Homework 2

Ulrich Atz

2021-03-06

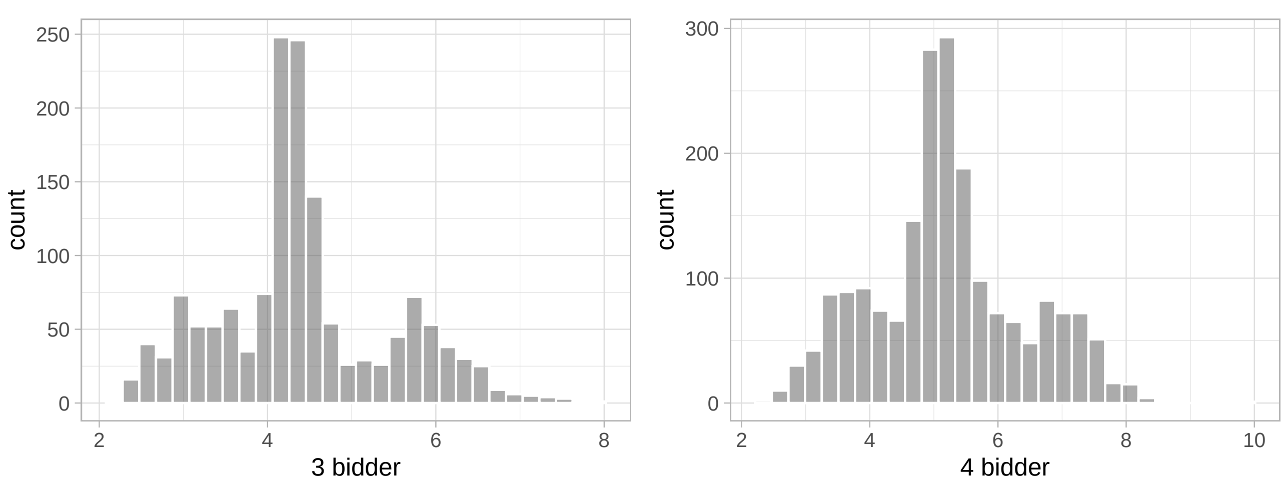
Part 1: Guerre Perrigne and Vuong (Econometrica 2000)

First we show that the general solution in (2) encompasses the simple uniform distribution for v_i, too. We used the distribution $v_i \sim U[0,1]$ to derive that $G(b_i) = v_i/a$ and $g(b_i) = 1/a$.

$$v_i = b_i + \frac{G(b_i)}{(N-1)g(b_i)} = b_i + \frac{v_i/a}{(N-1)/a}$$
$$b_i = v_i - \frac{v_i}{(N-1)} = \frac{v_i(N-1) - v_i}{(N-1)} = \frac{v_i(N-2)}{(N-1)} \approx \frac{(N-1)}{N} v_i$$

The histogram:

