Package ‘FAmle’

October 12, 2022

Type Package

Title Maximum Likelihood and Bayesian Estimation of Univariate Probability Distributions

Version 1.3.7

Date 2022-03-03

Author Francois Aucoin

Maintainer Thomas Petzoldt <thomas.petzoldt@tu-dresden.de>

Depends mvtnorm

Imports graphics, stats, utils

Suggests FAdist

Description Estimate parameters of univariate probability distributions with maximum likelihood and Bayesian methods.

License GPL-2

URL https://github.com/tpetzoldt/FAmle

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-03-04 00:20:02 UTC

R topics documented:

FAmle-package 2
boot.mle 2
ColesData 3
distr 4
FAmle-internal 5
floodsNB 6
metropolis 7
mle 10
plot.metropolis 11
plot.mle 12
print.metropolis 13

print.mle 14
Q.boot.ci 15
Q.conf.int 16
station01AJ010 17
yarns 17

Index **18**

`FAmle-package` *Maximum Likelihood and Bayesian Estimation of Univariate Probability Distributions*

Description

This package contains a series of functions that might be useful in carrying out maximum likelihood and Bayesian estimations of univariate probability distributions.

Author(s)

Francois Aucoin (author and original maintainer), Thomas Petzoldt (actual maintainer, applied formal changes to pass CRANs package check). Many thanks to the original author for his agreement.

`boot.mle` *Bootstrap Distribution for Fitted Model*

Description

This function allows the user to obtain draws from the (parametric) bootstrap distribution of the fitted model's parameters.

Usage

```
boot.mle(model, B = 200, seed = NULL, start = NULL,
 method = "Nelder-Mead")
```

Arguments

`model`	`mle` object corresponding to the fitted model.
`B`	Requested number of bootstrap samples.
`seed`	A seed may be specified (see `set.seed`)
`start`	Starting values for the optimization algorithm (if `is.null(start)==TRUE`, the fitted model's parameters are used as starting values).
`method`	The optimization method to be used (see `optim` and `mle`).

Details

Parametric bootstrap – see References.

Value

`model`	`mle` object corresponding to the fitted model.
`B`	Requested number of bootstrap samples.
`seed`	The specified seed (see `set.seed`)
`par.star`	Array containing realized values from the bootstrap distribution of the maximum likelihood parameter estimators.
`gof`	The bootstrap distributions of two goodness-of-fit statistics: Anderson-Darling statistic and Pearson's correlation coefficient for the pair ("observed quantiles","fitted quantiles").
`p.value`	Bootstrap p-values for the two goodness-of-fit statistics.
`failure.rate`	The proportion of bootstrap samples for which optimization failed using the specified starting values.
`total.time`	The total amount of time required to generate `B` bootstrap samples.

References

Davison, A.C., and Hinkley, D.V. (1997). Bootstrap methods and their application. Cambridge University Press.

See Also

`mle`, `Q.conf.int`, `Q.boot.ci`

Examples

```
data(yarns)
x <- yarns$x
fit.x <- mle(x,'weibull',c(.1,.1))
boot.x <- boot.mle(fit.x,B=10)
boot.x$par.star
boot.x$p.value
```

`ColesData` *Annual Maximum Sea Levels at Port Pirie, South Australia*

Description

This dataset is taken from Coles (2001) (also see references therein), and consists of 64 sea level (in meters) yearly maxima for the time period 1923-1987.

Usage

```
data(ColesData)
```

Format

A `data.frame` containing two columns named `year` and `sea.level` (in meters).

Source

Coles (2001), page 4 (also see references therein).

References

Coles, S. (2001). An introduction to statistical modeling of extreme values. Springer.

---

distr — *Distribution functions 4-in-1*

Description

This function can be used to call any of the 4 functions specific to a given probability distribution available in R.

Usage

```
distr(x, dist, param, type = "d", model = NULL, ...)
```

Arguments

`x`	Vector (or array) of quantiles, vector (or array) of probabilities, or number of observations.
`dist`	Distribution name.
`param`	Vector (or array) of parameters.
`type`	Type of function to be called (`'d'`, `'p'`, `'q'`, or `'r'`).
`model`	Object from the class `mle` - may be specified instead of `param` and `dist`.
`...`	Additional arguments `log`, `lower.tail`, and `log.p`, depending on `type`.

