Quantile Regression Averaging

Evan L. Ray

20 May, 2020

```r
knitr::opts_chunk$set(message=FALSE, warning=FALSE)
```

Introduction

The purpose of this document is to demonstrate and verify the functionality of the package with a few simple simulated examples.

We will use the following packages:

```r
library(covidEnsembles)
library(zeallot)
library(tidyverse)
library(gridExtra)
```

Simple QRA Example

We consider QRA for the following scenario:

- The observed training data are $X_1, \ldots, X_{100} \overset{\text{i.i.d.}}{\sim} \text{Normal}(0, 1)$
- We have three misspecified predictive models:
  - **wide:** $P_1 = \text{Normal}(0.0, \sqrt{2})$
  - **narrow:** $P_2 = \text{Normal}(0.0, \sqrt{0.5})$
  - **bias_wide:** $P_3 = \text{Normal}(0.2, \sqrt{2})$

The pdfs for these distributions are illustrated below:

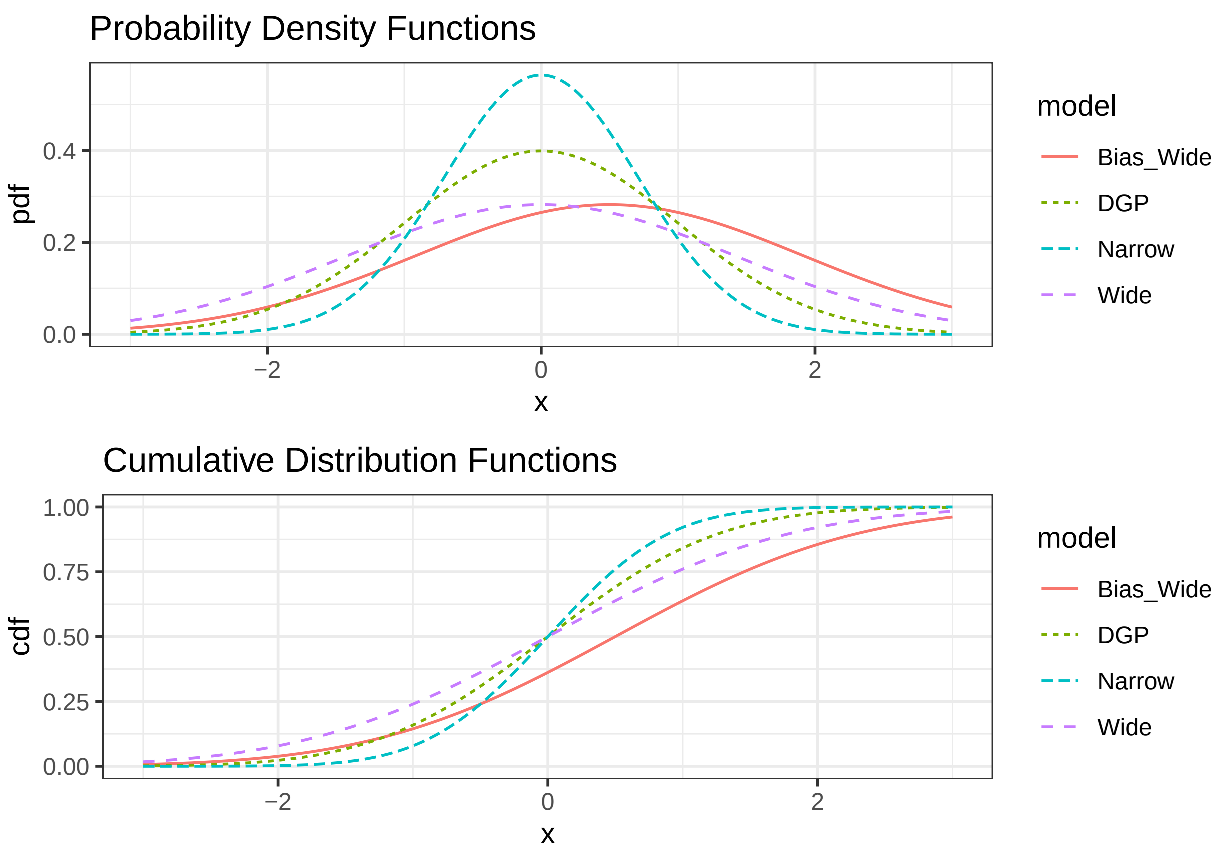
```r
model_params <- data.frame(
  model = c('DGP', 'Wide', 'Narrow', 'Bias_Wide'),
  mean = c(0.0, 0.0, 0.0, 0.5),
  sd = c(1.0, sqrt(2), sqrt(0.5), sqrt(2)),
  stringsAsFactors = FALSE
)

x_grid <- seq(from=-3, to=3, length=201)
curves <- purrr::pmap_dfr(
  model_params,
  function(model, mean, sd) {
    data.frame(
      model = model,
      x = x_grid,
      pdf = dnorm(x_grid, mean=mean, sd=sd),
      cdf = pnorm(x_grid, mean=mean, sd=sd)
    )
  })

p1 <- ggplot(data = curves,
        mapping = aes(x = x, y = pdf, color=model, linetype=model)) +
  geom_line() +
  theme_bw() +
  ggtitle('Probability Density Functions')

p2 <- ggplot(data = curves,
        mapping = aes(x = x, y = cdf, color=model, linetype=model)) +
  geom_line() +
  theme_bw() +
  ggtitle('Cumulative Distribution Functions')

grid.arrange(p1, p2)
```



