

# Package ‘activegp’

October 12, 2022

**Type** Package

**Title** Gaussian Process Based Design and Analysis for the Active Subspace Method

**Version** 1.1.0

**Date** 2022-06-27

**Author** Nathan Wycoff, Mickael Binois

**Maintainer** Nathan Wycoff <nathan.wycoff@georgetown.edu>

**Description** The active subspace method is a sensitivity analysis technique that finds important linear combinations of input variables for a simulator. This package provides functions allowing estimation of the active subspace without gradient information using Gaussian processes as well as sequential experimental design tools to minimize the amount of data required to do so. Implements Wycoff et al. (2019) <[arXiv:1907.11572](https://arxiv.org/abs/1907.11572)>.

**License** BSD\_3\_clause + file LICENSE

**Depends** R (>= 3.4.0)

**Imports** Rcpp (>= 0.12.18), hetGP (>= 1.1.1), lhs, numDeriv, methods, MASS, RcppProgress

**LinkingTo** Rcpp, RcppArmadillo, RcppProgress

**RoxygenNote** 7.1.2

**Suggests** testthat

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-06-27 21:00:02 UTC

## R topics documented:

activegp . . . . .	2
as.matrix.const_C . . . . .	4
C_GP . . . . .	5
C_GP_ci . . . . .	7
C_tr . . . . .	8
C_var . . . . .	9

C_var2 . . . . .	11
domain_to_R . . . . .	12
domain_to_unit . . . . .	13
grad_est_subspace . . . . .	13
logLikHessian . . . . .	14
Lt_GP . . . . .	14
n11_2_01 . . . . .	15
plot.const_C . . . . .	15
print.const_C . . . . .	16
subspace_dist . . . . .	16
update.const_C . . . . .	17
update_C2 . . . . .	18

<b>Index</b>	<b>19</b>
--------------	-----------

---

activegp	<i>Package activegp</i>
----------	-------------------------

---

## Description

Active subspace estimation with Gaussian processes

## Details

The primary function for analysis of the active subspace given some set of function evaluations is C\_GP.

C\_var, C\_var2, and C\_tr give three possible acquisition functions for sequential design. Either C\_var or C\_var2 is recommended, see Wycoff et al for details and the example below for usage.

## Author(s)

Nathan Wycoff, Mickael Binois

## References

N. Wycoff, M. Binois, S. Wild (2019+), Sequential Learning of Active Subspaces, preprint.  
 P. Constantine (2015) Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies, SIAM Spotlights

## Examples

```
#####
### Sequential learning of active subspaces
#####
library(hetGP); library(lhs)
set.seed(42)

nvar <- 2
```

```

n <- 20
nits <- 20

# theta gives the subspace direction
f <- function(x, theta, nugget = 1e-6){
  if(is.null(dim(x))) x <- matrix(x, 1)
  xact <- cos(theta) * x[,1] - sin(theta) * x[,2]
  return(hetGP::f1d(xact) + rnorm(n = nrow(x), sd = rep(nugget, nrow(x))))
  return(100*erf((xact + 0.5)*5) + hetGP::f1d(xact) +
    rnorm(n = nrow(x), sd = rep(nugget, nrow(x))))
}

theta_dir <- pi/6
act_dir <- c(cos(theta_dir), -sin(theta_dir))

# Create design of experiments and initial GP model
design <- X <- matrix(signif(maximinLHS(n, nvar), 2), ncol = nvar)
response <- Y <- apply(design, 1, f, theta = theta_dir)
model <- mleHomGP(design, response, lower = rep(1e-4, nvar),
  upper = rep(0.5, nvar), known = list(g = 1e-6, beta0 = 0))

C_hat <- C_GP(model)

ngrid <- 51
xgrid <- seq(0, 1, , ngrid)
Xgrid <- as.matrix(expand.grid(xgrid, xgrid))
filled.contour(matrix(f(Xgrid, theta = theta_dir), ngrid))
ssd <- rep(NA, nits)

# Main loop
for(nit in 1:nits) {
  cat(nit)
  cat(" ")

  af <- function(x, C) C_var(C, x, grad = FALSE)
  af_gr <- function(x, C) C_var(C, x, grad = TRUE)
  Ctr_grid <- apply(Xgrid, 1, af, C = C_hat) # CVAR

  # Best candidate point
  opt_cand <- matrix(Xgrid[which.max(Ctr_grid),], 1)

  # Refine with gradient based optimization
  opt <- optim(opt_cand, af, af_gr, method = 'L-BFGS-B', lower = rep(0, nvar), C = C_hat,
    upper = rep(1, nvar), hessian = TRUE,
    control = list(fnscale=-1, trace = 0, maxit = 10))

  # Criterion surface with best initial point and corresponding local optimum
  filled.contour(matrix(Ctr_grid, ngrid), color.palette = terrain.colors,
    plot.axes = {axis(1); axis(2); points(X, pch = 20);
      points(opt_cand, pch = 20, col = 'blue');
      points(opt$par, pch = 20, col = 'red')}})

  X <- rbind(X, opt$par)

