Nonparametric Analysis of US Dairy Production and Consumption

Conformal prediction

Teo Bucci* Filippo Cipriani† Gabriele Corbo‡ Andrea Puricelli§

2023-02-17

Contents

1 Load libraries and data

```
library(mgcv)
library(conformalInference)
library(rgl)
library(dbscan)
library(pbapply)
library(beadplexr)
```

```
data_path = file.path('data_updated_2021')
output_path = file.path('output')
data_infl <-
    read.table(
        file.path(data_path, 'production_facts_inflated.csv'),
        header = T,
        sep = ';'
```

*teo.bucci@mail.polimi.it
†filippo.cipriani@mail.polimi.it
‡gabriele.corbo@mail.polimi.it
§andrea3.puricelli@mail.polimi.it

```
    )
y = data_infl$avg_price_milk
n_b = n = length(y)
```

2 Conformal prediction

```
grid_factor = 1.25
n_grid = 200
alpha = 0.10
```

2.1 Using GAM

Extract train and predict GAM function

- Adjust `var1`, `var2`, ecc. depending on the GAM
- `cbind` correctly the order of the variables in the `conformal.pred`

```
train_gam = function(x, y, out = NULL) {
    colnames(x) = c('var1', 'var2', 'var3', 'var4', 'var5')
    train_data = data.frame(y, x)
    model_gam = gam(y ~ s(var1, bs = 'cr') + s(var2, bs = 'cr') + var3 + var4 + var5,
                    data = train_data)
}

predict_gam = function(obj, new_x) {
    new_x = data.frame(new_x)
    colnames(new_x) = c('var1','var2','var3','var4','var5')
    predict.gam(obj, new_x)
}
```

Perform conformal prediction on the median of our dataset

```
point = c(
    median(data_infl$milk_per_cow),
    median(data_infl$dairy_ration),
    median(data_infl$milk_feed_price_ratio),
    median(data_infl$milk_cow_cost_per_animal),
    median(data_infl$milk_volume_to_buy_cow_in_lbs)
)

c_preds = conformal.pred(
    cbind(
        data_infl$milk_per_cow,
        data_infl$dairy_ration,
        data_infl$milk_feed_price_ratio,
        data_infl$milk_cow_cost_per_animal,
        data_infl$milk_volume_to_buy_cow_in_lbs
    ),
    data_infl$avg_price_milk,
```

```
    point,
    alpha = alpha,
    verbose = T,
    train.fun = train_gam ,
    predict.fun = predict_gam,
    num.grid.pts = n_grid
)
```

```
## Initial training on full data set ...
## Processing prediction point 1 (of 1) ...
```

```
c("LOWER" = c_preds$lo,
  "PRED" = c_preds$pred,
  "UPPER" = c_preds$up)
```

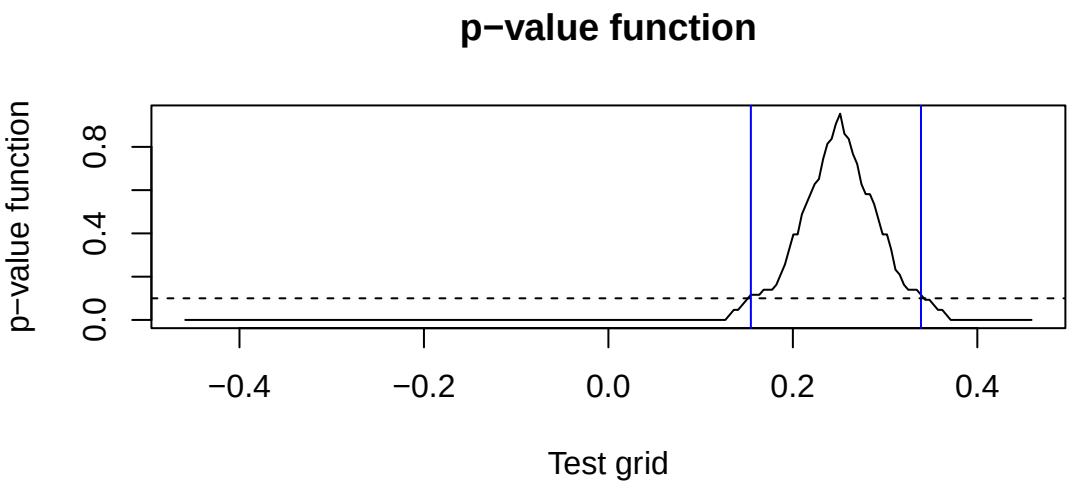
```
##     LOWER      PRED     UPPER
## 0.2374485 0.2439286 0.2512805
```