Details

For each distribution available in R, 4 functions can be called. For example, for the normal distribution, the following 4 functions are available: `dnorm`, `pnorm`, `qnorm`, and `rnorm`. For the normal distribution, based on the argument `type`, `distr` may be used to call any one of the previous four functions.

Value

Returns the density, the distribution function, the quantile function, or random variates.

Note

Most functions in FAmle rely upon distr.

Examples

```
## Example 1
dnorm(-4:4,0,1,log=TRUE)
distr(-4:4,'norm',c(0,1),type='d',log=TRUE)

## Example 2
mu.vec <- c(1,100,100)
sigma.vec <- c(1,11,111)
n <- 3
set.seed(123)
rnorm(n,mu.vec,sigma.vec)
set.seed(123)
distr(n,'norm',cbind(mu.vec,sigma.vec),'r')

## Example 3
qnorm(.9,mu.vec,sigma.vec)
distr(.9,'norm',cbind(mu.vec,sigma.vec),'q')
```

FAmle-internal *Internal Functions in the FAmle Package*

Description

Internal functions in the FAmle package .

Usage

```
cdf.plot(z)
delta.Q(p, model, ln = FALSE)
delta.QQ(model, alpha = 0.1, ln = FALSE)
Diff.1(x, f, h = 1e-04)
Diff.2(k, i, model, p, ln = FALSE)
Diff.3(i, model, p, ln = FALSE)
## S3 method for class 'metropolis'
hist(x, density = TRUE, ...)
## S3 method for class 'plot'
hist(x,...)
Plot.post.pred(x, ...)
post.pred(z, fun = NULL)
Quantile.plot(z, ci = FALSE, alpha = 0.05)
Return.plot(model, ci = FALSE, alpha = 0.05)
Carlin(x)
```

Arguments

`z`	A `mle` object.
`p`	A vector of probabilities.
`model`	A `mle` object.
`ln`	Whether or not (TRUE or FALSE) computations should be carried out on the natural logarithmic scale.
`alpha`	The significance level.
`x`	Value at which the numerical derivative should be evaluated. For the `Carlin` function (see References for `metropolis`), this `x` corresponds to an object from the class `metropolis`.
`f`	A function to be differentiated.
`h`	Small number representing a small change in `x`.
`k`	Parameter value at which the first derivative should be evaluated.
`i`	Position of the parameter, within a vector of parameters, with respect to which differentiation should be carried out.
`density`	Whether or not (TRUE or FALSE) a Kernel density should be added to the histogram - see `density`.
`...`	Additional arguments pertaining to `hist`.
`fun`	optional argument that may be used to modify the scale on which the histogram will be plotted.
`ci`	Whether or not (TRUE or FALSE) approximated *100*(1-*`alpha`*)* confidence intervals should be added to the plot (either `Quantile.plot` or `Return.plot`).

`floodsNB` *New Brunswick (Canada) Flood Dataset*

Description

`floodsNB` is a `list` object containing the hydrometric stations considered for analysis... Each element from the `list` corresponds to an hydrometric station located in the Canadian province of New Brunswick, for which the flow is unregulated. For each station, the following information is available:

- `data`: Maximum annual daily mean discharge (in m^3/s);
- `peak`: Maximum annual daily peak discharge (in m^3/s);
- `ln.drain`: Natural logarithm of the drainage area (in km^2);
- `coor`: Coordinates (in latitude and longitude) of the hydrometric station;
- `status`: Station's status - `Active` or `Inactive`;
- `Aucoin.2001`: Whether or not (`TRUE` or `FALSE`) the station is retained for analysis in ....

Usage

```
data(floodsNB)
```

Format

A `list` object whose elements correspond to distinct hydrometric stations.

Source

HYDAT database.

References

Environment and Climate Change Canada Historical Hydrometric Data web site, https://wateroffice.ec.gc.ca/mainmenu/historical_data_index_e.html

---

metropolis — *Bayesian Estimation of Univariate Probability Distributions*

Description

For a given dataset, this function serves to approximate (using a Metropolis algorithm) the posterior distribution of the parameters for some specified parametric probability distribution.

Usage

```
metropolis(model, iter = 1000, tun = 2, trans.list = NULL,
start = NULL, variance = NULL, prior = NULL, burn = 0,
uniroot.interval = c(-100, 100),pass.down.to.C=FALSE)
```

Arguments

`model` — `mle` object corresponding to the fitted (by maximum likelihood) model.

