Replication materials for Krüger, Lerch, Thorarinsdottir and Gneiting (2016), 'Probabilistic Forecasting and Comparative Model Assessment Based on Markov Chain Monte Carlo Output'

Fabian Krüger

17 August 2016

Motivation

The simulation study and data example in the paper are quite computer intensive. We nevertheless intend to make the paper's main results (Figures 1, 2 and 5) replicable. To this end, we offer R functions which can run the simulation study and data example, or possibly (and more realistically) smaller versions thereof. The relevant functions are part of the package `scoringRules` (Jordan et al, 2016); they are currently contained in the development version only (https://github.com/FK83/scoringRules), but we'll upload them to CRAN during the next weeks. Specifically, the main replication functions are:

- `run_mcstudy`, which runs the simulation study in Section 4;
- `run_casestudy`, which runs the case study in Section 5.

Both functions create an object that can be plotted via `plot`, as demonstrated below.

Replicating Figures 1 and 2

Running the simulation

The simulation study in the paper is based on 1000 Monte Carlo iterations; computing time is linear in the number of iterations. Therefore, the function `run_mcstudy` has an input argument (`nr_iterations`) which allows to run a small-scale study with fewer iterations (by default, `nr_iterations = 50`). Via another important input parameter, `zoom`, users can decide whether to run the large grid of sample sizes in Figure 1 (this corresponds to `zoom = FALSE`) or the finer grid in Figure 2 (`zoom = TRUE`). The following code chunk provides an example (produced using an Intel i7 processor on a Windows machine with 16 GB RAM).

```
library(scoringRules)

# Number of Monte Carlo iterations to be done
R <- 50

# Run simulation underlying Figure 1
run1 <- run_mcstudy(nr_iterations = R, zoom = FALSE)
```

```
## [1] "2016-08-18 10:28:33 - now starting run_mcstudy"
## [1] "2016-08-18 10:29:47 - now running iteration 10 out of 50"
## [1] "2016-08-18 10:31:07 - now running iteration 20 out of 50"
## [1] "2016-08-18 10:32:28 - now running iteration 30 out of 50"
```

```
## [1] "2016-08-18 10:33:49 - now running iteration 40 out of 50"
## [1] "2016-08-18 10:35:09 - now running iteration 50 out of 50"
## [1] "2016-08-18 10:35:17 - now finishing run_mcstudy"
```

The object `run1` is a list containing the simulation results:

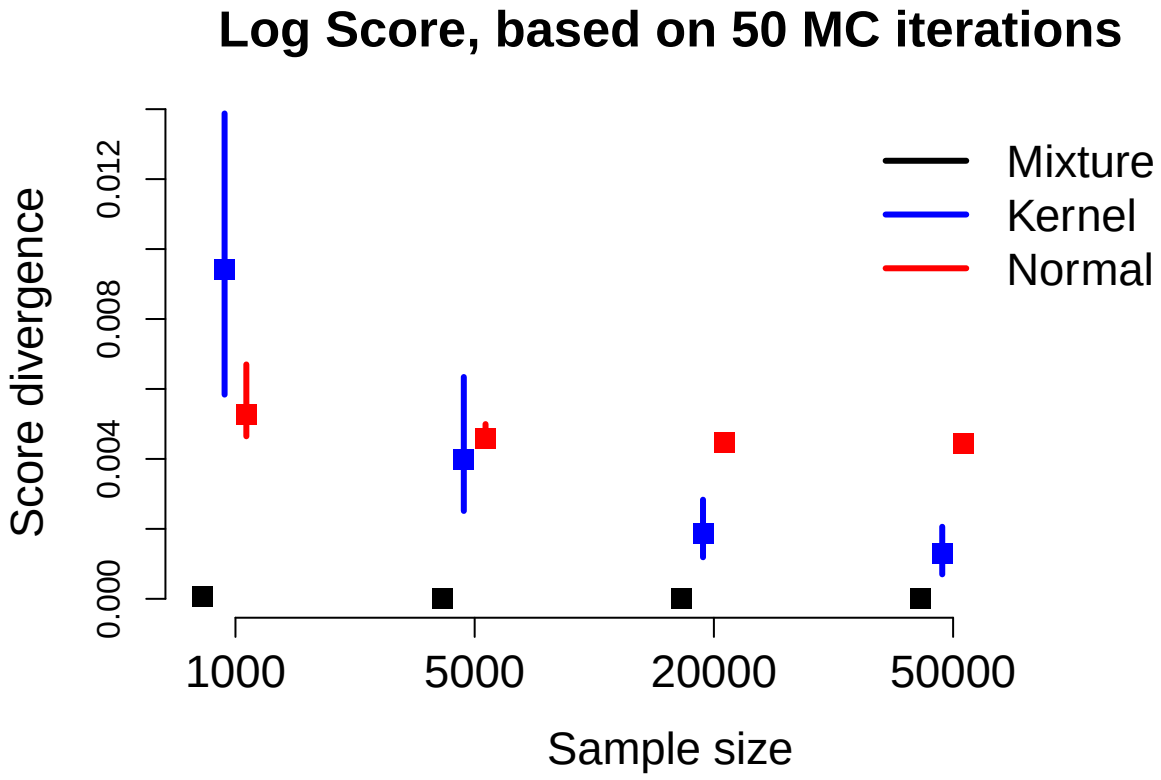
```
str(run1)
```

```
## List of 5
##  $ iteration  : int [1:1400] 1 1 1 1 1 1 1 1 1 1 ...
##  $ sample_size: num [1:1400] 1000 1000 1000 1000 1000 1000 1000 5000 5000 5000 ...
##  $ rule       : chr [1:1400] "logs" "crps" "logs" "crps" ...
##  $ approx     : chr [1:1400] "norm" "norm" "mixp" "mixp" ...
##  $ value      : num [1:1400] 4.99e-03 7.32e-04 4.27e-05 5.12e-06 8.80e-03 ...
##  - attr(*, "row.names")= int [1:1400] 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "class")= chr "mcstudy"
```

