Choosing an appropriate scalar transformation to normalise wpd

Sayani Gupta

- Data presented

Observations are generated from a N(0,1) distribution for each combination of nx and $nfacet$ from the following sets: $nx = nfacet = \{2, 3, 5, 7, 14, 20, 31, 50\}$ to cover a wide range of levels from very low to moderately high. Each combination is being referred to as a *panel*. That is, data is being generated for each of the panels $\{nx = 2, nfacet = 2\}, \{nx = 2, nfacet = 3\}, \{nx = 2, nfacet = 5\}, \dots, \{nx = 50, nfacet = 31\}, \{nx = 50, nfacet = 50\}$. For each of the 64 panels, $ntimes = 500$ observations are drawn for each combination of the categories. That is, if we consider the panel $\{nx = 2, nfacet = 2\}$, 500 observations are generated for each of the combination of categories from the panel, namely, $\{(1, 1), (1, 2), (2, 1), (2, 2)\}$. The values of λ is set to 0.67 and values of raw wpd is obtained.

- How the distribution of the raw wpd (without any transformation for normalization) looks across nfacets and nx? Both shape and scale of the distribution changes for different nx and nfacet categories as could be seen in Figure 1.

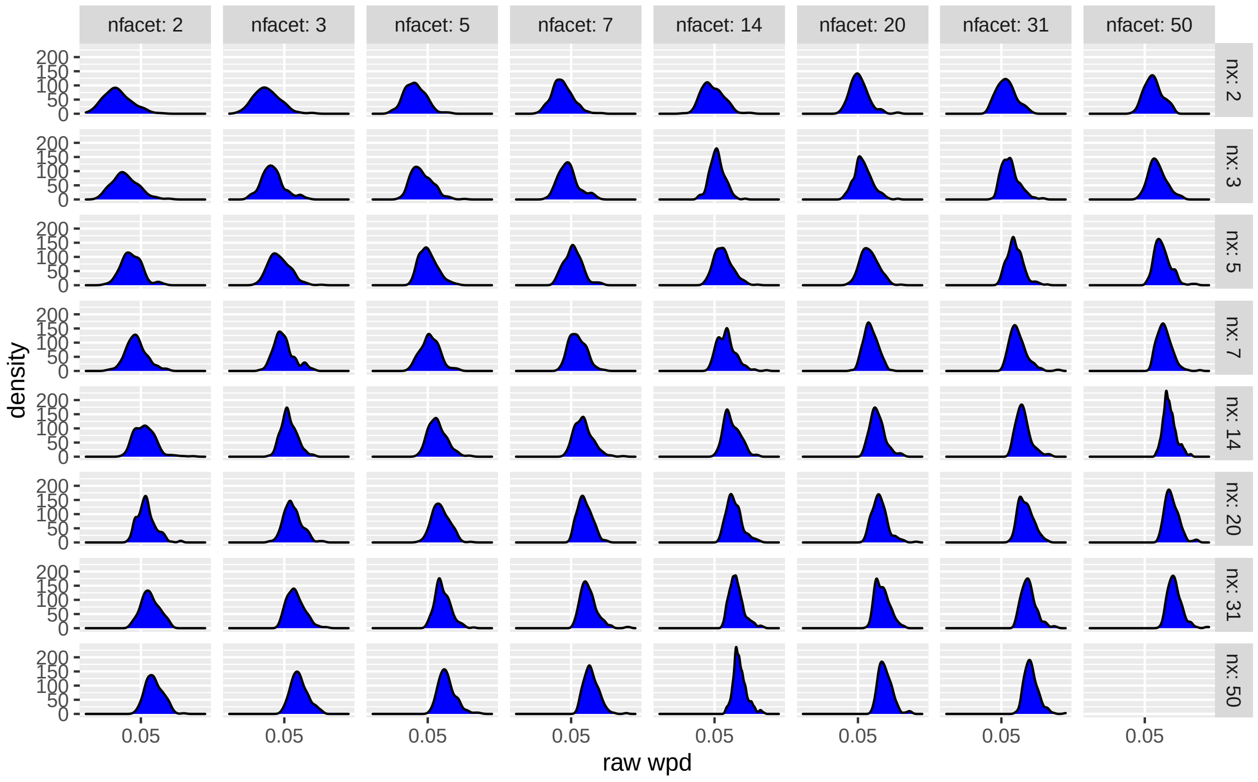


Figure 1: Distribution of raw wpd is plotted across different nx and nfacet categories. Both shape and scale of the distribution changes for different nx and nfacet categories.

