

# Package ‘AFM’

September 1, 2016

**Type** Package

**Version** 1.2.2

**Date** 2016-08-31

**Title** Atomic Force Microscope Image Analysis

**Author** Mathieu Beauvais [aut, cre],  
Irma Liascukiene [aut],  
Jessem Landoulsi [aut]

**Maintainer** Mathieu Beauvais <beauvais.escp@gmail.com>

**Description** Provides Atomic Force Microscope images analysis such as Power Spectral Density, roughness against lengthscale, experimental variogram and variogram models, fractal dimension and scale. The AFM images can be exported to STL format for 3D printing.

**NeedsCompilation** no

**Repository** CRAN

**License** AGPL-3

**Depends** R (>= 3.1.3)

**Imports** data.table(>= 1.9.6),stringr(>= 1.0.0),gstat(>= 1.0-26),fractaldim(>= 0.8-4),rgl(>= 0.96),pracma(>= 1.8.6),fftwtools(>= 0.9-7),grid(>= 3.1.3),gridExtra(>= 2.0.0),moments(>= 0.14),ggplot2(>= 1.0.1),sp(>= 1.2-0),png(>= 0.1-7),plyr(>= 1.8.3),igraph(>= 1.0.1),methods(>= 3.1.3),shiny(>= 0.12.2),shinyjs(>= 0.4.0),scales(>= 0.4.0),dbscan(>= 0.9-8)

**Collate** 'AFM3DPrinter.R' 'AFMFractalDimensionAnalyser.R' 'AFMImage.R'  
'AFMNetworksAnalyser.R' 'AFMPSDAnalyser.R'  
'AFMVariogramAnalyser.R' 'AFMImageAnalyser.R'  
'AFMReportMaker.R' 'pkgname.R' 'runAFMApp.R'

**RoxygenNote** 5.0.1

**Date/Publication** 2016-09-01 01:33:25

**R topics documented:**

AFM	4
AFMImage-class	5
AFMImage3DModelAnalysis-class	7
AFMImageAnalyser-class	8
AFMImageFractalDimensionMethod-class	9
AFMImageFractalDimensionsAnalysis-class	10
AFMImageNetworksAnalysis-class	11
AFMImageOfAluminiumInterface	12
AFMImageOfNormallyDistributedHeights	13
AFMImageOfOnePeak	13
AFMImageOfRegularPeaks	13
AFMImagePSDAnalysis-class	14
AFMImagePSDSlopesAnalysis-class	15
AFMImageVariogramAnalysis-class	16
AFMImageVariogramModel-class	18
AFMImageVariogramSlopesAnalysis-class	19
analyse	20
AreNodesConnected	21
calculate3DModel	21
calculateDirectionalVariograms	22
calculateIgraph	23
calculateNetworks	24
calculateNetworkSkeleton	24
calculateOmnidirectionalVariogram	25
calculatePhysicalDistanceFromPath	26
calculateShortestPaths	26
canBeRemoved	27
checkIsotropy	27
checkNormality	28
createGraph	29
displaygridIgraphPlot	30
displaygridIgraphPlotFromEdges	30
displayIn3D	31
evaluateVariogramModels	32
existsEdge	33
existsSegment	33
exportToSTL	34
extractAFMImage	35
filterAFMImage	36
fusionCloseNodes	36
generateAFMImageReport	37
generateCheckReport	37
generateReport	38
generateReportFromNanoscopeImageDirectory	39
getAngle	40
getAutoIntersectionForOmnidirectionalVariogram	40

getAutoIntersectionForRoughnessAgainstLengthscale . . . . .	41
getAutomaticWidthForVariogramCalculation . . . . .	42
getBresenham2DSegment . . . . .	43
getCircleSpatialPoints . . . . .	43
getCoordinatesFromVertexId . . . . .	44
getDTModelEvaluation . . . . .	44
getDTModelSillRange . . . . .	45
getFractalDimensions . . . . .	45
getIntersectionPointWithBorder . . . . .	46
getListOfDiameters . . . . .	47
getLogLogOmnidirectionalSlopeGraph . . . . .	47
getNetworkGridLayout . . . . .	48
getNyquistSpatialFrequency . . . . .	49
getPaddedAFMImage . . . . .	50
getRoughnessParameters . . . . .	50
getSpplotFromAFMImage . . . . .	51
getSurroundingVertexesList . . . . .	52
getTopologyAFMImage . . . . .	53
getTriangle . . . . .	53
getVertexId . . . . .	54
gridIgraphPlot . . . . .	54
identifyEdgesFromCircles . . . . .	55
identifyIsolatedNodes . . . . .	55
identifyNodesWithCircles . . . . .	56
importFromNanoscope . . . . .	56
initialize.AFMImageAnalyser-method . . . . .	57
isAdjacentToBetterVertex . . . . .	58
isAngleBetweenEdgesAlwaysSuperiorToMinAngle . . . . .	59
makeBinaryAFMImage . . . . .	59
multiplyHeightsAFMImage . . . . .	60
performAllPSDCalculation . . . . .	61
printVariogramModelEvaluations . . . . .	62
PSD1DAgainstFrequency . . . . .	62
PSD2DAgainstFrequency . . . . .	64
putAnalysisOnDisk . . . . .	65
putImagesFromAnalysisOnDisk . . . . .	65
RoughnessByLengthScale . . . . .	66
runAFMApp . . . . .	67
sampleAFMImage . . . . .	68
saveOnDisk . . . . .	69
saveOnDiskIntersectionForRoughnessAgainstLengthscale . . . . .	70
saveSpplotFromAFMImage . . . . .	70
shiftedPSDuv . . . . .	71
shiftFFT2D . . . . .	73
simplifyAFMImage . . . . .	73
totalRMSRoughness . . . . .	74
transformAFMImageForNetworkAnalysis . . . . .	75
updateProgress . . . . .	76

## Description

The AFM package provides statistics analysis tools for Atomic Force Microscopy image analysis.  
Licence: Affero GPL v3

## Details

A graphical user interface is available by using `runAFMApp` command.

Several high level functions are :

- create your AFM image from a list of measured heights (see example section of `AFMImage`)
- import your image from Nanoscope Analysis (TM) tool (`importFromNanoscope`)
- check if your sample is normally distributed and isotropic and get a pdf report (`generateCheckReport`)
- perform variance (variogram), roughness against lengthscale, fractal analysis and get a pdf report (`generateReport`)

Other functions are :

- check sample: for normality (`checkNormality`) and for isotropy (`checkIsotropy`)
- calculate total RMS roughness: quick calculation of total root mean square roughness(`totalRMSRoughness`)
- calculate omnidirectional variogram: calculate estimated variogram (`calculateOmnidirectionalVariogram`)
- calculate roughness against lengthscale and Power Spectrum Density (PSD): calculate roughness against length scale (`RoughnessByLengthScale`), PSD 1D (`PSD1DAgainstFrequency`) or PSD 2D (`PSD2DAgainstFrequency`) against frequencies
- calculate fractal dimension and scale: use (`getFractalDimensions`) function
- print in 3D (3D print) (`exportToSTL`) your AFM image

An EC2 instance is available for basic testing at the following address: <http://www.afmist.org>

Note: To use with a Bruker(TM) Atomic Force Microscope, use nanoscope analysis(TM) software and

- Use the "Flatten" function.
- Save the flattened image.
- Use the "Browse Data Files" windows, right click on image name and then Export the AFM image with the headers and the "Export> ASCII" contextual menu option.

## Author(s)

M.Beauvais, J.Landoulsi, I.Liascukiene

## References

Gneiting2012, Tilmann Gneiting, Hana Sevcikova and Donald B. Percival 'Estimators of Fractal Dimension: Assessing the Roughness of Time Series and Spatial Data - Statistics in statistical Science, 2012, Vol. 27, No. 2, 247-277'

Olea2006, Ricardo A. Olea "A six-step practical approach to semivariogram modeling", 2006, "Stochastic Environmental Research and Risk Assessment, Volume 20, Issue 5 , pp 307-318"

Sidick2009, Erkin Sidick "Power Spectral Density Specification and Analysis of Large Optical Surfaces", 2009, "Modeling Aspects in Optical Metrology II, Proc. of SPIE Vol. 7390 73900L-1"

## See Also

[gstat](#), [fractaldim](#), [rgl](#)

## Examples

```
## Not run:
library(AFM)
# Analyse the AFMImageOfRegularPeaks AFM Image from this package
data("AFMImageOfRegularPeaks")
AFMImage<-AFMImageOfRegularPeaks
# exportDirectory="C:/Users/my_windows_login" or exportDirectory="/home/ubuntu"
exportDirectory=tempdir()
AFMImage@fullfilename<-paste(exportDirectory,"AFMImageOfRegularPeaks.txt",sep="/")

# Start to check if your sample is normally distributed and isotropic.
generateCheckReport(AFMImage)

# If the sample is normally distributed and isotropic, generate a full report
generateReport(AFMImage)

## End(Not run)
```

---

AFMImage-class

*AFM image class*

---

## Description

A S4 class to store and manipulate images from Atomic Force Microscopes.

Constructor method of AFMImage Class.

Wrapper function AFMImage

**Usage**

```
AFMImage(data, samplesperline, lines, hscansize, vscansize, scansize,
          fullfilename)
```

```
## S4 method for signature 'AFMImage'
initialize(.Object, data, samplesperline, lines, hscansize,
          vscansize, scansize, fullfilename)
```

```
AFMImage(data, samplesperline, lines, hscansize, vscansize, scansize,
          fullfilename)
```

**Arguments**

data	(\$x,\$y,\$h): a data.table storing the coordinates of the sample and the measured heights
samplesperline	number of samples per line (e.g.: 512)
lines	number of line (e.g.: 512)
hscansize	horizontal size of scan usually in nanometer (e.g.: hscansize=1000 for a scan size of 1000 nm)
vscansize	vertical size of scan usually in nanometer (e.g.: vscansize=1000 for a scan size of 1000 nm)
scansize	if hscansize equals vscansize, scansize is the size of scan usually in nanometer (e.g.: scansize=1000 for a scan size of 1000 nm)
fullfilename	directory and filename on the disk (e.g.: /users/ubuntu/flatten-image.txt)
.Object	an AFMImage object

**Slots**

data	(\$x,\$y,\$h): a data.table storing the coordinates of the sample and the measured heights
samplesperline	number of samples per line (e.g.: 512)
lines	number of line (e.g.: 512)
hscansize	horizontal size of scan usually in nanometer (e.g.: hscansize=1000 for a scan size of 1000 nm)
vscansize	vertical size of scan usually in nanometer (e.g.: vscansize=1000 for a scan size of 1000 nm)
scansize	if hscansize equals vscansize, scansize is the size of scan usually in nanometer (e.g.: scansize=1000 for a scan size of 1000 nm)
fullfilename	directory and filename on the disk (e.g.: /users/ubuntu/flatten-image.txt)

**Author(s)**

M.Beauvais

**Examples**

```
library(AFM)
library(data.table)

# create a 128 pixels by 128 pixels AFM image
Lines=128
Samplesperline=128
fullfilename="RandomFakeAFMImage"
# the size of scan is 128 nm
ScanSize=128
# the heights is a normal distribution in nanometers
nm<-c(rnorm(128*128, mean=0, sd=1 ))

scanby<-ScanSize/Samplesperline
endScan<-ScanSize*(1-1/Samplesperline)
RandomFakeAFMImage<-AFMImage(
  data = data.table(x = rep(seq(0,endScan, by= scanby), times = Lines),
                    y = rep(seq(0,endScan, by= scanby), each = Samplesperline),
                    h = nm),
  samplesperline = Samplesperline, lines = Lines,
  vscansize = ScanSize, hscansize = ScanSize, scansize = ScanSize,
  fullfilename = fullfilename )
```

---

AFMImage3DModelAnalysis-class

*AFM image Power Spectrum Density analysis class*

---

**Description**

AFMImage3DModelAnalysis

**Slots**

f1 a face of the 3D model

f2 a face of the 3D model

f3 a face of the 3D model

f4 a face of the 3D model

**Author(s)**

M.Beauvais

---

AFMImageAnalyser-class

*AFM image analyser class*

---

### **Description**

A S4 class to handle the analysis of one AFM Image.

Wrapper function AFMImageAnalyser

### **Usage**

AFMImageAnalyser(AFMImage)

AFMImageAnalyser(AFMImage)

### **Arguments**

AFMImage            an AFMImage

### **Slots**

AFMImage [AFMImage](#) to be analysed

variogramAnalysis [AFMImageVariogramAnalysis](#)

psdAnalysis [AFMImagePSDAnalysis](#)

fdAnalysis [AFMImageFractalDimensionsAnalysis](#)

networksAnalysis [AFMImageNetworksAnalysis](#)

mean the mean of heights of the [AFMImage](#)

variance the variance of heights of the [AFMImage](#)

TotalRrms the total Root Mean Square Roughness of the [AFMImage](#) calculated from variance

Ra mean roughness or mean of absolute values of heights

fullfilename to be removed ?

updateProgress a function to update a graphical user interface

### **Author(s)**

M.Beauvais



---

 AFMImageFractalDimensionMethod-class

*AFM image fractal dimension method class*


---

**Description**

AFMImageFractalDimensionMethod stores calculation from one fractal dimension method

Constructor method of AFMImageFractalDimensionMethod Class.