2.2 Using T Prediction Intervals

```
wrapper_full = function(grid_point) {
    aug_y = c(grid_point, y)
    mu = mean(aug_y)
    ncm = abs(mu - aug_y)
    sum((ncm[-1] >= ncm[1])) / (n + 1)
}

test_grid = seq(-grid_factor * max(abs(y)), +grid_factor * max(abs(y)),
                length.out = n_grid)

pval_fun = sapply(test_grid, wrapper_full)
index_in = pval_fun > alpha
pred_t_interval = range(test_grid[index_in])
```

Plot p-value function

```
plot_pval = function(test_grid, pval_fun, pred, alpha) {
    plot(
        test_grid,
        pval_fun,
        type = 'l',
        main = "p-value function",
        xlab = "Test grid",
        ylab = "p-value function"
    )
    abline(v = pred, col = 'blue')
    abline(h = alpha, lty = 2)
}

plot_pval(test_grid, pval_fun, pred_t_interval, alpha)
```

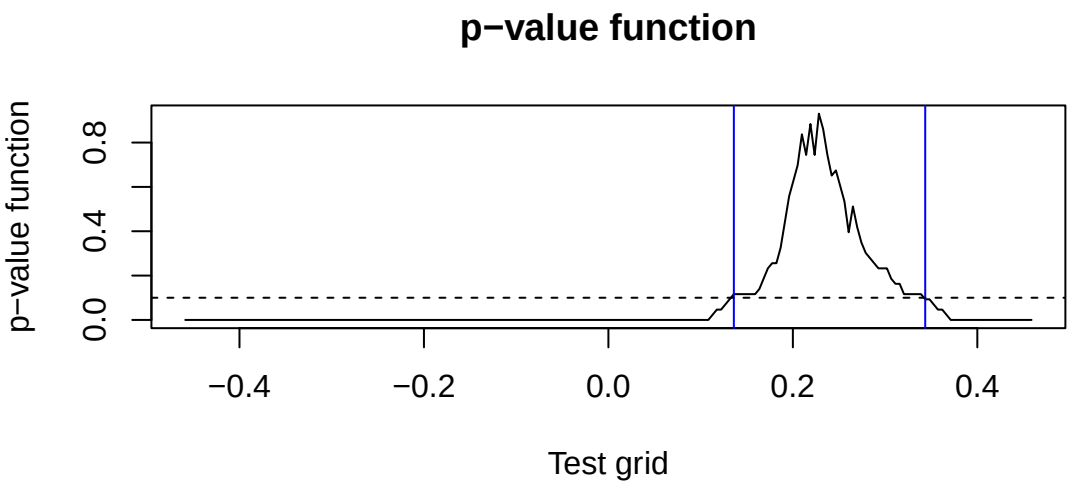


2.3 Using KNN distance

```
pval_fun = numeric(n_grid)
k_s = 0.46
wrapper_knn = function(grid_point) {
    aug_y = c(grid_point, y)
    ncm = kNNdist(matrix(aug_y), k_s * n)
    sum((ncm[-1] >= ncm[1])) / (n_b + 1)
}

pval_fun = sapply(test_grid, wrapper_knn)
index_in = pval_fun > alpha
pred_knn = test_grid[as.logical(c(0, abs(diff(index_in))))]
```

