Vignette for fanplot package

Guy J. Abel
Wittgenstein Centre (IIASA, VID/ÖAW, WU),
Vienna Institute of Demography/Austrian Academy of Sciences
g.j.abel@gmail.com

February 7, 2013

1 Introduction

The fanplot package contains a collection of R (R Development Core Team, 2012) functions to effectively display plots of sequential distributions such as probabilistic forecasts. The plotting of distributions are based around two functions. The first, `pn`, calculates the percentiles for a set of sequential distributions over a specified time period. The second, `fan`, plots the calculated percentiles of the sequential distributions. The resulting plot is a set of coloured polygon, with shadings corresponding to the percentile values.

This document illustrates these two core functions using MCMC simulation results from fitted stochastic volatility models. These MCMC simulations can be recreated from data and BUGS model also contained in the fanplot package, via the R2OpenBUGS package of (Sturtz et al., 2005). These are first shown for `dataframe` type object where there is no time series type attributes, followed by plots based on a `ts` object.

2 Volatility Plots

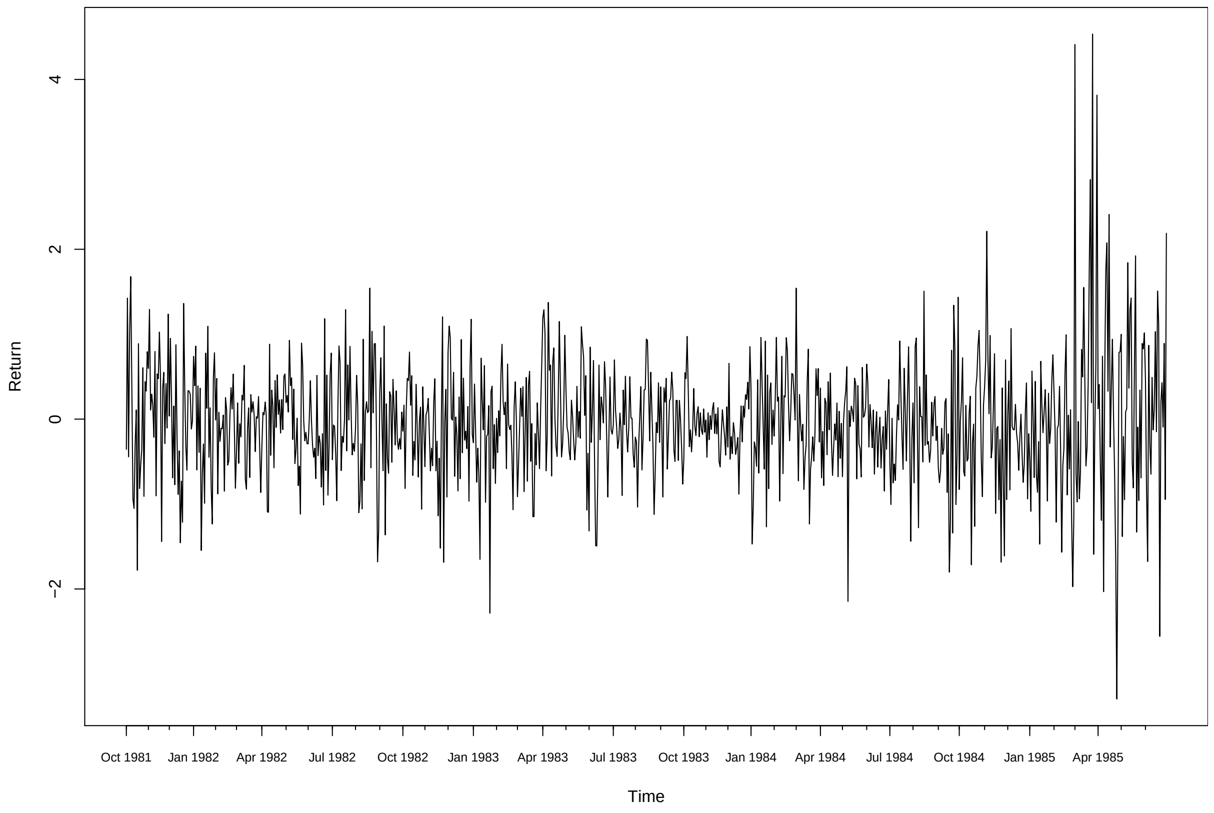
To illustrate the basics of the fanplot package consider the `svpdx` data contained in the tsbugs (Abel, 2013) package. This contains information on the log return of the Pound-Dollar exchange rate from 2nd October 1981 to 28th June 1985. For a view of the first part of the data, use the `head` function, after loading the fanplot package.

```
> library("tsbugs")
> head(svpdx)
```

```
        date        pdx
1 1981-10-02 -0.3555316
2 1981-10-05  1.4254090
3 1981-10-06 -0.4439399
4 1981-10-07  1.0256500
5 1981-10-08  1.6775790
6 1981-10-09  0.3690041
```

This data is a `dataframe` object, and is difficult to express it as a standard `ts` object due to the irregular nature of the data. It is best plot initially without an x-axis, which can then be added later using the `axis` function:

```
> #plot
> plot(svpdx$pdx, type = "l", xaxt = "n", xlab = "Time", ylab = "Return")
> #x-axis
> svpdx$rdate<-format(svpdx$date, format = "%b %Y")
> mth <- unique(svpdx$rdate)
> qtr <- mth[seq(1,length(mth),3)]
> axis(1, at = match(qtr, svpdx$rdate), labels = qtr, cex.axis = 0.75)
> axis(1, at = match(mth, svpdx$rdate), labels = FALSE, tcl = -0.2)
```



If the xaxt argument was not used in the plot function and the labels not added later, the x-axis would be based on the row number of each observation in the svpdx data, i.e. an index sequence from 1 to 945.

