Package 'fanplot'

October 7, 2015

Type Package

Title Visualisation of Sequential Probability Distributions Using Fan Charts

Version 3.4.1

Author Guy J. Abel

Maintainer ``Guy J. Abel'' <g.j.abel@gmail.com>

Description Visualise sequential distributions using a range of plotting styles. Sequential distribution data can be input as either simulations or values corresponding to percentiles over time. Plots are added to existing graphic devices using the fan function. Users can choose from four different styles, including fan chart type plots, where a set of coloured polygon, with shadings corresponding to the percentile values are layered to represent different uncertainty levels.

License GPL-2

URL http://gjabel.wordpress.com/category/r/fanplot/

Depends R (>= 2.10)

Suggests shiny, tsbugs,

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-07 22:43:43

R topics documented:

fanplot-package *Visualisation of Sequential Probability Distributions Using Fan Charts.*

Description

Visualise sequential distributions using a range of plotting styles. Sequential distribution data can be input as either simulations or values corresponding to percentiles over time. Plots are added to existing graphic devices using the fan function. Users can choose from four different styles, including fan chart type plots, where a set of coloured polygon, with shadings corresponding to the percentile values are layered to represent different uncertainty levels.

Details

Package:	fanplot
Type:	Package
License:	GPL-2

Blog posts with some additional details of the implementation of functions in the package can be found at http://gjabel.wordpress.com/category/r/fanplot/

Github repo: http://github.com/gjabel/fanplot

R Journal article: http://journal.r-project.org/archive/2015-1/abel.pdf

Author(s)

Guy J. Abel

References

Abel, G. J. (2015). fanplot: An R Package for visualising sequential distributions. *The R Journal* 7 (2) 15–23.

boe *Parameters for MPC CPI Inflation Projections from Q1 2004 to Q4 2013.*

Description

Numerical parameters for inflation report of the Bank of England used to specify the probability distributions for forecast charts of CPI inflation. Data formatted from the November 2013 Bank of England Inflation Report.

Usage

```
data(boe)
```

Format

A data frame with 512 observations on the following 5 variables.

`time0` Publication time of parameters

`time` Future time of projected parameter

`mode` Central location parameter of split-normal distribution

`uncertainty` Uncertainty parameter of split-normal distribution

`skew` Skew parameter of split-normal distribution

Details

mode, uncertainty and skew parameters relate to those given in `dsplitnorm`, where uncertainty is the standard deviation.

Source

Bank of England Inflation Report November 2013. Retrieved from "Parameters for MPC CPI Inflation Projections from February 2004" spreadsheet at:

