

# Package ‘farmR’

February 16, 2010

**Type** Package

**Title** Mixed Integer model of Arable Farms

**Version** 1.1

**Date** 2010-02-16

**Author** Ira R. Cooke

**Maintainer** Ira R. Cooke <iracooke@gmail.com>

**Description** A mixed integer description of an arable farm. Finds optimal farming plans given economic and social preference information.

**License** GPL-3

**Suggests** sp

**LazyLoad** yes

**Depends** rJava,methods

**URL** <http://code.google.com/p/javawfm/>

**Repository** CRAN

**Date/Publication** 2010-02-16 12:04:28

## R topics documented:

CompositeFarm . . . . .	2
CompositeFarm-class . . . . .	3
constrainArea-methods . . . . .	4
cropArea-methods . . . . .	4
cropNames-methods . . . . .	5
defaultArableFarmParameters . . . . .	5
document-methods . . . . .	6
eo-methods . . . . .	6
Farm . . . . .	6

Farm-class . . . . .	7
FarmParameters . . . . .	8
FarmParameters-class . . . . .	9
FarmRepresentation-class . . . . .	10
guts-methods . . . . .	11
isSolved-methods . . . . .	12
model-methods . . . . .	12
objectiveNames-methods . . . . .	12
ObjectiveParameters . . . . .	13
ObjectiveParameters-class . . . . .	14
objectiveScaleFactors-methods . . . . .	15
objectiveValues-methods . . . . .	15
Parameters-class . . . . .	15
profit-methods . . . . .	16
SBFactories . . . . .	16
setInputCost-methods . . . . .	17
solvelp-methods . . . . .	17
<b>Index</b>	<b>18</b>

---

CompositeFarm	<i>CompositeFarm</i>
---------------	----------------------

---

## Description

Creates Composite Farm Objects

## Usage

```
CompositeFarm(farmParams, mou = NULL, mouweights = c(1), soildata, SBFactories = NU
```

## Arguments

<code>farmParams</code>	A <code>FarmParameters</code> object to be used for all component farms
<code>mou</code>	A single <code>ObjectiveParameters</code> object or a list of such objects specifying the objective parameters used to define the component farms. If a list is supplied the total number of component farms will be the length of this list multiplied by the number of soil classes.
<code>mouweights</code>	If supplied this specifies the weightings to be used for each of the objective parameters supplied in the <code>mou</code> argument. The lengths of <code>mouweights</code> and <code>mou</code> must be equal.
<code>soildata</code>	A <code>data.frame</code> or <code>SpatialDataFrame</code> with a single row and items named ( <code>RF,X0.5,X0.75,X1.0,X1.25,X1.5,X1.75,X2.0,X2.25,X2.5</code> ) representing the yearly rainfall in mm and the proportion of land in each of the soiltype classes. All items must be present.

`SBFactories` SpatialDataFrame object with the Locations of sugarbeet factories to be used to calculate sugarbeet transport costs. `data(SBFactories)` provides an appropriate dataframe. Requires the `sp` package

`haulagePerTonnePerKm`  
Cost of sugarbeet haulage in pounds

`maxSBHaulageDistance`  
Maximum allowable haulage distance for sugarbeet in km

**Details**

Constructs itself from component farms, each of which will represent a particular combination of soiltype and objective parameters. The total number of component farms constructed will be `length(mou)*length(soildata)`

**Note**

If using spatial data as arguments (eg for `soildata` or `SBFactories`) this requires the `sp` package.

**Author(s)**

Ira Cooke

**Examples**

```
fp=FarmParameters()
op=ObjectiveParameters()
localConditions=data.frame(RF=600.0,X0.5=0,X0.75=0.1,X1.0=0.1,X1.25=0.5,X1.5=0.1,X1.75=0,X2.
cfarm=CompositeFarm(fp,op,1.0,localConditions)
solvelp(cfarm)
show(cfarm)
```

---

CompositeFarm-class

*Class "CompositeFarm"*

---

**Description**

A collection of Farm objects representing a farm with heterogenous soil or objective types

**Objects from the Class**

Objects can be created by calls to the function `CompositeFarm`.

**Slots**

`model`: Object of class "`jobjRef`" which is a reference to the internal java object representing the farm

`cropNames`: Object of class "`vector`" representing the crop names

**Extends**

Class "[FarmRepresentation](#)", directly.