Wrapper function AFMImageFractalDimensionMethod

**Usage**

```
AFMImageFractalDimensionMethod(fd_method, fd, fd_scale)
```

```
## S4 method for signature 'AFMImageFractalDimensionMethod'
initialize(.Object, fd_method, fd,
          fd_scale)
```

```
AFMImageFractalDimensionMethod(fd_method, fd, fd_scale)
```

**Arguments**

fd_method	Two dimensional function names used to evaluate the fractal dimension and fractal scale
fd	the value of the fractal dimension
fd_scale	the value of the fractal scale
.Object	an AFMImageFractalDimensionMethod object

**Slots**

fd\_method Two dimensional function names used to evaluate the fractal dimension and fractal scale

fd the value of the fractal dimension

fd\_scale the value of the fractal scale

**Author(s)**

M.Beauvais

**See Also**

[fractaldim](#)

---

AFMImageFractalDimensionsAnalysis-class  
*AFM image fractal dimensions analysis class*

---

### Description

A S4 class to handle the fractal dimension calculation with several fractal dimension methods

Constructor method of AFMImageFractalDimensionsAnalysis Class.

Wrapper function AFMImageFractalDimensionsAnalysis

Method fractalDimensionMethods returns a list of FractalDimensionMethod objects

### Usage

```
AFMImageFractalDimensionsAnalysis()

## S4 method for signature 'AFMImageFractalDimensionsAnalysis'
initialize(.Object,
  fractalDimensionMethods, csvFullfilename)

AFMImageFractalDimensionsAnalysis()

fractalDimensionMethods(object)

## S4 method for signature 'AFMImageFractalDimensionsAnalysis'
fractalDimensionMethods(object)
```

### Arguments

```
.Object      an AFMImageFractalDimensionsAnalysis Class
fractalDimensionMethods
              a list of AFMImageFractalDimensionMethod
csvFullfilename
              To be removed ?
object       a AFMImageFractalDimensionsAnalysis
```

### Slots

```
fractalDimensionMethods a list of AFMImageFractalDimensionMethod
csvFullfilename To be removed ?
updateProgress a function to update a graphical user interface
```

### Author(s)

M.Beauvais

---

 AFMImageNetworksAnalysis-class

*AFM image networks analysis class*


---

### Description

A S4 class to handle the networks calculation

Constructor method of AFMImageNetworksAnalysis Class.

Wrapper function AFMImageNetworksAnalysis

### Usage

```
AFMImageNetworksAnalysis()
```

```
## S4 method for signature 'AFMImageNetworksAnalysis'
initialize(Object, binaryAFMImage,
  binaryAFMImageWithCircles, circlesTable, edgesTable,
  fusionedNodesCorrespondance, fusionedNodesEdgesTable, isolatedNodesList,
  heightNetworkslider, filterNetworksliderMin, filterNetworksliderMax,
  originalGraph, skeletonGraph, shortestPaths, libVersion)
```

```
AFMImageNetworksAnalysis()
```

### Arguments

```
.Object          an AFMImageNetworksAnalysis Class
binaryAFMImage  the AFMImage after transformation before analysis
binaryAFMImageWithCircles
                  the AFMImage after transformation with the spotted circles
circlesTable    a data.table of identified circles
edgesTable      a data.table of edges
fusionedNodesCorrespondance
                  a data.table of correspon
fusionedNodesEdgesTable
                  a data.table of correspondance between intial node and fusioned node
isolatedNodesList
                  a data.table of isolated nodes
heightNetworkslider
                  used multiplier of heights to facilitate analysis
filterNetworksliderMin
                  used filter minimum value to facilitate analysis
filterNetworksliderMax
                  used filter maximum value to facilitate analysis
originalGraph   a list of igraph
```

skeletonGraph a list of [igraph](#)  
 shortestPaths a data.table of shortest path  
 libVersion version of the AFM library used to perform the analysis

### Slots

binaryAFMImage the AFMImage after transformation before analysis  
 binaryAFMImageWithCircles the AFMImage after transformation with the spotted circles  
 circlesTable a data.table of identified circles  
 edgesTable a data.table of edges  
 fusionedNodesCorrespondance a data.table of correspondance between intial node and fusioned node  
 fusionedNodesEdgesTable a data.table of nodes fusioned because of intersecting  
 isolatedNodesTable a data.table of isolated nodes  
 heightNetworkslider used multiplier of heights to facilitate analysis  
 filterNetworksliderMin used filter minimum value to facilitate analysis  
 filterNetworksliderMax used filter maximum value to facilitate analysis  
 originalGraph a list of [igraph](#)  
 skeletonGraph a list of [igraph](#)  
 shortestPaths a data.table of shortest paths  
 libVersion version of the AFM library used to perform the analysis  
 updateProgress a function to update a graphical user interface

### Author(s)

M.Beauvais

---

AFMImageOfAluminiumInterface

*AFM image sample*

---

### Description

A real dataset containing an [AFMImage](#) of an Aluminium interface. The image is made of 512\*512 samples of a 1000 nm \* 1000 nm surface. samplesperline=512 lines=512 hscansize=1000 vscan-size=1000

### Author(s)

J.Landoulsi, I.Liascukiene

---

AFMImageOfNormallyDistributedHeights  
*AFM image sample*

---

**Description**

A fake dataset containing a manually generated [AFMImage](#) (a normal distribution of heights). The image is made of 128\*128 samples of a 128 nm \* 128 nm surface. samplesperline= 128 lines= 128 hscansize= 128 vscansize= 128

---

AFMImageOfOnePeak      *AFM image sample*

---

**Description**

A fake dataset containing a manually generated [AFMImage](#) (one peak positioned on the surface). The image is made of 128\*128 samples of a 128 nm \* 128 nm surface. samplesperline= 128 lines= 128 hscansize= 128 vscansize= 128

---

AFMImageOfRegularPeaks  
*AFM image sample*

---

**Description**

A fake dataset containing a manually generated [AFMImage](#) (peaks regularly positioned on the surface). The image is made of 128\*128 samples of a 128 nm \* 128 nm surface. samplesperline= 128 lines= 128 hscansize= 128 vscansize= 128

---

 AFMImagePSDAnalysis-class

*AFM image Power Spectrum Density analysis class*


---

### Description

AFMImagePSDAnalysis handles an [AFMImage](#) roughness against lengthscale analysis

Constructor method of AFMImagePSDAnalysis Class.

Wrapper function AFMImagePSDAnalysis

Method psd1d\_breaks returns a number of breaks to calculate PSD1D from PSD2D

Method psd2d\_maxHighLengthScale returns the maximum lengthscale to be managed by PSD2D

Method psd2d\_truncHighLengthScale returns if the lengthscale of PSD2D should be truncated

Method psd1d returns a data.table of psd in 1D

Method psd2d returns a data.table of psd in 1D

Method roughnessAgainstLengthscale returns a data.table of roughnesses versus lengthscale

Method intersections returns a intersection numeric value

### Usage

```
AFMImagePSDAnalysis()
```

```
## S4 method for signature 'AFMImagePSDAnalysis'
initialize(.Object)
```

```
AFMImagePSDAnalysis()
```

```
psd1d_breaks(object)
```

```
## S4 method for signature 'AFMImagePSDAnalysis'
psd1d_breaks(object)
```

```
psd2d_maxHighLengthScale(object)
```

```
## S4 method for signature 'AFMImagePSDAnalysis'
psd2d_maxHighLengthScale(object)
```

```
psd2d_truncHighLengthScale(object)
```

```
## S4 method for signature 'AFMImagePSDAnalysis'
psd2d_truncHighLengthScale(object)
```

```
psd1d(object)
```

```
## S4 method for signature 'AFMImagePSDAnalysis'
```

```

psd1d(object)

psd2d(object)

## S4 method for signature 'AFMImagePSDAnalysis'
psd2d(object)

roughnessAgainstLengthscale(object)

## S4 method for signature 'AFMImagePSDAnalysis'
roughnessAgainstLengthscale(object)

intersections(object)

## S4 method for signature 'AFMImagePSDAnalysis'
intersections(object)

```

**Arguments**

```

.Object      an AFMImagePSDAnalysis object
object      a AFMImagePSDAnalysis

```

**Slots**

```

roughnessAgainstLengthscale a data.table to store the roughness against lengthscale data
intersections a list to store the lengthscale values as the intersections between slopes and the
                sill in roughness against lengthscale graph
updateProgress a function to update a graphical user interface

```

**Author(s)**

M.Beauvais

---

```

AFMImagePSDSlopesAnalysis-class
      AFM Image psd slope analysis

```

---

**Description**

AFMImagePSDSlopesAnalysis stores the analysis of the second slope in roughness against lengthscale  
 Constructor method of AFMImagePSDSlopesAnalysis Class.  
 Wrapper function AFMImagePSDSlopesAnalysis

**Usage**

```
AFMImagePSDSlopesAnalysis()

## S4 method for signature 'AFMImagePSDSlopesAnalysis'
initialize(.Object)

AFMImagePSDSlopesAnalysis()
```

**Arguments**

.Object            an AFMImagePSDSlopesAnalysis object

**Slots**

lc   to be removed ?  
 wsat   to be removed ?  
 slope   to be removed ?  
 yintersept   to be removed ?

**Author(s)**

M.Beauvais

---

AFMImageVariogramAnalysis-class  
*AFM image variogram analysis class*

---

**Description**

AFMImageVariogramAnalysis manages the variogram analysis of an [AFMImage](#)  
 Constructor method of AFMImageVariogramAnalysis Class.  
 Wrapper function AFMImageVariogramAnalysis  
 Method variogramModels returns a list of variogram model evaluation  
 Method omnidirectionalVariogram returns the omnidirectional variogram  
 Method directionalVariograms returns the directional variograms

**Usage**

```
AFMImageVariogramAnalysis(sampleFitPercentage)

## S4 method for signature 'AFMImageVariogramAnalysis'
initialize(.Object, sampleFitPercentage,
  updateProgress)
```



```

AFMImageVariogramAnalysis(sampleFitPercentage)

variogramModels(object)

## S4 method for signature 'AFMImageVariogramAnalysis'
variogramModels(object)

omnidirectionalVariogram(object)

## S4 method for signature 'AFMImageVariogramAnalysis'
omnidirectionalVariogram(object)

directionalVariograms(object)

## S4 method for signature 'AFMImageVariogramAnalysis'
directionalVariograms(object)

```

### Arguments

sampleFitPercentage a sample size as a percentage (e.g. "5" for 5 percents) of random points in the [AFMImage](#). These points will be used to fit the variogram models.

.Object an AFMImageVariogramAnalysis class

updateProgress a function to update a graphical user interface

object a AFMImageVariogramAnalysis object

### Slots

width (optional) a distance step for the calculation of the variograms (e.g.: width= integer of (scan Size divided by number of lines)= ceil(1000 / 512) for AFMImageOfAluminiumInterface

omnidirectionalVariogram a data.table to store the omnidirectional variogram

directionalVariograms a data.table to store the directional variograms

sampleFitPercentage a sample size as a percentage of random points in the [AFMImage](#). These points will be used to fit the variogram models.

chosenFitSample the chosen random points of the [AFMImage](#) to perform the fitting of the variogram models.

cuts the cuts for spplot of the [AFMImage](#). The same cuts will be used for the predicted [AFMImage](#)

variogramModels A list of [AFMImageVariogramModel](#) containing the various evaluated variogram models.

fullfilename to be removed ?

updateProgress a function to update a graphical user interface

### Author(s)

M.Beauvais

---

 AFMImageVariogramModel-class

*AFM Image Variogram Model class*


---

### Description

AFMImageVariogramModel stores the evaluation of one experimental variogram model

Constructor method of AFMImageVariogramModel Class.

Wrapper function AFMImageVariogramModel

### Usage

```
AFMImageVariogramModel()
```

```
## S4 method for signature 'AFMImageVariogramModel'
initialize(Object, model,
  fit.v = data.table(), mykrige, res = data.table(), cor, press, sill,
  imageFullfilename)
```

```
AFMImageVariogramModel()
```

### Arguments

.Object	an AFMImageVariogramModel object
model	the variogram model name
fit.v	the values from the <a href="#">fit.variogram</a> function in the gstat package
mykrige	the values from the <a href="#">krige</a> function in the gstat library
res	a data.table to store: (cor) the correlation between the predicted sample and the real sample (press) the sum of the square of the differences between real and predicted values for each point of the sample
cor	to be removed ?
press	to be removed ?
sill	to be removed ?
imageFullfilename	to be removed ?

### Slots

model the variogram model name

fit.v the values from the [fit.variogram](#) function in the gstat package

mykrige the values from the [krige](#) function in the gstat library

res a data.table to store: (cor) the correlation between the predicted sample and the real sample (press) the sum of the square of the differences between real and predicted values for each point of the sample

cor to be removed ?  
press to be removed ?  
sill to be removed ?  
imageFullfilename to be removed ?