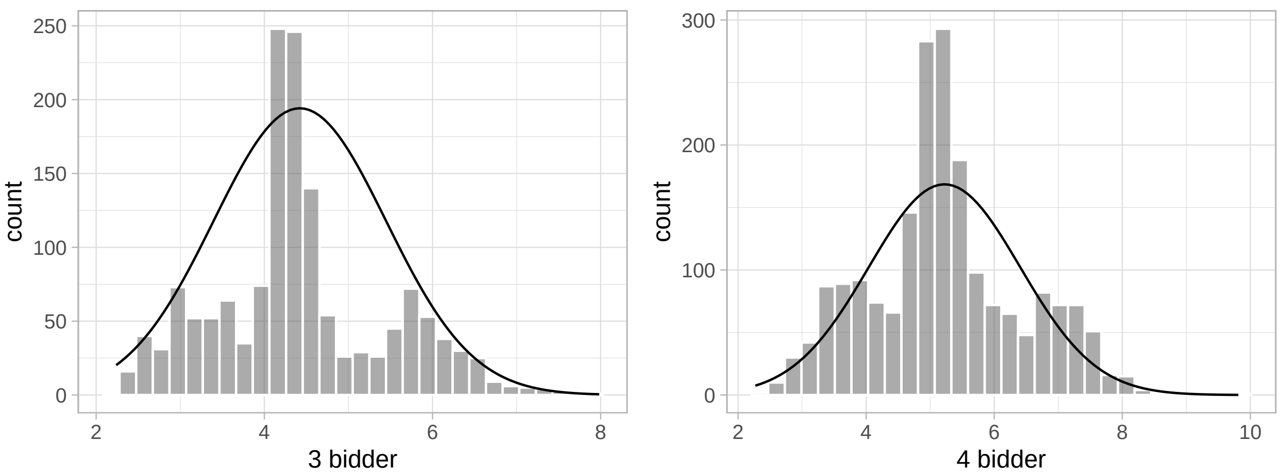
```
# We drop the one missing observation
d3 <- read_csv("auction_3bidder.csv", col_names = F) %>%
  pivot_longer(everything(), names_to = "bidder", values_to = "bid") %>% drop_na()

d4 <- read_csv("auction_4bidder.csv", col_names = F)%>%
  pivot_longer(everything(), names_to = "bidder", values_to = "bid") %>% drop_na()

h3 <- ggplot(d3, aes(bid)) + geom_histogram(alpha = 0.5, color = 'white') + xlab("3 bidder")
h4 <- ggplot(d4, aes(bid)) + geom_histogram(alpha = 0.5, color = 'white') + xlab("4 bidder")

gridExtra::grid.arrange(h3, h4, nrow = 1)
```




```
summ3 <- d3 %>% summarize(mean = mean(bid), sd = sd(bid)) %>% round(2)
summ4 <- d4 %>% summarize(mean = mean(bid), sd = sd(bid)) %>% round(2)
```

For the 3-bidder auction the mean is 4.42 and the standard deviation is 1.03. For the 4-bidder auction the mean is 5.22 and the standard deviation is 1.18.

```
h3_n <- h3 + geom_function(fun = dnorm,
                           args = list(mean(d3$bid, na.rm = T), sd(d3$bid, na.rm = T)),
                           aes(y = after_stat(y*nrow(d3)/3)))
h4_n <- h4 + geom_function(fun = dnorm,
                           args = list(mean(d4$bid, na.rm = T), sd(d4$bid, na.rm = T)),
                           aes(y = after_stat(y*nrow(d4)/4)))

gridExtra::grid.arrange(h3_n, h4_n, nrow = 1)
```

