations before adaption (default: 300)

**Value**

An object of class ‘CalibSetup‘

---

setTemperatureLadder    *Set Temperature Ladder*

---

**Description**

This function setups up temperature ladder for tempering

**Usage**

```

setTemperatureLadder(obj, temperature_ladder, start_temper = 1000)

```

**Arguments**

obj                    CalibSetup Object  
 temperature\_ladder    array of temperatures  
 start\_temper        what MCMC sample to start tempering (default: 1000)

**Value**

An object of class ‘CalibSetup‘

---

tran_unif	<i>Transform parameters</i>
-----------	-----------------------------

---

**Description**

Transforms variables from the scale 0 to 1 to variable provided by bounds

**Usage**

```
tran_unif(th, bounds, names)
```

**Arguments**

th                    list of parameters  
 bounds               list of bounds for each parameter that is a two parameter vector with high and low  
 names                vector of variable names

**Value**

a list of parameters

# Index

`addVecExperiments`, [2](#)

`calibPool`, [3](#)

`CalibSetup`, [4](#)

`cf_bounds`, [5](#)

`evalm`, [6](#)

`ModelBassPca_func`, [6](#)

`ModelmvBayes`, [7](#)

`ModelmvBayes_elastic`, [7](#)

`ModelmvBayes_elastic_GP`, [8](#)

`ModelmvBayes_GP`, [9](#)

`setMCMC`, [9](#)

`setTemperatureLadder`, [10](#)

`tran_unif`, [11](#)