To produce the x-axis with date information in the above code, a new column is added to svpdx for the month-year combination of each observation. Objects mth and qtr are then created to mark each month and quarter in the data series respectively. Major axis ticks are then plotted on the unseen 1 to 945 index for the beginning of every quarter with a corresponding label, whilst minor axis tick are also plotted for beginning of every month.

Meyer and Yu (2002) used the above Pound-Dollar exchange rate data to fit various stochastic volatility models in WinBUGS (Lunn et al., 2000). One such stochastic model they fitted to the data is contained in my1.txt in the model directory of the fanplot package. The model of Meyer and Yu (2002) can be refitted in BUGS via R, using the R2OpenBUGS package (Sturtz et al., 2005):

```
> library("R2OpenBUGS")
> # write model file:
> my1.bug <- dget(system.file("model", "my1.txt", package = "fanplot"))
> write.model(my1.bug, "my1.txt")
> # take a look:
> file.show("my1.txt")
> # run openbugs
> my1<-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)
```

Here, the same initial parameter values as Meyer and Yu (2002) are set. One chain of the MCMC simulation is run for 11000 iterations, with the first 1000 used for burn in. The resulting `bugs` object contains the MCMC simulation results for the parameters in the stochastic volatility model, including the time dependent volatility parameters (θ_t). This set of sequential posterior distributions are of interest when studying the variation in the data over time.

The fanplot package can effectively display the entire posterior distribution of θ_t. The separate MCMC simulation of the volatility be obtained from `my1` using[1],

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

A plot of the entire posterior distribution of θ_t first requires a calculation of the percentiles over all t using the `pn` function,

```
> library("fanplot")
> th.pn <- pn(sims = th.mcmc)
```

This produces a `pn` type object, where rows represent the time index and columns the percentiles calculated.

```
> head(th.pn[,c(1:3, 97:99)])
```

```
           1%        2%      3%     97%        98%       99%
[1,] -1.05400 -0.999006 -0.9811 -0.2069 -0.1615000 -0.091476
[2,] -1.00907 -0.947200 -0.9127 -0.2230 -0.1829000 -0.116800
[3,] -1.01600 -0.945000 -0.9030 -0.1869 -0.1776000 -0.151200
[4,] -1.02800 -0.932900 -0.8911 -0.1714 -0.1466000 -0.082590
[5,] -1.00600 -0.917500 -0.8740 -0.1367 -0.0973464 -0.047270
[6,] -1.05101 -0.935700 -0.8998 -0.1297 -0.0543700  0.029130
```

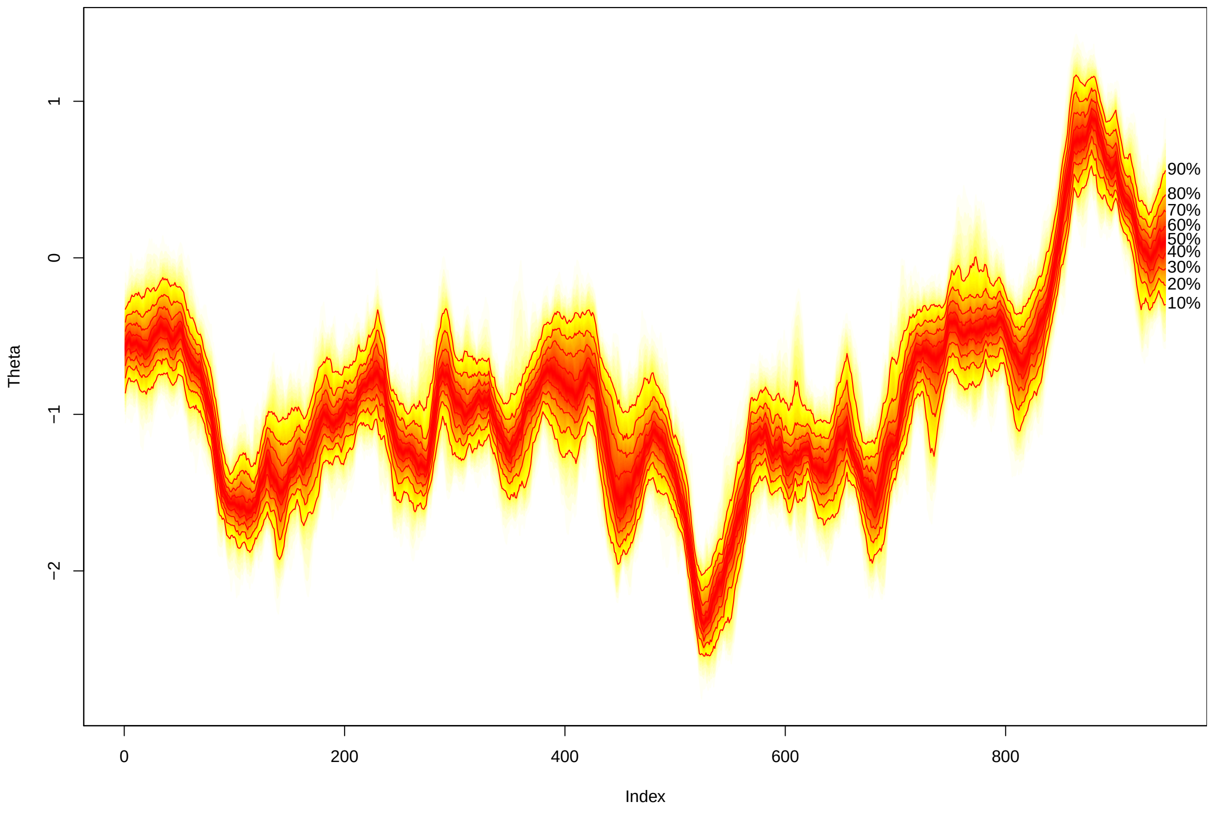
Every percentile between 1st and 99th are calculated by default, however, more or less percentiles can be calculated via the `p` argument in the `pn` function. The number of percentiles calculated has a direct impact on the plotting of the sequential distributions, as we shall see later. Additional arguments to control the indexing of the rows, which is of use when the time series are from regular intervals, will also be discussed later.