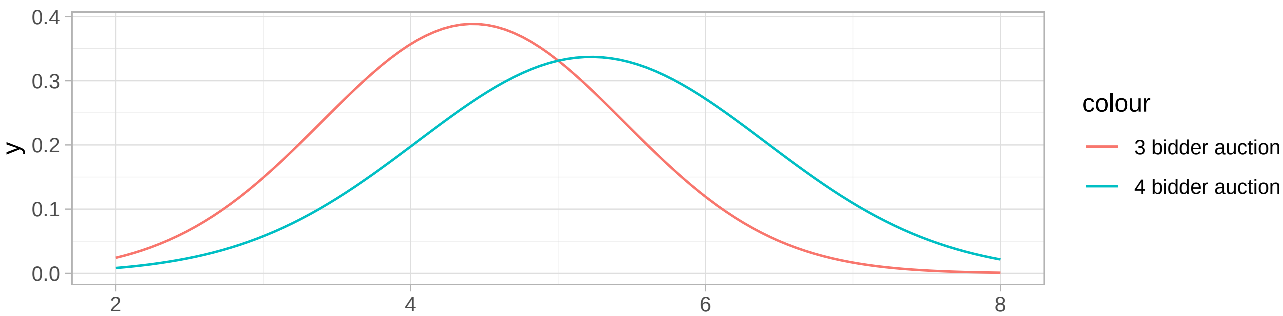


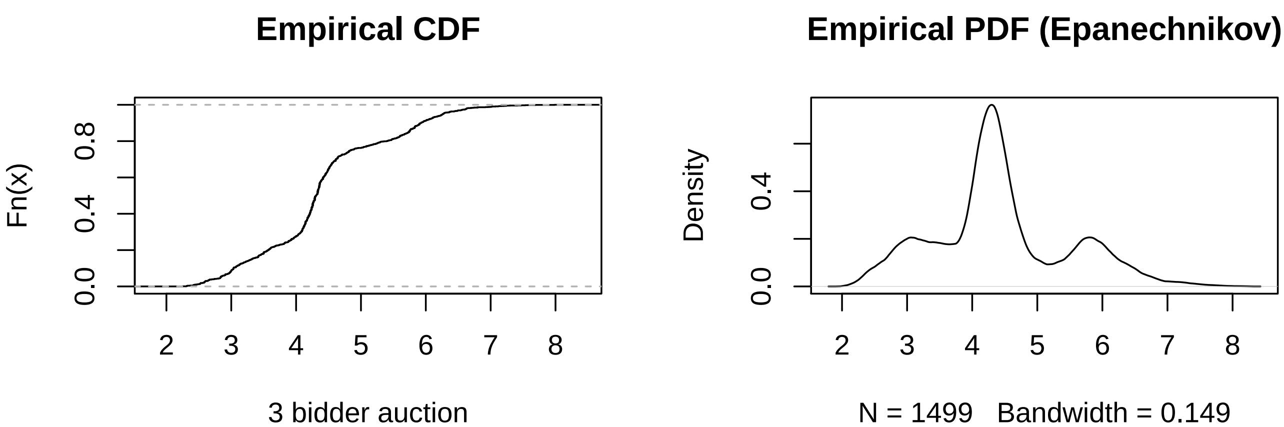
The two normal distributions do not appear to be similar, at least for their central moments.

```
ggplot() +
  xlim(2, 8) +
  geom_function(fun = dnorm, args = list(mean(d3$bid, na.rm = T), sd(d3$bid, na.rm = T)),
                aes(color = "3 bidder auction")) +
  geom_function(fun = dnorm, args = list(mean(d4$bid, na.rm = T), sd(d4$bid, na.rm = T)),
                aes(color = "4 bidder auction"))
```

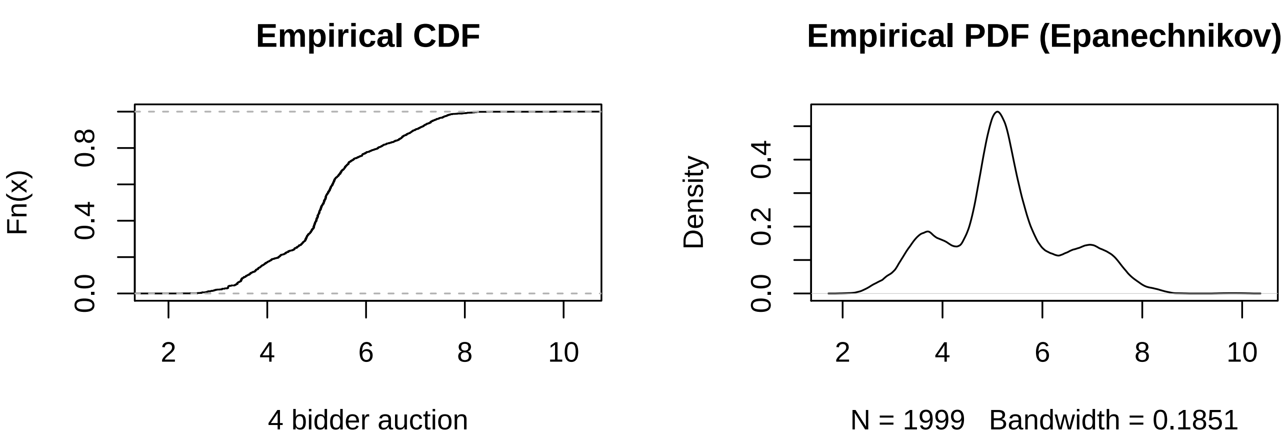
The empirical cumulative distribution function is a step function.

```
# Quick eyeball for 3 bidder auction
ecdf3 <- ecdf(d3$bid)
dens3 <- density(d3$bid, kernel = "epanechnikov")

par(mfrow = c(1,2))
ecdf3 %>% plot(main = "Empirical CDF", xlab = "3 bidder auction")
plot(dens3, main = "Empirical PDF (Epanechnikov)")
```

