

# Package ‘AHM’

October 12, 2022

**Type** Package

**Title** Additive Heredity Model: Method for the Mixture-of-Mixtures Experiments

**Version** 1.0.1

**Maintainer** Sumin Shen <sumin@vt.edu>

**Description** An implementation of the additive heredity model for the mixture-of-mixtures experiments of Shen et al. (2019) in Technometrics <doi:10.1080/00401706.2019.1630010>. The additive heredity model considers an additive structure to inherently connect the major components with the minor components. The additive heredity model has a meaningful interpretation for the estimated model because of the hierarchical and heredity principles applied and the nonnegative garrote technique used for variable selection.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** mixexp, plgp, devtools, dplyr, tibble, tidyr, Matrix

**Depends** R (>= 2.10), quadprog, glmnet

**Suggests** knitr, rmarkdown, partitions

**VignetteBuilder** knitr

**BuildVignettes** yes

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Author** Sumin Shen [aut, cre],  
Lulu Kang [aut],  
Xinwei Deng [aut]

**Repository** CRAN

**Date/Publication** 2019-07-28 09:00:05 UTC

## R topics documented:

ahm . . . . .	2
check_col_correlation . . . . .	3
coating . . . . .	4
coef.ahm . . . . .	4
coef.cv.ahm . . . . .	5
compute_aicc . . . . .	6
cv.ahm . . . . .	6
design_simplex_centroid_design_3_major_component . . . . .	8
enlist . . . . .	8
expand_interactions . . . . .	9
find_condition_num . . . . .	9
mapping_function . . . . .	10
mymaximin . . . . .	10
predict.ahm . . . . .	12
predict.cv.ahm . . . . .	13
pringles_candidates2search . . . . .	14
pringles_fat . . . . .	14
pringles_hardness . . . . .	15
summary.ahm . . . . .	15

<b>Index</b>	<b>17</b>
--------------	-----------

---

ahm	<i>This is one of the main functions. The function ahm computes the proposed additive heredity model.</i>
-----	---

---

### Description

This is one of the main functions. The function ahm computes the proposed additive heredity model.

### Usage

```
ahm(y, x, num_major = 3, dist_minor = c(2, 2, 1), type = "weak",
    alpha = 0, lambda_seq = seq(0, 5, 0.01), nfold = NULL,
    mapping_type = c("power"), powerh = 0, rep_gcv = 100)
```

### Arguments

y	numeric vector
x	data.frame Note the column names of the x should be in the order of major components, minor components, and no interactions are needed.
num_major	number of major components
dist_minor	the allocation of number of minor components nested under major components
type	heredity type, weak heredity is the current support type
alpha	0 is for the ridge in glmnet <a href="https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html">https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html</a>

lambda_seq	a numeric vector for the options of lambda used in ridge regression for estimating the initials
nfolds	used in cv.glmnet for initial value of parameters in the non-negative garrote method
mapping_type	the form of the coefficient function of major components in front of corresponding minor terms. Currently only support "power"
powerh	the power parameter used for the power function
rep_gcv	the number of choices of tuning parameter used in the GCV selection

**Value**

Return a list

**Examples**

```
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
y = data_fat[,1]
out = ahm (y, x, num_major = 3, dist_minor = c(2,2,1),
          type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfold = NULL,
          mapping_type = c("power"), powerh = h_tmp,
          rep_gcv=100)
summary(out)
```

---

check\_col\_correlation *Check column correlations*

---

**Description**

Check column correlations

**Usage**

```
check_col_correlation(dat)
```

**Arguments**

dat                    data.frame

**Examples**

```
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
check_col_correlation (dat=x)
```

---

coating	<i>Photoresist-coating experiment data</i>
---------	--

---

**Description**

Photoresist-coating experiment data

**Usage**

```
data(coating)
```

**Format**

data.frame

**References**

Cornell, J.A. and Ramsey, P.J. (1998). A Generalized mixture model for categorized-components problems with an application to a photoresist-coating experiment. *Technometrics*, 40(1), 48-61. ([tandfonline](#))

**Examples**

```
data(coating)
print(coating)
```

---

coef.ahm	<i>Coefficient method for the fitted ahm object</i>
----------	---

---

**Description**

Coefficient method for the fitted ahm object

**Usage**

```
## S3 method for class 'ahm'
coef(object, ...)
```

**Arguments**

object	ahm object
...	not used

**Value**

a numerical vector

**Examples**

```

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
y = data_fat[,1]
out = ahm (y, x, num_major = 3, dist_minor = c(2,2,1),
          type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfold = NULL,
          mapping_type = c("power"), powerh = h_tmp,
          rep_gcv=100)
coef(out)