In order to plot the `th.pn` percentile object the plot area must be first set. This can be done using `type = "n"` argument in `plot`. Both the `xlim` and `ylim` arguments need to be set appropriately. In the code below, the `xlim` argument is set between 1 and 945, the length of `th.pn`. The `ylim` argument is set to the range of `th.pn` to enable all percentiles calculated to be included in the plot area.

[1]The fanplot package contains a shorter MCMC simulation (due to overall package size constraints) of θ_t, which is overwritten when creating a new `th.mcmc` object

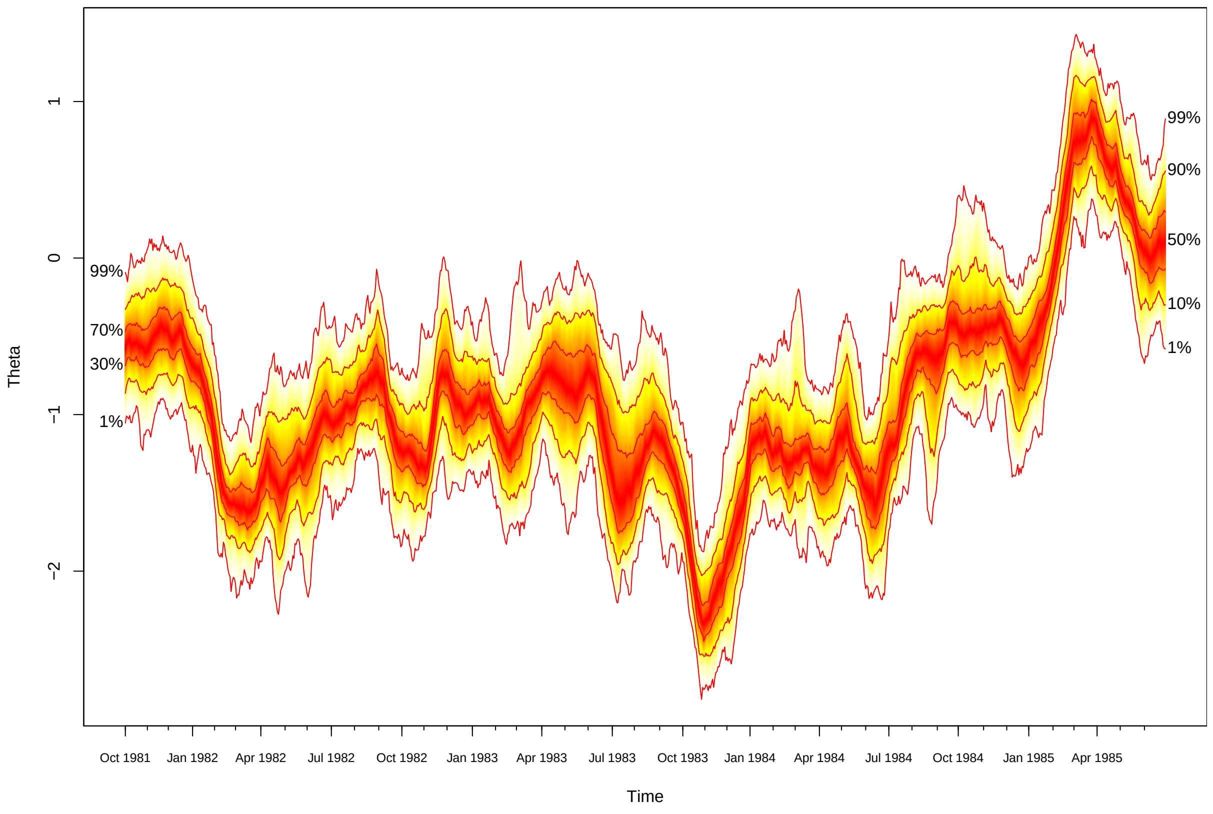
Once the plot area is set up, the `fan` function can be used to add the `th.pn` object. Each percentile in the plot is represented by a different shade of the default colour scale in the `fan.col` argument of fan. In addition, contour lines are drawn on every decile, with labels for these decides add to the right hand side.

```
> #empty plot
> plot(NULL, type = "n", xlim = c(1, 945), ylim = range(th.pn), ylab = "Theta")
> #add fan
> fan(th.pn)
```