http://www.bankofengland.co.uk/publications/pages/inflationreport/irprobab.aspx

Examples

```
##
##Q1 2013
##
#extract data for Q1 2013
y0 <- 2013
boe0<-subset(boe, time0==y0)
k <- nrow(boe0)

#guess work to set percentiles the boe are plotting
p <- seq(0.05, 0.95, 0.05)
p <- c(0.01, p, 0.99)

#estimate percentiles for future time period
pp <- matrix(NA, nrow = length(p), ncol = k)
for (i in 1:k)
  pp[, i] <- qsplitnorm(p, mode = boe0$mode[i], sd = boe0$uncertainty[i], skew = boe0$skew[i])
pp

#plot cpi
par(mar=rep(2,4))
plot(cpi, type = "l", xlim = floor(c(y0-5, y0+3)), ylim = c(-2, 7), las = 1,
 col="tomato", lwd=2, xaxt = "n", yaxt = "n")

#backround
```

```
rect(y0-0.25, par("usr")[3] - 1, y0+3, par("usr")[4] + 1, border = "gray90", col = "gray90")

#fan
pal <- colorRampPalette(c("tomato", "gray90"))
fan(data=pp, probs=p, sim.data=FALSE, start=y0, frequency=4,
 anchor=cpi[time(cpi)==y0-0.25], fan.col=pal, ln=NULL, rlab=FALSE)

#aesthetics for boe axis
axis(2, at = -2:7, las = 2, tcl = 0.5, labels = FALSE)
axis(4, at = -2:7, las = 2, tcl = 0.5)
axis(1, at = 2008:2016, tcl = 0.5)
axis(1, at = seq(2008, 2016, 0.25), labels = FALSE, tcl = 0.2)
abline(h = 2) #cpi target
abline(v = y0 + 1.75, lty = 2) #2 year line


##
##Q4 2013 (coarser fan)
##
#extract data for Q4 2013
y0 <- 2013.75
boe0<-subset(boe, time0==y0)
k <- nrow(boe0)

#guess work at which percentiles the boe are plotting
p <- seq(0.2, 0.8, 0.2)
p <- c(0.05, p, 0.95)
pp <- matrix(NA, nrow = length(p), ncol = k)
for (i in 1:k)
  pp[, i] <- qsplitnorm(p, mode = boe0$mode[i], sd = boe0$uncertainty[i], skew = boe0$skew[i])
pp

#define prediction intervals for labels
p.int<-p[4:6]-p[3:1]
p.int

#plot cpi
par(mar=rep(2,4))
plot(cpi, type = "l", xlim = c(y0-5, y0+3), ylim = c(-2, 7), las = 1,
 col="tomato", lwd=2, xaxt = "n", yaxt = "n")

#backround
rect(y0-0.25, par("usr")[3] - 1, y0+3, par("usr")[4] + 1, border = "gray90", col = "gray90")

# add fan
pal <- colorRampPalette(c("tomato", "gray90"))
fan(data=pp, probs=p.int, sim.data=FALSE, start=y0, frequency=4,
 anchor=cpi[time(cpi)==y0-0.25], fan.col=pal, ln=NULL, rlab=pi, nfan=4, type="interval")

#aesthetics for boe axis
axis(2, at = -2:7, las = 2, tcl = 0.5, labels = FALSE)
axis(4, at = -2:7, las = 2, tcl = 0.5)
axis(1, at = 2008:2016, tcl = 0.5)
```

```
axis(1, at = seq(2008, 2016, 0.25), labels = FALSE, tcl = 0.2)
abline(h = 2) #cpi target
abline(v = y0 + 1.75, lty = 2) #2 year line
```

`cpi` *Percentage Change (over 12 months) of United Kingdom Consumer Price Inflation*

Description

Time series of quarterly UK CPI from Q1 1997 to Q3 2013. Data formatted from the October 2013 release of the CPI data by the Office of National Statistics. Q1 are taken from February values, Q2 from May, Q3 from August and Q4 from November.

Usage

```
data(cpi)
```

Format

The format is: Time-Series [1:67] from 1997 to 2014: 88.8 89.6 90 90.4 90.3 91.5 91.2 91.7 91.5 92.7 ...

Source

October 2013 CPI data by the Office of National Statistics. Retrieved from "Consumer Price Inflation Reference Tables, October 2013" spreadsheet at:

http://www.ons.gov.uk/ons/publications/re-reference-tables.html?edition=tcm

Examples

```
data(cpi)
```

`dsplitnorm` *The Split Normal Distribution (or 2 Piece Normal Distribution).*

Description

Density, distribution function, quantile function and random generation for the split normal distribution with mode equal to `mode`, uncertainty indicator equal to `sd` and inverse skewness equal to `skew`.

Usage

```
dsplitnorm(x, mode = 0, sd = 1, skew = 0, sd1 = NULL, sd2 = NULL)
psplitnorm(x, mode = 0, sd = 1, skew = 0, sd1 = NULL, sd2 = NULL)
qsplitnorm(p, mode = 0, sd = 1, skew = 0, sd1 = NULL, sd2 = NULL)
rsplitnorm(n, mode = 0, sd = 1, skew = 0, sd1 = NULL, sd2 = NULL)
```

Arguments

x	Vector of quantiles.
p	Vector of probabilities
n	Number of observations required.
mode	Vector of modes.
sd	Vector of uncertainty indicators.
skew	Vector of inverse skewnewss indicators. Must range between -1 and 1
sd1	Vector of standard deviations for left hand side. NULL by default.
sd2	Vector of standard deviations for right hand side. NULL by default.

Details

If mode, sd or skew are not specified they assume the default values of 0, 1 and 1, respectively. This results in identical values as a those obtained from a normal distribution.

The probability density function is:

$$f(x;\mu,\sigma_1,\sigma_2) = \frac{\sqrt{2}}{\sqrt{\pi}(\sigma_1+\sigma_2)} e^{-\frac{1}{2\sigma_1^2}(x-\mu)^2}$$

for -Inf$< x < \mu$, and

$$f(x;\mu,\sigma_1,\sigma_2) = \frac{\sqrt{2}}{\sqrt{\pi}(\sigma_1+\sigma_2)} e^{-\frac{1}{2\sigma_2^2}(x-\mu)^2}$$

for $\mu < x <$Inf, where, if not specified (in sd1 and sd2) σ_1 and σ_2 are derived as

$$\sigma_1 = \sigma/\sqrt{(1-\gamma)}$$

$$\sigma_2 = \sigma/\sqrt{(1+\gamma)}$$

from σ_1 is the overall uncertainty indicator sd and γ is the inverse skewness indicator skew.