A `list(x=dataset, dist=distribution)` object may also be provided, but the user will then have to make sure to specify the arguments `start` and `variance`. Moreover, the latter two arguments will have to be specified on their transformed scales (see `trans.list`).

`iter` — The requested number of iterations - the Markov Chain's length.

`tun` — A tuning constant; value by which the covariance matrix of the multivariate normal proposal will be multiplied - see References.

`trans.list` — A `list` object containing a function for each parameter that is to be estimated. For each parameter, the function must correspond to the inverse transformation that will determine the parametrization for which the simulation will be carried out (see Example and Details).

- `start` — A vector of starting values for the algorithm. If `NULL`, the maximum likelihood parameter estimates will be used as starting values for the Markov Chain. If `model` is not an object from the class `mle`, this argument will have to be specified, along with the argument `variance`. Moreover, as already stated above, the user will have to make sure that both `start` and `variance` are those for the transformed parameters (see `trans.list`).
- `variance` — Covariance matrix of the multivariate normal proposal distribution. If `NULL`, the observed Fisher's information will be used and multiplied by the specified `tun`. As for `start`, this argument needs to be specified if `model` is not from the class `mle`.
- `prior` — A function that corresponds to the joint prior distribution (see Example). Note that the prior distribution will be evaluated on the transformed parameter space(s).
- `burn` — Burn-in period (see References).
- `uniroot.interval` — Default is c(-100,100). This interval is used by `R`'s function `uniroot` to search for the inverse of each element in `trans.list`.
- `pass.down.to.C` — If `TRUE`, the iterative task is passed down to a C program for faster implementation of the MCMC algorithm.

Details

This function uses a single block Metropolis algorithm with multivariate normal proposal. For this function to work properly, all parameters should be defined on the real line - parameter transformation(s) might be required. If `trans.list` is not specified, the function will assume that the parameter distributions are all defined on the real line (i.e., `function(x) x` will be used for each parameter). If no prior distribution is provided, an improper prior distribution - uniform on the interval)-Inf,+Inf(- will be used for all parameters (i.e., prior distribution proportional to 1 - function(x) 1).

In order to minimize the number of arguments for `metropolis`, the function automatically computes the inverse of `trans.list`: this suppresses the need for the user to provide both the "inverse transformation" and the "transformation". However, problems may occur, and it is why the user is allowed to alter `uniroot.interval`. Depending on the number of errors reported, future versions of this package may end up requesting that a list for both the "inverse transformation" and the "transformation" be provided by the user.

A nice list of references is provided below for more information on topics such as: MCMC algorithms, tuning of Metropolis-Hastings algorithms, MCMC convergence diagnostics, the Bayesian paradigm ...

Value

- `rate` — MCMC acceptance rate. This value is computed before applying the burn-in; i.e., it is computed for `sims.all`.
- `total.time` — Total computation time.
- `sims.all` — Array containing all iterations.
- `sims` — Array containing iterations after burn-in.
- `input` — Inputted `mle` object.

iter	Number of iterations.
prior	Prior distribution.
burn	Integer corresponding to the number of iterations to be discarded - burn-in period.
M	Parameter vector whose elements correspond to the parameter values (on the scales specified by `trans.list`) obtained at the last iteration of the Metropolis sampler; i.e. `sims[iter,]`.
V	Covariance matrix computed using, after removing the burn-in period, the joint posterior distribution of the parameters (on the scales specified by `trans.list`). This matrix might be used to tune the MCMC algorithm.

References

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. (2004). Bayesian data analysis, 2nd edition, Chapman & Hall/CRC.

Carlin, B.P, and Louis, T.A. (2009). Bayesian methods for data analysis. Chapman & Hall/CRC.

Gamerman, D., and Lopes H.F. (2006). Markov Chain Monte Carlo: Stochastic simulation for Bayesian inference. 2nd edition, Chapman & Hall/CRC.

Gilks, W.R., Richardson, S., and Spiegelhalter, D.J. (1996). Markov Chain Monte Carlo in Practice. Chapman & Hall.