Throughout, we evaluate forecast skill with the weighted interval score (WIS) based on the following quantile levels:

```
c(0.01, 0.025, seq(0.05, 0.95, by = 0.05), 0.975, 0.99)
```

```
##  [1] 0.010 0.025 0.050 0.100 0.150 0.200 0.250 0.300 0.350 0.400 0.450 0.500
## [13] 0.550 0.600 0.650 0.700 0.750 0.800 0.850 0.900 0.950 0.975 0.990
```

Train and test set set up, component model predictions

We simulate $n_{train} = 10^2$ observations for the training set used to estimate quantile regression averaging parameters, and $n_{test} = 10^4$ observations for the purpose of evaluating performance.

```
set.seed(7462934) # mashed keyboard for science
n_train <- 10^2
y_train <- rnorm(n_train)

n_test <- 10^4
y_test <- rnorm(n_test)
```

For all observations, the predictive quantiles from the three component models are the same. We assemble training and test set QuantileForecastMatrix objects representing these predictions.

```
make_predictive_quantile_objects <- function(n) {
  predictive_quantiles_df <- expand.grid(
    obs_ind = seq_len(n),
    model = model_params$model,
    quantile_level = c(0.01, 0.025, seq(0.05, 0.95, by = 0.05), 0.975, 0.99),
    stringsAsFactors = FALSE
  ) %>%
    left_join(model_params, by='model') %>%
    mutate(quantile_value = qnorm(quantile_level, mean=mean, sd=sd)) %>%
    select(-mean, -sd)

  qfm <- new_QuantileForecastMatrix_from_df(
    predictive_quantiles_df %>% filter(model != 'DGP'),
    model_col = 'model',
    id_cols = c('obs_ind'),
    quantile_name_col = 'quantile_level',
    quantile_value_col = 'quantile_value'
  )

  qfm_per_model <- purrr::map(
    model_params$model,
    function(model) {
      new_QuantileForecastMatrix_from_df(
        predictive_quantiles_df %>% filter(model == UQ(model)),
        model_col = 'model',
        id_cols = c('obs_ind'),
        quantile_name_col = 'quantile_level',
        quantile_value_col = 'quantile_value'
      )
    }
  ) %>%
    `names<-`(model_params$model)

  return(c(list(predictive_quantiles_df, qfm), qfm_per_model))
}

c(train_predictive_quantiles_df,
  train_qfm,
  train_qfm_DGP,
  train_qfm_Wide,
  train_qfm_Narrow,
  train_qfm_Bias_Wide) %<-% make_predictive_quantile_objects(n_train)

c(test_predictive_quantiles_df,
  test_qfm,
  test_qfm_DGP,
  test_qfm_Wide,
  test_qfm_Narrow,
  test_qfm_Bias_Wide) %<-% make_predictive_quantile_objects(n_test)
```