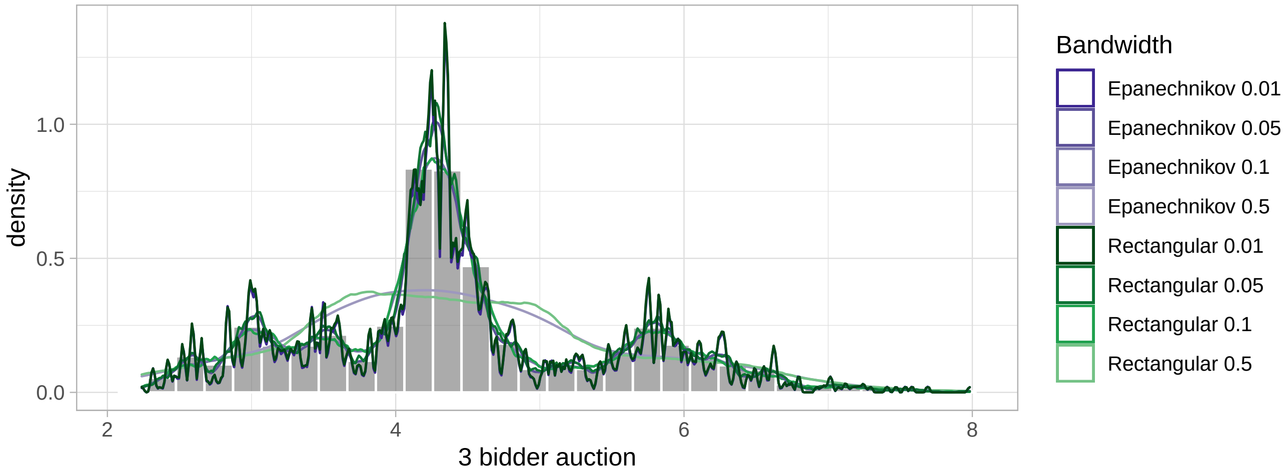




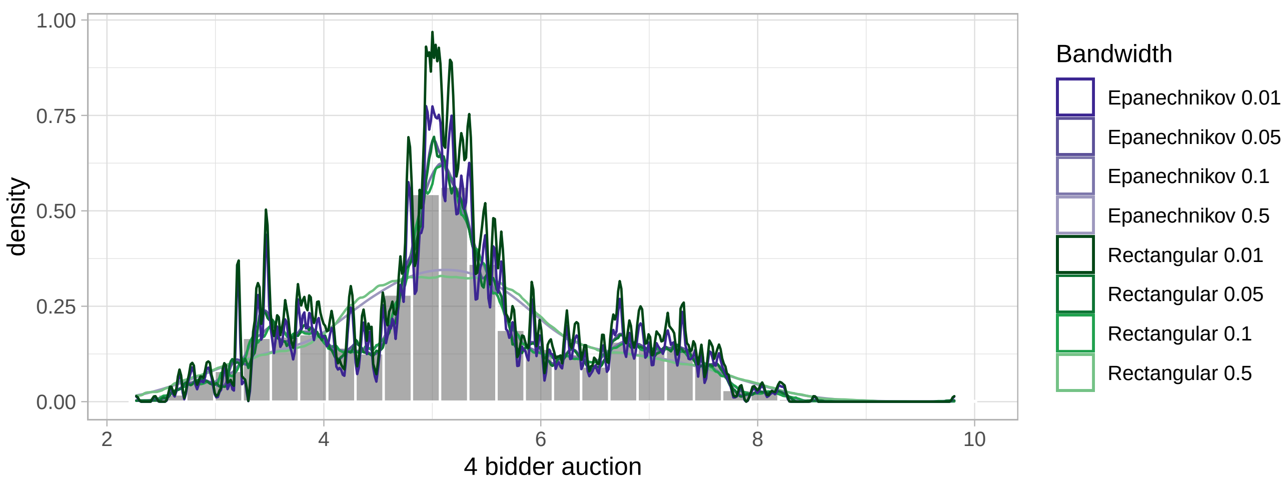

```r
# Quick eyeball for 4 bidder auction
ecdf4 <- ecdf(d4$bid)
dens4 <- density(d4$bid, kernel = "epanechnikov")

par(mfrow = c(1,2))
ecdf4 %>% plot(main = "Empirical CDF", xlab = "4 bidder auction")
plot(dens4, main = "Empirical PDF (Epanechnikov)")
```



```r
# Create ECDF and estimated prices data
n3 <- 3
n4 <- 4

prices3 <- d3 %>%
  mutate(G = ecdf3(bid)) %>%
  arrange(G) %>%
  mutate(g =  approx(dens3, xout = bid)$y) %>%
  mutate(G_int = cumsum(g) / sum(g)) %>%
  mutate(prices = bid + G / ((n3 - 1) * g) ) %>%
  mutate(prices_int = bid + G_int / ((n3 - 1) * g) )

prices4 <- d4 %>%
  mutate(G = ecdf3(bid)) %>%
  arrange(G) %>%
  mutate(g =  approx(dens4, xout = bid)$y) %>%
  mutate(G_int = cumsum(g) / sum(g)) %>%
  mutate(prices = bid + G / ((n4 - 1) * g) ) %>%
  mutate(prices_int = bid + G_int / ((n4 - 1) * g) )
# Set up histogram & density plot
bws <- c(0.5, 0.1, 0.05, 0.01)

epa4 <- colorspace::sequential_hcl(7, palette = "Purples 2")
rec4 <- colorspace::sequential_hcl(7, palette = "Greens 3")

p <- ggplot(prices3, aes(bid)) +
  geom_histogram(alpha = 0.5, color = 'white', aes(y = ..density..)) +
  labs(x = "3 bidder auction",
       color = "Bandwidth") +
    scale_colour_manual(values = c(epa4[1:4], rec4[1:4]))

walk(bws, function(x) {
  p <<- p +
    geom_density(kernel = "epanechnikov", bw = x, aes(color = paste("Epanechnikov", x))) +
    geom_density(kernel = "rectangular", bw = x, aes(color = paste("Rectangular", x)))
  }
)

p


# Set up histogram & density plot
p <- ggplot(prices4, aes(bid)) +
  geom_histogram(alpha = 0.5, color = 'white', aes(y = ..density..)) +
```