**Author(s)**

M.Beauvais

---

AFMImageVariogramSlopesAnalysis-class  
*AFM Image psd slope analysis*

---

**Description**

AFMImageVariogramSlopesAnalysis stores the analysis of the second slope in roughness against lengthscale  
Constructor method of AFMImageVariogramSlopesAnalysis Class.  
Wrapper function AFMImageVariogramSlopesAnalysis

**Usage**

```
AFMImageVariogramSlopesAnalysis()  
  
## S4 method for signature 'AFMImageVariogramSlopesAnalysis'  
initialize(.Object)  
  
AFMImageVariogramSlopesAnalysis()
```

**Arguments**

.Object            an AFMImageVariogramSlopesAnalysis object

**Slots**

intersection\_sill to be removed ?  
sill to be removed ?  
slope to be removed ?  
yintersept to be removed ?

**Author(s)**

M.Beauvais

---

analyse

*Analyse an AFMImage*

---

## Description

A function to wrap all the analysis of an [AFMImage](#)

- variogram analysis including evaluation of basic variogram models with sill and range calculation
- power spectrum density analysis including roughness against lengthscale calculation
- fractal dimension analysis including fractal dimensions calculation
- basic roughness parameters analysis such as mean, variance, Rrms, Ra

## Usage

```
analyse(AFMImageAnalyser)
```

## Arguments

AFMImageAnalyser  
a [AFMImageAnalyser](#) to manage and store image analysis

## Value

an [AFMImageAnalyser](#) containing all the analysis

## Author(s)

M.Beauvais

## Examples

```
library(AFM)

data(AFMImageOfAluminiumInterface)
AFMImage<-extractAFMImage(AFMImageOfAluminiumInterface, 0, 0, 32)
AFMImageAnalyser<-new("AFMImageAnalyser", AFMImage= AFMImage, fullfilename = AFMImage@fullfilename)
AFMImageAnalyser<-analyse(AFMImageAnalyser)
print(AFMImageAnalyser@fdAnalysis)
```

---

AreNodesConnected	<i>check if nodes represented by circles are connected. The function defines all the possible segments between the circles and check if at least one segment exists.</i>
-------------------	--

---

### Description

check if nodes represented by circles are connected. The function defines all the possible segments between the circles and check if at least one segment exists.

### Usage

```
AreNodesConnected(binaryAFMImage, center1, radius1, center2, radius2)
```

### Arguments

binaryAFMImage	a binary <a href="#">AFMImage</a> from Atomic Force Microscopy
center1	the center of the circle with center\$lon as the x coordinates and center\$lat as the y coordinates
radius1	the radius of the circle
center2	the center of the circle with center\$lon as the x coordinates and center\$lat as the y coordinates
radius2	the radius of the circle

### Value

TRUE if the nodes are connected

### Author(s)

M.Beauvais

---

calculate3DModel	<i>Calculate the 3D model for 3D printing</i>
------------------	---

---

### Description

calculate3DModel update [AFMImage3DModelAnalysis](#)

### Usage

```
calculate3DModel(AFMImage3DModelAnalysis, AFMImage)

## S4 method for signature 'AFMImage3DModelAnalysis'
calculate3DModel(AFMImage3DModelAnalysis,
  AFMImage)
```

**Arguments**

AFMImage3DModelAnalysis  
n [AFMImage3DModelAnalysis](#) to store the setup and results of PSD analysis

AFMImage  
an [AFMImage](#) from Atomic Force Microscopy

**Author(s)**

M.Beauvais

---

calculateDirectionalVariograms

*Calculate experimental directional semi-variograms*

---

**Description**

calculate four experimental directional variograms of an [AFMImage](#) with the [variogram](#) function of the `gstat` package. The directional semi-variogram can be used to check the isotropy of the sample. Note: The sample will be isotropic if the slopes of the four variograms are similar.

**Usage**

```
calculateDirectionalVariograms(AFMImageVariogramAnalysis, AFMImage)
```

**Arguments**

AFMImageVariogramAnalysis  
an [AFMImageVariogramAnalysis](#) to manage and store the result of variogram analysis

AFMImage  
an [AFMImage](#) from Atomic Force Microscopy

**Details**

calculateDirectionalVariograms returns the directional variograms

**Value**

Four directional variograms

**Author(s)**

M.Beauvais

**Examples**

```
## Not run:
library(AFM)
library(ggplot2)

data(AFMImageOfRegularPeaks)
variogramAnalysis<-AFMImageVariogramAnalysis(sampleFitPercentage=3.43/100)
varios<-AFM::calculateDirectionalVariograms(AFMImage= AFMImageOfRegularPeaks,
                                             AFMImageVariogramAnalysis= variogramAnalysis)

dist<-gamma<-NULL
p <- ggplot(varios, aes(x=dist, y=gamma,
                       color= as.factor(dir.hor),
                       shape=as.factor(dir.hor)))

p <- p + expand_limits(y = 0)
p <- p + geom_point()
p <- p + geom_line()
p <- p + ylab("semivariance (nm^2)")
p <- p + xlab("distance (nm)")
p <- p + ggtitle("Directional")
p

## End(Not run)
```

---

calculateIgraph

*Calculate iGraph from AFMImage*

---

**Description**

calculateIgraph return

**Usage**

```
calculateIgraph(AFMImage, AFMImageNetworksAnalysis)
```

**Arguments**

AFMImage            an [AFMImage](#) from Atomic Force Microscopy  
AFMImageNetworksAnalysis    an [AFMImageNetworksAnalysis](#) from Atomic Force Microscopy

**Author(s)**

M.Beauvais

---

calculateNetworks      *Calculate networks on the surface*

---

### Description

calculateNetworks update [AFMImageNetworksAnalysis](#)

### Usage

```
calculateNetworks(AFMImageNetworksAnalysis, AFMImage)
```

```
## S4 method for signature 'AFMImageNetworksAnalysis'
calculateNetworks(AFMImageNetworksAnalysis,
  AFMImage)
```

### Arguments

AFMImageNetworksAnalysis  
                                   n [AFMImageNetworksAnalysis](#) to store the results of networks analysis

AFMImage                        an [AFMImage](#) from Atomic Force Microscopy

### Author(s)

M.Beauvais

---

calculateNetworkSkeleton  
                                   *calculateNetworkSkeleton*

---

### Description

calculateNetworkSkeleton return

### Usage

```
calculateNetworkSkeleton(AFMImage, AFMImageNetworksAnalysis)
```

### Arguments

AFMImage                        an [AFMImage](#) from Atomic Force Microscopy

AFMImageNetworksAnalysis  
                                   an [AFMImageNetworksAnalysis](#) from Atomic Force Microscopy

### Author(s)

M.Beauvais



---

 calculateOmnidirectionalVariogram

*Calculate experimental omnidirectional semi-variogram*


---

## Description

calculateOmnidirectionalVariogram returns the semivariance calculated for all the directions calculate the experimental omnidirectional variogram of an [AFMImage](#) with the [variogram](#) function of the `gstat` package. The experimental semi-variogram is used to fit (find the best sill and range) the theoretical variogram models. With 512\*512 images, it takes several minutes to calculate.

## Usage

```
calculateOmnidirectionalVariogram(AFMImageVariogramAnalysis, AFMImage)
```

## Arguments

`AFMImageVariogramAnalysis` an [AFMImageVariogramAnalysis](#) to manage and store the result of variogram analysis

`AFMImage` an [AFMImage](#) from Atomic Force Microscopy

## Value

the semivariance calculated in all the directions

## Author(s)

M.Beauvais

## Examples

```
## Not run:
library(AFM)
library(ggplot2)

data(AFMImageOfRegularPeaks)
variogramAnalysis<-AFMImageVariogramAnalysis(sampleFitPercentage=3.43/100)
avario<-AFM::calculateOmnidirectionalVariogram(AFMImageVariogramAnalysis= variogramAnalysis,
                                              AFMImage= AFMImageOfRegularPeaks)

dist<-gamma<-NULL
p <- ggplot(avario, aes(x=dist, y=gamma))
p <- p + geom_point()
p <- p + geom_line()
p <- p + ylab("semivariance")
p <- p + xlab("distance (nm)")
p <- p + ggtitle("Experimental semivariogram")
p

## End(Not run)
```

---

calculatePhysicalDistanceFromPath

*calculate the physical distances between nodes*

---

### Description

calculate the physical distances between nodes

### Usage

calculatePhysicalDistanceFromPath(AFMImage, pathVidVector)

### Arguments

AFMImage      an [AFMImage](#) from Atomic Force Microscopy

pathVidVector   a network path

### Value

the physical distance the extrmitities of the path

### Author(s)

M.Beauvais

---

calculateShortestPaths

*calculate the shortest path between adjacent nodes Calculateshrotest path between all nodes of degree superior to 3 identify nodes that are connected by nodes which degrees is inferior to 2 and calculate the distance between these nodes*

---

### Description

calculate the shortest path between adjacent nodes Calculateshrotest path between all nodes of degree superior to 3 identify nodes that are connected by nodes which degrees is inferior to 2 and calculate the distance between these nodes

### Usage

calculateShortestPaths(AFMImageNetworksAnalysis, fusioned = TRUE)

### Arguments

AFMImageNetworksAnalysis

a [AFMImageNetworksAnalysis](#)

fusioned      TRUE if graph is created from fusioned nodes

**Author(s)**

M.Beauvais

---

canBeRemoved	<i>canBeRemoved</i>
--------------	---------------------

---

**Description**

canBeRemoved return

**Usage**

canBeRemoved(vertexId, g, allVertices, DEGREE\_LIMIT\_FOR\_CANDIDATE\_VERTICE)

**Arguments**

vertexId	a vertex id
g	a igrph
allVertices	list of all vertices
DEGREE_LIMIT_FOR_CANDIDATE_VERTICE	degree

**Author(s)**

M.Beauvais

---

checkIsotropy	<i>Check the isotropy of a sample</i>
---------------	---------------------------------------

---

**Description**

checkIsotropy is used to check the isotropy of an [AFMImage](#). A directional variogram is calculated for various directions. If the variogram is very similar for all the directions then the sample is isotropic.

**Usage**

checkIsotropy(AFMImage, AFMImageAnalyser)

**Arguments**

AFMImage	an <a href="#">AFMImage</a> to be analysed
AFMImageAnalyser	an <a href="#">AFMImageAnalyser</a> to perform the analysis

**Value**

an [AFMImageAnalyser](#) containing the directional variograms

**Author(s)**

M.Beauvais

**Examples**

```
library(AFM)
library(ggplot2)

data(AFMImageOfAluminiumInterface)
AFMImage<-extractAFMImage(AFMImageOfAluminiumInterface, 0, 0, 32)
AFMImageAnalyser<-new("AFMImageAnalyser", AFMImage= AFMImage, fullfilename = AFMImage@fullfilename)
AFMImageAnalyser<-checkIsotropy(AFMImage,AFMImageAnalyser)
varios<-AFMImageAnalyser@variogramAnalysis@directionalVariograms
p2 <- ggplot(varios, aes(x=dist, y=gamma,
                        color= as.factor(dir.hor), shape=as.factor(dir.hor)))
p2 <- p2 + expand_limits(y = 0)
p2 <- p2 + geom_point()
p2 <- p2 + geom_line()
p2 <- p2 + ylab("semivariance (nm^2)")
p2 <- p2 + xlab("distance (nm)")
p2 <- p2 + ggtitle("Directional")
p2
```

---

checkNormality

*Check visually of the normality of the sample*

---

**Description**

checkNormality performs a visual check to know if the distribution of heights of an [AFMImage](#) follows a normal distribution. The function displays Quantile/Quantile and distribution plots.

**Usage**

```
checkNormality(..., AFMImage)
```

**Arguments**

...                      pngfullfilename (optional): directory and filename to save the visual check to png or pdf  
                            pdffullfilename(optional): directory and filename to save the visual check to pdf

AFMImage                an [AFMImage](#) from Atomic Force Microscopy

**Author(s)**

M.Beauvais

**References**

Olea2006, Ricardo A. Olea "A six-step practical approach to semivariogram modeling", 2006, "Stochastic Environmental Research and Risk Assessment, Volume 20, Issue 5 , pp 307-318"

**Examples**

```
## Not run:
library(AFM)

# display Quantile/Quantile and distribution plots.
data(AFMImageOfNormallyDistributedHeights)
checkNormality(AFMImage= AFMImageOfNormallyDistributedHeights)

# display and save on disk Quantile/Quantile and distribution plots.
data(AFMImageOfNormallyDistributedHeights)
checkNormality(AFMImage= AFMImageOfNormallyDistributedHeights,
               pngfilename=paste(tempdir(), "checkNormality.png", sep="/"))

## End(Not run)
```

---

`createGraph`*create the igraph graph from the fusionned nodes and the edges*

---

**Description**

create the igraph graph from the fusionned nodes and the edges

**Usage**`createGraph(AFMImageNetworksAnalysis, fusioned = TRUE)`**Arguments**

AFMImageNetworksAnalysis

a `AFMImageNetworksAnalysis`

fusioned

TRUE if graph is created from fusionned nodes

**Author(s)**

M.Beauvais

displaygridIgraphPlot *display the network of nodes and edges*

---

**Description**

display the network of nodes and edges

**Usage**

```
displaygridIgraphPlot(AFMImageNetworksAnalysis)
```

**Arguments**

AFMImageNetworksAnalysis  
an [AFMImageNetworksAnalysis](#)

**Author(s)**

M.Beauvais

---

displaygridIgraphPlotFromEdges  
*display the network of nodes and edges*

---

**Description**

display the network of nodes and edges

**Usage**

```
displaygridIgraphPlotFromEdges(AFMImage, edges, isolates)
```

**Arguments**

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
edges	list of edges
isolates	list of isolated edges

**Author(s)**

M.Beauvais

---

displayIn3D	<i>Display a 3D image of an AFMImage and store it on disk.</i>
-------------	--

---

### Description

Display a 3D image of an AFMImage and store it on disk if fullfilename variable is set. It uses the [rgl](#) package.