Value

dsplitnorm gives the density, psplitnorm gives the distribution function, qsplitnorm gives the quantile function, and rsplitnorm generates random deviates.

The length of the result is determined by n for rsplitnorm, and is the maximum of the lengths of the numerical parameters for the other functions.

The numerical parameters other than n are recycled to the length of the result.

Note

Tested against the fan chart package in MATLAB (http://www.mathworks.de/matlabcentral/fileexchange/27702-fan-chart). Obtained the same results for a set of simple comparisons.

Author(s)

Guy J. Abel

References

Source for all functions based on:

Julio, J. M. (2007). The Fan Chart: The Technical Details Of The New Implementation. Bogota, Colombia. Retrieved from http://www.banrep.gov.co/docum/ftp/borra468.pdf

Examples

```
x<-seq(-5,5,length=110)
plot(x,dsplitnorm(x),type="l")

#compare to normal density
lines(x,dnorm(x), lty=2, col="red", lwd=5)

#add positive skew
lines(x,dsplitnorm(x, mode=0, sd=1, skew=0.8))

#add negative skew
lines(x,dsplitnorm(x, mode=0, sd=1, skew=-0.5))

#add left and right hand sd
lines(x,dsplitnorm(x, mode=0, sd1=1, sd2=2), col="blue")

#psplitnorm
x<-seq(-5,5,length=100)
plot(x,pnorm(x),type="l")
lines(x, psplitnorm(x, skew=-0.9), col="red")

#qsplitnorm
x<-seq(0,1,length=100)
plot(qnorm(x),type="l",x)
lines(qsplitnorm(x), x, lty=2, col="blue")
lines(qsplitnorm(x, skew=-0.3), x, col="red")

#rsplitnorm
hist(rsplitnorm(n=10000, mode=1, sd=1, skew=0.9),100)
```

---

fan *Fan Plot of Distributions Percentiles Over Time.*

Description

Visualise sequential distributions using a range of plotting styles.

Usage

```
fan(data = NULL, data.type="simulations", style = "fan", type = "percentile",
  probs = if(type=="percentile") seq(0.01, 0.99, 0.01) else c(0.5, 0.8, 0.95),
  start = 1, frequency = 1, anchor = NULL, anchor.time=NULL,
  fan.col = heat.colors, alpha = if (style == "spaghetti") 0.5 else 1,
```

```
  n.fan = NULL,
  ln = if(length(probs)<10) probs else
    probs[round(probs,2) %in% round(seq(0.1, 0.9, 0.1),2)],
  ln.col = if(style=="spaghetti") "gray" else NULL,
  med.ln = if(type=="interval") TRUE else FALSE,
  med.col= "orange",
  rlab = ln, rpos = 4, roffset = 0.1, rcex = 0.8, rcol = NULL,
  llab = FALSE, lpos = 2, loffset = roffset, lcex = rcex, lcol = rcol,
  upplab = "U", lowlab = "L", medlab=if(type == "interval") "M" else NULL,
  n.spag = 30,
  space = if(style=="boxplot") 1/frequency else 0.9/frequency,
  add = FALSE, ylim = range(data)*0.8, ...)
fan0(data = NULL, data.type = "simulations", style = "fan", type = "percentile",
  probs = if(type=="percentile") seq(0.01, 0.99, 0.01) else c(0.5, 0.8, 0.95),
  start = 1, frequency = 1, anchor = NULL, anchor.time=NULL,
  fan.col = heat.colors, alpha = if (style == "spaghetti") 0.5 else 1,
  n.fan = NULL,
  ln = NULL,
  ln.col = if(style=="spaghetti") "gray" else NULL,
  med.ln = if(type=="interval") TRUE else FALSE,
  med.col= "orange",
  rlab = ln, rpos = 4, roffset = 0.1, rcex = 0.8, rcol = NULL,
  llab = FALSE, lpos = 2, loffset = roffset, lcex = rcex, lcol = rcol,
  upplab = "U", lowlab = "L", medlab=if(type == "interval") "M" else NULL,
  n.spag = 30,
  space = if(style=="boxplot") 1/frequency else 0.9/frequency,
  add = TRUE, ylim = range(data)*0.8, ...)
```