The `fan.txt` function can be add more labels to a set of sequential distributions plotted using `fan`. When used in addition to the `fan` function, labels for closely spaced deciles can be controlled to allow a more spacious display of labels in comparison to those shown in the above plot. It can also be used to add text labels to percentiles that are not of a unit of 10, such as the 1st or 99th. The code below demonstrates these features. First a empty plot area is created with the x-axis changed to dates using the same method as in the plot of the original data. Second, the percentiles of the sequential distribution is plotted for θ_t with no text labels (setting `txt = NA`) and contour lines for the 1st and 99th percentiles alongside some selected deciles. The `ln` argument in the `fan` function is set to include contour lines at percentiles where future text labels are to be added.

```
> #empty plot with x-axis added later
> plot(NULL, type = "l", xlim = c(1, 945), xlab = "Time", xaxt = "n",
       ylim = range(th.pn), ylab = "Theta")
> axis(1, at = match(qtr, svpdx$rdate), labels = qtr, cex.axis = 0.75)
> axis(1, at = match(mth, svpdx$rdate), labels = FALSE, tcl = -0.2)
> #add fan
> fan(th.pn, txt = NA, ln = c(1,10,30,50,70,90,99))
> #add text labels for percentiles
> fan.txt(th.pn, pn.r = c(1,10,50,90,99))
> fan.txt(th.pn, pn.l = c(1,30,70,99))
```



The colour of the percentiles in the sequential distributions can be easily altered from the default `heat.colors` scheme. A new set of graded colours can be passed to the `fan` function using the `fan.col` argument. The number of colours should be half the number of percentiles (columns) in the `pn` object. New graded colour schemes can be constructed in a number of ways. For example, using the `colorRampPalette` a new shading from blue to white, via grey can be created.

```
> pal <- colorRampPalette(c("royalblue", "grey", "white"))
```

Using this palette, 50 colours (approximately half the number of percentiles calculated in `pn`) can be defined:

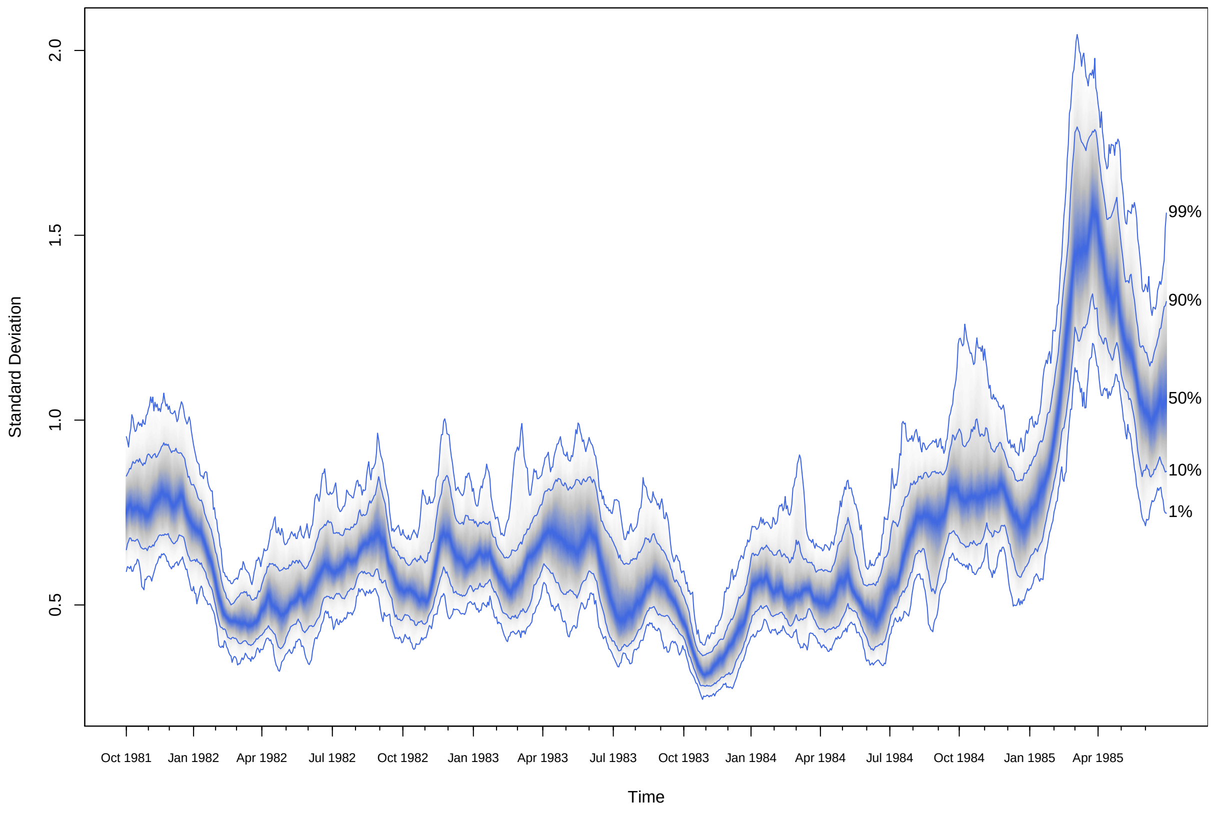
```
> fancol <- pal(50)
```

For a change, this new colour scheme is used to plot the posterior distribution of the standard deviation over time, σ_t, which is derived as such:

```
> sigma.pn <- pn(sims = sqrt(exp(th.mcmc)))
```

The new colour scheme can then be passed to the fan function for `sigma.pn`, with contour lines on selected percentiles:

```
> #empty plot with x-axis added later
> plot(NULL, type = "l", xlim = c(1, 945), xlab = "Time", xaxt = "n",
       ylim = range(sigma.pn), ylab = "Standard Deviation")
> axis(1, at = match(qtr, svpdx$rdate), labels = qtr, cex.axis = 0.75)
> axis(1, at = match(mth, svpdx$rdate), labels = FALSE, tcl = -0.2)
> #add fan
> fan(sigma.pn, fan.col = fancol, ln = c(1, 10, 50, 90, 99))
```



3 Model Fits

To illustrate plots in the fanplot package that use time series objects (`ts`) based on regularly spaced data, a stochastic volatility model is fitted to the change of population growth rate of England and Wales, similar to that in Abel et al. (2010).