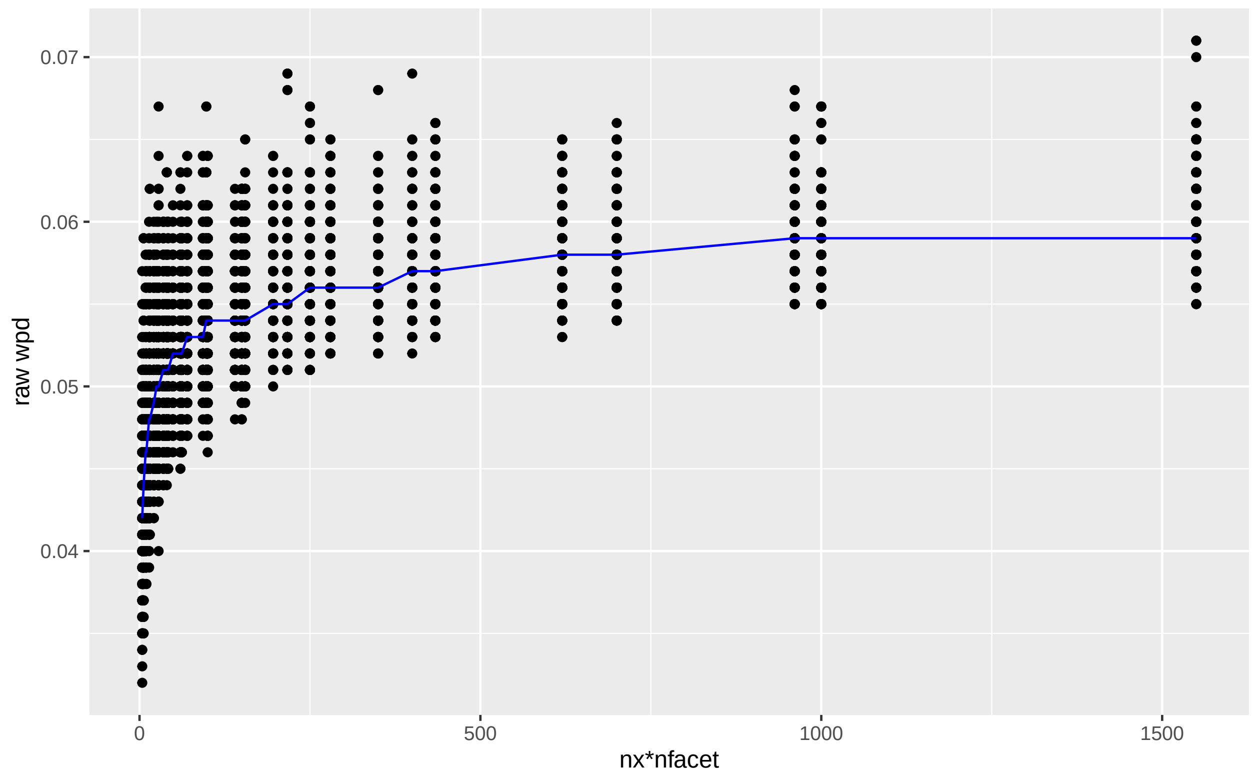


Figure 2: The raw wpd is plotted against nx*nfacet and the blue line represents the median of the multiple values for each nx*nfacet levels.

- Plot the values of raw wpd against nx*nfacet to see the rough relationship (2).

Current attempt

- Fit a log-linear model of median(values) to log(nx*nfacet) as discussed today. The fit is pretty good and the summary of the fit is shown below.