```

coef.cv.ahm

*Coefficient method for the fitted cv.ahm object***Description**

Coefficient method for the fitted cv.ahm object

**Usage**

```

## S3 method for class 'cv.ahm'
coef(object, metric = "mse", ...)

```

**Arguments**

object	cv.ahm object
metric	"mse" or "aicc"
...	not used

**Value**

a numerical vector

**Examples**

```

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
y = data_fat[,1]
powerh_path = round(seq(0.001,2,length.out =15),3)
num_major = 3; dist_minor = c(2,2,1)
res = cv.ahm (y, x, powerh_path=powerh_path, metric = "mse", num_major, dist_minor, type = "weak"
, alpha=0, lambda_seq=seq(0,5,0.01), nfolds=NULL, mapping_type = c("power"), rep_gcv=100)
coefficients = coef(res)

```

---

compute_aicc	<i>compute AICc</i>
--------------	---------------------

---

**Description**

compute AICc

**Usage**

```
compute_aicc(rss, n, p, type = "AICc")
```

**Arguments**

rss	residual sum of squares
n	number of observation
p	number of nonzero parameters
type	character "AICc"

**References**

[Calculating AIC “by hand” in R in Stack Overflow](#)

**Examples**

```
compute_aicc (rss=10, n=30, p=6, type = "AICc")
```

---

cv.ahm	<i>This is one of the main functions. It perform the cross validation on ahm models to select the optimal setting of hyper parameter h</i>
--------	--

---

**Description**

This is one of the main functions. It perform the cross validation on ahm models to select the optimal setting of hyper parameter h

**Usage**

```
cv.ahm(y, x, powerh_path = NULL, metric = c("mse", "AICc"), num_major = 3,
  dist_minor = c(2, 2, 1), type = "weak", alpha = 0, lambda_seq = seq(0,
  5, 0.01), nfolds = NULL, mapping_type = c("power"), rep_gcv = 100)
```

**Arguments**

y	numeric vector
x	data.frame Note the column names of the x should be in the order of major components, minor components, and no interactions between major or minor components are needed.
powerh_path	if NULL, then the default is the vector: round(seq(0.001,2,length.out =15),3)
metric	"mse" or "AICc" the metric used in cross validation where the minimum is selected as the optimal
num_major	number of major components
dist_minor	the allocation of number of minor components nested under major components
type	heredity type, weak heredity is the current support type
alpha	0 is for the ridge in glmnet <a href="https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html">https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html</a>
lambda_seq	a numeric vector for the options of lambda used in ridge regression for estimating the initials
nfolds	used in cv.glmnet for initial value of parameters in the non-negative garrote method
mapping_type	the form of the coefficient function of major components in front of corresponding minor terms. Currently only support "power"
rep_gcv	the number of choices of tuning parameter used in the GCV selection

**Value**

Return a list

**Examples**

```
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
y = data_fat[,1]
powerh_path = round(seq(0.001,2,length.out =15),3)
num_major = 3; dist_minor = c(2,2,1)
res = cv.ahm(y, x, powerh_path=powerh_path, metric = "mse", num_major, dist_minor, type = "weak",
, alpha=0, lambda_seq=seq(0,5,0.01), nfolds=NULL, mapping_type = c("power"), rep_gcv=100)
object = res$metric_mse
```

---

```
design_simplex_centroid_design_3_major_component
```

*Design points for the simplex centroid design with 3 components*

---

**Description**

Design points for the simplex centroid design with 3 components

**Usage**

```
data(design_simplex_centroid_design_3_major_component)
```

**Format**

```
data.frame
```

**Examples**

```
data(design_simplex_centroid_design_3_major_component)
print(design_simplex_centroid_design_3_major_component)
```

---

```
enlist
```

*Create a list*

---

**Description**

Create a list

**Usage**

```
enlist(...)
```

**Arguments**

... object to be included as elements in the list

**Examples**

```
item = c(1:10)
enlist(item)
```



---

expand_interactions	<i>Expand the interaction terms for each subset group, say x11, x12, or c1, c2, c3</i>
---------------------	--