See Also

`plot.metropolis`, `mle`

Examples

```
### These examples should be re-run with, e.g., iter > 2000.
data(yarns)
x <- yarns$x
fit.x <- mle(x,'gamma',c(.1,.1))
bayes.x.no.prior <- metropolis(model=fit.x,iter=150,
trans.list=list(function(x) x,function(x) exp(x)))
plot(bayes.x.no.prior)

# examples of prior distributions (note that these prior distribution
# are specified for the transformated parameters;
# i.e., in this case, 'meanlog' -> 'meanlog' and 'sdlog' -> 'ln.sdlog')
# for the scale parameter only
prior.1 <- function(x) dnorm(x[2],.8,.1)
# for both parameters (joint but independent in this case)
prior.2 <- function(x) dunif(x[1],3.4,3.6)*dnorm(x[2],1,1)

bayes.x.prior.2 <- metropolis(model=fit.x,iter=150,
trans.list=list(function(x) x,function(x) exp(x)),prior=prior.2)
plot(bayes.x.prior.2)

# Example where 'model' is not from the class 'mle'; i.e.
```

```
# both 'start' and 'variance' need to be specified!
#x <- rweibull(5,2,1)
x <- c(0.9303492,1.0894917,0.9628029,0.6145032,0.4756699)
# Here 'fit.x <- mle(x,'weibull',c(.1,.1))' is not used,
model.x <- list(x=x,dist='weibull')
# and an informative prior distribution is considered to ensure a proper posterior distribution
prior.x <- function(x) dnorm(x[1],log(2),.1)*dnorm(x[2],log(1),.1)
trans.list.x <- list(function(x) exp(x), function(x) exp(x))
bayes.x <- metropolis(model=model.x,iter=150,prior=prior.x,trans.list=trans.list.x,
           pass.down.to.C=TRUE,start=c(0,0),variance=diag(.1,2,2))
```

`mle` *Maximum Likelihood Estimation of Univariate Probability Distributions*

Description

For a given dataset, this function serves to find maximum likelihood parameter estimates for some specified parametric probability distribution.

Usage

```
mle(x, dist, start = NULL, method = "Nelder-Mead")
```

Arguments

`x`	A univariate dataset (a vector).
`dist`	Distribution to be fitted to `x`.
`start`	Starting parameter values for the optimization algorithm (see `optim`).
`method`	The optimization method to be used (see `optim`).

Value

`fit`	`optim` output (see `optim`).
`x.info`	Array that contains the following columns: `i`: (`1:length(x)`), `x`: (original dataset), `z`: (sorted dataset), `Fx`: (CDF of `x` evaluated at the estimated parameter values), `Fz`: (sorted values of `Fx`), `Emp`: (`i/(length(x)+1)`), `zF`: (`distr(Emp,'dist',par.hat,'q')` evaluated at estimated parameter values (`par.hat`)), `fx`: (PDF of `x` evaluated at the estimated parameter values), `fz`: (PDF of `z` evaluated at the estimated parameter values)
`dist`	Distribution fitted to `x`.

`par.hat`	Vector of estiamted parameters.
`cov.hat`	Observed Fisher's information matrix.
`k`	Number of parameters
`n`	Number of observations (i.e., `length(x)`).
`log.like`	Log-likelihood value evaluated at the estimated parameter (i.e. `par.hat`).
`aic`	Akaike information criterion computed as `2*k - 2*log.like`.
`ad`	Anderson Darling statistic evaluated at the estimated parameter values.
`data.name`	Name for `x`.
`rho`	Pearson's correlation coefficient computed as `cor(x.info[,'z'],x.info[,'zF'])`.

See Also

`optim`, `distr`, `boot.mle`, `metropolis`, `Q.conf.int`

Examples

```
data(yarns)
x <- yarns$x
fit.x <- mle(x,'weibull',c(.1,.1))
fit.x
names(fit.x)
#plot(fit.x)
#plot(fit.x,TRUE,alpha=.01)
p <- c(.9,.95,.99)
distr(p,model=fit.x,type='q')
Q.conf.int(p,fit.x,.01)
Q.conf.int(p,fit.x,.01,TRUE)
```

`plot.metropolis` *A Function to Plot metropolis objects*

Description

This function allows to user to call different plots for visual assessment of the posterior distribution(s).

Usage

```
## S3 method for class 'metropolis'
plot(x, plot.type = "carlin", pos = 1:x$iter, ...)
```

Arguments

x	mle object corresponding to the fitted model.
plot.type	The user may choose betweew: carlin returns the same plot as in Carlin and Louis (2009) (see References); ts returns plot.ts; pairs returns a pairs; hist returns an hist for each marginal posterior distribution; post.pred returns an histogram of the data's posterior predictive distribution.
pos	May be used by the user to plot a subset (i.e. a random subset, sample)) of the posterior distribution when pairs is called. This avoids using too much memory while building the plot.
...	Additional arguments pertaining to function plot.default.