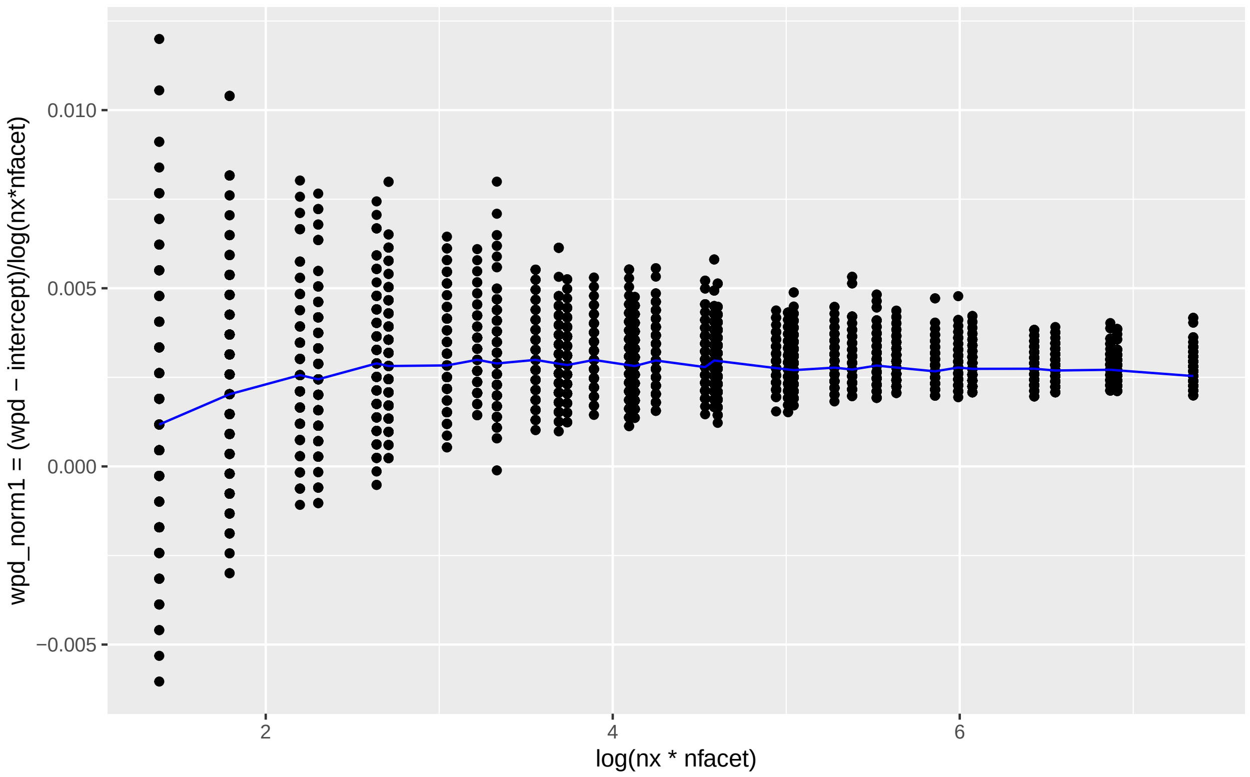
```
N01_median <- N01 %>%
  group_by(nx*nfacet) %>%
  summarise(actual = median(value))

fit_lm2 <- lm(actual ~ poly(log(`nx * nfacet`) ,1, raw=TRUE), data = N01_median)
summary(fit_lm2)
```

```
##
## Call:
## lm(formula = actual ~ poly(log(`nx * nfacet`), 1, raw = TRUE),
##     data = N01_median)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0021910 -0.0002941  0.0001063  0.0003966  0.0009917
##
## Coefficients:
```

```
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                            4.037e-02  3.814e-04  105.85   <2e-16
## poly(log(`nx * nfacet`), 1, raw = TRUE) 2.757e-03  8.043e-05   34.27   <2e-16
##
## (Intercept)                            ***
## poly(log(`nx * nfacet`), 1, raw = TRUE) ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0007206 on 32 degrees of freedom
## Multiple R-squared:  0.9735, Adjusted R-squared:  0.9727
## F-statistic:  1175 on 1 and 32 DF,  p-value: < 2.2e-16
```

- Make it a horizontal line so that the values are not affected by nx*nfacet by plotting $wpd_{norm} = (value - intercept)/log(nx * nfacet)$ against log(nx*nfacet). The line actually becomes horizontal after this transformation as could be seen in Figure **??**.