### Usage

```
displayIn3D(AFMImage, width, fullfilename, changeViewpoint)
```

### Arguments

AFMImage	the AFM image to be displayed in three dimensions.
width	(optional) width of the image. Default is 512 pixels. Note: width can't be superior to screen resolution.
fullfilename	(optional) the directory and filename to save the png of the 3D image. If this variable is missing, the function will not save on disk the 3D image.
changeViewpoint	(optional) if TRUE, the viewpoint is changed. Default is TRUE.

### Author(s)

M.Beauvais

### Examples

```
library(AFM)

data(AFMImageOfAluminiumInterface)
newAFMImage<-extractAFMImage(AFMImageOfAluminiumInterface, cornerX=50, cornerY=50, size=256)
displayIn3D(newAFMImage, 1024)

## Not run:
AFMImage<-importFromNanoscope("/user/ubuntu/myNanoscopeAnalysisAFMImage.txt")
displayIn3D(AFMImage, 1024, "/users/ubuntu/myRglAFMImage.png")

## End(Not run)
```

---

```
evaluateVariogramModels
```

```
evaluateVariogramModels method to evaluate the basic variogram models
```

---

### Description

evaluateVariogramModels method to evaluate the basic variogram models available in the [gstat](#) package A [AFMImageVariogramAnalysis](#) method to handle the variogram analysis of an [AFMImage](#). The variogram models used can be seen with the `show.vgms()` function from the [gstat](#) package.

### Usage

```
evaluateVariogramModels(AFMImageVariogramAnalysis, AFMImage)
```

```
## S4 method for signature 'AFMImageVariogramAnalysis'
evaluateVariogramModels(AFMImageVariogramAnalysis,
  AFMImage)
```

### Arguments

```
AFMImageVariogramAnalysis  an object
AFMImage                    an AFMImage
```

### Examples

```
library(AFM)

data("AFMImageOfRegularPeaks")
# take an extract of the image to fasten the calculation
AFMImage<-extractAFMImage(AFMImageOfRegularPeaks, 40, 40, 32)
# e.g. AFMImage@fullfilename<-"/users/ubuntu/AFMImageOfRegularPeaks-extract.txt"
AFMImage@fullfilename<-paste(tempdir(), "AFMImageOfRegularPeaks-extract.txt", sep="/")

AFMImageAnalyser<-AFMImageAnalyser(AFMImage)

# Variogram analysis
sampleFitPercentage<-3.43/100
variogramAnalysis<-AFMImageVariogramAnalysis(sampleFitPercentage)
variogramAnalysis@omnidirectionalVariogram<-
  AFM:::calculateOmnidirectionalVariogram(AFMImage=AFMImage,
    AFMImageVariogramAnalysis=variogramAnalysis)
variogramAnalysis@directionalVariograms<-
  AFM:::calculateDirectionalVariograms(AFMImage=AFMImage,
    AFMImageVariogramAnalysis=variogramAnalysis)

# manage model evaluations
```



```

AFMImageVariogram<-variogramAnalysis@omnidirectionalVariogram
class(AFMImageVariogram)=c("gstatVariogram","data.frame")
variogramAnalysis<-evaluateVariogramModels(variogramAnalysis, AFMImage)

mergedDT<-getDTModelEvaluation(variogramAnalysis)
mergedDT
sillRangeDT<-getDTModelSillRange(variogramAnalysis)
sillRangeDT

```

---

existsEdge	<i>Does an edge exist ?</i>
------------	-----------------------------

---

### Description

existsEdge return TRUE if an edge exists for this vertex id

### Usage

```
existsEdge(AFMImage, vertexId)
```

### Arguments

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
vertexId	the vertex id

### Author(s)

M.Beauvais

---

existsSegment	<i>existsSegment checks if a segment exists in an AFMImage; check if all the heights at the segment coordinates are different to zero.</i>
---------------	--

---

### Description

existsSegment return a boolean

### Usage

```
existsSegment(AFMImage, segment)
```

### Arguments

AFMImage	a <a href="#">AFMImage</a> from Atomic Force Microscopy or a binary <a href="#">AFMImage</a>
segment	a data.table coming from the getBresenham2Dsegment #TODO Segment class

**Value**

TRUE if all the heights of the segment are different from zero

**Author(s)**

M.Beauvais

---

exportToSTL

*Export an AFM Image as a STL format file.*

---

**Description**

Export an [AFMImage](#) as a STL format file thanks to the [rgl](#) package. The STL file can be used as an input for a 3D printing software tool.

exportToSTL is compatible with slicr (<http://slic3r.org>) version 1.2.9 (GPL v3 licence). In order to 3D print the AFM Image with slic3r, do as following:

- Use "File> Repair STL file..." menu option to create a file with the obj extension.
- Use "Add" button below the menu to display your AFM Image on the print board
- Right click on your AFM image. Use "Scale> uniformly" option, Set "15"

**Usage**

```
exportToSTL(AFMImage3DModelAnalysis, AFMImage, stlfullfilename)
```

**Arguments**

AFMImage3DModelAnalysis

an [AFMImage3DModelAnalysis](#)

AFMImage

an [AFMImage](#) from Atomic Force Microscopy

stlfullfilename

directory and filename to save as a stl file

**Author(s)**

M.Beauvais

**Examples**

```
## Not run:
library(AFM)
```

```
data(AFMImageOfAluminiumInterface)
newAFMImage<-extractAFMImage(AFMImageOfAluminiumInterface, cornerX=50, cornerY=50, size=64)
exportToSTL(newAFMImage, paste(tempdir(), "myFile.stl", sep="/"))
```

```
## End(Not run)
```

---

extractAFMImage	<i>Extract a portion of an AFM image.</i>
-----------------	---

---

### Description

The extract will be a square of the specified size. If the size is too large for the original [AFMImage](#), only the biggest valid size will be kept.

### Usage

```
extractAFMImage(AFMImage, cornerX, cornerY, size)
```

### Arguments

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
cornerX	horizontal coordinates of the extract
cornerY	vertical coordinates of the extract
size	square size of the extract in number of pixels

### Details

extractAFMImage returns an extract of the AFMImage

### Value

a new [AFMImage](#) sample

### Author(s)

M.Beauvais

### Examples

```
data(AFMImageOfAluminiumInterface)
anAFMImageExtract<-extractAFMImage(AFMImageOfAluminiumInterface, 15, 15, 256)
```

---

filterAFMImage	<i>filter the heights of an AFMImage with a minimum and a maximum value</i>
----------------	---

---

**Description**

filterAFMImage returns a filtered AFMImage

**Usage**

```
filterAFMImage(AFMImage, Min, Max)
```

**Arguments**

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
Min	the minimum height value to keep
Max	the maximum height value to keep

**Value**

an [AFMImage](#)

**Author(s)**

M.Beauvais

---

fusionCloseNodes	<i>fusion the nodes that are intersecting</i>
------------------	---

---

**Description**

fusion the nodes that are intersecting

**Usage**

```
fusionCloseNodes(AFMImageNetworksAnalysis)
```

**Arguments**

AFMImageNetworksAnalysis	the AFMImageNetworksAnalysis instance
--------------------------	---------------------------------------

**Value**

a list of edges with fused nodes

**Author(s)**

M.Beauvais

---

generateAFMImageReport

*Generate an analysis report from an AFMImageAnalyser object*

---

**Description**

generateAFMImageReport generates a report from an AFMImageAnalyser object

**Usage**

generateAFMImageReport(AFMImageAnalyser, reportFullfilename, isCheckReport)

**Arguments**

AFMImageAnalyser

an [AFMImageAnalyser](#) to be used to produce report

reportFullfilename

location on disk where to save the generated report

isCheckReport

TRUE to generate a check report must be generated, FALSE to generate a full report

**Author(s)**

M.Beauvais

---

generateCheckReport

*Generate a check report for one AFMImage*

---

**Description**

Generate a check report in pdf format in order to analyse the distribution and the isotropy of heights of the [AFMImage](#).

**Usage**

generateCheckReport(AFMImage)

**Arguments**

AFMImage

an [AFMImage](#) imported from Nanoscope Analysis(TM) with `importFromNanoscope` or created manually [AFMImage](#)

**Author(s)**

M.Beauvais

**Examples**

```
## Not run:
library(AFM)

# Analyse the AFMImageOfRegularPeaks AFMImage sample from this package
data("AFMImageOfRegularPeaks")
AFMImage<-AFMImageOfRegularPeaks
# exportDirectory="C:/Users/my_windows_login" or exportDirectory="/home/ubuntu"
exportDirectory=tempdir()
AFMImage@fullfilename<-paste(exportDirectory,"AFMImageOfRegularPeaks.txt",sep="/")

# Start to check if your sample is normaly distributed and isotropic.
generateCheckReport(AFMImage)
# If the sample is normaly distributed and isotropic, generate a full report
generateReport(AFMImage)

# Analyse your own AFM image from nanoscope analysis (TM) software tool
anotherAFMImage<-importFromNanoscope("c:/users/me/myimage.txt")
# Start to check if your sample is normaly distributed and isotropic.
generateCheckReport(anotherAFMImage)
# If your sample is normaly distributed and isotropic, generate a full report
generateReport(anotherAFMImage)

## End(Not run)
```

---

`generateReport`*Generate an analysis report for one AFMImage*

---

**Description**

A function to analyse an [AFMImage](#) and save on disk the analysis. The analysis are saved in outputs directory located in the image directory. All the rdata and image files in the reportDirectory directory are loaded to generate one report for one [AFMImage](#).

**Usage**

```
generateReport(AFMImage)
```

**Arguments**

AFMImage      an [AFMImage](#) to be analysed

**Author(s)**

M.Beauvais

## Examples

```
## Not run:
library(AFM)

# Analyse the AFMImageOfRegularPeaks AFMImage sample from this package
data("AFMImageOfRegularPeaks")
AFMImage<-AFMImageOfRegularPeaks

# exportDirectory="C:/Users/my_windows_login" or exportDirectory="/home/ubuntu"
exportDirectory=tempdir()
AFMImage@fullfilename<-paste(exportDirectory,"AFMImageOfRegularPeaks.txt",sep="/")

# Start to check if your sample is normally distributed and isotropic.
generateCheckReport(AFMImage)
# If the sample is normally distributed and isotropic, generate a full report
generateReport(AFMImage)

# Analyse your own AFM image from nanoscope analysis (TM) software tool
anotherAFMImage<-importFromNanoscope("c:/users/my_windows_login/myimage.txt")
# Start to check if your sample is normally distributed and isotropic.
generateCheckReport(anotherAFMImage)
# If your sample is normally distributed and isotropic, generate a full report
generateReport(anotherAFMImage)

## End(Not run)
```

---

generateReportFromNanoscopeImageDirectory

*Generate a pdf report for all AFM images in a directory*

---

## Description

A function to generate a pdf report for each [AFMImage](#) in a directory. Images should be in export Nanoscope format as the [importFromNanoscope](#) function will be used.

## Usage

```
generateReportFromNanoscopeImageDirectory(imageDirectory, imageNumber)
```

## Arguments

`imageDirectory` a directory where are located image as Nanoscope export format  
`imageNumber` (optional) an image number in the directory. If it is set only the selected image will be processed.