Arguments

data	Set of sequential simulation data, where rows represent simulation number and columns represent some form of time index. If `data.type = "values"`, data must instead be a set of quantile values by rows for a set of probabilities (which need to be provided in `probs` and by column for some form of time index. Data can take multiple classes, where the contents are converted to a `matrix`. If the input is a `mts` or `zoo`, the time series properties will be inherted (and `start` and `frequency` arguments will be ignored).
data.type	Indicates if `data` are sets of pre-calculated values based for defined probabilities `data.type = "values"` or simulated data `data.type = "simulations"`. That later is the default.
style	Plot style, choose from `fan` (default), `spaghetti` `boxplot` or `boxfan`. See Examples and Details Sections for further explanation.
type	Type of percentiles to plot in `fan` or `boxfan`. Choose from, `percentile` (default) or `interval`.
probs	Probabilities related to percentiles or prediction intervals to be plotted (dependent on the `type` function. These values control the number of shades used in the `fan` or `boxfan`. These must be between 0 and 100 (inclusive) or 0 and 1.

	Percentiles greater than 50 (or 0.5), if not given, are automatically calculated as 100-p, to ensure symmetric fan. Values can be non-integers. Default to single percentile values when `type="percetinle"` and the 50th, 80th and 95th prediction interval when `type="interval"` is set.
`start`	The time of the first distribution in `sims`. Similar to use in `ts`.
`frequency`	The number of distribution in `sims` per unit of time in `sims`. Similar to use in `ts`.
`anchor`	Optional data value to anchor a forecast fan on. Typically this will be the last observation of the observed data series.
`anchor.time`	Optional data value for the time of the anchor. Useful for irregular time series.
`fan.col`	Palette of colours used in the `fan` or `boxfan`.
`n.fan`	The number of colours to use in the fan.
`alpha`	Factor modifying the opacity alpha; typically in [0,1].
`ln`	Vector of number to plot contour lines on-top `fan` or `boxfan`. Must correspond to calculated percentiles in `probs`. By default for `fan` takes either every percentile or prediction interval given in the `probs` argument (if less then 10) or, when there are more than 10 `probs` passed, the either every decile or every prediction interval that are a multiple of 10 and given in `probs`. No lines plotted by default for `fan0`
`med.ln`	Add a median line to fan. Might be of particular use if `type="interval"`. Only works when `data.type = "simulations"` and one of `fan`, `boxfan` or `spaghetti` styles.
`ln.col`	Line colour to be imposed on top of the fan. By default takes the darkest colour from `fan.col` argument, unless `style="spaghetti"`
`med.col`	Median Line colour. By default this is set to the first colour in `fan.col`. Users might wish to change to highlight the median.
`rlab`	Vector of labels at the end (right) of corresponding percentiles or prediction intervals of the `fan` or `boxfan`. Must be in calculated in the `probs` argument. By default plotted alongside values provided to the `ln` argument.
`rpos`	Position of right labels for the `fan` or `boxfan`. See `text`.
`roffset`	Offset of right labels for the `fan` or `boxfan`. See `text`.
`rcex`	Text size of right labels for the `fan` or `boxfan`. See `text`.
`rcol`	Colour of text for right labels for the `fan` or `boxfan`. See `text`.
`llab`	Can take either 1) a `TRUE` or `FALSE` value to plot label at the start (right) of the corresponding percentiles or prediction intervals given in `rlab`, default is `FALSE` or 2) a original vector of percentiles or prediction intervals. Must be in calculated in the `probs` argument. Only works for `fan` or `boxfan` styles.
`lpos`	Position of left labels for the `fan` or `boxfan`. See `text`.
`loffset`	Offset of left labels for the `fan` or `boxfan`.By default takes the same value as `roffset`. See `text`.
`lcex`	Text size of left labels for the `fan` or `boxfan`. By default takes the same value as `rex`. See `text`.

lcol	Colour of text for left labels for the fan or boxfan. By default takes the same value as rcol. See text.
upplab	Prefix character string for upper labels to be used for the fan or boxfan when type="interval".
lowlab	Prefix character string for lower labels to be used for the fan or boxfan when type="interval" .
medlab	Character string for median label.
n.spag	Number of simulations to plot in the spaghetti plot.
space	Space between boxes in the boxfan plot.
add	Add to active plot. By default FALSE for fan and TRUE for fan0.
ylim	Passed to plot when add = TRUE.
...	Additional arguments passed to boxplot for fan and to plot for fan0.