Population data from 1841 to 2007 from Human Mortality Database (2012) are included in the fanplot package (`ew`). The growth rate, that the stochastic volatility model is based on can be derived following Rogers (1995) as such

```
> r <- diff(log(ew))
```

Mean stationarity can be obtained by differencing the series

```
> y <- diff(r)
```

Using this differenced growth rate time series, we can build a BUGS stochastic volatility model using the tsbugs package Abel (2013).

```
> pop.bug <- sv.bugs(y, h=25, sim=TRUE,
                     sv.mean.prior2 = "dgamma(0.000001,0.000001)",
                     sv.ar.prior2 = "dunif(-0.999, 0.999)")
```

The `sv.bugs` function speficies for 25 future values to be forecast and simulations of the model to be taken. Specifications for alternative prior distirbutions for the parameters in the volaritliy process are also stated. This BUGS model can be run using R2OpenBUGS,

```
> library("R2OpenBUGS")
> # write model file:
> writeLines(pop.bug$bug, "pop.txt")
> # take a look:
> file.show("pop.txt")
> # run openbugs
> pop <- bugs(data = pop.bug$data,
              inits = list(list(psi0.star=exp(12), psi1=0.5, itau2=0.5)),
              param = c("psi0", "psi1", "tau", "y.new", "y.sim"),
              model = "pop.txt",
              n.iter = 11000, n.burnin = 1000, n.chains = 1)
```

As was the case with the exchange rate data, one chain of the MCMC simulation is run for 11000 iterations, with the first 1000 used for burn in. The resulting `bugs` object contains the MCMC simulation results for the parameters in the stochastic volatility model, including the time dependent model fits ($E(y_t)$), forecasts of the future population growth rate ($\hat{r}_{t+h|t}$) and population ($\hat{p}_{t+h|t}$).

The fanplot package can efficiently display these sequential distributions in a number of ways. Consider the MCMC simulation of the model fit, which can be extracted from the `pop` using,

```
> y.mcmc <- pop$sims.list$y.sim
```

A plot of the entire posterior distribution of the model fits requires the calculation of percentiles over all t using the `pn` function.

```
> y0 <- start(y)[1]
> y.pn <- pn(sims = y.mcmc, start = y0)
```

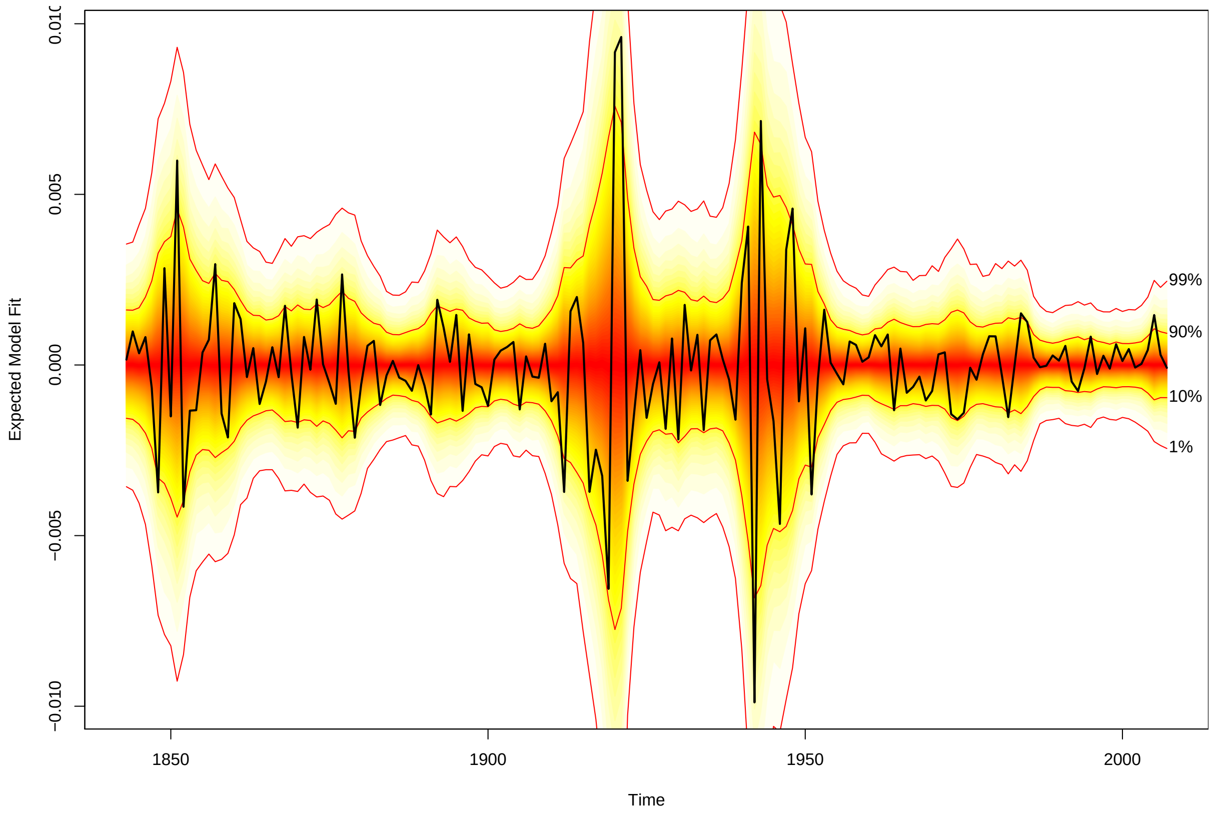
Here, the corresponding start date is also given so that the `pn` object takes the relevant time series properties when plotted. This saves on the previous effort for the non-regularly spaced exchange rate data, in setting up date labels on the x-axis. The time series properties are stored in the `tsp` attributes of the new `y.pn` object.

```
> str(y.pn)

 pn [1:165, 1:99] -0.00357 -0.00367 -0.00405 -0.00467 -0.00588 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:99] "1%" "2%" "3%" "4%" ...
 - attr(*, "tsp")= num [1:3] 1843 2007 1
```