- See distribution of the transformed variable wpd_{norm} across nx and nfacets. For higher values of nx and nfacet, the distribution is similar, but not for lower values.

It was suggested that we can do permutation normalization for smaller levels and this scalar transformation for higher levels as this transformation is working well when both nx and nfacet are high enough. There are potentially two problems if we follow that approach:

- Since the normalized wpd's for different harmonies will be on different scales, harmonies could not be ranked even for the same objects (households) to select the most important household for that object - which was essentially the idea. Hence, the normalized wpd's would have to be brought on the same scale irrespective of the approach (whether permutation approach or scalar transformation).
- We have to think about the cut-offs, that is, for which level of nx and nfacets, we want to move to scalar transformation like this.

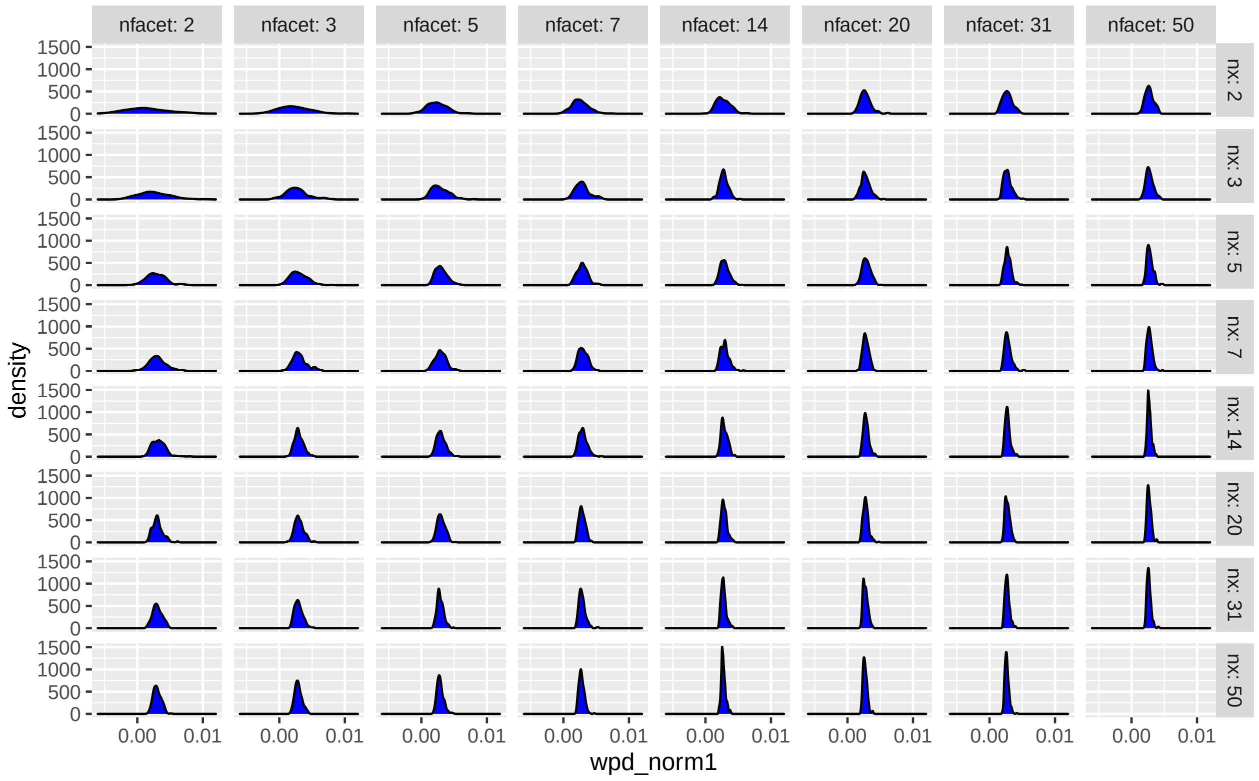


Figure 3: The distribution of the transformed wpd is plotted. The shape and the scale of the distribution still changes across different nx and nfacet levels, but seems uniform for higher levels.