```

```

Ynew <- f(opt$par, theta = theta_dir)
Y <- c(Y, Ynew)
model <- update(model, Xnew = opt$par, Znew = Ynew)

## periodically restart model fit
if(nit %% 5 == 0){
  mod2 <- mleHomGP(X = list(X0 = model$X0, Z0 = model$Z0, mult = model$mult), Z = model$Z,
                  known = model$used_args$known, lower = model$used_args$lower,
                  upper = model$used_args$upper)
  if(mod2$ll > model$ll) model <- mod2
}
C_hat <- C_GP(model)
# Compute subspace distance
ssd[nit] <- subspace_dist(C_hat, matrix(act_dir, nrow = nvar), r = 1)
}
plot(ssd, type = 'b')

```

---

as.matrix.const\_C      *Extract Matrix*

---

### Description

Given a const\_C object, extracts the actual matrix itself.

### Usage

```

## S3 method for class 'const_C'
as.matrix(x, ...)

```

### Arguments

x	A const_C object with field 'mat'.
...	Additional parameters. Not used.

### Value

The mat entry of C, a matrix.

C\_GP

*C matrix closed form expression for a GP.***Description**

Computes the integral over the input domain of the outer product of the gradients of a Gaussian process. The corresponding matrix is the C matrix central in active subspace methodology.

**Usage**

```
C_GP(
  modelX,
  y,
  measure = "lebesgue",
  xm = NULL,
  xv = NULL,
  S = NULL,
  verbose = TRUE
)
```

**Arguments**

modelX	This may be either 1) a homGP or hetGP GP model, see <a href="#">hetGP-package</a> containing, e.g., a vector of thetas, type of covariance ct, an inverse covariance matrix Ki, a design matrix X0, and response vector Z0. 2) A matrix of design locations, in which case a vector of responses must be given as the y argument, and this function will fit a default model for you.
y	A vector of responses corresponding to the design matrix; may be omitted if a GP fit is provided in the modelX argument.
measure	One of c("lebesgue", "gaussian", "trunc_gaussian", "sample", "discrete"), indicating the probability distribution with respect to which the input points are drawn in the definition of the active subspace. "lebesgue" uses the Lebesgue or Uniform measure over the unit hypercube $[0,1]^d$ . "gaussian" uses a Gaussian or Normal distribution, in which case xm and xv should be specified. "trunc_gaussian" gives a truncated Gaussian or Normal distribution over the unit hypercube $[0,1]^d$ , in which case xm and xv should be specified. "sample" gives the Sample or Empirical measure (dirac deltas located at each design point), which is equivalent to calculating the average expected gradient outer product at the design points. "discrete" gives a measure which puts equal weight at points in the input space specified via the S parameter, which should be a matrix with one row for each atom of the measure.
xm	If measure is "gaussian" or "trunc_gaussian", gives the mean vector.
xv	If measure is "gaussian" or "trunc_gaussian", gives the marginal variance vector. The covariance matrix is assumed to be diagonal.
S	If measure is "discrete", gives the locations of the measure's atoms.
verbose	Should we print progress?

**Value**

a const\_C object with elements

- model: GP model provided or estimated;
- mat: C matrix estimated;
- Wij: list of W matrices, of size number of variables;
- ct: covariance type (1 for "Gaussian", 2 for "Matern3\_2", 3 for "Matern5\_2").