## Author(s)

M.Beauvais

**Examples**

```

library(AFM)
# A report will be generated for all the images in imageDirectory directory
# imageDirectory="c:/images"
imageDirectory=tempdir()
exit<-generateReportFromNanoscopeImageDirectory(imageDirectory)

# A report will be generated for the fifth image in the imageDirectory directory
exit<-generateReportFromNanoscopeImageDirectory(imageDirectory,5)

```

---

`getAngle`                      *calculate the angle between two vectors*

---

**Description**

calculate the angle between two vectors

**Usage**

```
getAngle(x, y)
```

**Arguments**

x	a vector
y	a vector

**Value**

the angle between the vectors

**Author(s)**

M.Beauvais

---

```

getAutoIntersectionForOmnidirectionalVariogram
           Calculate slopes and intersections in variogram
getAutoIntersectionForOmnidirectionalVariogram returns
           the slope in the omnidirectional variograms

```

---

**Description**

Calculate slopes and intersections in variogram `getAutoIntersectionForOmnidirectionalVariogram` returns the slope in the omnidirectional variograms



**Usage**

```
getAutoIntersectionForOmnidirectionalVariogram(AFMImageAnalyser)
```

**Arguments**

AFMImageAnalyser  
an [AFMImageAnalyser](#)

**Value**

an [AFMImageVariogramSlopesAnalysis](#)

**Author(s)**

M.Beauvais

---

*getAutoIntersectionForRoughnessAgainstLengthscale*  
*get the intersection between tangente and plateau*

---

**Description**

[getAutoIntersectionForRoughnessAgainstLengthscale](#) get the intersection between tangente and plateau

**Usage**

```
getAutoIntersectionForRoughnessAgainstLengthscale(AFMImageAnalyser,  
second_slope = FALSE)
```

**Arguments**

AFMImageAnalyser  
an [AFMImageAnalyser](#) to get Roughness against lengthscale calculation

second\_slope a boolean to manage first or second slope in the roughness against lengthscale curve

**Value**

a [AFMImagePSDSlopesAnalysis](#)

**Author(s)**

M.Beauvais

---

`getAutomaticWidthForVariogramCalculation`  
*calculate a width to be used for experimental variogram calculation*

---

### **Description**

calculate a width to be used for experimental variogram calculation in order to generate a line instead of a cloud of points. If the chosen width is too small, the experimental variogram will be a cloud of points instead of a line.

### **Usage**

```
getAutomaticWidthForVariogramCalculation(AFMImage)
```

### **Arguments**

AFMImage      an [AFMImage](#) from Atomic Force Microscopy

### **Details**

`getAutomaticWidthForVariogramCalculation` returns the width to be used for variogram calculation

### **Value**

the smallest width to be used for variogram calculation

### **Author(s)**

M.Beauvais

### **Examples**

```
library(AFM)  
  
data(AFMImageOfAluminiumInterface)  
print(getAutomaticWidthForVariogramCalculation(AFMImageOfAluminiumInterface))
```

---

getBresenham2DSegment *get a segment of points thanks to Bresenham line algorithm*

---

**Description**

getBresenham2DSegment return the Bresenham segment in 2D from extremities coordinates

**Usage**

```
getBresenham2DSegment(x1, y1, x2, y2)
```

**Arguments**

x1	abscissa coordinates of the first point
y1	ordinate coordinates of the first point
x2	abscissa coordinates of the second point
y2	ordinate coordinates of the second point

**Value**

a data.table of points - data.table(x, y)

**Author(s)**

M.Beauvais

---

getCircleSpatialPoints  
*get the spatial points on the circle including the center of the circle*

---

**Description**

get the spatial points on the circle including the center of the circle

**Usage**

```
getCircleSpatialPoints(binaryAFMImage, center, circleRadius)
```

**Arguments**

binaryAFMImage	a binary <a href="#">AFMImage</a> from Atomic Force Microscopy
center	the center of the circle with center\$lon as the x coordinates and center\$lat as the y coordinates
circleRadius	the radius of the circle

**Value**

a [SpatialPoints](#) object of all the points of the circle including the center of the circle

**Author(s)**

M.Beauvais

---

getCoordinatesFromVertexId

*Get x,y coordinates from vertex id*

---

**Description**

getCoordinatesFromVertexId return a list x,y coordinates

**Usage**

```
getCoordinatesFromVertexId(AFMImage, vId)
```

**Arguments**

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
vId	the vertex id

**Author(s)**

M.Beauvais

---

getDTModelEvaluation *getDTModelEvaluation method*

---

**Description**

getDTModelEvaluation method

**Usage**

```
getDTModelEvaluation(AFMImageVariogramAnalysis)
```

```
## S4 method for signature 'AFMImageVariogramAnalysis'
getDTModelEvaluation(AFMImageVariogramAnalysis)
```

**Arguments**

AFMImageVariogramAnalysis	an <a href="#">AFMImageVariogramAnalysis</a> object
---------------------------	---

---

getDTModelSillRange    *getDTModelSillRange method*

---

### Description

getDTModelSillRange method

### Usage

```
getDTModelSillRange(AFMImageVariogramAnalysis)
```

```
## S4 method for signature 'AFMImageVariogramAnalysis'  
getDTModelSillRange(AFMImageVariogramAnalysis)
```

### Arguments

AFMImageVariogramAnalysis  
    an AFMImageVariogramAnalysis object

---

getFractalDimensions    *Calculate 2D fractal dimensions and scales of an AFM Image*

---

### Description

getFractalDimensions calculates fractal dimensions and scales of an [AFMImage](#) with [fd.estim.method](#) from the [fractaldim](#) package.

### Usage

```
getFractalDimensions(AFMImage, AFMImageFractalDimensionsAnalysis)
```

### Arguments

AFMImage            an [AFMImage](#) from Atomic Force Microscopy  
AFMImageFractalDimensionsAnalysis  
    an [AFMImageFractalDimensionsAnalysis](#) to store the results of the fractal analysis

### Value

a list of [AFMImageFractalDimensionMethod](#) objects with the calculated fractal dimensions and scales

### Author(s)

M.Beauvais

**References**

Gneiting2012, Tilmann Gneiting, Hana Sevcikova and Donald B. Percival 'Estimators of Fractal Dimension: Assessing the Roughness of Time Series and Spatial Data - Statistics in statistical Science, 2012, Vol. 27, No. 2, 247-277'

**See Also**

[fractaldim](#)

**Examples**

```
library(AFM)

data(AFMImageOfAluminiumInterface)
print(getFractalDimensions(AFMImageOfAluminiumInterface))
```

---

`getIntersectionPointWithBorder`  
*getIntersectionPointWithBorder to be described*

---

**Description**

`getIntersectionPointWithBorder` return a `data.table`

**Usage**

```
getIntersectionPointWithBorder(AFMImage, center, r, deg)
```

**Arguments**

AFMImage	a <a href="#">AFMImage</a> from Atomic Force Microscopy
center	center
r	radius
deg	degree

**Author(s)**

M.Beauvais

---

```
getListOfDiameters    getListOfDiameters
```

---

**Description**

getListOfDiameters return

**Usage**

```
getListOfDiameters(g)
```

**Arguments**

g                    list of igraph networks

**Author(s)**

M.Beauvais

---

```
getLogLogOmnidirectionalSlopeGraph
```

*Get the graph of the Log Log omnidirection variogram  
getLogLogOmnidirectionalSlopeGraph returns Get the graph  
of the Log Log omnidirectional variogram*

---

**Description**

Get the graph of the Log Log omnidirection variogram getLogLogOmnidirectionalSlopeGraph returns Get the graph of the Log Log omnidirectional variogram

**Usage**

```
getLogLogOmnidirectionalSlopeGraph(AFMImageAnalyser, withFratcalSlope = FALSE)
```

**Arguments**

AFMImageAnalyser

an [AFMImageAnalyser](#)

withFratcalSlope

a boolean to indicate if the graph should contain a line representating the slope for the calculation of the fractal index and topothesy

**Value**

a ggplot2 graph

**Author(s)**

M.Beauvais

**Examples**

```
## Not run:
library(AFM)
library(ggplot2)

data(AFMImageOfRegularPeaks)

AFMImageAnalyser = new("AFMImageAnalyser",
  fullfilename="/home/ubuntu/AFMImageOfRegularPeaks-Analyser.txt")
variogramAnalysis<-AFMImageVariogramAnalysis(sampleFitPercentage=3.43/100)
AFMImageAnalyser@variogramAnalysis<-variogramAnalysis
AFMImageAnalyser@variogramAnalysis@omnidirectionalVariogram<-
  calculateOmnidirectionalVariogram(AFMImage= AFMImageOfRegularPeaks,
    AFMImageVariogramAnalysis= variogramAnalysis)
p<-getLogLogOmnidirectionalSlopeGraph(AFMImageAnalyser, withFratcalSlope=TRUE)
p

## End(Not run)
```

---

```
getNetworkGridLayout #' @export getCoordinatesFromVertexId2<-function(AFMImage,
  vId) vertexId<-as.numeric(vId) y<-floor(vertexId/HASHSIZE)
  x<-vertexId-y*HASHSIZE return(data.table(vId=vId, co-
  ords.x1=x,coords.x2=y)) Get getNetworkGridLayout
```

---

**Description**

getNetworkGridLayout return a list x,y coordinates

**Usage**

```
getNetworkGridLayout(AFMImage, vId)
```

**Arguments**

AFMImage        an [AFMImage](#) from Atomic Force Microscopy  
vId             the vertex id

**Author(s)**

M.Beauvais



---

`getNyquistSpatialFrequency`

*Get the Nyquist spatial frequency*

---

### **Description**

Get the Nyquist spatial frequency of an [AFMImage](#) calculated as following:  
0.5 multiplied by the minimum between the horizontal scansize divided by the number of samples per line and the vertical scansize divided by the number of lines

### **Usage**

```
getNyquistSpatialFrequency(AFMImage)
```

```
## S4 method for signature 'AFMImage'  
getNyquistSpatialFrequency(AFMImage)
```

### **Arguments**

AFMImage            an [AFMImage](#) from Atomic Force Microscopy

### **Details**

`getNyquistSpatialFrequency` returns the Nyquist spatial frequency as a numeric

### **Value**

the Nyquist spatial frequency of the [AFMImage](#)

### **Author(s)**

M.Beauvais

### **Examples**

```
library(AFM)  
  
data(AFMImageOfNormallyDistributedHeights)  
NyquistSpatialFrequency<-getNyquistSpatialFrequency(AFMImageOfNormallyDistributedHeights)  
print(NyquistSpatialFrequency)
```

getPaddedAFMImage      *Get a zero padded AFMImage*

---

### Description

Get a zero padded [AFMImage](#) useful in Power Spectral Density analysis. The original [AFMImage](#) is padded with zero in order to get a larger square [AFMImage](#) which size is a power of 2.

### Usage

```
getPaddedAFMImage(AFMImage)
```

### Arguments

AFMImage      an [AFMImage](#) from Atomic Force Microscopy

### Value

a zero-padded [AFMImage](#) with a fullfilename equals to the original fullfilename pasted with padded-to-"ScanSize".txt

### Author(s)

M.Beauvais

### Examples

```
library(AFM)

data(AFMImageOfNormallyDistributedHeights)
paddedAFMImage<-getPaddedAFMImage(AFMImageOfNormallyDistributedHeights)
displayIn3D(AFMImage= paddedAFMImage, width= 1024)
```

---

getRoughnessParameters  
*Get Roughness parameters*

---

### Description

Get basic roughness parameters as amplitude parameters: Total root mean square Roughness or Total Rrms or totalRMSRoughness\_TotalRrms  
Mean roughness or Ra or MeanRoughness\_Ra

### Usage

```
getRoughnessParameters(AFMImage)  
  
## S4 method for signature 'AFMImage'  
getRoughnessParameters(AFMImage)
```

### Arguments

AFMImage            an [AFMImage](#) from Atomic Force Microscopy

### Details

getRoughnessParameters returns a data.table of roughness parameters

### Value

a data.table of roughness parameters:

- totalRMSRoughness\_TotalRrms the total RMS Roughness as the square root of the variance of heights
- MeanRoughness\_Ra the average roughness as the mean of absolute value of heights

### Author(s)

M.Beauvais

### Examples

```
library(AFM)  
  
data(AFMImageOfAluminiumInterface)  
roughnessParameters<-getRoughnessParameters(AFMImageOfAluminiumInterface)  
print(roughnessParameters)
```

---

getSpplotFromAFMImage *Get an AFMImage as a Lattice (trellis) plot*

---

### Description

get a Lattice (trellis) plot of an [AFMImage](#) using the [spplot](#) method of the [sp](#) package. This function is used to evaluate visually the quality of the predicted surface when a variogram model is used.

### Usage

```
getSpplotFromAFMImage(AFMImage, expectedWidth, expectHeight, withoutLegend)
```

**Arguments**

AFMImage            an [AFMImage](#) from Atomic Force Microscopy  
 expectedWidth    (optional) expected width of the saved image. Default is 400px.  
 expectHeight     (optional) expected height of the saved image. Default is 300px.  
 withoutLegend    (optional) set at FALSE, the cuts legend will be included in the plot. Default is FALSE.