1 Reduce the heteroskedasticity by the transformation y - a - b*log(n)

In Figure 4, we see that the transformed wpd value $wpd_{norm2} = wpd - intercept - slope * log(nx * nfacet)$ has lower heteroscedasticity as compared to wpd_{norm1} when plotted across $log(nx * nfacet)$.

```r
N01 %>%
  ggplot(aes(x=log(nx*nfacet), y = (value - intercept - slope*log(nx*nfacet)))) +
  geom_point() + stat_summary(fun=mean, geom="line", aes(group=1), color = "blue") +
  xlab("wpd_norm2 =  value - intercept - slope*log(nx*nfacet)")
```

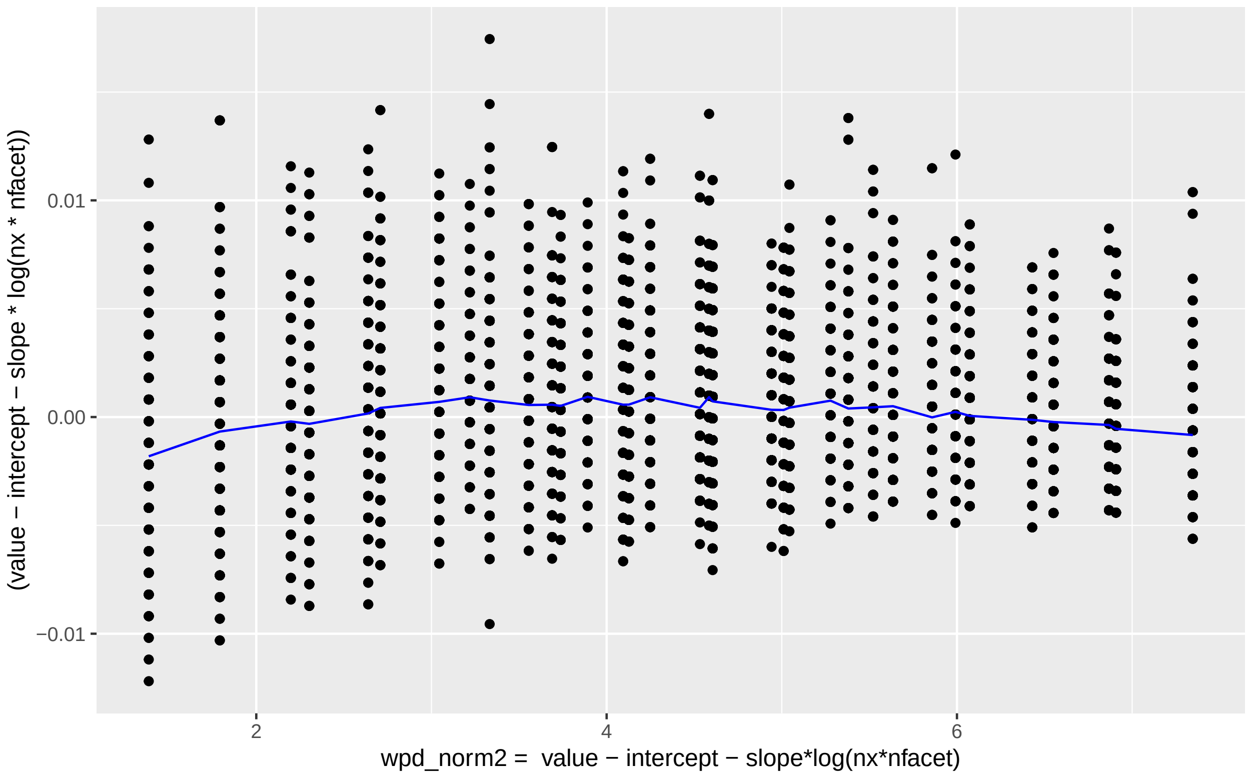


Figure 4: wpd_{norm2} is plotted against log(nx*nfacet). The relationship is horizontal with reduced heteroscedasticity.