**References**

N. Wycoff, M. Binois, S. Wild (2019+), Sequential Learning of Active Subspaces, preprint.

P. Constantine (2015), Active Subspaces, Philadelphia, PA: SIAM.

**See Also**

[print.const\\_C](#), [plot.const\\_C](#)

**Examples**

```
#####
### Active subspace of a Gaussian process
#####

library(hetGP); library(lhs)
set.seed(42)

nvar <- 2
n <- 100

# theta gives the subspace direction
f <- function(x, theta, nugget = 1e-3){
  if(is.null(dim(x))) x <- matrix(x, 1)
  xact <- cos(theta) * x[,1] - sin(theta) * x[,2]
  return(hetGP::f1d(xact) + rnorm(n = nrow(x), sd = rep(nugget, nrow(x))))
}

theta_dir <- pi/6
act_dir <- c(cos(theta_dir), -sin(theta_dir))

# Create design of experiments and initial GP model
design <- X <- matrix(signif(maximinLHS(n, nvar), 2), ncol = nvar)
response <- Y <- apply(design, 1, f, theta = theta_dir)
model <- mleHomGP(design, response, known = list(beta0 = 0))

C_hat <- C_GP(model)

# Subspace distance to true subspace:
print(subspace_dist(C_hat, matrix(act_dir, nrow = nvar), r = 1))
plot(design %% eigen(C_hat$mat)$vectors[,1], response,
```

```

    main = "Projection along estimated active direction")
plot(design %% eigen(C_hat$mat)$vectors[,2], response,
     main = "Projection along estimated inactive direction")

# For other plots:
# par(mfrow = c(1, 3)) # uncomment to have all plots together
plot(C_hat)
# par(mfrow = c(1, 1)) # restore graphical window

```

---

C\_GP\_ci

*CI on Eigenvalues via Monte Carlo/GP*


---

## Description

CI on Eigenvalues via Monte Carlo/GP

## Usage

```
C_GP_ci(model, B = 100)
```

## Arguments

model	A homGP model
B	Monte Carlo iterates

## Value

A list with elements ci giving 95

## Examples

```

#####
## Example of uncertainty quantification on C estimate
#####
library(hetGP); library(lhs)
set.seed(42)

nvar <- 2
n <- 20
nits <- 20

# theta gives the subspace direction
f <- function(x, theta, nugget = 1e-6){
  if(is.null(dim(x))) x <- matrix(x, 1)
  xact <- cos(theta) * x[,1] - sin(theta) * x[,2]
  return(hetGP::f1d(xact) + rnorm(n = nrow(x), sd = rep(nugget, nrow(x))))
}

```

```

theta_dir <- pi/6
act_dir <- c(cos(theta_dir), -sin(theta_dir))

# Create design of experiments and initial GP model
design <- X <- matrix(signif(maximinLHS(n, nvar), 2), ncol = nvar)
response <- Y <- apply(design, 1, f, theta = theta_dir)
model <- mleHomGP(design, response, known = list(beta0 = 0))

res <- C_GP_ci(model)

plot(c(1, 2), log(c(mean(res$eigen_draws[,1]), mean(res$eigen_draws[,2]))),
      ylim = range(log(res$eigen_draws)), ylab = "Eigenvalue", xlab = "Index")
      segments(1, log(res$ci[1,1]), 1, log(res$ci[2,1]))
      segments(2, log(res$ci[1,2]), 2, log(res$ci[2,2]))

```

---

C\_tr

*Expected variance of trace of C*


---

### Description

Expected variance of trace of C

### Usage

```
C_tr(C, xnew, grad = FALSE)
```

### Arguments

C	A const_C object, the result of a call to <code>C_GP</code> .
xnew	The new design point
grad	If FALSE, calculate variance of trace after update. If TRUE, returns the gradient.

### Value

A real number giving the expected variance of the trace of C given the current design.