These attributes can be directly used to set up the x-axis limits in the plot area, alongside y-axis limits based on the difference in the growth rate, which was the basis for the stochastic variance model. The `fan` function can then be used to add the sequential posterior distributions with contour lines at the 1st, 10th, 90th and 99th percentiles using the `ln` argument. Note, the `fan` function will automatically only draw and label contour lines given in the `ln` argument. If the user wishes to subdue this and add text labels later, they can do so by adding the `txt = NA` as was demonstrated earlier. The original data can also be plotted on top of the posterior distributions using the `lines` function:

```
> #empty plot
> plot(NULL, type = "l", xlim = range(time(y.pn)), xlab = "Time",
      ylim = range(y), ylab = "Expected Model Fit")
> #add fan
> fan(y.pn, ln = c(1, 10, 90, 99))
> #add data
> lines(diff(r), lwd = 2)
```



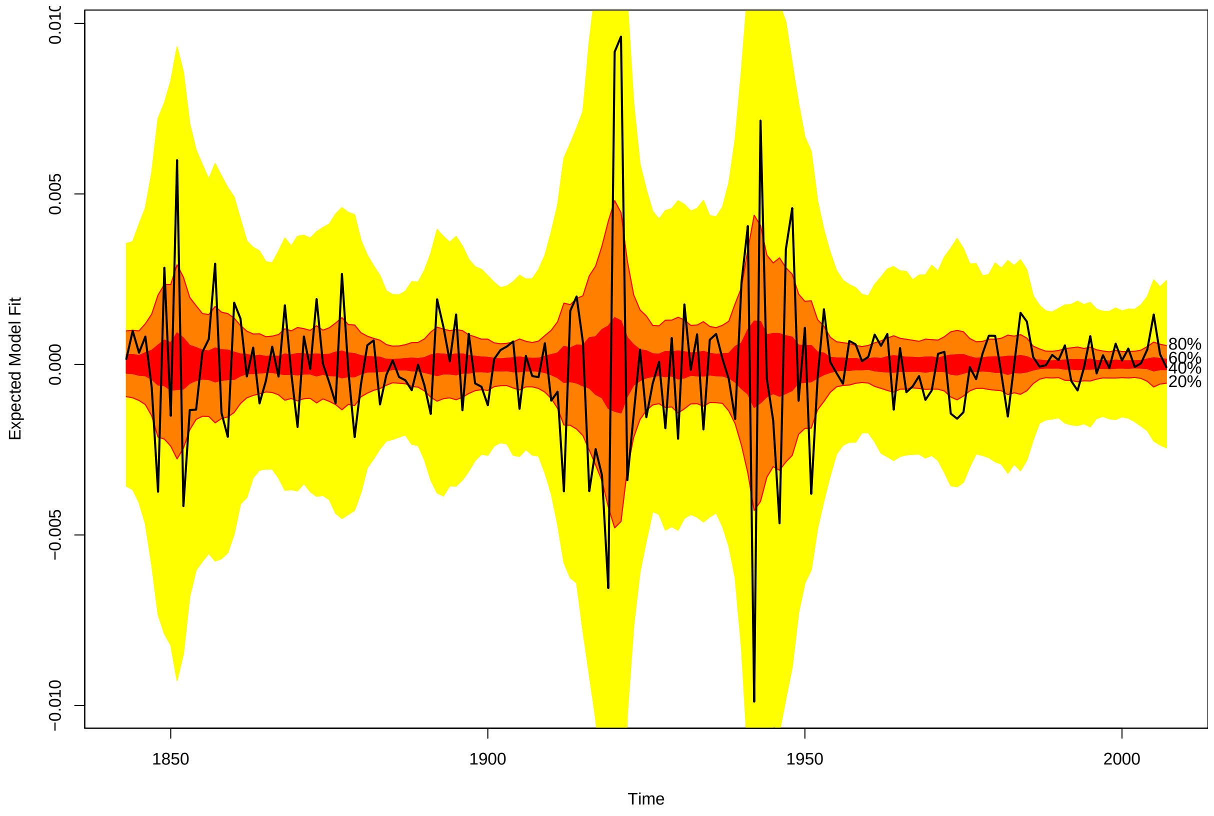
A coarser set of colours can be plotted by creating a pn object with fewer percentiles. This is done by defining the p argument of the pn function with the only the percentiles for which colour changes are desired.

```
> y.pn2 <- pn(sims = y.mcmc, p = c(1, 20, 40, 60, 80, 99), start = y0)
```

Note, that elements of p will ultimately be adjusted to be symmetric around 50. For example if a user set `p = c(1, 40, 80)` a pn object identical the one above would be returned. This allows the user, if desired, to only define percentiles either above or below 50.