---

**Description**

Expand the interaction terms for each subset group, say x11, x12, or c1, c2, c3

**Usage**

```
expand_interactions(dat, sel_names)
```

**Arguments**

dat	data frame
sel_names	characters

**Examples**

```
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
expand_interactions (dat=x, sel_names=c("c1", "c2", "c3"))
```

---

find_condition_num	<i>Compute the conditional number of design matrix</i>
--------------------	--

---

**Description**

Compute the conditional number of design matrix

**Usage**

```
find_condition_num(x)
```

**Arguments**

x	matrix to be used in svd
---	--------------------------

**Examples**

```
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
find_condition_num (x)
```

---

mapping_function	<i>Mapping_function is a function to add the functional coefficients of major components in front of minor components terms</i>
------------------	---

---

### Description

Mapping\_function is a function to add the functional coefficients of major components in front of minor components terms

### Usage

```
mapping_function(x, num_major = 3, dist_minor = C(2, 2, 1),
  mapping_type = c("power"), powerh = 0)
```

### Arguments

x	data.frame Note the column names of the x should be in the order of major components, minor components, and no interactions are needed.
num_major	number of major components
dist_minor	the allocation of number of minor components nested under major components
mapping_type	the form of the coefficient function of major components in front of corresponding minor terms. Currently only support "power"
powerh	the power parameter used for the power function

### Value

data frame

### Examples

```
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
mapping_function(x=x, num_major=3, dist_minor=c(2,2,1), mapping_type = c("power"), powerh=0)
```

---

mymaximin	<i>The mymaximin function generates the matrix of maximin design points. It uses the simplex centroid design as the base design, then in a stochastic way sample the candidate design points generated by the function partition.</i>
-----------	---

---

### Description

This method is modified based on Prof. Bobby Gramacy's Computer Experiment lecture at Virginia Tech. [Prof. Gramacy's lecture website](#)

**Usage**

```
mymaximin(pool, n = 50, m = 3, iter = 1e+05, Xorig = NULL)
```

**Arguments**

pool,	partition the base design points provided to the function
n	numeric, sample size
m	numeric, 3 stands for 3 components, i.e. c1, c2, and c3
iter	numeric, iterations used in the stochastic sampling
Xorig	matrix, initial design points

**Value**

Return a matrix of the design points for the major components

**Examples**

```
# The case of unconstrained experiments
library(mixexp)
num_size = 8 # num points in the design for the major component
Xorig = as.matrix(SCD(3))
# all possible combinations sum to 1
pool_3d = partitions::compositions(1000, 3, include.zero = TRUE)/1000
res_C = mymaximin(pool=pool_3d, n=num_size, m=3, iter=1e5, Xorig=Xorig)
DesignPoints(res_C, cornerlabs = c("c3", "c2", "c1"), axislabs=c("c1", "c2", "c3"))

# The case of constrained experiments
library(mixexp)
num_size = 8 # num points in the design for the major component
# all possible combinations sum to 1
pool_3d = partitions::compositions(1000, 3, include.zero = TRUE)/1000
c1_min=0.2
c1_max=0.45
c2_min=0.4
c2_max=0.6
c3_min=0.1
c3_max=0.25
tmp = Xvert(nfac=3, lc=c(c1_min, c2_min, c3_min), uc=c(c1_max, c2_max, c3_max), ndm=1, pseudo=FALSE)
Xorig=tmp[c(1:6, 13), c(1:3)]
colnames(Xorig)=c("V1", "V2", "V3")
pool_3d = t(dplyr::filter(as.data.frame(t(as.matrix(pool_3d))), t(pool_3d)[,1] > c1_min &
  t(pool_3d)[,1] <= c1_max &
  t(pool_3d)[,2] > c2_min &
  t(pool_3d)[,2] <= c2_max &
  t(pool_3d)[,3] > c3_min &
  t(pool_3d)[,3] <= c3_max
)
)
res_C = mymaximin(pool=pool_3d, n=num_size, m=3, iter=1e5, Xorig=Xorig)
```

```
DesignPoints(res_C,cornerlabs = c("c3","c2","c1"),axislabs=c("c1","c2","c3")
             ,x1lower=c1_min,x2lower=c2_min,x3lower=c3_min
             ,x1upper=c1_max,x2upper=c2_max,x3upper=c3_max, pseudo=FALSE)
```

---

predict.ahm

*Predict method for the fitted ahm object*

---

### Description

Predict method for the fitted ahm object

### Usage

```
## S3 method for class 'ahm'
predict(object, newx, ...)
```

### Arguments

object	ahm object
newx	Matrix of new values for x at which predictions are to be made.
...	not used