### References

N. Wycoff, M. Binois, S. Wild (2019+), Sequential Learning of Active Subspaces, preprint.



## Examples

```
#####
### Variance of trace criterion landscape
#####
library(hetGP)
set.seed(42)
nvar <- 2
n <- 20

# theta gives the subspace direction
f <- function(x, theta = pi/6, nugget = 1e-6){
  if(is.null(dim(x))) x <- matrix(x, 1)
  xact <- cos(theta) * x[,1] - sin(theta) * x[,2]
  return(hetGP::f1d(xact) +
         rnorm(n = nrow(x), sd = rep(nugget, nrow(x))))
}

design <- matrix(signif(runif(nvar*n), 2), ncol = nvar)
response <- apply(design, 1, f)
model <- mleHomGP(design, response, lower = rep(1e-4, nvar),
                 upper = rep(0.5,nvar), known = list(g = 1e-4))

C_hat <- C_GP(model)

ngrid <- 101
xgrid <- seq(0, 1,, ngrid)
Xgrid <- as.matrix(expand.grid(xgrid, xgrid))
filled.contour(matrix(f(Xgrid), ngrid))

Ctr_grid <- apply(Xgrid, 1, C_tr, C = C_hat)
filled.contour(matrix(Ctr_grid, ngrid), color.palette = terrain.colors,
                    plot.axes = {axis(1); axis(2); points(design, pch = 20)})
```

---

C\_var

*Element-wise  $C_{n+1}$  variance*

---

### Description

Element-wise  $C_{n+1}$  variance

### Usage

```
C_var(C, xnew, grad = FALSE)
```

### Arguments

C                    A const\_C object, the result of a call to [C\\_GP](#).

xnew            The new design point  
 grad            If FALSE, calculate variance of update. If TRUE, returns the gradient.

### Value

A real number giving the expected elementwise variance of C given the current design.

### References

N. Wycoff, M. Binois, S. Wild (2019+), Sequential Learning of Active Subspaces, preprint.

### Examples

```
#####
### Norm of the variance of C criterion landscape
#####
library(hetGP)
set.seed(42)
nvar <- 2
n <- 20

# theta gives the subspace direction
f <- function(x, theta = pi/6, nugget = 1e-6){
  if(is.null(dim(x))) x <- matrix(x, 1)
  xact <- cos(theta) * x[,1] - sin(theta) * x[,2]
  return(hetGP::f1d(xact)
    + rnorm(n = nrow(x), sd = rep(nugget, nrow(x))))
}

design <- matrix(signif(runif(nvar*n), 2), ncol = nvar)
response <- apply(design, 1, f)
model <- mleHomGP(design, response, lower = rep(1e-4, nvar),
  upper = rep(0.5,nvar), known = list(g = 1e-4))

C_hat <- C_GP(model)

ngrid <- 51
xgrid <- seq(0, 1,, ngrid)
Xgrid <- as.matrix(expand.grid(xgrid, xgrid))
filled.contour(matrix(f(Xgrid), ngrid))

cvar_crit <- function(C, xnew){
  return(sqrt(sum(C_var(C, xnew)^2)))
}

Cvar_grid <- apply(Xgrid, 1, cvar_crit, C = C_hat)
filled.contour(matrix(Cvar_grid, ngrid), color.palette = terrain.colors,
  plot.axes = {axis(1); axis(2); points(design, pch = 20)})
```

---

C\_var2 *Alternative Variance of Update*

---

**Description**

Defined as  $E[(C - E[C])^2]$ , where  $A^2 = AA$  (not elementwise multiplication).

**Usage**

```
C_var2(C, xnew, grad = FALSE)
```

**Arguments**

C	A const_C object, the result of a call to <code>C_GP</code> .
xnew	The new design point
grad	If FALSE, calculate variance of update. If TRUE, returns the gradient.

**Value**

A real number giving the expected variance of C defined via matrix multiplication given the current design.

**References**

N. Wycoff, M. Binois, S. Wild (2019+), Sequential Learning of Active Subspaces, preprint.

**Examples**

```
#####
### Norm of the variance of C criterion landscape
#####

library(hetGP)
set.seed(42)
nvar <- 2
n <- 20

# theta gives the subspace direction
f <- function(x, theta = pi/6, nugget = 1e-6){
  if(is.null(dim(x))) x <- matrix(x, 1)
  xact <- cos(theta) * x[,1] - sin(theta) * x[,2]
  return(hetGP::f1d(xact) + rnorm(n = nrow(x), sd = rep(nugget, nrow(x))))
}

design <- matrix(signif(runif(nvar*n), 2), ncol = nvar)
response <- apply(design, 1, f)
model <- mleHomGP(design, response, lower = rep(1e-4, nvar),
                 upper = rep(0.5,nvar), known = list(g = 1e-4))
```

```
C_hat <- C_GP(model)

ngrid <- 51
xgrid <- seq(0, 1,, ngrid)
Xgrid <- as.matrix(expand.grid(xgrid, xgrid))
filled.contour(matrix(f(Xgrid), ngrid))

cvar_crit <- function(C, xnew){
  return(sqrt(sum(C_var(C, xnew)^2)))
}

Cvar_grid <- apply(Xgrid, 1, cvar_crit, C = C_hat)
filled.contour(matrix(Cvar_grid, ngrid), color.palette = terrain.colors,
  plot.axes = {axis(1); axis(2); points(design, pch = 20)})
```