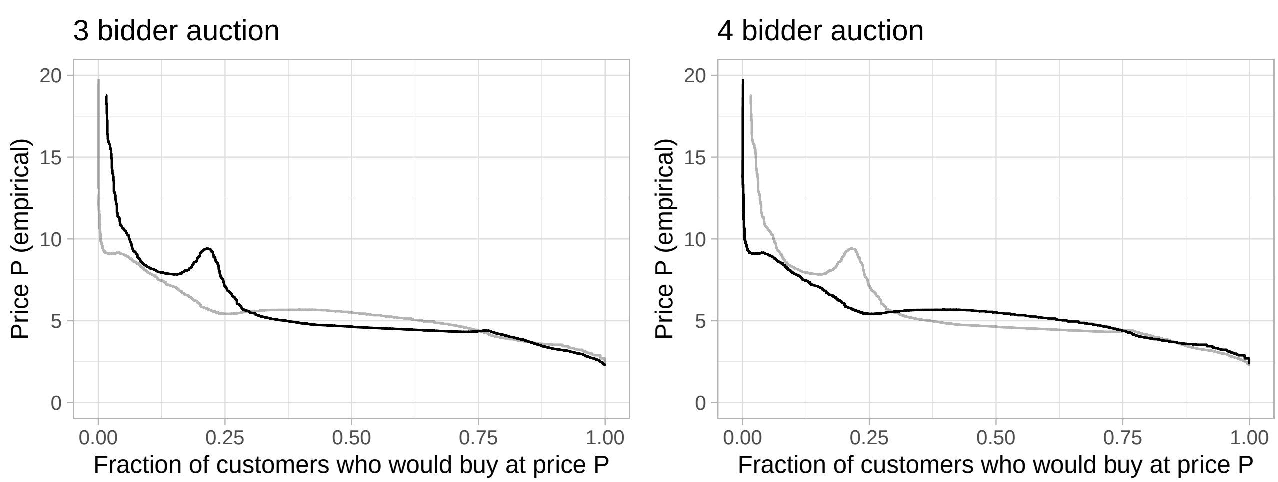

```
  labs(x = "4 bidder auction",
       color = "Bandwidth") +
    scale_colour_manual(values = c(epa4[1:4], rec4[1:4]))

walk(bws, function(x) {
  p <<- p +
    geom_density(kernel = "epanechnikov", bw = x, aes(color = paste("Epanechnikov", x))) +
    geom_density(kernel = "rectangular", bw = x, aes(color = paste("Rectangular", x)))
  }
)

p
```