References

See list of references for metropolis.

See Also

metropolis

Examples

```
data(yarns)
x <- yarns$x
fit.x <- mle(x,'gamma',c(.1,.1))
bayes.x <- metropolis(model=fit.x,iter=100,
trans.list=list(function(x) exp(x),function(x) exp(x)))
plot(bayes.x)
plot(bayes.x,'hist',col='cyan')
plot(bayes.x,'pairs',cex=.1,pch=19)
plot(bayes.x,'pairs',pos=sample(1:bayes.x$iter,20),col='red')
plot(bayes.x,'post.pred',col='green')
```

---

plot.mle *Diagnostic Plots for the Fitted Model*

Description

This function returns diagnotic plots for a mle object.

Usage

```
## S3 method for class 'mle'
plot(x, ci = FALSE, alpha = 0.05,...)
```

Arguments

x	mle object corresponding to the fitted model.
ci	Whether or not approximate confidence intervals should be added to the return period and quantile plots.
alpha	1-alpha is the requested coverage probability for the confidence interval.
...	none...

See Also

mle, Q.conf.int

Examples

```
data(yarns)
x <- yarns$x
fit.1 <- mle(x,'weibull',c(.1,.1))
fit.2 <- mle(x,'logis',c(.1,.1))
plot(fit.1,TRUE,.05)
dev.new();plot(fit.2,TRUE,.05)
```

---

print.metropolis *Bayesian Estimation of Univariate Probability Distributions*

Description

See metropolis.

Usage

```
## S3 method for class 'metropolis'
print(x, stats.fun = NULL,...)
```

Arguments

x	metropolis object corresponding to the fitted model.
stats.fun	An optional function that may be provided by the user in order to obtain a posterior summary (see Example).
...	none...

See Also

metropolis, print

Examples

```
data(yarns)
x <- yarns$x
fit.x <- mle(x,'gamma',c(.1,.1))
bayes.x <- metropolis(fit.x,50,trans.list=
list(function(x) exp(x), function(x) exp(x)))
print(bayes.x)
print(bayes.x,stats.fun=function(x) c(mean=mean(x),CV=sd(x)/mean(x)))
```

`print.mle` *Maximum Likelihood Estimation of Univariate Probability Distributions*

Description

See `mle`.

Usage

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

Arguments

`x`	`mle` object corresponding to the fitted model.
`...`	none...

See Also

`mle`, `print`

Examples

```
data(yarns)
x <- yarns$x
fit.x <- mle(x,'gamma',c(.1,.1))
print(fit.x)
print.mle(fit.x)
```

Q.boot.ci *Parametric Bootstrap Confidence Intervals for p-th Quantile*

Description

This function can be used to derive parametric bootstrap confidence intervals for the p-th quantile of the fitted distribution (see `mle`).

Usage

```
Q.boot.ci(p,boot,alpha=.1)
```

Arguments

`p`	Vector of probabilities.
`boot`	An object obtained using `boot.mle`.
`alpha`	`1-alpha` is the interval's coverage probability.

Value

This functions returns two types of bootstrap confidence intervals for the p-th quantile - one is based on the "percentile" method, while the other corresponds to the basis bootstrap interval or "reflexion" (see References).

Note

See References for other means of deriving bootstrap intervals.

References

Davison, A.C., and Hinkley, D.V. (1997) Bootstrap methods and their application. Cambridge University Press.

See Also

`boot.mle`, `mle`, `Q.conf.int`

Examples

```
data(yarns)
x <- yarns$x
fit.x <- mle(x,'gamma',c(.1,.1))
Q.conf.int(p=c(.5,.9,.95,.99),model=fit.x,alpha=.01,ln=FALSE)
# should be run again with B = 1000, for example...
boot.x <- boot.mle(model=fit.x,B=50)
Q.boot.ci(p=c(.5,.9,.95,.99),boot=boot.x,alpha=.01)
```

Q.conf.int *Approximate Confidence Intervals for p-th Quantile*

Description

This function can be used to derive approximate confidence intervals for the p-th quantile of the fitted distribution (see mle).

Usage

```
Q.conf.int(p, model, alpha = 0.1, ln = FALSE)
```

Arguments

p	Vector of probabilities.
model	mle object corresponding to the fitted model.
alpha	1-alpha is the interval's coverage probability.
ln	whether or not the confidence interval of the p-th quantile should be computed on the natural logarithmic scale (see Details).