The new y.pn2 object can be plotted using the default arguments for contour lines and text labels. Lines are never drawn for percentiles not calculated in the pn object. As a result, in the plotting of y.pn2 there are no contour lines on the 10th, 30th, 50th, 70th and 90th deciles.

```
> #empty plot
> plot(NULL, type = "l", xlim = range(time(y.pn)), xlab = "Time",
      ylim = range(diff(r)), ylab = "Expected Model Fit")
> #add fan
> fan(y.pn2)
> #add data
> lines(diff(r), lwd = 2)
```



4 Forecast Fans

To illustrate the plotting of forecast fans we use the MCMC predictive distributions in the `pop` object. These are used to derive posterior predictive distributions of the population growth rate and population total using the `diffinv` function,

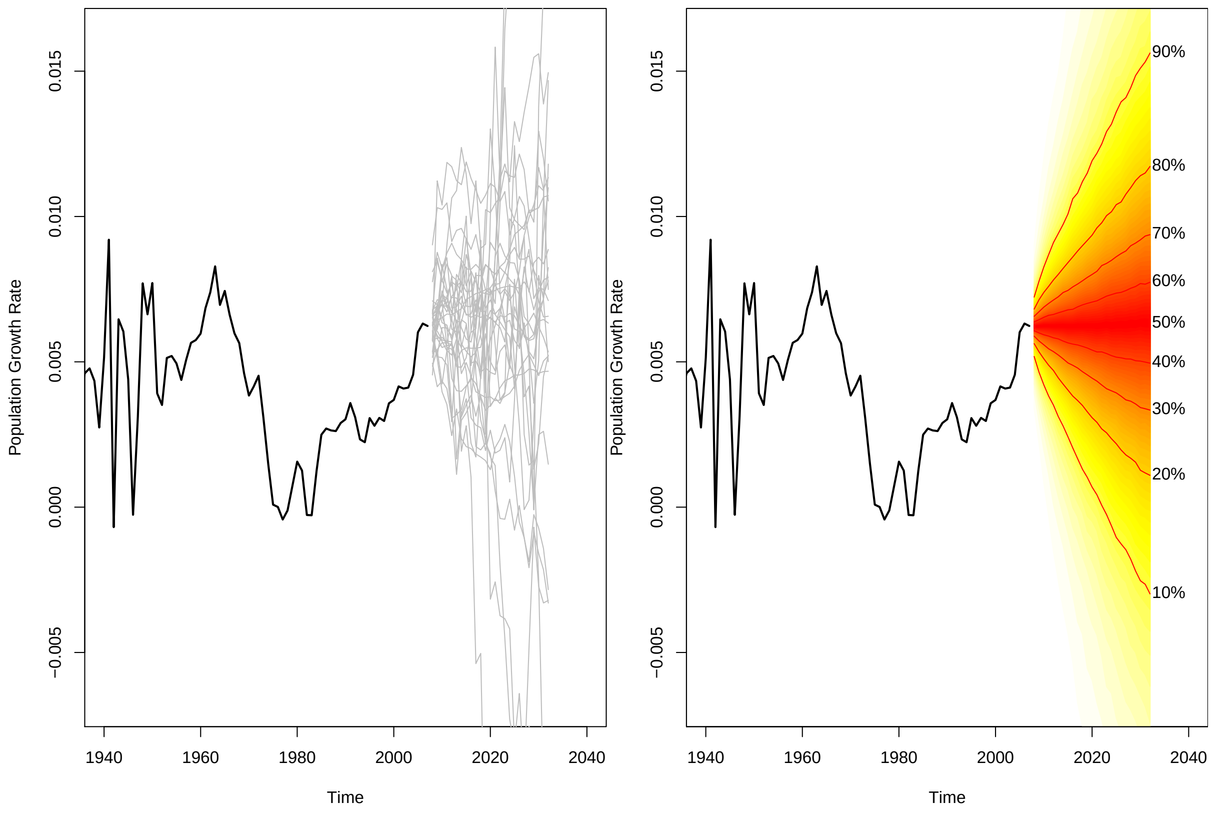
```
> ynew.mcmc <- pop$sims.list$y.new
> rnew.mcmc <- apply(ynew.mcmc, 1, diffinv, xi = tail(r,1))
> rnew.mcmc <- t(rnew.mcmc[-1,])
> pnew.mcmc <- apply(rnew.mcmc, 1, diffinv, xi = log(tail(ew,1)))
> pnew.mcmc <- t(pnew.mcmc[-1,])
> pnew.mcmc <- exp(pnew.mcmc)
```

Percentiles for `rnew.mcmc` can be derived as

```
> h0 <- end(ew)[1]
> rnew.pn <- pn(sims = rnew.mcmc, start = h0 + 1)
```

Note, that as `rnew.mcmc` is a simulation of forecasts, the start argument set to the year after the last observation of the population growth rate. Forecast fans are plotted after estimating the percentile objects, much in the same way as they were for the volatility and model fit in the previous sections. However, some extra care must be taken in setting up the `xlim` to allow space for the fan to appear in the right hand side of plotting area. For the population growth rate this is demonstrated alongside a small section of the underlying simulation data (the first 30 MCMC samples of predictive distribution) that are represent a small part of the underlying data used to calculate the percentiles.

```
> par(mfrow = c(1 ,2))
> #sample of underlying simulation data
> plot(r, ylim = range(r), xlim = c(1940, 2040),
      ylab = "Population Growth Rate", lwd = 2)
> for (i in 1:30) lines(ts(rnew.mcmc[i, ], h0 + 1), col = "grey")
> #plot r
> plot(r, ylim = range(r), xlim = c(1940, 2040),
      ylab = "Population Growth Rate", lwd = 2)
> #add fan
> fan(rnew.pn)
```