**Details**

getSpplotFromAFMImage get a Lattice (trellis) plot of an [AFMImage](#) on disk

**Author(s)**

M.Beauvais

**Examples**

```
## Not run:
library(AFM)

data(AFMImageOfAluminiumInterface)
p<-getSpplotFromAFMImage(AFMImageOfAluminiumInterface, 800,800, TRUE)
print(p)

## End(Not run)
```

---

getSurroundingVertexesList

*Get surrounding vertexes from x,y coordinates*

---

**Description**

getSurroundingVertexesList return the vertexId

**Usage**

```
getSurroundingVertexesList(AFMImage, x, y)
```

**Arguments**

AFMImage            an [AFMImage](#) from Atomic Force Microscopy  
 x                    coordinates in x axis  
 y                    coordinates in y axis

**Author(s)**

M.Beauvais

---

getTopologyAFMImage     *Calculate topology image (TBC)*

---

**Description**

getTopologyAFMImage return the global topological distance

**Usage**

getTopologyAFMImage(BinaryAFMImage, AFMImageNetworksAnalysis)

**Arguments**

BinaryAFMImage     an [AFMImage](#) from Atomic Force Microscopy in a binary format 0 or 1 values for heights

AFMImageNetworksAnalysis  
                          an [AFMImageNetworksAnalysis](#) from Atomic Force Microscopy

**Author(s)**

M.Beauvais

---

getTriangle             *get a triangle starting from center, two segments of length r with angles deg1 and deg2*

---

**Description**

getTriangle return a data.table points of a triangle

**Usage**

getTriangle(AFMImage, center, r, deg1, deg2)

**Arguments**

AFMImage             a [AFMImage](#) from Atomic Force Microscopy

center                center

r                        length of segment

deg1                  angle 1

deg2                  angel 2

**Author(s)**

M.Beauvais

getVertexId                    *Get vertex id from x,y coordinates*

---

**Description**

getVertexId return the vertexId

**Usage**

```
getVertexId(AFMImage, x, y)
```

**Arguments**

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
x	coordinates in x axis
y	coordinates in y axis

**Author(s)**

M.Beauvais

---

gridIgraphPlot                *gridIgraphPlot*

---

**Description**

gridIgraphPlot return TRUE if vertex is adjacent to a better vertex

**Usage**

```
gridIgraphPlot(AFMImage, g)
```

**Arguments**

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
g	the networks

**Author(s)**

M.Beauvais

---

identifyEdgesFromCircles  
*display the network of nodes and edges*

---

**Description**

display the network of nodes and edges

**Usage**

```
identifyEdgesFromCircles(AFMImageNetworksAnalysis, MAX_DISTANCE = 40,  
    ANGLE_MIN = (pi/3.25))
```

**Arguments**

AFMImageNetworksAnalysis  
a [AFMImageNetworksAnalysis](#)

MAX\_DISTANCE the maximum distance between nodes to check if nodes are connected. Default value is 40.

ANGLE\_MIN the minimum angle between edges to accept or reject an edge

**Author(s)**

M.Beauvais

---

identifyIsolatedNodes *identify isolated nodes comparing the list of edges and the list of nodes*

---

**Description**

identify isolated nodes comparing the list of edges and the list of nodes

**Usage**

```
identifyIsolatedNodes(AFMImageNetworksAnalysis)
```

**Arguments**

AFMImageNetworksAnalysis  
the AFMImageNetworksAnalysis instance

**Value**

the updated instance of AFMImageNetworksAnalysis

**Author(s)**

M.Beauvais

---

identifyNodesWithCircles

*identify largest circles in binary image*

---

### Description

identifyCirclesInBinaryImage return TRUE if vertex is adjacent to a better vertex

### Usage

```
identifyNodesWithCircles(AFMImageNetworksAnalysis, smallBranchesTreatment)
```

### Arguments

AFMImageNetworksAnalysis

a [AFMImageNetworksAnalysis](#)

smallBranchesTreatment

TRUE if the algorithm should try to identify very small branches

### Value

AFMImageNetworksAnalysis the [AFMImageNetworksAnalysis](#) instance

### Author(s)

M.Beauvais

---

importFromNanoscope *Import data from nanoscope analysis(tm) tool*

---

### Description

The imported file should contain a header and list of heights The header should contain the following fields:

- Lines: number of scanned lines (e.g. 512)
- Sampslines: number of scan per line (e.g. 512)
- ScanSize: the sample size (e.g. 1000nm) the extension nm is mandatory and will be removed

### Usage

```
importFromNanoscope(fullfilename)
```



**Arguments**

fullfilename a concatenated string of the directory and filename exported with Nanoscope analysis(TM) software

**Details**

importFromNanoscope returns an [AFMImage](#)

**Author(s)**

M.Beauvais

**Examples**

```
## Not run:
library(AFM)

fullfilename<-" /user/ubuntu/NanoscopeFlattenExportedFile.txt"
myAFMimage<-importFromNanoscope(fullfilename)
displayIn3D(myAFMimage, 1024)

## End(Not run)
```

---

initialize,AFMImageAnalyser-method

*Constructor method of AFMImageAnalyser Class.*

---

**Description**

Constructor method of AFMImageAnalyser Class.

**Usage**

```
## S4 method for signature 'AFMImageAnalyser'
initialize(.Object, AFMImage, variogramAnalysis,
  psdAnalysis, fdAnalysis, networksAnalysis, threeDimensionAnalysis, mean,
  variance, TotalRrms, Ra, fullfilename)
```

**Arguments**

.Object an AFMImageAnalyser object  
AFMImage an AFMImage  
variogramAnalysis [AFMImageVariogramAnalysis](#)  
psdAnalysis [AFMImagePSDAnalysis](#)  
fdAnalysis [AFMImageFractalDimensionsAnalysis](#)

networksAnalysis	<a href="#">AFMImageNetworksAnalysis</a>
threeDimensionAnalysis	<a href="#">AFMImage3DModelAnalysis</a>
mean	the mean of heights of the <a href="#">AFMImage</a>
variance	the variance of heights of the <a href="#">AFMImage</a>
TotalRrms	the total Root Mean Square Roughness of the <a href="#">AFMImage</a> calculated from variance
Ra	mean roughness or mean of absolute values of heights
fullfilename	to be removed?

---

`isAdjacentToBetterVertex`

*isAdjacentToBetterVertex*

---

### **Description**

`isAdjacentToBetterVertex` return TRUE if vertex is adjacent to a better vertex

### **Usage**

`isAdjacentToBetterVertex(AFMImage, x, y)`

### **Arguments**

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
x	coordinates in x axis
y	coordinates in y axis

### **Author(s)**

M.Beauvais

---

isAngleBetweenEdgesAlwaysSuperiorToMinAngle

*check if all the angles between one edge and a list of edges is superior to a specified value.*

---

### Description

check if all the angles between one edge and a list of edges is superior to a specified value.

### Usage

```
isAngleBetweenEdgesAlwaysSuperiorToMinAngle(binaryAFMImage, edge1, edges2,
minAngle)
```

### Arguments

binaryAFMImage	a binary <a href="#">AFMImage</a> from Atomic Force Microscopy
edge1	one edge
edges2	list of edges
minAngle	the minimum angle value

### Value

TRUE if all the angle are superior to the specified value

### Author(s)

M.Beauvais

---

makeBinaryAFMImage	<i>make a binary AFMImage setting all the heights different to 0 to 1.</i>
--------------------	--

---

### Description

makeBinaryAFMImage returns a binary AFMImage

### Usage

```
makeBinaryAFMImage(AFMImage)
```

### Arguments

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
----------	--

**Value**

an [AFMImage](#)

**Author(s)**

M.Beauvais

---

multiplyHeightsAFMImage

*multiply the heights of an AFMImage*

---

**Description**

multiplyHeightsAFMImage returns a simplified AFMImage

**Usage**

```
multiplyHeightsAFMImage(AFMImage, multiplier)
```

**Arguments**

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
multiplier	the number to multiply the heights with

**Value**

an [AFMImage](#)

**Author(s)**

M.Beauvais

**Examples**

```
data(AFMImageOfAluminiumInterface)
newAFMImage<-multiplyHeightsAFMImage(AFMImageOfAluminiumInterface,10)
displayIn3D(newAFMImage)
```

---

```
performAllPSDCalculation
    Perform all the calculation for PSD exploitation
```

---

## Description

`performAllPSDCalculation` perform all the calculation for PSD exploitation

## Usage

```
performAllPSDCalculation(AFMImagePSDAnalysis, AFMImage)
```

## Arguments

AFMImagePSDAnalysis      an [AFMImagePSDAnalysis](#) to manage and store the results of PSD analysis

AFMImage                  an [AFMImage](#) from Atomic Force Microscopy

## Author(s)

M.Beauvais

## Examples

```
## Not run:
library(AFM)

data(AFMImageOfNormallyDistributedHeights)

newAFMImage<-AFMImageOfNormallyDistributedHeights
newAFMImage@fullfilename<-"C:/Users/one/AFMImageOfNormallyDistributedHeights.txt"
psdAnalysis<-AFMImagePSDAnalysis()
# Create a closure to update progress
psdAnalysis@updateProgress<- function(value = NULL, detail = NULL, message = NULL) {
  if (exists("progressPSD")){
    if (!is.null(message)) {
      progressPSD$set(message = message, value = 0)
    }else{
      progressPSD$set(value = value, detail = detail)
    }
  }
}
psdAnalysis@psd1d_breaks<-2^3
psdAnalysis@psd2d_truncHighLengthScale<-TRUE
psdAnalysis<-performAllPSDCalculation(AFMImagePSDAnalysis= psdAnalysis, AFMImage= newAFMImage)
print("done psdAnalysis")

## End(Not run)
```

---

```
printVariogramModelEvaluations
    printVariogramModelEvaluations
```

---

### Description

printVariogramModelEvaluations generates a graphic element containing the evaluation of all variogram models

### Usage

```
printVariogramModelEvaluations(AFMImageAnalyser, sampleDT,
    numberOfModelsPerPage)
```

### Arguments

AFMImageAnalyser  
                                   an [AFMImageAnalyser](#) to be used to produce report

sampleDT                  a data.table containing the evaluation information

numberOfModelsPerPage  
                                   numeric to specify the number of model evaluations per pages

### Author(s)

M.Beauvais

---

PSD1DAgainstFrequency *Calculate the 1D Power Spectral Density; returns a data table of PSD 1D and PSD 2D values against spatial frequencies. As mentionned in Sidick2009, this function calculates the PSD against spatial frequencies in 1D from [PSD2DAgainstFrequency](#) by using breaks in the log space to sum PSD 2D and frequency values.*

---

### Description

Calculate the 1D Power Spectral Density; returns a data table of PSD 1D and PSD 2D values against spatial frequencies.

As mentionned in Sidick2009, this function calculates the PSD against spatial frequencies in 1D from [PSD2DAgainstFrequency](#) by using breaks in the log space to sum PSD 2D and frequency values.

### Usage

```
PSD1DAgainstFrequency(AFMImage, AFMImagePSDAnalysis)
```

```
## S4 method for signature 'AFMImage'
```

```
PSD1DAgainstFrequency(AFMImage, AFMImagePSDAnalysis)
```

**Arguments**

AFMImage            an AFMImage to be analysed  
 AFMImagePSDAnalysis  
                       n AFMImagePSDAnalysis to store the setup and results of PSD analysis

**Value**

PSD1DAgainstFrequency returns a data table of frequencies and PSD values

- freq: the considered frequency
- PSD: the considered PSD value
- type: PSD-1D
- fullfilename: directory and filename on the disk

**Examples**

```
## Not run:
library(AFM)
library(ggplot2)
library(plyr)
library(scales)
data("AFMImageOfNormallyDistributedHeights")
newAFMImage<-AFMImageOfNormallyDistributedHeights
newAFMImage@fullfilename<-"C:/Users/one/AFMImageOfNormallyDistributedHeights.txt"
psdAnalysis<-AFMImagePSDAnalysis()
# Create a closure to update progress
psdAnalysis@updateProgress<- function(value = NULL, detail = NULL, message = NULL) {
  if (exists("progressPSD")){
    if (!is.null(message)) {
      progressPSD$set(message = message, value = 0)
    }else{
      progressPSD$set(value = value, detail = detail)
    }
  }
}
psdAnalysis@psd1d_breaks<-2^3
psdAnalysis@psd2d_truncHighLengthScale<-TRUE
psdAnalysis<-performAllPSDCalculation(AFMImagePSDAnalysis= psdAnalysis, AFMImage= newAFMImage)
datap<-psdAnalysis@psd1d
p <- ggplot(data=datap)
p <- p + geom_point(aes(freq, PSD, color=type),data=datap[datap$type %in% c("PSD-2D")])
p <- p + geom_line(aes(freq, PSD, color=type),data=datap[datap$type %in% c("PSD-1D")],size=1.1)
p <- p + scale_x_log10()
p <- p + scale_y_log10()
p <- p + ylab("PSD (nm^4)")
p <- p + xlab("Frequency (nm^-1)")
p

## End(Not run)
```

---

PSD2DAgainstFrequency *Calculate the 2D Power Spectral Density*

---

### Description

PSD2DAgainstFrequency returns a data table of PSD 2D values against spatial frequencies

### Usage

```
PSD2DAgainstFrequency(AFMImage, AFMImagePSDAnalysis)
```

```
## S4 method for signature 'AFMImage'
PSD2DAgainstFrequency(AFMImage, AFMImagePSDAnalysis)
```

### Arguments

AFMImage            an AFMImage to be analysed  
 AFMImagePSDAnalysis            an AFMImagePSDAnalysis to store PSD analysis results

### Value

PSD2DAgainstFrequency returns a data table of frequencies and PSD values

- freq: the considered frequency
- PSD: the considered PSD value
- type: PSD-2D
- fullfilename: directory and filename on the disk