---

domain\_to\_R

*Rectangular Domain -> Unbounded Domain*

---

### Description

Given an  $m$  dimensional function whose inputs live in bounded intervals  $[a_1, b_1]$ , ...,  $[a_m, b_m]$ , return a wrapped version of the function whose inputs live in  $\mathbb{R}^m$ . Transformed using the logit function.

### Usage

```
domain_to_R(f, domain)
```

### Arguments

**f**                    The function to wrap, should have a single vector-valued input.

**domain**                A list of real tuples, indicating the original domain of the function.

### Value

A function wrapping  $f$ .

---

domain_to_unit	<i>Change a function's inputs to live in [-1, 1]</i>
----------------	--

---

**Description**

Given an  $m$  dimensional function whose inputs live in bounded intervals  $[a_1, b_1], \dots, [a_m, b_m]$ , return a wrapped version of the function whose inputs live in  $[-1, 1], \dots, [-1, 1]$ .

**Usage**

```
domain_to_unit(f, domain)
```

**Arguments**

<code>f</code>	The function to wrap, should have a single vector-valued input.
<code>domain</code>	A list of real tuples, indicating the original domain of the function.

**Value**

A function wrapping `f`.

---

grad_est_subspace	<i>Estimate the Active Subspace of a Cheap Function using Gradients</i>
-------------------	---

---

**Description**

Looks between  $[-1, 1]$

**Usage**

```
grad_est_subspace(f, r, m, M = NULL, scale = FALSE)
```

**Arguments**

<code>f</code>	The function to eval
<code>r</code>	The max dim of the active subspace
<code>m</code>	The dimension of the underlying/embedding space.
<code>M</code>	optional budget of evaluations, default to $2 * r * \log(m)$
<code>scale</code>	Scale all gradients to have norm 1?

**Value**

A list with `sub`, the active subspace, `sv`, the singular values (all  $m$  of them), `fs`, which gives function values, `gs`, function grads, and `X`, which gives sampled locations.

---

logLikHessian	<i>Hessian of the log-likelihood with respect to lengthscales hyperparameters Works for homGP and hetGP models from the hetGP package for now.</i>
---------------	--

---

**Description**

Hessian of the log-likelihood with respect to lengthscales hyperparameters Works for homGP and hetGP models from the hetGP package for now.

**Usage**

```
logLikHessian(model)
```

**Arguments**

model	homGP model
-------	-------------

**Value**

A matrix giving the Hessian of the GP loglikelihood.

---

Lt_GP	<i>Active Subspace Prewarping</i>
-------	-----------------------------------

---

**Description**

Computes a matrix square root of  $C = Lt$

**Usage**

```
Lt_GP(..., C)
```

**Arguments**

...	Parameters to be passed to C_GP, if C was not provided.
C	the result of a call to C_GP. If provided, all other arguments are ignored.

**Value**

The matrix Lt which can be used for sensitivity prewarping, i.e. by computing  $Xw = X$

---

n11_2_01	<i>f:[-1, 1] -&gt; R Becomes f:[0,1] -&gt; R</i>
----------	--

---

**Description**

f:[-1, 1] -> R Becomes f:[0,1] -> R

**Usage**

n11\_2\_01(f)

**Arguments**

f                    initial function

**Value**

The same function with domain shifted.

---

plot.const_C	<i>Plot const_C objectc</i>
--------------	-----------------------------

---

**Description**

Plot const\_C objectc

**Usage**

```
## S3 method for class 'const_C'
plot(x, output = c("all", "matrix", "logvals", "projfn"), ...)
```

**Arguments**

x	A const_C object, the result of a call to C_GP
output	one of "image" (image of the C matrix), "logvals" (log-eigen values), "projfn" projected function on first eigen vector or all plots at once (default).
...	Additional parameters. Not used.