Details

Visualise sequential distributions using a range of plotting styles. Sequential distribution data can be input as either simulations or pre-computed values over time (columns). For the later, the user should declare input data as percentiles by setting data.type = "values". Plots are added to existing graphic devices. Users can choose from four different styles.

The fan and boxfan style plot distributions based on used-defined shading scheme, controlled by the fan.col argument. Additional lines and text are added to illustrate major contours on the probability distribution. Lines and labels can be suppressed by adding ln = NULL and rlab = NULL. Labels to the left of the fan can also be specified using the llab argument. Colours are by default taken from the heat.colors palette. Alternatives can be specified using fan.col (see the example below). The joining of a forecast fan to data is controlled be the anchor argument.

The spaghetti style, plots random draws (when data.type = "simulations" is set) along the sequence of distributions. The number of draws is controlled by the n.spag argument. The transparency of the lines is controlled by alpha.

The boxplot style, adds a box plot for simulated data at the appropriate location, according to the start and frquency arguments. Gaps between box plots are controlled by space argument. Additional arguments are passed to boxplot.

Value

See details

Author(s)

Guy J. Abel

References

Abel, G. J. (2015). fanplot: An R Package for visualising sequential distributions. *The R Journal* 7 (2) 15–23.

Examples

```
##
## Basic Fan: fan0()
##
fan0(th.mcmc)

##
## Basic Fan: fan()
##
# empty plot
plot(NULL, xlim = c(1, 945), ylim = range(th.mcmc)*0.85)

# add fan
fan(th.mcmc)

##
## 20 or so examples of fan charts and
## spaghetti plots based on the th.mcmc object
##
## Make sure you have zoo, tsbugs, RColorBrewer and
## colorspace packages installed
##
## Not run:
demo("sv_fan", "fanplot")

## End(Not run)

##
## Fans for forecasted values
##
## Not run:
#create time series
net <- ts(ips$net, start=1975)

# fit model
library("forecast")
m <- auto.arima(net)

# plot in forecast package (limited customisation possible)
plot(forecast(m, h=5))

# another plot in forecast (with some customisation, no
# labels or anchoring possible at the moment)
plot(forecast(m, h=5, level=c(50,80,95)),
     shadecols=rev(heat.colors(3)))

# simulate future values
mm <- matrix(NA, nrow=1000, ncol=5)
for(i in 1:1000)
  mm[i,] <- simulate(m, nsim=5)

# interval fan chart
```

```
plot(net, xlim=c(1975,2020), ylim=c(-100,300))
fan(mm, type="interval", start=2013)

# anchor fan chart
plot(net, xlim=c(1975,2020), ylim=c(-100,300))
fan(mm, type="interval", start=2013,
    anchor=net[time(net)==2012])

# anchor spaghetti plot with underlying fan chart
plot(net, xlim=c(1975,2020), ylim=c(-100,300))
fan(mm, type="interval", start=2013,
    anchor=net[time(net)==2012], alpha=0, ln.col="orange")
fan(mm, type="interval", start=2013,
    anchor=net[time(net)==2012], alpha=0.5, style="spaghetti")

## End(Not run)

##
## Box Plots
##
# sample every 21st day of theta_t
th.mcmc21 <- th.mcmc[, seq(1, 945, 21)]
plot(NULL, xlim = c(1, 945), ylim = range(th.mcmc21))
fan(th.mcmc21, style = "boxplot", frequency = 1/21)

# additional arguments for boxplot
plot(NULL, xlim = c(1, 945), ylim = range(th.mcmc21))
fan(th.mcmc21, style = "boxplot", frequency = 1/21,
 outline = FALSE, col = "red", notch = TRUE)

##
## Fan Boxes
##
plot(NULL, xlim = c(1, 945), ylim = range(th.mcmc21))
fan(th.mcmc21, style = "boxfan", type = "interval", frequency = 1/21)

# more space between boxes
plot(NULL, xlim = c(1, 945), ylim = range(th.mcmc21))
fan(th.mcmc21, style = "boxfan", type = "interval",
 frequency = 1/21, space = 10)

# overlay spaghetti
fan(th.mcmc21, style = "spaghetti",
 frequency = 1/21, n.spag = 50, ln.col = "red", alpha=0.2)
```

`ips` *ONS International Passenger Survey Long-Term International Migration 1975-2012*

Description

Immigration, emigration and net migration flow counts (and their confidence intervals) for the UK from the International Passenger Survey (IPS) conducted by the Office of National Statistics. Data formatted from the 2012 release of the Long-Term International Migration Statistics.