Here are test set scores for the four models individually:

```
test_scores <- model_params['model'] %>%
  mutate(
    wis = purrr::map_dbl(
      model,
      function(model) {
        mean(wis(y_test, get(paste0('test_qfm_', model))))
      }
    )
  )
test_scores
```

```
##       model       wis
## 1       DGP 0.5103418
## 2      Wide 0.5304833
## 3    Narrow 0.5244168
## 4 Bias_Wide 0.5811568
```

QRA Method 1: Equally weighted combination of misspecified models

```
ew_qra_fit <- estimate_qra(qfm_train = train_qfm, qra_model = 'ew')
test_qfm_EW_QRA <- predict(ew_qra_fit, test_qfm)
```

```r
test_scores <- data.frame(model = c('DGP', 'Wide', 'Narrow', 'Bias_Wide', 'EW_QRA')) %>%
  mutate(
    wis = purrr::map_dbl(
      model,
      function(model) {
        mean(wis(y_test, get(paste0('test_qfm_', model))))
      }
    )
  )
test_scores
```

```
##       model       wis
## 1       DGP 0.5103418
## 2      Wide 0.5304833
## 3    Narrow 0.5244168
## 4 Bias_Wide 0.5811568
## 5    EW_QRA 0.5208812
```

QRA Method 2: Convex Combination of misspecified models, one weight per model across all quantile levels

```r
convex_qra_fit <- estimate_qra(qfm_train = train_qfm, y_train = y_train, qra_model = 'convex_pe
test_qfm_Convex_QRA <- predict(convex_qra_fit[[2]], test_qfm)
```

```r
test_scores <- data.frame(model = c('DGP', 'Wide', 'Narrow', 'Bias_Wide', 'EW_QRA', 'Convex_QRA
  mutate(
    wis = purrr::map_dbl(
      model,
      function(model) {
        mean(wis(y_test, get(paste0('test_qfm_', model))))
      }
    )
  )
test_scores
```

```
##        model       wis
## 1        DGP 0.5103418
## 2       Wide 0.5304833
## 3     Narrow 0.5244168
## 4  Bias_Wide 0.5811568
## 5     EW_QRA 0.5208812
## 6 Convex_QRA 0.5104026
```

QRA Method 2: Convex Combination of misspecified models, one weight per model across all quantile levels

```
unconstrained_qra_fit <- estimate_qra(qfm_train = train_qfm, y_train = y_train, qra_model = 'un
test_qfm_Unconstrained_QRA <- predict(unconstrained_qra_fit[[2]], test_qfm)
```

All Test Set Scores

```
all_qra_approaches <- c('DGP', 'Wide', 'Narrow', 'Bias_Wide', 'EW_QRA', 'Convex_QRA', 'Unconst
test_scores <- data.frame(model = all_qra_approaches) %>%
  mutate(
    wis = purrr::map_dbl(
      model,
      function(model) {
        mean(wis(y_test, get(paste0('test_qfm_', model))))
      }
    )
  )
test_scores
```

```
##               model       wis
## 1               DGP 0.5103418
## 2              Wide 0.5304833
## 3            Narrow 0.5244168
## 4         Bias_Wide 0.5811568
## 5            EW_QRA 0.5208812
## 6        Convex_QRA 0.5104026
## 7 Unconstrained_QRA 0.5171923
```