Details

The p-th quantile confidence interval is derived using the observed Fisher's information matrix in conjuction with the well-known delta method. Here, Q.conf.int allows the user to chose between two types of confidence intervals: one that is computed on the original scale and one that is computed on the quantile's natural logarithmic scale.

Value

The function returns a 3-by-length(p) array containing, for each value of p, the confidence interval's lower and upper bounds, as well as the quantile point estimate (maximum likelihood).

References

Rice, J.A. (2006) Mathematical statistics and data analysis. Duxbury Press, 3rd edition (regarding the Delta method).

See Also

plot.mle

Examples

```
data(yarns)
x <- yarns$x
fit.x <- mle(x,'gamma',c(.1,.1))
Q.conf.int(p=c(.5,.9,.95,.99),model=fit.x,alpha=.01,ln=FALSE)
Q.conf.int(p=c(.5,.9,.95,.99),model=fit.x,alpha=.01,ln=TRUE)
```

station01AJ010 *Annual Maximum Daily Mean Flow Data (NB, Canada)*

Description

This dataset is taken from the HYDAT database, and corresponds to realized values of annual maximum daily mean flows (in m^3/s).

Usage

```
data(station01AJ010)
```

Format

A vector of observations.

Source

Hydrometric station 01AJ010

References

Environment Canada: `https://wateroffice.ec.gc.ca/`

yarns *Yarns Failure Data*

Description

This dataset is taken from Gamerman and Lopes (2006) (also see references therein), and consists of 100 cycles-to-failure times for airplane yarns.

Usage

```
data(yarns)
```

Format

A `data.frame` object - one column of 100 observations.

Source

Gamerman and Lopes (2006), page 255 (also see references therein).

References

Gamerman, D., and Lopes H.F. (2006). Markov Chain Monte Carlo: Stochastic simulation for Bayesian inference. 2nd edition, Chapman & Hall/CRC.

Index

∗ **aplot**
  plot.metropolis, 11
  plot.mle, 12
∗ **datasets**
  ColesData, 3
  floodsNB, 6
  station01AJ010, 17
  yarns, 17
∗ **distribution**
  distr, 4
∗ **models**
  boot.mle, 2
  metropolis, 7
  mle, 10
  plot.metropolis, 11
  plot.mle, 12
  print.metropolis, 13
  print.mle, 14
  Q.boot.ci, 15
  Q.conf.int, 16
∗ **optimize**
  boot.mle, 2
  mle, 10
∗ **package**
  FAmle-internal, 5
  FAmle-package, 2
∗ **print**
  print.metropolis, 13
  print.mle, 14
∗ **robust**
  boot.mle, 2

boot.mle, 2, *11*, *15*

Carlin *(FAmle-internal)*, 5
cdf.plot *(FAmle-internal)*, 5
ColesData, 3

data.frame, *4*, *17*
delta.Q *(FAmle-internal)*, 5
delta.QQ *(FAmle-internal)*, 5
density, *6*
Diff.1 *(FAmle-internal)*, 5
Diff.2 *(FAmle-internal)*, 5
Diff.3 *(FAmle-internal)*, 5
distr, 4, *11*
dnorm, *4*

FAmle, *5*
FAmle *(FAmle-package)*, 2
FAmle-internal, 5
FAmle-package, 2
floodsNB, 6

hist, *6*, *12*
hist.metropolis *(FAmle-internal)*, 5
hist.plot *(FAmle-internal)*, 5

list, *7*

metropolis, *6*, 7, *11–13*
mle, *2–4*, *6–9*, 10, *12–16*

optim, *2*, *10*, *11*

pairs, *12*
plot.default, *12*
plot.metropolis, *9*, 11
plot.mle, 12, *16*
Plot.post.pred *(FAmle-internal)*, 5
plot.ts, *12*
pnorm, *4*
post.pred *(FAmle-internal)*, 5
print, *13*, *14*
print.metropolis, 13
print.mle, 14

Q.boot.ci, *3*, 15
Q.conf.int, *3*, *11*, *13*, *15*, 16
qnorm, *4*
Quantile.plot *(FAmle-internal)*, 5

Return.plot *(*FAmle-internal*)*, 5
rnorm, *4*

set.seed, *2, 3*
station01AJ010, 17

uniroot, *8*

yarns, 17