### References

Sidick2009, Erkin Sidick "Power Spectral Density Specification and Analysis of Large Optical Surfaces", 2009, "Modeling Aspects in Optical Metrology II, Proc. of SPIE Vol. 7390 73900L-1"

### Examples

```
## Not run:
library(AFM)
library(ggplot2)
library(plyr)

# Calculate Power Spectrum Density in 2D against frequency
data("AFMImageOfNormallyDistributedHeights")
oneAFMImage<-AFMImageOfNormallyDistributedHeights
psd2d<-PSD2DAgainstFrequency(oneAFMImage)
p <- ggplot(data=psd2d)
p <- p + geom_point(aes(freq, PSD, color=type),subset = .(type %in% c("PSD-2D")))
p <- p + geom_line(aes(freq, PSD, color=type),subset = .(type %in% c("PSD-1D")),size=1.1)
```



```
p <- p + scale_x_log10()
p <- p + scale_y_log10()
p <- p + ylab("PSD (nm^4)")
p <- p + xlab("Frequency (nm^-1)")
p <- p + ggtitle(basename(oneAFMImage@fullfilename))
p

## End(Not run)
```

---

putAnalysisOnDisk      *Export all data from an analysis of an AFM Image as rdata file*

---

### Description

A function to export to all the data from all analysis of an [AFMImage](#) and put them on disk as rdata file

### Usage

```
putAnalysisOnDisk(AFMImageAnalyser, AFMImage)

## S4 method for signature 'AFMImageAnalyser'
putAnalysisOnDisk(AFMImageAnalyser, AFMImage)
```

### Arguments

AFMImageAnalyser      an [AFMImageAnalyser](#)  
AFMImage              an [AFMImage](#)

### Author(s)

M.Beauvais

---

putImagesFromAnalysisOnDisk  
*Put the images from all analysis on disk*

---

### Description

A function to put on disk all the images from variogram, PSD Analysis of an [AFMImage](#) An AFM Image 3D representation is saved on disk thanks to the [rgl](#) package. On Unix system, it is necessary to have a X server connection to be able to use the [rgl](#) package.

### Usage

```
putImagesFromAnalysisOnDisk(AFMImageAnalyser, AFMImage, exportDirectory)
```

**Arguments**

AFMImageAnalyser  
   an AFMImageAnalyser  
 AFMImage                              an AFMImage  
 exportDirectory  
   where the images will be stored

**Author(s)**

M.Beauvais

---

RoughnessByLengthScale

*Calculate the roughness of the sample against length scale*

---

**Description**

The calculation of the roughness against lengthscale is performed through a FFT 2D calculation, PSD 2D calculation and a meshgrid of frequencies. RoughnessByLengthScale returns a data.table of roughnesses against length scales

**Usage**

```
RoughnessByLengthScale(AFMImage, AFMImagePSDAnalysis)

## S4 method for signature 'AFMImage'
RoughnessByLengthScale(AFMImage, AFMImagePSDAnalysis)
```

**Arguments**

AFMImage                              an AFMImage from Atomic Force Microscopy  
 AFMImagePSDAnalysis  
   n AFMImagePSDAnalysis to store the setup and results of PSD analysis

**Value**

a data table of length scale (r) and roughness values (roughness)

- roughness: roughnesses
- r: length scales
- filename: fullfilename slot of the AFMImage

**Author(s)**

M.Beauvais

**Examples**

```
library(AFM)
library(ggplot2)

data("AFMImageOfNormallyDistributedHeights")
oneAFMImage<-AFMImageOfNormallyDistributedHeights
AFMImagePSDAnalysis<-AFMImagePSDAnalysis()
data<-RoughnessByLengthScale(oneAFMImage, AFMImagePSDAnalysis)
r<-roughness<-filename<-NULL
p1 <- ggplot(data, aes(x=r, y=roughness, colour= basename(filename)))
p1 <- p1 + geom_point()
p1 <- p1 + geom_line()
p1 <- p1 + ylab("roughness (nm)")
p1 <- p1 + xlab("lengthscale (nm)")
p1
```

---

`runAFMApp`*Launch the AFM shiny application*

---

**Description**

Launch the AFM shiny graphical interface to access most of the functionalities of the AFM library

**Usage**

```
runAFMApp()
```

**Author(s)**

M.Beauvais

**Examples**

```
## Not run:
install.packages("AFM")
AFM::runAFMApp()

## End(Not run)
```

---

sampleAFMImage	<i>Get a sample of an AFM image.</i>
----------------	--------------------------------------

---

## Description

Random selection of heights to keep in an [AFMImage](#). This function can be used to calculate quickly an approximated variogram of a large image.

## Usage

```
sampleAFMImage(AFMImage, percentage)
```

## Arguments

AFMImage	an <a href="#">AFMImage</a> from Atomic Force Microscopy
percentage	percentage of heights to keep

## Details

sampleAFMImage returns a sample of the AFMImage to boost calculation time of variogram

## Value

a sample of an [AFMImage](#)

## Author(s)

M.Beauvais

## Examples

```
## Not run:
library(AFM)
library(ggplot2)

data(AFMImageOfAluminiumInterface)
anAFMImageSample<-sampleAFMImage(AFMImageOfAluminiumInterface,15)
variogramAnalysis<-AFMImageVariogramAnalysis(sampleFitPercentage=3.43)
avario<-AFM::calculateOmnidirectionalVariogram(AFMImage= anAFMImageSample,
                                               AFMImageVariogramAnalysis= variogramAnalysis)

dist<-gamma<-NULL
p1 <- ggplot(avario, aes(x=dist, y=gamma))
p1 <- p1 + geom_point()
p1 <- p1 + geom_line()
p1 <- p1 + ylab("semivariance")
p1 <- p1 + xlab("distance (nm)")
p1 <- p1 + ggtitle("Approximation of variogram thanks to sampling")
p1
```

```
## End(Not run)
```

---

saveOnDisk	<i>Save an AFM image on disk.</i>
------------	-----------------------------------

---

### Description

The function saves the an [AFMImage](#) as a rdata file. It uses the fullfilename param of the [AFMImage](#) and add "AFMImage.rda" extension to save the rdata file on disk.

### Usage

```
saveOnDisk(AFMImage, exportDirectory)
```

### Arguments

AFMImage            an [AFMImage](#) from Atomic Force Microscopy  
exportDirectory    an optional argument to change the directory where the rdata file will be stored on disk

### Details

saveOnDisk save on disk an [AFMImage](#) as rdata file

### Author(s)

M.Beauvais

### Examples

```
library(AFM)  
  
data(AFMImageOfAluminiumInterface)  
# save the rdata file of the AFMImage in the tempdir() directory;  
# select another directory to save it permanently on your hard drive  
saveOnDisk(AFMImageOfAluminiumInterface, tempdir())
```

saveOnDiskIntersectionForRoughnessAgainstLengthscale  
*save an image of the roughness against lengthscale calculations*

---

### Description

[saveOnDiskIntersectionForRoughnessAgainstLengthscale](#) save an image of the roughness against lengthscale calculations

### Usage

```
saveOnDiskIntersectionForRoughnessAgainstLengthscale(AFMImageAnalyser,  
  exportDirectory)
```

### Arguments

AFMImageAnalyser  
    an [AFMImageAnalyser](#) to get Roughness against lengthscale calculation

exportDirectory  
    a directory on the file system

### Author(s)

M.Beauvais

---

saveSpplotFromAFMImage  
*Save on disk an AFMImage as a Lattice (trellis) plot*

---

### Description

save a Lattice (trellis) plot of an [AFMImage](#) using the [spplot](#) method of the sp package. This function is used to evaluate visually the quality of the predicted surface when a variogram model is used.

### Usage

```
saveSpplotFromAFMImage(AFMImage, fullfilename, expectedWidth, expectHeight,  
  withoutLegend)
```

**Arguments**

AFMImage            an [AFMImage](#) from Atomic Force Microscopy  
 fullfilename        directory and filename to save to png  
 expectedWidth      (optional) expected width of the saved image. Default is 400px.  
 expectHeight        (optional) expected height of the saved image. Default is 300px.  
 withoutLegend      (optional) set at FALSE, the cuts legend will be included in the plot. Default is FALSE.

**Details**

saveSpplotFromAFMImage save a a Lattice (trellis) plot of an [AFMImage](#) on disk

**Author(s)**

M.Beauvais

**Examples**

```
## Not run:
library(AFM)

data(AFMImageOfAluminiumInterface)
saveSpplotFromAFMImage(AFMImageOfAluminiumInterface,
                        paste(tempdir(), "myFileWithoutLegend.png", sep="/"), 800,800, TRUE)
saveSpplotFromAFMImage(AFMImageOfAluminiumInterface,
                        paste(tempdir(), "myFileWithLegend.png", sep="/"), 800,800, FALSE)

## End(Not run)
```

---

shiftedPSDuv

*Calculate the shifted PSD matrix*

---

**Description**

shiftedPSDuv returns the Power Spectral Density matrix in the frequency space from shifted FFT 2D

**Usage**

```
shiftedPSDuv(AFMImage)
```

**Arguments**

AFMImage            an [AFMImage](#) from Atomic Force Microscopy

**Value**

$(1/NM^2) * \text{abs}(\text{shiftedFFT2Ddata})^2$  with N the number of lines of the sample and M the number of samples per line of the sample

**Author(s)**

M.Beauvais

**Examples**

```
library(AFM)
library(ggplot2)

data(AFMImageOfRegularPeaks)
AFMImage<-AFMImageOfRegularPeaks
nMheightsData= matrix(AFMImage@data$h, nrow=AFMImage@samplesperline)
shiftedPSDuv<-shiftedPSDuv(AFMImage)
a=AFMImage@scansize
b=AFMImage@scansize

M=AFMImage@sampslines
N=AFMImage@lines
NM=N*M # pixels^2
MN = M*N
A=a*b
ab=a*b

dx=a/M
dy=b/N

um = seq( (1-(M+1)/2)/(M*dx), (M-(M+1)/2)/(M*dx), by=1/(M*dx))
vn = seq( (1-(N+1)/2)/(N*dy), (N-(N+1)/2)/(N*dy), by=1/(N*dy))
x = rep(um, times = AFMImage@lines)
y = rep(vn, each = AFMImage@sampslines)
z = as.vector(shiftedPSDuv)

data<-data.frame(x=x, y=y, z=z)

p5 <- qplot(x, y, data=data, colour=log10(z))
p5 <- p5 + scale_colour_gradientn(colours = rainbow(7))
p5 <- p5 + ylab("v")
p5 <- p5 + xlab("u")
title<-paste("shifted PSD of", basename(AFMImage@fullfilename))
p5 <- p5 + ggtitle(title)
# Hide all the horizontal gridlines
p5 <- p5 + theme(panel.grid.minor.x=element_blank(), panel.grid.major.x=element_blank())
# Hide all the vertical gridlines
p5 <- p5 + theme(panel.grid.minor.y=element_blank(), panel.grid.major.y=element_blank())
p5 <- p5 + theme(panel.background = element_rect(fill = 'white', colour = 'black'))
p5
```



---

shiftFFT2D	<i>Shift the quadrants of the FFT 2D</i>
------------	--

---

**Description**

shiftFFT2D returns the FFT 2D matrix shifted to put zero frequencies in the middle.

**Usage**

```
shiftFFT2D(fft2data)
```

**Arguments**

fft2data            the FFT 2D of the AFM image

**Value**

The shifted matrix

**Author(s)**

M.Beauvais

**Examples**

```
library(AFM)
library(ffttools)

data(AFMImageOfNormallyDistributedHeights)
AFMImage<-AFMImageOfNormallyDistributedHeights
nMheightsData= matrix(AFMImage@data$h, nrow=AFMImage@samplesperline)
shiftedFFT2D<-shiftFFT2D(fft2d(nMheightsData))
```

---

simplifyAFMImage	<i>simplify an AFM image.</i>
------------------	-------------------------------

---

**Description**

The simplification is taking a very simple gridded sample of the image. It can be useful to speed up display.