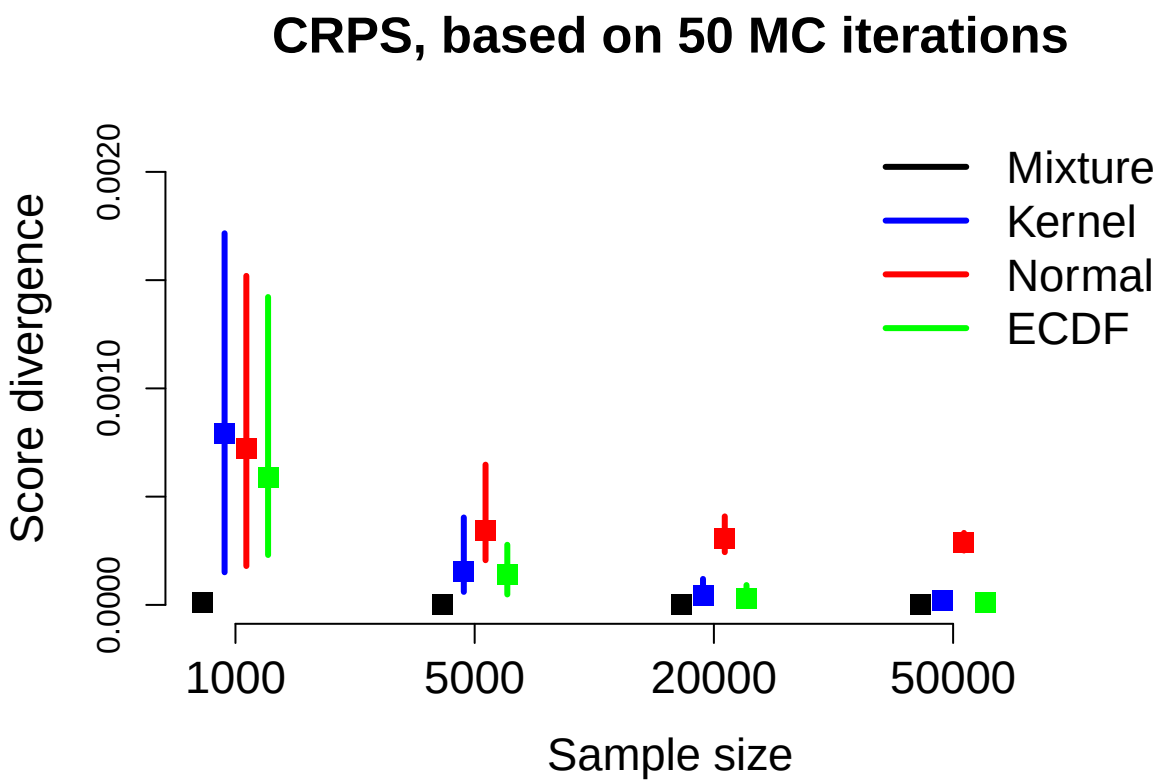
Plotting the results

The data in `run1` can be plotted via `plot`, see `?plot.mcstudy`. Some examples follow.

```
# Produce a plot that's similar to Figure 1 (left panel)
plot(run1, scoring_rule = "logs")
```



```
# ... Figure 1 (right panel)
plot(run1, scoring_rule = "crps")
```



The code for replicating Figure 2 is suppressed here for brevity; it consists of running `run2 <- run_mcstudy(nr_iterations = R, zoom = TRUE)`, followed by `plot(run2)`.

Replicating Figure 5

Producing the results

The data example in the paper is a recursive forecasting exercise, i.e., a statistical model is fitted to an expanding sample of data, making a prediction for the next time step in each case. In the paper we consider various sizes of the MCMC sample, as well as 16 parallel runs using different random seeds. The function `replicate_casestudy` allows to run a reduced (or full) version of the case study. Computation time can be reduced by considering a smaller number of parallel chains (the