Plot p-value function

```
plot_pval(test_grid, pval_fun, pred_knn, alpha)
```

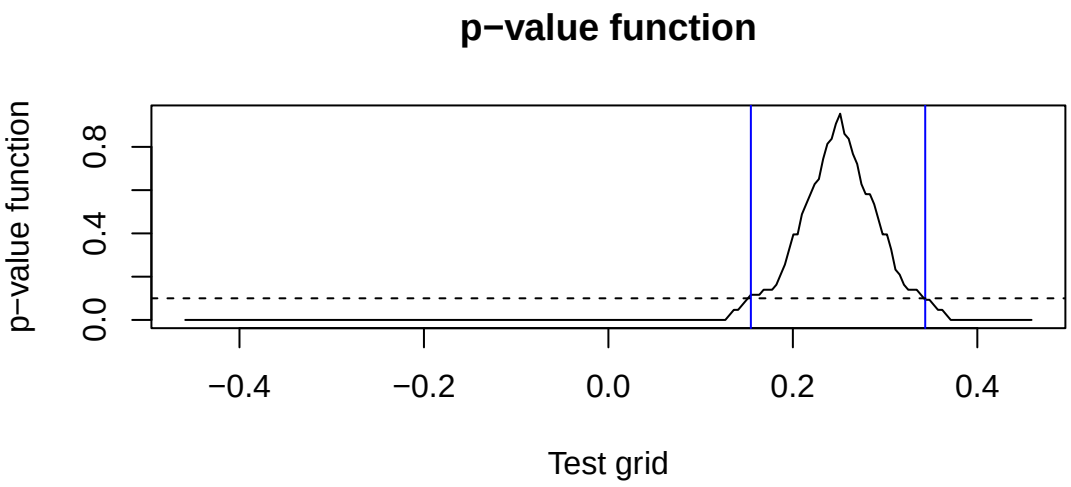


2.4 Using Mahalanobis distance

```
pval_fun = numeric(n_grid)
wrapper_mal = function(grid_point) {
    aug_y = c(grid_point, y)
    ncm = mahalanobis(matrix(aug_y), colMeans(matrix(aug_y)), cov(matrix(aug_y)))
    sum((ncm[-1] >= ncm[1])) / (n_b + 1)
}

pval_fun = sapply(test_grid, wrapper_mal)
index_in = pval_fun > alpha
pred_mahalanobis = test_grid[as.logical(c(0, abs(diff(index_in))))]
```

Plot p-value function

```
plot_pval(test_grid, pval_fun, pred_mahalanobis, alpha)
```



3 Show result

Plot histogram of target variable

```
hist(
    y,
    breaks = 10,
    freq = FALSE,
    main = 'Histogram of Milk Price',
    xlab = 'Milk Price',
    xlim = c(0.1, 0.4),
    border = NA
)
lines(density(y))

abline(v = c(c_preds$lo, c_preds$up),col = 'red',lwd = 2)
abline(v = jitter(pred_t_interval, amount=0.003), col = 'blue', lwd = 1)
abline(v = jitter(pred_knn, amount=0.003), col = 'orange', lwd = 1)
abline(v = jitter(pred_mahalanobis, amount=0.003), col = 'green', lwd = 2)

legend("topright",
       legend = c("GAM", "T Prediction Interval", "KNN", "Mahalanobis"),
       fill = c("red", "blue", "orange", "green"))
```


```
result = data.frame(
    rbind(
        "GAM"=c(c_preds$lo, c_preds$up),
        "T Prediction Interval"=pred_t_interval,
        "KNN"=pred_knn,
        "Mahalanobis"=pred_mahalanobis
    )
)
names(result) = c("LOWER", "UPPER")

knitr::kable(result)
```

	LOWER	UPPER
GAM	0.2374485	0.2512805
T Prediction Interval	0.1544568	0.3388828
KNN	0.1360142	0.3434935
Mahalanobis	0.1544568	0.3434935