Usage

```
data(ips)
```

Format

A data frame with 38 observations on the following 7 variables.

`year` a numeric vector

`imm` a numeric vector

`imm.ci` a numeric vector

`emi` a numeric vector

`emi.ci` a numeric vector

`net` a numeric vector

`net.ci` a numeric vector

Details

Data differ slightly from the final adjusted migration estimates published by the ONS, that take account of certain types of migration that the IPS doesn't pick up, such as asylum seekers, people migrating for longer or shorter than they thought they would, and migration over land to and from Northern Ireland.

Source

Annual statistics on flows of international migrants to and from the UK and England and Wales by the Office of National Statistics. Retrieved from "1.02 IPS Margins of Error, 1975-2012" spreadsheet at:

http://www.ons.gov.uk/ons/publications/re-reference-tables.html?edition=tcm

Examples

```
#standard plot
net<-ts(ips$net, start=1975)
plot(net, ylim=range(net-ips$net.ci, net+ips$net.ci))
lines(net+ips$net.ci, lty=2, col="red")
lines(net-ips$net.ci, lty=2, col="red")

#simulate values
ips.sim <- matrix(NA, nrow = 10000, ncol=length(net))
for (i in 1:length(net))
  ips.sim[, i] <- rnorm(10000, mean = ips$net[i], sd =ips$net.ci[i]/1.96)
```

```
#spaghetti plot
plot(net, ylim=range(net-ips$net.ci, net+ips$net.ci), type = "n")
fan(ips.sim, style="spaghetti", start=tsp(net)[1], n.spag=50)

#box plot
plot(net, ylim=range(net-ips$net.ci, net+ips$net.ci), type = "n")
fan(ips.sim, style="boxplot", start=tsp(net)[1], llab=TRUE, outline=FALSE)

#box fan
plot(net, ylim=range(net-ips$net.ci, net+ips$net.ci), type = "n")
fan(ips.sim, style="boxfan", type="interval", start=tsp(net)[1])
```

---

th.mcmc *1000 MCMC Simulations of Estimated Volatility from Pound Dollar Exchange Rate Data.*

Description

MCMC simulations of volatility obtained from the bugs function in the R2OpenBUGS package. Estimates based on the stochastic volatility model for Pound-Dollar exchange rate data presented in the appendix of Meyer and Yu (2002). The MCMC was ran for 1100 simulations, thining to keep every 10th iteration, and treating the first 100 simulations as burn in.

Larger simulations (without thinning) can be obtained using the data (svpdx) and the my1.txt BUGS file contained in this package, see example below.

Details

See Meyer and Yu (2010) for model specification.

References

Meyer, R. and J. Yu (2002). BUGS for a Bayesian analysis of stochastic volatility models. *Econometrics Journal* 3 (2), 198–215.

Sturtz, S., U. Ligges, and A. Gelman (2005). R2WinBUGS: a package for running WinBUGS from R. *Journal of Statistical Software* 12 (3), 1–16.

Examples

```
## Not run:
# empty plot
plot(NULL, type = "n", xlim = c(1, 945), ylim = range(th.mcmc), ylab = "Theta")

# add fan
fan(th.mcmc)

##
##Create your own (longer) MCMC sample:
##
```

```
library("tsbugs")
library("R2OpenBUGS")
# write model file:
my1.bug <- dget(system.file("model", "my1.R", package = "fanplot"))
write.model(my1.bug, "my1.txt")
# take a look:
file.show("my1.txt")
# run openbugs, remember to include theta as a param otherwise will not
# have anything to plot
my1.mcmc<-bugs(data=list(n=length(svpdx$pdx),y=svpdx$pdx),
               inits=list(list(phistar=0.975,mu=0,itau2=50)),
               param=c("mu","phi","tau","theta"),
               model="my1.txt",
               n.iter=11000, n.burnin=1000, n.chains=1)

th.mcmc <- my1.mcmc$sims.list$theta

## End(Not run)
```

Index