**Author(s)**

Ira Cooke

**See Also**

See also [Farm](#) and [FarmRepresentation](#)

**Examples**

```
showClass("CompositeFarm")
```

---

```
constrainArea-methods
```

*Methods for Function constrainArea in Package 'farmR'*

---

**Description**

Constrain the area of a specific crop type

**Methods**

**farm = "FarmRepresentation", cropName = "character", lb = "numeric"** Constrain the area of a particular crop on a farm to be the value lb. If an optional ub argument is supplied the constraint is to hold the area of the crop between lb and ub

---

```
cropArea-methods
```

*Methods for Function cropArea in Package 'farmR'*

---

**Description**

Methods for function cropArea in Package 'farmR'

**Methods**

**farm = "FarmRepresentation", cropName = "character"** Get the area cropped

---

cropNames-methods *Extract lists of crop names*

---

### Description

Methods to extract lists of crop names

### Methods

**object = "FarmParameters"** Get a string vector of crop names defined in the Parameters object

**object = "FarmRepresentation"** Get a string vector of crop names defined in the FarmRepresentation object

---

defaultArableFarmParameters

*Functions to create default Parameters*

---

### Description

Create FarmParameters or ObjectiveParameters objects from default values

### Usage

```
defaultArableFarmParameters()  
defaultArableObjectiveParameters()
```

### Value

Both functions return an object of class Parameters. defaultArableFarmParameters returns a FarmParameters object defaultArableObjectiveParameters returns a ObjectiveParameters object

### Author(s)

Ira Cooke

### See Also

See also [ObjectiveParameters](#) and [FarmParameters](#)

### Examples

```
op=defaultArableObjectiveParameters()  
fp=defaultArableFarmParameters()  
show(op) # Dump Objective parameters to screen in xml format  
show(fp) # Dump the Farm Parameters to screen in xml format
```

---

document-methods     *Methods to extract java references to document objects*

---

### Description

Extract the underlying java xml document representing a `Parameters` object

### Methods

**params = "Parameters"** Extract java references to xml document objects

---

eo-methods     *Methods for Function eo in Package 'farmR'*

---

### Description

Extract values of the "enterprise output" from `FarmRepresentation` objects

### Methods

**farm = "FarmRepresentation"** Get the Enterprise output of a `FarmRepresentation` Object

---

Farm     *Farm*

---

### Description

Create `Farm` objects

### Usage

```
Farm(farm.params = defaultArableFarmParameters(), obj.params = NULL)
```

### Arguments

`farm.params`     An object of class `FarmParameters` or the name of an xml file from which to read the parameters

`obj.params`     An object of class `ObjectiveParameters` or the name an xml file from which to read the parameters

### Value

Returns a `Farm` object

**Author(s)**

Ira Cooke

**See Also**See Also [CompositeFarm](#)**Examples**

```
fm=Farm() # Create a default Farm object
solvelp(fm) # Solve the farm
show(fm)
```

---

 Farm-class

---

 Class "Farm"
 

---

**Description**

Represents a farm model for a single farm

**Objects from the Class**Objects can be created by calls to [Farm](#)**Slots**

**model:** Object of class "jobjRef" which is a reference to the internal java object representing the farm

**cropNames:** Object of class "vector" representing the crop names

**Extends**Class "[FarmRepresentation](#)", directly.**Methods**

**set** signature(farm = "Farm", input = "jobjRef"): Sets the multi objective preferences using an ObjectiveParameters document object, typically created using `document (ObjectiveParameters)`

**set** signature(farm = "Farm", input = "ObjectiveParameters"): Sets the multi objective preferences using an ObjectiveParameters object, typically created using `ObjectiveParameters ()`

**set** signature(farm = "Farm", input = "character"): Sets the multi objective preferences using an xml preference file

**Author(s)**

Ira Cooke

**See Also**

See also [CompositeFarm](#) and [FarmRepresentation](#)

**Examples**

```
showClass("Farm")
```

---

FarmParameters      *Create and set values on FarmParameters objects*

---

**Description**

Create FarmParameters objects from xml files or from the defaults and customise by setting parameters

**Usage**

```
FarmParameters(file = defaultArableFarmParameters())
setSolverType(params, solver)
getSolverType(params)
setPriceForCrop(params, newPrice, cropName)
setSubsidyForCrop(params, newSubsidy, cropName)
setRelativePriceForCrop(params, cropName, val)
setRelativeRotationPenalties(params, val)
setRelativeLabourRequirements(params, val)
setRelativeYieldForCrop(params, cropName, val)
setRelativeCost(params, val, costType="Input", inputName="N fertiliser")
```