---

<code>print.const_C</code>	<i>Print const_C objects</i>
----------------------------	------------------------------

---

**Description**

Print const\_C objects

**Usage**

```
## S3 method for class 'const_C'
print(x, ...)
```

**Arguments**

<code>x</code>	A const_C object, the result of a call to C_GP
<code>...</code>	Additional parameters. Not used.

---

<code>subspace_dist</code>	<i>Get the distance between subspaces defined as the ranges of A and B</i>
----------------------------	--

---

**Description**

Get the distance between subspaces defined as the ranges of A and B

**Usage**

```
subspace_dist(A, B, r)
```

**Arguments**

<code>A</code>	A matrix or const_C object.
<code>B</code>	Another matrix with the same number of rows as A, or const_C object of the same dimension.
<code>r</code>	A scalar integer, the dimension of the subspace to compare (only necessary if either A or B is a const_C object).

**Value**

A nonnegative scalar giving the cosine of the first principle angle between the two subspaces.



---

update.const\_C            *C update with new observations*

---

## Description

Update Constantine's C with new point(s) for a GP

## Usage

```
## S3 method for class 'const_C'
update(object, Xnew, Znew, ...)
```

## Arguments

object	A const_C object, the result of a call to the C_GP function.
Xnew	matrix (one point per row) corresponding to the new designs
Znew	vector of size nrow(Xnew) for the new responses at Xnew
...	not used (for consistency of update method)

## Value

The updated const\_C object originally provided.

## See Also

[C\\_GP](#) to generate const\_C objects from [mleHomGP](#) objects; [update\\_C2](#) for an update using faster expressions.

## Examples

```
#####
### Active subspace of a Gaussian process
#####
library(hetGP); library(lhs)
set.seed(42)

nvar <- 2
n <- 100

# theta gives the subspace direction
f <- function(x, theta, nugget = 1e-3){
  if(is.null(dim(x))) x <- matrix(x, 1)
  xact <- cos(theta) * x[,1] - sin(theta) * x[,2]
  return(hetGP::f1d(xact) +
         rnorm(n = nrow(x), sd = rep(nugget, nrow(x))))
}
```

```

theta_dir <- pi/6
act_dir <- c(cos(theta_dir), -sin(theta_dir))

# Create design of experiments and initial GP model
design <- X <- matrix(signif(maximinLHS(n, nvar), 2), ncol = nvar)
response <- Y <- apply(design, 1, f, theta = theta_dir)
model <- mleHomGP(design, response, known = list(beta0 = 0))

C_hat <- C_GP(model)

print(C_hat)
print(subspace_dist(C_hat, matrix(act_dir, nrow = nvar), r = 1))

# New designs
Xnew <- matrix(runif(2), 1)
Znew <- f(Xnew, theta_dir)

C_new <- update(C_hat, Xnew, Znew)
print(C_new)
subspace_dist(C_new, matrix(act_dir, nrow = nvar), r = 1)

```

---

update\_C2

*Update Constantine's C, using update formula*


---

## Description

Update Constantine's C, using update formula

## Usage

```
update_C2(C, xnew, ynew)
```

## Arguments

C	A const_C object, the result of a call to <code>C_GP</code> .
xnew	The new design point
ynew	The new response

## Value

Updated C matrix, a const\_C object.

## References

N. Wycoff, M. Binois, S. Wild (2019+), Sequential Learning of Active Subspaces, preprint.

# Index

activegp, [2](#)  
as.matrix.const\_C, [4](#)  
  
C\_GP, [5](#), [8](#), [9](#), [11](#), [17](#), [18](#)  
C\_GP\_ci, [7](#)  
C\_tr, [8](#)  
C\_var, [9](#)  
C\_var2, [11](#)  
  
domain\_to\_R, [12](#)  
domain\_to\_unit, [13](#)  
  
grad\_est\_subspace, [13](#)  
  
logLikHessian, [14](#)  
Lt\_GP, [14](#)  
  
mleHomGP, [17](#)  
  
n11\_2\_01, [15](#)  
  
plot.const\_C, [6](#), [15](#)  
print.const\_C, [6](#), [16](#)  
  
subspace\_dist, [16](#)  
  
update.const\_C, [17](#)  
update\_C2, [17](#), [18](#)