default input is `nr_of_chains = 3`), or by reducing the maximal size of the MCMC chain at each point in time (the default is `max_mcmc_sample_size = 10000`). Furthermore, in contrast to the results in the paper's Section 4, `replicate_casestudy` uses the computational shortcut described in Appendix B of the paper.

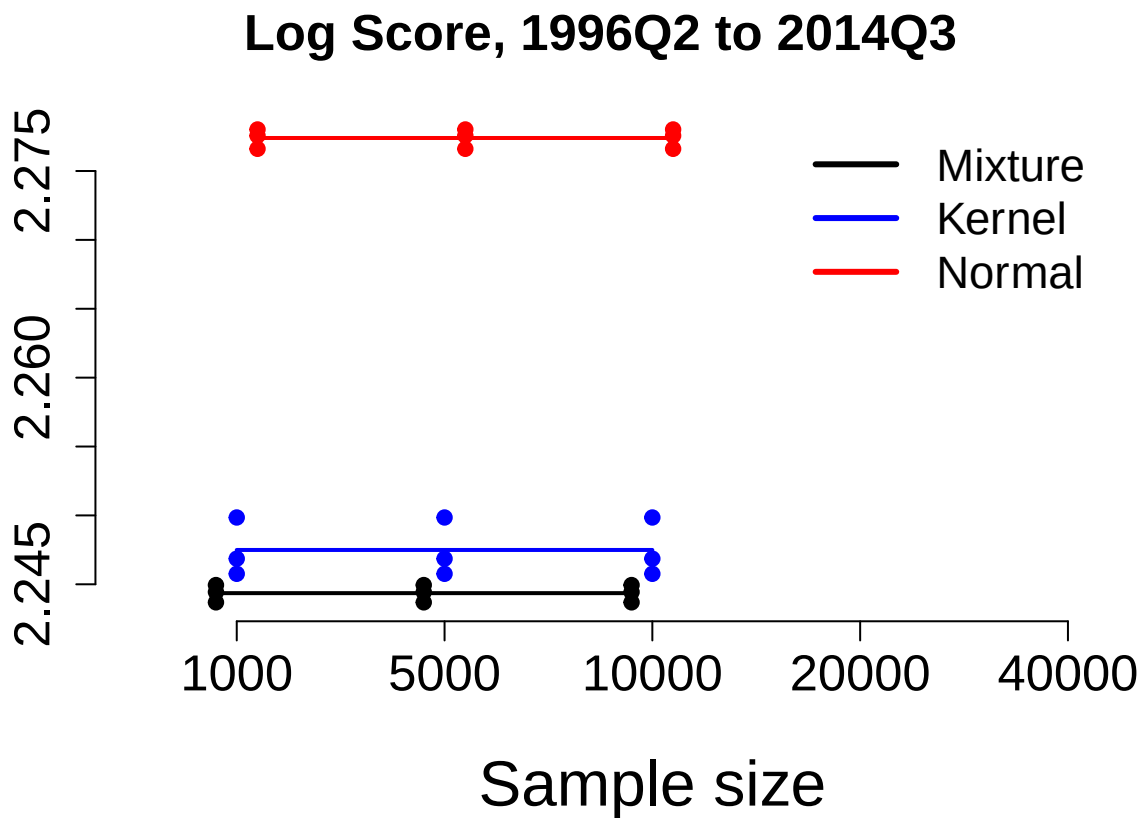
```
data(gdp)
# Run the case study underlying Figure 5
run5 <- run_casestudy(data_df = gdp, max_mcmc_sample_size = 10000)
```

```
## [1] "2016-08-18 10:35:18 - now starting run_casestudy"
## [1] "2016-08-18 10:42:07 - now running date 10 out of 74"
## [1] "2016-08-18 10:49:51 - now running date 20 out of 74"
## [1] "2016-08-18 10:57:44 - now running date 30 out of 74"
## [1] "2016-08-18 11:05:54 - now running date 40 out of 74"
## [1] "2016-08-18 11:14:06 - now running date 50 out of 74"
## [1] "2016-08-18 11:22:37 - now running date 60 out of 74"
## [1] "2016-08-18 11:30:58 - now running date 70 out of 74"
## [1] "2016-08-18 11:35:14 - now finishing run_casestudy"
```