**Usage**

```
simplifyAFMImage(AFMImage, newSamplesperline, newLines)
```

**Arguments**

AFMImage            an [AFMImage](#) from Atomic Force Microscopy  
newSamplesperline    the new number of samplesperline of the AFMImage  
newLines            the new number of lines of the AFMImage

**Details**

simplifyAFMImage returns a simplified AFMImage

**Value**

a new simplified [AFMImage](#)

**Author(s)**

M.Beauvais

**Examples**

```
data(AFMImageOfAluminiumInterface)
anAFMImageExtract<-simplifyAFMImage(AFMImageOfAluminiumInterface,16,16)
```

---

totalRMSRoughness	<i>Calculate the total Root Mean Square Roughness (Rrms total)</i>
-------------------	--

---

**Description**

totalRMSRoughness returns the total RMS roughness calculated from the variance of heights

**Usage**

```
totalRMSRoughness(AFMImage)
```

**Arguments**

AFMImage            an [AFMImage](#) from Atomic Force Microscopy

**Value**

a numeric as the square root of the variance of heights

**Author(s)**

M.Beauvais

## Examples

```
library(AFM)

data(AFMImageOfAluminiumInterface)
totalRMSRoughness<-totalRMSRoughness(AFMImageOfAluminiumInterface)
print(totalRMSRoughness)
```

---

transformAFMImageForNetworkAnalysis

*Multiply, filter the heights and make a binary AFMImage from the transformed AFMImage*

---

## Description

transformAFMImageForNetworkAnalysis update [AFMImageNetworksAnalysis](#) making a binary AFMImage

## Usage

```
transformAFMImageForNetworkAnalysis(AFMImageNetworksAnalysis, AFMImage)
```

```
## S4 method for signature 'AFMImageNetworksAnalysis'
transformAFMImageForNetworkAnalysis(AFMImageNetworksAnalysis,
  AFMImage)
```

## Arguments

AFMImageNetworksAnalysis  
n [AFMImageNetworksAnalysis](#) to store the results of the transformation

AFMImage  
an [AFMImage](#) from Atomic Force Microscopy

## Author(s)

M.Beauvais

---

updateProgress	<i>updateProgress</i>
----------------	-----------------------

---

**Description**

is a function used by a GUI such as shiny GUI

**Usage**

```
updateProgress(AFMImageVariogramAnalysis, value, detail, message)
```

**Arguments**

AFMImageVariogramAnalysis	an <a href="#">AFMImageVariogramAnalysis</a>
value	shiny progress bar value
detail	shiny progress bar detail
message	shiny progress bar message

# Index

- AFM, [4](#)
- AFM-package (AFM), [4](#)
- AFMImage, [4](#), [8](#), [12–14](#), [16](#), [17](#), [20–28](#), [30](#),  
[32–39](#), [42–46](#), [48–54](#), [57–61](#), [65](#), [66](#),  
[68–71](#), [74](#), [75](#)
- AFMImage (AFMImage-class), [5](#)
- AFMImage-class, [5](#)
- AFMImage3DModelAnalysis, [21](#), [22](#), [34](#), [58](#)
- AFMImage3DModelAnalysis  
(AFMImage3DModelAnalysis-class),  
[7](#)
- AFMImage3DModelAnalysis-class, [7](#)
- AFMImageAnalyser, [20](#), [27](#), [28](#), [37](#), [41](#), [47](#), [62](#),  
[65](#), [66](#), [70](#)
- AFMImageAnalyser  
(AFMImageAnalyser-class), [8](#)
- AFMImageAnalyser-class, [8](#)
- AFMImageFractalDimensionMethod, [10](#), [45](#)
- AFMImageFractalDimensionMethod  
(AFMImageFractalDimensionMethod-class),  
[9](#)
- AFMImageFractalDimensionMethod-class,  
[9](#)
- AFMImageFractalDimensionsAnalysis, [8](#),  
[10](#), [45](#), [57](#)
- AFMImageFractalDimensionsAnalysis  
(AFMImageFractalDimensionsAnalysis-class),  
[10](#)
- AFMImageFractalDimensionsAnalysis-class,  
[10](#)
- AFMImageNetworksAnalysis, [8](#), [23](#), [24](#), [26](#),  
[29](#), [30](#), [53](#), [55](#), [56](#), [58](#), [75](#)
- AFMImageNetworksAnalysis  
(AFMImageNetworksAnalysis-class),  
[11](#)
- AFMImageNetworksAnalysis-class, [11](#)
- AFMImageOfAluminiumInterface, [12](#)
- AFMImageOfNormallyDistributedHeights,  
[13](#)
- AFMImageOfOnePeak, [13](#)
- AFMImageOfRegularPeaks, [13](#)
- AFMImagePSDAnalysis, [8](#), [15](#), [57](#), [61](#)
- AFMImagePSDAnalysis  
(AFMImagePSDAnalysis-class), [14](#)
- AFMImagePSDAnalysis-class, [14](#)
- AFMImagePSDSlopesAnalysis, [41](#)
- AFMImagePSDSlopesAnalysis  
(AFMImagePSDSlopesAnalysis-class),  
[15](#)
- AFMImagePSDSlopesAnalysis-class, [15](#)
- AFMImageVarioqramAnalysis, [8](#), [22](#), [25](#), [32](#),  
[57](#), [76](#)
- AFMImageVarioqramAnalysis  
(AFMImageVarioqramAnalysis-class),  
[16](#)
- AFMImageVarioqramAnalysis-class, [16](#)
- AFMImageVarioqramModel, [17](#)
- AFMImageVarioqramModel  
(AFMImageVarioqramModel-class),  
[18](#)
- AFMImageVarioqramModel-class, [18](#)
- AFMImageVarioqramSlopesAnalysis, [41](#)
- AFMImageVarioqramSlopesAnalysis  
(AFMImageVarioqramSlopesAnalysis-class),  
[19](#)
- AFMImageVarioqramSlopesAnalysis-class,  
[19](#)
- analyse, [20](#)
- AreNodesConnected, [21](#)
- calculate3DModel, [21](#)
- calculate3DModel, AFMImage-method  
(calculate3DModel), [21](#)
- calculate3DModel, AFMImage3DModelAnalysis-method  
(calculate3DModel), [21](#)
- calculateDirectionalVarioqrams, [22](#)
- calculateIgraph, [23](#)
- calculateNetworks, [24](#)

- calculateNetworks, AFMImage-method  
(calculateNetworks), 24
- calculateNetworks, AFMImageNetworksAnalysis-method  
(calculateNetworks), 24
- calculateNetworkSkeleton, 24
- calculateOmnidirectionalVariogram, 4, 25
- calculatePhysicalDistanceFromPath, 26
- calculateShortestPaths, 26
- canBeRemoved, 27
- checkIsotropy, 4, 27
- checkNormality, 4, 28
- createGraph, 29
- directionalVariograms  
(AFMImageVariogramAnalysis-class), 16
- directionalVariograms, AFMImageVariogramAnalysis-method  
(AFMImageVariogramAnalysis-class), 16
- displaygridIgraphPlot, 30
- displaygridIgraphPlotFromEdges, 30
- displayIn3D, 31
- evaluateVariogramModels, 32
- evaluateVariogramModels, AFMImageVariogramAnalysis-method  
(evaluateVariogramModels), 32
- existsEdge, 33
- existsSegment, 33
- exportToSTL, 4, 34
- extractAFMImage, 35
- fd.estim.method, 45
- filterAFMImage, 36
- fit.variogram, 18
- fractaldim, 5, 9, 45, 46
- fractalDimensionMethods  
(AFMImageFractalDimensionsAnalysis-class), 10
- fractalDimensionMethods, AFMImageFractalDimensionsAnalysis-method  
(AFMImageFractalDimensionsAnalysis-class), 10
- fusionCloseNodes, 36
- generateAFMImageReport, 37
- generateCheckReport, 4, 37
- generateReport, 4, 38
- generateReportFromNanoscopeImageDirectory, 39
- getAngle, 40
- getAutoIntersectionForOmnidirectionalVariogram, 40
- getAutoIntersectionForRoughnessAgainstLengthscale, 41, 41
- getAutomaticWidthForVariogramCalculation, 42
- getBresenham2DSegment, 43
- getCircleSpatialPoints, 43
- getCoordinatesFromVertexId, 44
- getDTModelEvaluation, 44
- getDTModelEvaluation, AFMImageVariogramAnalysis-method  
(getDTModelEvaluation), 44
- getDTModelSillRange, 45
- getDTModelSillRange, AFMImageVariogramAnalysis-method  
(getDTModelSillRange), 45
- getFractalDimensions, 4, 45
- getIntersectionPointWithBorder, 46
- getListOfDiameters, 47
- getLogLogOmnidirectionalSlopeGraph, 47
- getNetworkGridLayout, 48
- getNyquistSpatialFrequency, 49
- getNyquistSpatialFrequency, AFMImage-method  
(getNyquistSpatialFrequency), 49
- getPaddedAFMImage, 50
- getRoughnessParameters, 50
- getRoughnessParameters, AFMImage-method  
(getRoughnessParameters), 50
- getSpplotFromAFMImage, 51
- getSurroundingVertexesList, 52
- getTopologyAFMImage, 53
- getTriangle, 53
- getVertexId, 54
- gridIgraphPlot, 54
- gstat, 5, 32
- identifyEdgesFromCircles, 55
- identifyIsolatedNodes, 55
- identifyNodesWithCircles, 56
- igraph, 11, 12
- importFromNanoscope, 4, 37, 39, 56
- initialize, AFMImage-method  
(AFMImage-class), 5
- initialize, AFMImageAnalyser-method, 57
- initialize, AFMImageFractalDimensionMethod-method  
(AFMImageFractalDimensionMethod-class), 9

- initialize, AFMImageFractalDimensionsAnalysis-method (AFMImageFractalDimensionsAnalysis-class), 10
- initialize, AFMImageNetworksAnalysis-method (AFMImageNetworksAnalysis-class), 11
- initialize, AFMImagePSDAnalysis-method (AFMImagePSDAnalysis-class), 14
- initialize, AFMImagePSDSlopesAnalysis-method (AFMImagePSDSlopesAnalysis-class), 15
- initialize, AFMImageVariogramAnalysis-method (AFMImageVariogramAnalysis-class), 16
- initialize, AFMImageVariogramModel-method (AFMImageVariogramModel-class), 18
- initialize, AFMImageVariogramSlopesAnalysis-method (AFMImageVariogramSlopesAnalysis-class), 19
- intersections (AFMImagePSDAnalysis-class), 14
- intersections, AFMImagePSDAnalysis-method (AFMImagePSDAnalysis-class), 14
- isAdjacentToBetterVertex, 58
- isAngleBetweenEdgesAlwaysSuperiorToMinAngle, 59
- krige, 18
- makeBinaryAFMImage, 59
- multiplyHeightsAFMImage, 60
- omnidirectionalVariogram (AFMImageVariogramAnalysis-class), 16
- omnidirectionalVariogram, AFMImageVariogramAnalysis-method (AFMImageVariogramAnalysis-class), 16
- performAllPSDCalculation, 61, 61
- printVariogramModelEvaluations, 62
- psd1d (AFMImagePSDAnalysis-class), 14
- psd1d, AFMImagePSDAnalysis-method (AFMImagePSDAnalysis-class), 14
- psd1d\_breaks (AFMImagePSDAnalysis-class), 14
- psd1d\_breaks, AFMImagePSDAnalysis-method (AFMImagePSDAnalysis-class), 14
- psd1dAgainstFrequency, 4, 62
- PSD1DAgainstFrequency, AFMImage-method (PSD1DAgainstFrequency), 62
- psd2d (AFMImagePSDAnalysis-class), 14
- psd2d, AFMImagePSDAnalysis-method (AFMImagePSDAnalysis-class), 14
- psd2d\_maxHighLengthScale (AFMImagePSDAnalysis-class), 14
- psd2d\_maxHighLengthScale, AFMImagePSDAnalysis-method (AFMImagePSDAnalysis-class), 14
- psd2d\_truncHighLengthScale (AFMImagePSDAnalysis-class), 14
- psd2d\_truncHighLengthScale, AFMImagePSDAnalysis-method (AFMImagePSDAnalysis-class), 14
- PSD2DAgainstFrequency, 4, 62, 64
- PSD2DAgainstFrequency, AFMImage-method (PSD2DAgainstFrequency), 64
- putAnalysisOnDisk, 65
- putAnalysisOnDisk, AFMImageAnalyser-method (putAnalysisOnDisk), 65
- putImagesFromAnalysisOnDisk, 65
- rgl, 5, 31, 34, 65
- roughnessAgainstLengthscale (AFMImagePSDAnalysis-class), 14
- roughnessAgainstLengthscale, AFMImagePSDAnalysis-method (AFMImagePSDAnalysis-class), 14
- RoughnessByLengthScale, 4, 66
- RoughnessByLengthScale, AFMImage-method (RoughnessByLengthScale), 66
- runAFMApp, 4, 67
- sampleAFMImage, 68
- saveOnDisk, 69
- saveOnDiskIntersectionForRoughnessAgainstLengthscale, 70, 70
- saveOnDiskIntersectionForRoughnessAgainstLengthscale, AFMImage-method (saveOnDiskIntersectionForRoughnessAgainstLengthscale), 70
- shiftedPSDuv, 71
- shiftFFT2D, 73
- simplifyAFMImage, 73
- SpatialPoints, 44
- spplot, 51, 70
- totalRMSRoughness, 4, 74
- transformAFMImageForNetworkAnalysis, 75
- transformAFMImageForNetworkAnalysis, AFMImage-method (transformAFMImageForNetworkAnalysis), 75

transformAFMImageForNetworkAnalysis, AFMImageNetworksAnalysis-method  
(transformAFMImageForNetworkAnalysis),  
[75](#)

updateProgress, [76](#)

updateProgress, AFMImageVariogramAnalysis-method  
(updateProgress), [76](#)

variogram, [22](#), [25](#)

variogramModels  
(AFMImageVariogramAnalysis-class),  
[16](#)

variogramModels, AFMImageVariogramAnalysis-method  
(AFMImageVariogramAnalysis-class),  
[16](#)