### Value

predicted value(s) at newx

### Examples

```
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
y = data_fat[,1]
out = ahm (y, x, num_major = 3, dist_minor = c(2,2,1),
          type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfold = NULL,
          mapping_type = c("power"), powerh = h_tmp,
          rep_gcv=100)
predict(out)
```

---

predict.cv.ahm	<i>Predict method for the fitted cv.ahm object</i>
----------------	--

---

## Description

Predict method for the fitted cv.ahm object

## Usage

```
## S3 method for class 'cv.ahm'  
predict(object, newx, metric = "mse", ...)
```

## Arguments

object	cv.ahm object
newx	Matrix of new values for x at which predictions are to be made.
metric	"mse" or "aicc"
...	not used

## Value

Return a list

## Examples

```
data("pringles_fat")  
data_fat = pringles_fat  
h_tmp = 1.3  
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]  
y = data_fat[,1]  
powerh_path = round(seq(0.001, 2, length.out = 15), 3)  
num_major = 3; dist_minor = c(2, 2, 1)  
res = cv.ahm(y, x, powerh_path=powerh_path, metric = "mse", num_major, dist_minor, type = "weak",  
  alpha=0, lambda_seq=seq(0, 5, 0.01), nfolds=NULL, mapping_type = c("power"), rep_gcv=100)  
pred = predict(res)
```

---

```
pringles_candidates2search
```

*The candidate search points in the nonlinear optimization for the optimal value in the Pringles experiment*

---

**Description**

The candidate search points in the nonlinear optimization for the optimal value in the Pringles experiment

**Usage**

```
data(pringles_candidates2search)
```

**Format**

matrix

**Examples**

```
data(pringles_candidates2search)
print(pringles_candidates2search)
```

---

```
pringles_fat
```

*Pringles experiment data set with the percent of Fat as the response*

---

**Description**

Pringles experiment data set with the percent of Fat as the response

**Usage**

```
data(pringles_fat)
```

**Format**

data.frame

**References**

Kang, L., Joseph, V.R. and Brenneman, W.A. (2011). Design and modeling strategies for mixture-of-mixtures experiments. *Technometrics*, 53(2), 125–36. ([tandfonline](#))

**Examples**

```
data(pringles_fat)
print(pringles_fat)
```

---

pringles_hardness	<i>Pringles experiment data set with the Hardness as the response</i>
-------------------	---

---

**Description**

Pringles experiment data set with the Hardness as the response

**Usage**

```
data(pringles_hardness)
```

**Format**

```
data.frame
```

**References**

Kang, L., Joseph, V.R. and Brenneman, W.A. (2011). Design and modeling strategies for mixture-of-mixtures experiments. *Technometrics*, 53(2), 125–36. ([tandfonline](#))

**Examples**

```
data(pringles_hardness)
print(pringles_hardness)
```

---

summary.ahm	<i>Summary method for the fitted ahm object</i>
-------------	---

---

**Description**

Summary method for the fitted ahm object

**Usage**

```
## S3 method for class 'ahm'
summary(object, ...)
```

**Arguments**

object	fitted ahm object
...	not used

**Examples**

```
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1", "c2", "c3", "x11", "x12", "x21", "x22")]
y = data_fat[,1]
out = ahm (y, x, num_major = 3, dist_minor = c(2,2,1),
          type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfold = NULL,
          mapping_type = c("power"), powerh = h_tmp,
          rep_gcv=100)
summary(out)
```



# Index

## \* datasets

- coating, [4](#)
- design\_simplex\_centroid\_design\_3\_major\_component, [8](#)
- pringles\_candidates2search, [14](#)
- pringles\_fat, [14](#)
- pringles\_hardness, [15](#)

ahm, [2](#)

check\_col\_correlation, [3](#)

coating, [4](#)

coef.ahm, [4](#)

coef.cv.ahm, [5](#)

compute\_aicc, [6](#)

cv.ahm, [6](#)

design\_simplex\_centroid\_design\_3\_major\_component, [8](#)

enlist, [8](#)

expand\_interactions, [9](#)

find\_condition\_num, [9](#)

mapping\_function, [10](#)

mymaximin, [10](#)

predict.ahm, [12](#)

predict.cv.ahm, [13](#)

pringles\_candidates2search, [14](#)

pringles\_fat, [14](#)

pringles\_hardness, [15](#)

summary.ahm, [15](#)