**Arguments**

file	A character string with the full path to an xml document containing the farm parameters
params	A FarmParameters object
solver	Character string with the name of the core solver to be used. Available values are "cbc" or "glpk"
newPrice	Numeric value of the absolute price per tonne for a crop
cropName	Character string with the name of the crop. A list of possible values can be obtained with <code>cropNames(params)</code>
newSubsidy	Numeric value of the absolute subsidy value per hectare for a crop
val	Numeric multiplier to be applied to the existing value of a parameter. A value of 1.0 leaves the parameter unchanged.
costType	A character string with one of the following values "Input", "Machinery", "Fuel", "Labour", corresponding to different types of farming costs

`inputName` Name of an input cost (required for `costType="Input"`). Available values depend on which values have been defined in `params` but defaults are "N fertiliser", "P fertiliser", "K fertiliser", "BGHerbicide", "WOHerbicide", "Seed Amount of crop"

### Details

When creating `FarmParameters` objects `xml` document provided is parsed and stored internally as a java object. The raw `xml` can be viewed using the `show` command. Values in the `xml` document can be changed using the various `set` commands.

Use functions of the form `setXXX()` to set absolute values Use functions of the form `setRelativeXXX()` to set a multiplier on an existing value

### Value

`FarmParameters` returns an object of the class `FarmParameters`

### Author(s)

Ira Cooke

### Examples

```
params=FarmParameters()
show(params) # Dumps xml to the screen. It may be more useful to write to a file using the w
```

---

`FarmParameters-class`

*Class "FarmParameters"*

---

### Description

Encapsulates an `xml` document containing detailed parameter information for the Farm model

### Objects from the Class

Objects can be created by calls to the function `FarmParameters`.

### Slots

`document`: Object of class "jobjRef": a reference to an underlying java object containing the `xml` file

### Extends

Class "`Parameters`", directly.

**Methods**

**cropNames** signature(object = "FarmParameters"): returns a vector of strings with the names of crops defined in the parameter file

**Author(s)**

Ira Cooke

**See Also**

See also [ObjectiveParameters](#)

**Examples**

```
showClass("FarmParameters")
```

---

```
FarmRepresentation-class
      Class "FarmRepresentation"
```

---

**Description**

Abstract Class providing generic methods to access properties of Farm objects

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Methods**

**cropArea** signature(farm = "FarmRepresentation", cropName = "character"): The area in hectares of the named crop. Should only be called on solved farm objects

**cropNames** signature(object = "FarmRepresentation"): A string vector with the names of all crops defined for this farm

**eo** signature(farm = "FarmRepresentation"): Enterprise output of the farm

**guts** signature(object = "FarmRepresentation"): Dump the internal representation of the farm to the screen. This is not pretty

**isSolved** signature(farm = "FarmRepresentation"): Returns 1 if the farm has been solved, 0 if not

**model** signature(farm = "FarmRepresentation"): Get a reference to the internal java reference representing this farm

**objectiveNames** signature(object = "FarmRepresentation"): A string vector with the names of objectives that will be optimised in a call to solvelp

**objectiveScaleFactors** signature(farm = "FarmRepresentation"): A numeric vector with the relative weights (normalized) of objectives

- objectiveValues** signature(farm = "FarmRepresentation"): Get the values of the objectives. Should only be called for solved farms
- profit** signature(farm = "FarmRepresentation"): Get the value of the profit objective
- setInputCost** signature(farm = "FarmRepresentation", inputName = "character", inputCost = "numeric"): Set the value of a particular input (potential values are "Input", "Machinery", "Fuel", "Labour", "AreaSubsidy")
- constrainArea** signature(farm = "FarmRepresentation", cropName = "character", lb = "numeric"): Constrain the area of a particular crop on a farm to be the value lb. If an optional ub argument is supplied the constraint is to hold the area of the crop between lb and ub
- show** signature(object = "FarmRepresentation"): Print a summary of the Farm to the screen
- solvelp** signature(farm = "FarmRepresentation"): Solve the Model

**Author(s)**

Ira Cooke

**See Also**

See also [Farm CompositeFarm](#) and for concrete subclasses

**Examples**

```
showClass("FarmRepresentation")
```

---

guts-methods

*Methods for Function guts in Package 'farmR'*

---

**Description**

Spill the guts of the Farm or Composite farm all over the screen

**Methods**

**object = "FarmRepresentation"** Dump very detailed information about the internals of the model to the screen

---

isSolved-methods      *Methods for Function isSolved in Package 'farmR'*

---

### Description

Determine whether

### Methods

**farm = "FarmRepresentation"** Returns 1 if the farm has been solved 0 if not

---

model-methods      *Methods for Function model in Package 'farmR'*

---

### Description

Extract the internal java model object from a FarmRepresentation

### Methods

**farm = "FarmRepresentation"** Returns a java object reference holding the model

---

objectiveNames-methods  
*Methods for Function objectiveNames in Package 'farmR'*