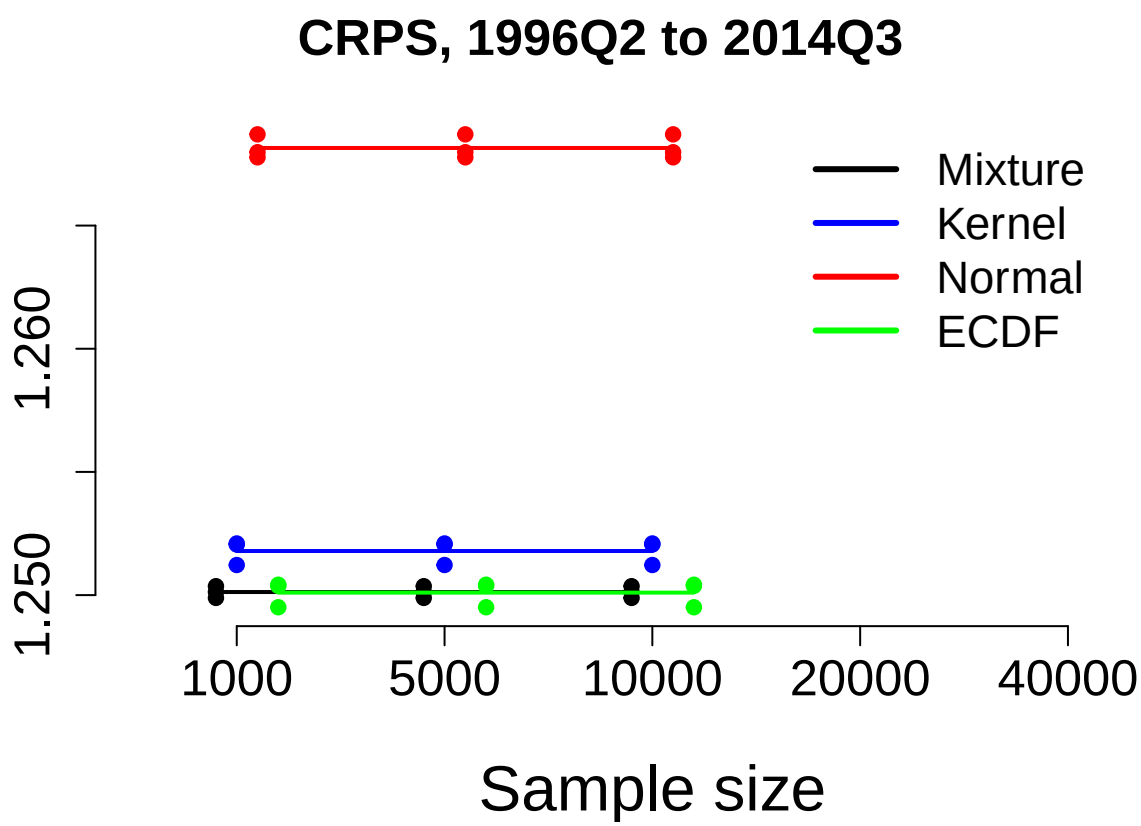
Plotting

Similar to `run_mcstudy`, `run_casestudy` produces a list of data that can be visualized via `plot`:

```
# Produce a plot similar to Figure 5 (left panel)
plot(run5, scoring_rule = "logs")
```




```
# ...Figure 5 (right panel)
plot(run5, scoring_rule = "crps")
```



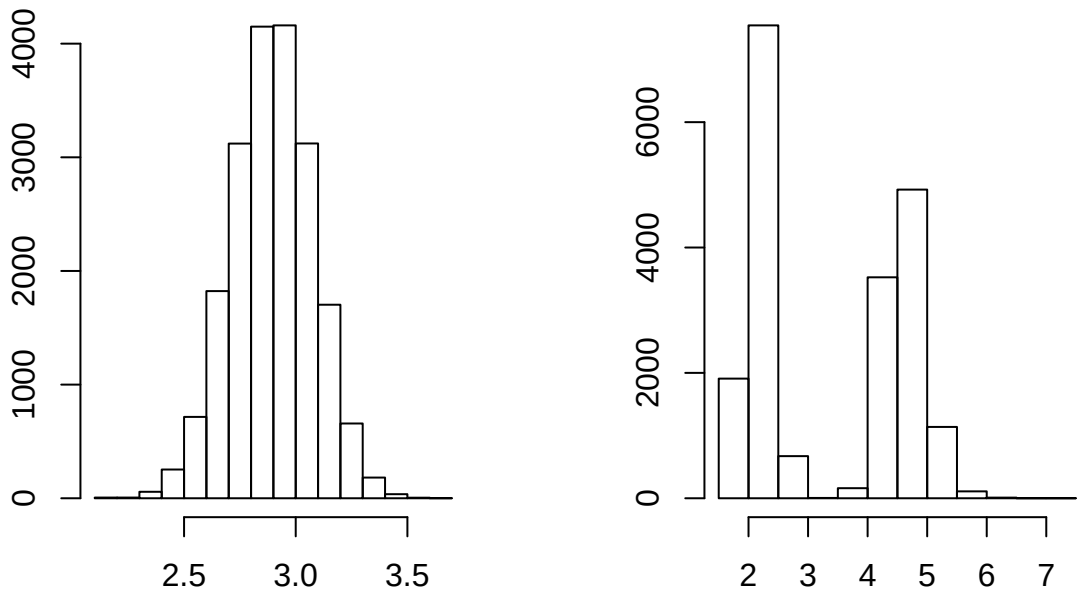
Since `max_mcmc_sample_size = 10000` (smaller than the value of 40000 used in the paper), the axes of the figures are not filled completely; instead, the sample sizes not computed (20000 and 40000) are left blank.

Replicating Figure 4

The paper's main results are contained in Figures 2, 3, and 5, all of which concern the comparison of various approximation methods. Nevertheless, fitting the autoregressive Markov Switching model featured in the paper's Section 5 (and Figure 4) may be of independent interest. We therefore provide the function `ar_ms`. The following example produces a figure similar to the paper's Figure 4, but based on 20000 (rather than 40000) MCMC draws:

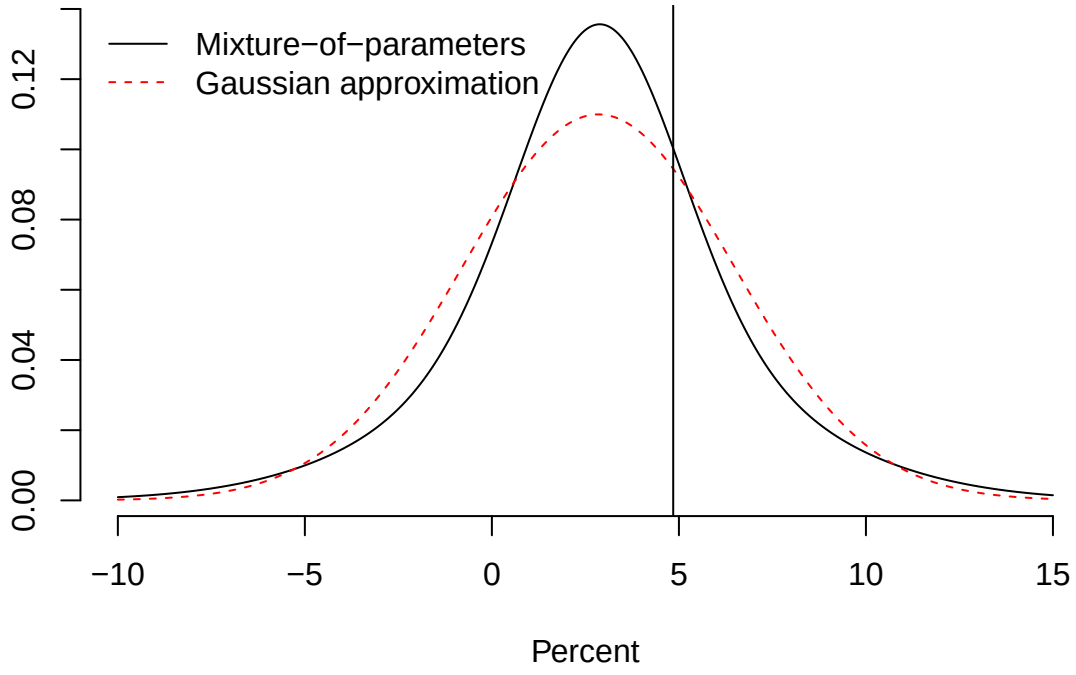
```
# Get 2014Q3 vintage of GDP data
library(dplyr)
data(gdp)
dat <- gdp %>% filter(vint == "2014Q3") %>% arrange(dt)
fit <- ar_ms(dat$val)

par(mfrow = c(1, 2))

# Posterior draws for forecast means
# (similar to top left panel of Figure 4)
hist(fit$fcMeans[, 1], xlab = "", ylab = "", main = "")

# Posterior draws for forecast standard deviations
```

```r
# (similar to top right panel of Figure 4)
hist(fit$fcSds[, 1], xlab = "", ylab = "", main = "")
```




```r
# Posterior predictive density
# (similar to bottom panel of Figure 4)
# Plot for distributions

# Mixture-of-parameters estimator
fmix <- function(z, m, s){
  sapply(z, function(r) mean(dnorm(r, mean = m, sd = s)))
}
xax <- seq(from = -10, to = 15, length.out = 1000)
y <- fmix(xax, m = fit$fcMeans[, 1], s = fit$fcSds[, 1])

# Gaussian approximation
x <- rnorm(nrow(fit$fcMeans),
           mean = fit$fcMeans[, 1], sd = fit$fcSds[, 1])
y2 <- dnorm(xax, mean = mean(x), sd = sd(x))

# Realization
rlz <- gdp %>% filter(dt == "2014Q3" & vint == "2015Q1") %>%
  select(val) %>% unlist %>% unname

# Draw plot
par(mfrow = c(1, 1))
```

```
plot(x = xax, y = y, type = "l", bty = "n",
     xlab = "Percent", ylab = "", main = "")
lines(x = xax, y = y2, col = "red", lty = 2)
abline(v = rlz)
legend("topleft", c("Mixture-of-parameters",
                    "Gaussian approximation"),
       col = c("black", "red"),
       lty = 1:2, bty = "n")
```



References

Jordan, A., Krüger, F. and Lerch, S. (2016): 'scoringRules: Scoring Rules for Parametric and Simulated Distribution Forecasts', R package, available on https://cran.r-project.org/web/packages/scoringRules/index.html (stable version) and https://github.com/FK83/scoringRules (development version).

Krüger, F., Lerch, S., Thorarinsdottir, T. and Gneiting, T. (2016): 'Probabilistic Forecasting and Comparative Model Assessment Based on Markov Chain Monte Carlo Output', working paper, Heidelberg Institute for Theoretical Studies.