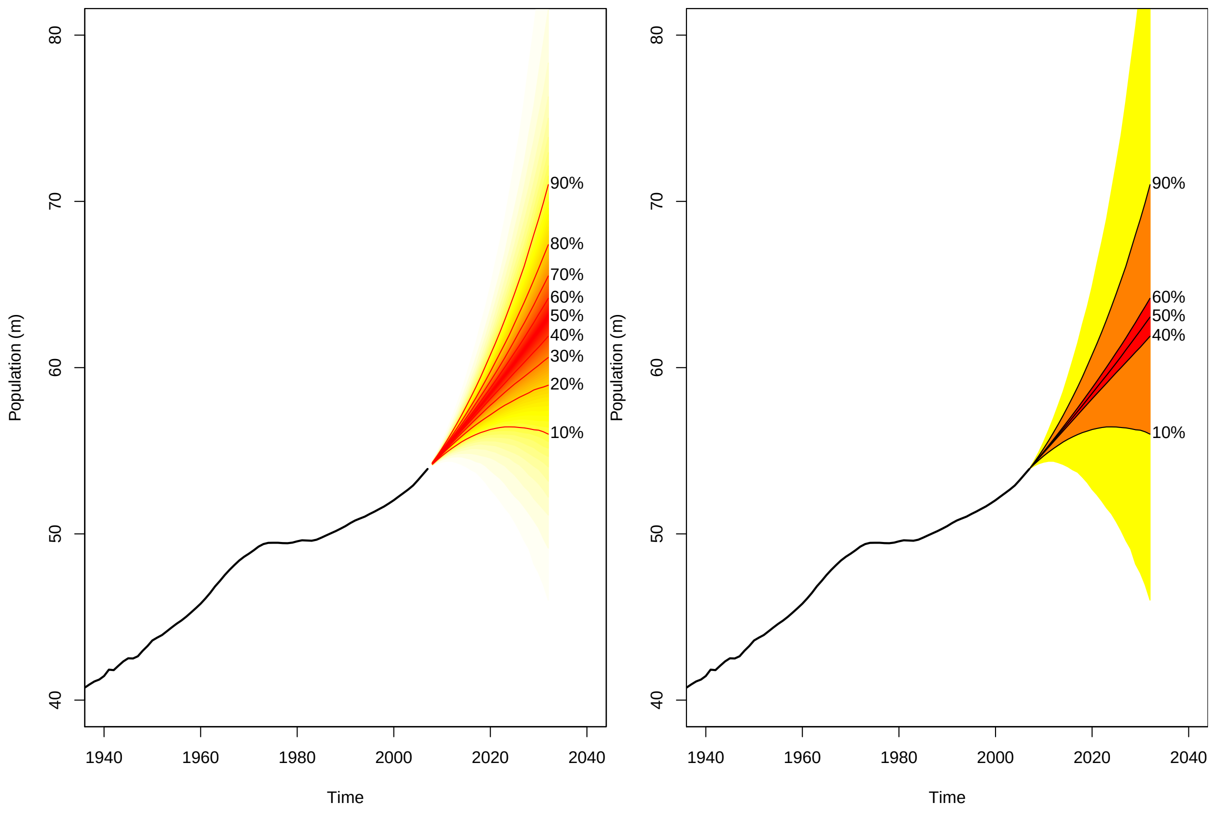
The anchor argument in pn can be utilised to bridge the gap between the predictive distribution fan and the final data point. The associated starting point for creating an time series type object for plotting must also be adjusted to account for the anchoring. For the pnew.mcmc the two alternative sets of percentile calculations, with or without an anchoring can be derived as such:

```
> pnew.pn <- pn(sims = pnew.mcmc/1e+06, start = h0 + 1)
> pnew.pn2 <- pn(sims = pnew.mcmc/1e+06, p = c(1, 10, 40, 50),
     anchor = ew[length(ew)]/1e+06, start = h0)
```

For the pnew.pn2 only a few percentiles are calculated, which will provide a coarser set of shade in the plotting of the predictive distribution. In addition, the anchor is set to the last observed population count, hence the start point is now on the last observation h0, not at h0 + 1. In both calculations simulations are divided by one million to provide easier interpretation.

Both pn objects can be plotted side by side using the fan function. Contour line colours can be altered directly using the ln.col argument for the display of pnew.pn2 on the right hand side below, in comparison to the default plot on the left hand side.

```
> par(mfrow = c(1 ,2))
> #plot ew
> plot(ew/1e+06, ylim = c(40, 80), xlim = c(1940, 2040),
     ylab = "Population (m)", lwd = 2)
> #add fan
> fan(pnew.pn)
> #plot ew
> plot(ew/1e+06, ylim = c(40, 80), xlim = c(1940, 2040),
     ylab = "Population (m)", lwd = 2)
> #add fan
> fan(pnew.pn2, ln.col = "black")
```



References

Abel, G. J. (2013). *tsbugs: Create time series BUGS models.* Retrieved 31 January 2013, from `http://cran.r-project.org/web/packages/tsbugs`.

Abel, G. J., J. Bijak, and J. Raymer (2010, October). A comparison of official population projections with Bayesian time series forecasts for England and Wales. *Population Trends*, 95–114.

Human Mortality Database (2012). Available at `http://www.mortality.org`. University of California, Berkeley (USA) and Max Planck Institute for Demographic Research (Germany).

Lunn, D. J., A. Thomas, N. Best, and D. Spiegelhalter (2000, October). WinBUGS - A

Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing 10*(4), 325–337.

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

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing.

Rogers, A. (1995). *Multiregional Demography: Principles, Methods and Extensions* (1 ed.). John Wiley & Sons.

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