---

### Description

Get a list of the names of all the objectives defined in a FarmRepresentation or ObjectiveParameters object

### Methods

**object = "FarmRepresentation"** Returns a list of the objectives

**object = "ObjectiveParameters"** Returns a list of the objectives

---

 ObjectiveParameters

*Create ObjectiveParameters objects*


---

### Description

Create ObjectiveParameters objects from xml files or from the defaults

### Usage

```
ObjectiveParameters(file = defaultArableObjectiveParameters())
setWeightForObjective(params, objective, value)
getWeightForObjective(params, objective)
```

### Arguments

file	A character string with the full path to an xml document containing the parameters
params	An ObjectiveParameters object
objective	A character string identifying the objective. Available values can be obtained via objectiveNames(params)
value	New weighting for objective. Objectives are renormalized so only the relative values assigned to different objectives are important.

### Details

Parses the xml document provided and stores it internally as a java object. The raw xml can be viewed using the `show` command

### Value

ObjectiveParameters returns an object of the class ObjectiveParameters  
 getWeightForObjective returns the relative weight of the specified objective  
 objectiveNames returns a string vector with the names of objectives

### Author(s)

Ira Cooke

### Examples

```
parameters=ObjectiveParameters()
show(parameters)

names=objectiveNames(parameters)
weights=sapply(names, function(x) getWeightForObjective(parameters, x))
```

---

ObjectiveParameters-class  
*Class "ObjectiveParameters"*

---

### Description

Represents multiple objective utility parameters for a Farm or CompositeFarm object

### Objects from the Class

Objects can be created by calls to the function [ObjectiveParameters](#).

### Slots

**document**: Object of class "jobRef" which is a reference to the underlying java object that holds the parameter data

### Extends

Class "[Parameters](#)", directly.

### Methods

**set** signature(farm = "Farm", input = "ObjectiveParameters"): Used to set the ObjectiveParameters for a single Farm object. Must be done before the first call to solveIp

**objectiveNames** signature(object="ObjectiveParameters"): Get a string vector with the objective parameters

### Author(s)

Ira Cooke

### See Also

See also [FarmParameters](#)

### Examples

```
showClass("ObjectiveParameters")
```

---

objectiveScaleFactors-methods

*Methods for Function objectiveScaleFactors in Package 'farmR'*

---

### Description

Get the normalised objective weights

### Methods

**farm = "FarmRepresentation"** Returns a list of normalised objective weights

---

objectiveValues-methods

*Methods for Function objectiveValues in Package 'farmR'*

---

### Description

Get a list of objective values

### Methods

**farm = "FarmRepresentation"** Get a list of objective values

---

Parameters-class    *Class "Parameters"*

---

### Description

Abstract class defining generic methods for Parameters objects

### Objects from the Class

A virtual Class: No objects may be created from it.

### Methods

**document** signature(params = "Parameters"): Get the underlying java object representing the Parameters

**show** signature(object = "Parameters"): Dump the parameters to screen

### Author(s)

Ira Cooke

**See Also**

See also [ObjectiveParameters](#) [FarmParameters](#) for concrete subclasses

**Examples**

```
showClass("Parameters")
```

---

<code>profit-methods</code>	<i>Methods for Function profit in Package 'farmR'</i>
-----------------------------	---

---

**Description**

Get the profit value for a solved farm

**Methods**

**farm = "FarmRepresentation"** Return the maximum solved profit

---

<code>SBFactories</code>	<i>Locations of Sugarbeet factories in the UK</i>
--------------------------	---

---

**Description**

SpatialPointsDataFrame with the locations of Sugarbeet factories in the UK

**Usage**

```
data(SBFactories)
```

**Format**

A SpatialPointsDataFrame with the locations and names of sugarbeet factories

**Examples**

```
data(SBFactories)
```

---

setInputCost-methods

*Methods for Function setInputCost in Package 'farmR'*

---

### **Description**

Set the cost of named inputs

### **Methods**

**farm = "FarmRepresentation", inputName = "character", inputCost = "numeric"** Set the cost of a named input.

---

solvelp-methods

*Methods for Function solvelp in Package 'farmR'*

---

### **Description**

Solve the model

### **Methods**

**farm = "FarmRepresentation"** Solve the model

# Index

## \*Topic **classes**

CompositeFarm-class, 3  
Farm-class, 7  
FarmParameters-class, 9  
FarmRepresentation-class, 10  
ObjectiveParameters-class, 13  
Parameters-class, 15