In Figure 5, we see that the distribution of wpd_{norm2} is more similar across nx and nfacets than wpd_{norm1}.

```r
N01 %>%
  ggplot(aes(x = ((value - intercept - slope*log(nx*nfacet))))) +
  geom_density(fill = "blue") +
  facet_grid(nx~nfacet,
             labeller = "label_both") +
  xlab("wpd_norm2 =  value - intercept - slope*log(nx*nfacet)") +
  scale_x_continuous(breaks = scales::breaks_extended(3))
```

In Figure 6, we see that the distribution of wpd_{norm2} is similar when normalised with true or approximate estimates of a and b.

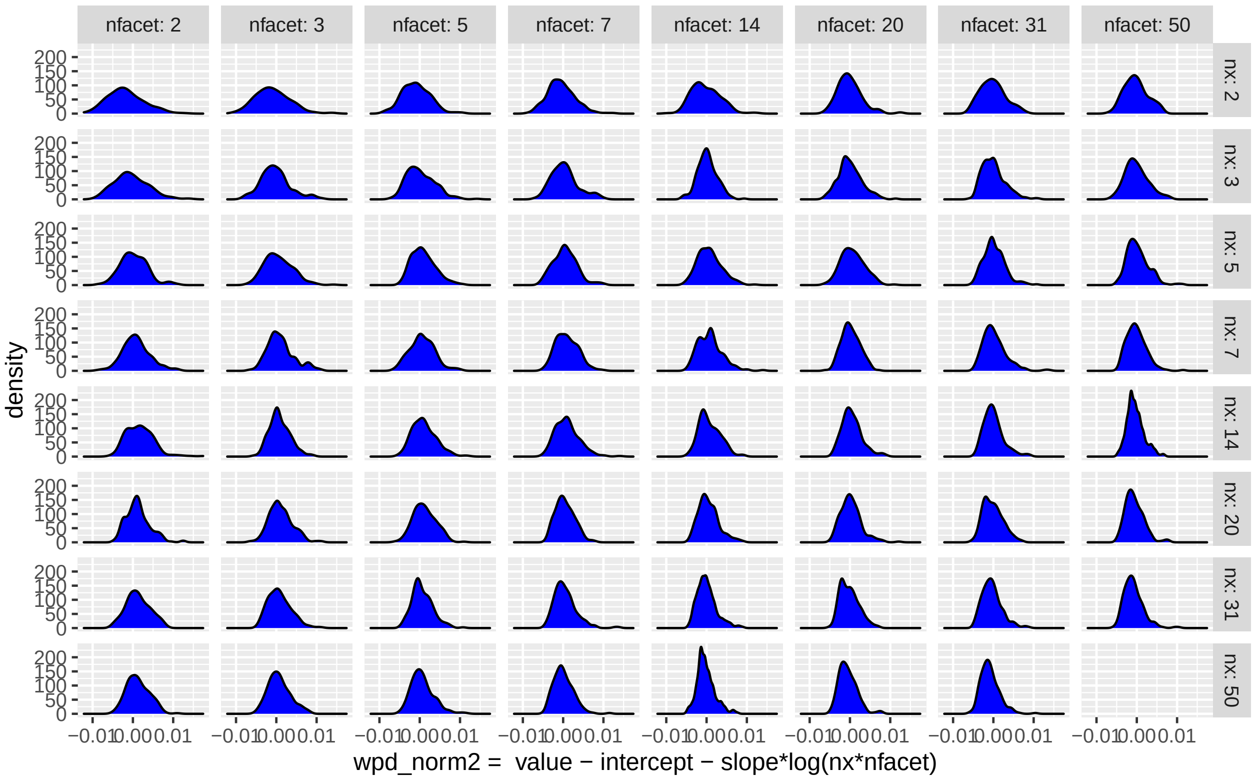


Figure 5: The distribution of wpd_{norm2} is plotted. The distributions are more similar across different nx and nfacets as compared to wpd_{norm1}.

```r
N01_compare <- N01 %>%
  mutate(with_true_est = (value - intercept - slope*log(nx*nfacet)),