## \*Topic **datasets**

SBFactories, 16

## \*Topic **methods**

constrainArea-methods, 4  
cropArea-methods, 4  
cropNames-methods, 4  
document-methods, 5  
eo-methods, 6  
guts-methods, 11  
isSolved-methods, 11  
model-methods, 12  
objectiveNames-methods, 12  
objectiveScaleFactors-methods, 14  
objectiveValues-methods, 15  
profit-methods, 16  
setInputCost-methods, 16  
solvelp-methods, 17

CompositeFarm, 2, 3, 6, 7, 11

CompositeFarm-class, 3

constrainArea  
(*constrainArea-methods*), 4

constrainArea, FarmRepresentation, character, numeric-method  
(*FarmRepresentation-class*), 10

constrainArea-methods, 4

cropArea (*cropArea-methods*), 4

cropArea, FarmRepresentation, character-method(*FarmRepresentation-class*), 10

cropArea-methods, 4

cropNames (*cropNames-methods*), 4

cropNames, FarmParameters-method  
(*FarmParameters-class*), 9

cropNames, FarmRepresentation-method  
(*FarmRepresentation-class*), 10

cropNames-methods, 4

defaultArableFarmParameters, 5

defaultArableObjectiveParameters  
(*defaultArableFarmParameters*), 5

document (*document-methods*), 5

document, Parameters-method  
(*Parameters-class*), 15

document-methods, 5

eo (*eo-methods*), 6

eo, FarmRepresentation-method  
(*FarmRepresentation-class*), 10

eo-methods, 6

Farm, 3, 6, 7, 11

Farm-class, 7

FarmParameters, 5, 8, 9, 14, 15

FarmParameters-class, 9

FarmRepresentation, 3, 7

FarmRepresentation-class, 10

getSolverType (*FarmParameters*), 8

getWeightForObjective  
(*ObjectiveParameters*), 12

guts (*guts-methods*), 11

guts, FarmRepresentation-method  
(*FarmRepresentation-class*), 10

guts-methods, 11

isSolved (*isSolved-methods*), 11

- isSolved, FarmRepresentation-method  
(*FarmRepresentation-class*),  
10
- isSolved-methods, 11
- model (*model-methods*), 12
- model, FarmRepresentation-method  
(*FarmRepresentation-class*),  
10
- model-methods, 12
- objectiveNames  
(*objectiveNames-methods*),  
12
- objectiveNames, FarmRepresentation-method  
(*FarmRepresentation-class*),  
10
- objectiveNames, ObjectiveParameters-method  
(*ObjectiveParameters-class*),  
13
- objectiveNames-methods, 12
- ObjectiveParameters, 5, 10, 12, 13, 15
- ObjectiveParameters-class, 13
- objectiveScaleFactors  
(*objectiveScaleFactors-methods*),  
14
- objectiveScaleFactors, FarmRepresentation-method  
(*FarmRepresentation-class*),  
10
- objectiveScaleFactors-methods, 14
- objectiveValues  
(*objectiveValues-methods*),  
15
- objectiveValues, FarmRepresentation-method  
(*FarmRepresentation-class*),  
10
- objectiveValues-methods, 15
- Parameters, 9, 14
- Parameters-class, 15
- profit (*profit-methods*), 16
- profit, FarmRepresentation-method  
(*FarmRepresentation-class*),  
10
- profit-methods, 16
- SBFactories, 16
- set, Farm, character-method  
(*Farm-class*), 7
- set, Farm, jobjRef-method  
(*Farm-class*), 7
- set, Farm, ObjectiveParameters-method  
(*ObjectiveParameters-class*),  
13
- setInputCost  
(*setInputCost-methods*), 16
- setInputCost, FarmRepresentation, character, numerical  
(*FarmRepresentation-class*),  
10
- setInputCost-methods, 16
- setPriceForCrop (*FarmParameters*),  
8
- setRelativeCost (*FarmParameters*),  
8
- setRelativeLabourRequirements  
(*FarmParameters*), 8
- setRelativePriceForCrop  
(*FarmParameters*), 8
- setRelativeRotationPenalties  
(*FarmParameters*), 8
- setRelativeYieldForCrop  
(*FarmParameters*), 8
- setSolverType (*FarmParameters*), 8
- setSubsidyForCrop  
(*FarmParameters*), 8
- setWeightForObjective  
(*ObjectiveParameters*), 12
- show, FarmRepresentation-method  
(*FarmRepresentation-class*),  
10
- show, Parameters-method  
(*Parameters-class*), 15
- solvelp (*solvelp-methods*), 17
- solvelp, FarmRepresentation-method  
(*FarmRepresentation-class*),  
10
- solvelp-methods, 17