                        with_approx_est = (value - 0.04 - 0.003*log(nx*nfacet))) %>%
  pivot_longer(cols = 5:6,
               names_to = "type_estimate",
               values_to = "value_estimate")

N01_compare %>%
  ggplot() +
  geom_density(aes(x = value_estimate, fill = type_estimate), alpha = 0.5) +
  facet_grid(nx~nfacet,
             labeller = "label_both") +
  xlab("value - intercept - slope*log(nx*nfacet)") +
  scale_x_continuous(breaks = scales::breaks_extended(3)) +
  theme(legend.position = "bottom") +
  scale_fill_manual(values = c("#0072B2", "#D55E00")) +
  xlab("wpd_norm2")
```

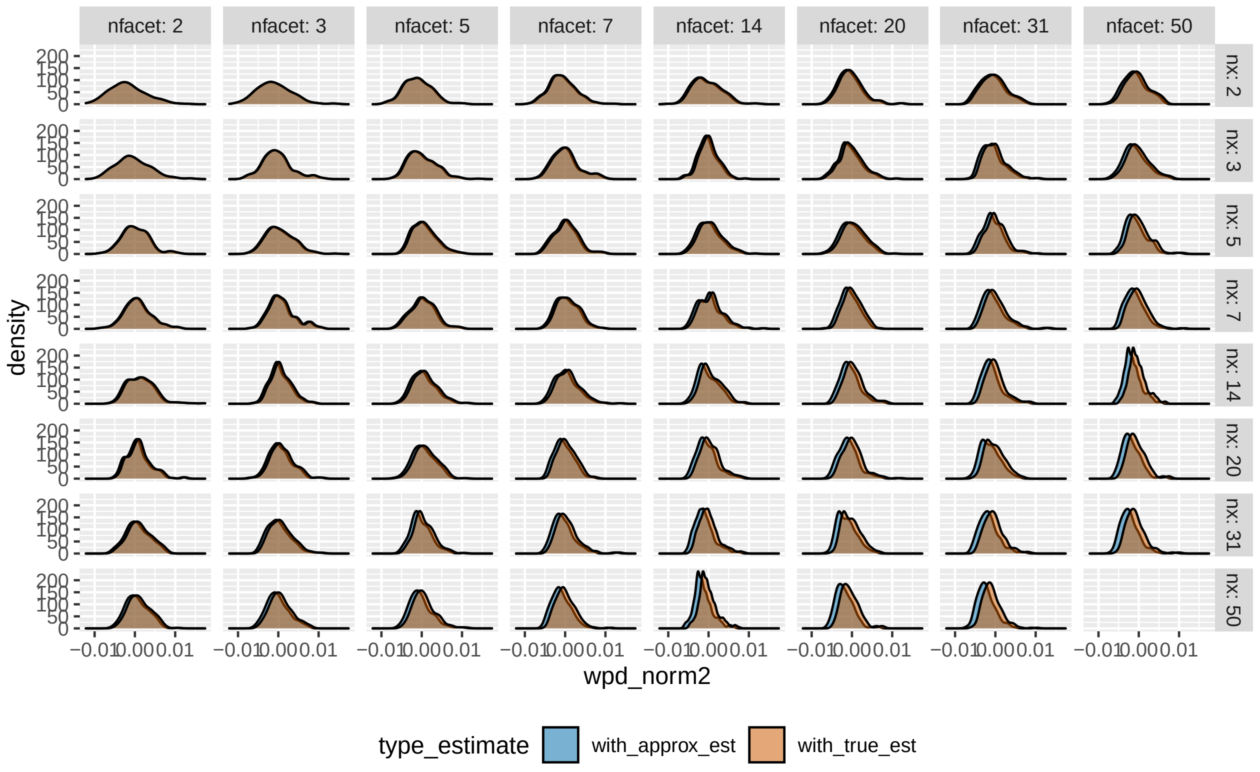


Figure 6: The distribution of wpd_{norm2} is plotted for approximate estimate of a and b. The distributions are similar when plotted against true estimate of and b.