The bandwidth is much more important than the choice of kernel in this example. For example, $bw = 0.5$ leads to a flat curve, whereas

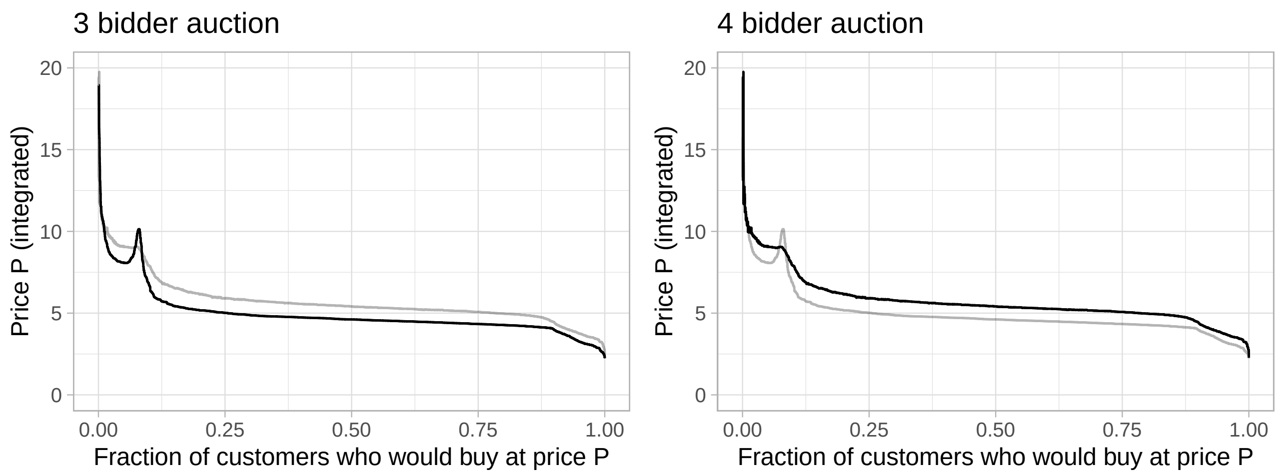
$bw = 0.01$ is very jagged. The optimal choice is somewhere in between.

```
# Demand curves
qp3 <- ggplot(prices3) +
  geom_step(aes(x = 1 - G, y = prices)) +
  labs(x = "Fraction of customers who would buy at price P",
       y = "Price P (empirical)",
       title = "3 bidder auction") +
  geom_step(data = prices4, aes(x = 1 - G, y = prices) , alpha = 0.3) + ylim(c(0, 20))

qp4 <- ggplot(prices4) +
  geom_step(aes(x = 1 - G, y = prices)) +
  labs(x = "Fraction of customers who would buy at price P",
       y = "Price P (empirical)",
       title = "4 bidder auction") +
  geom_step(data = prices3, aes(x = 1 - G, y = prices) , alpha = 0.3) + ylim(c(0, 20))

gridExtra::grid.arrange(qp3, qp4, nrow = 1)
```




```
# Demand curves
qp3i <- ggplot(prices3) +
  geom_step(aes(x = 1 - G_int, y = prices_int)) +
  labs(x = "Fraction of customers who would buy at price P",
       y = "Price P (integrated)",
       title = "3 bidder auction") +
  geom_step(data = prices4, aes(x = 1 - G_int, y = prices_int) , alpha = 0.3) +
  ylim(c(0, 20))

qp4i <- ggplot(prices4) +
```

```
  geom_step(aes(x = 1 - G_int, y = prices_int)) +
  labs(x = "Fraction of customers who would buy at price P",
       y = "Price P (integrated)",
       title = "4 bidder auction") +
  geom_step(data = prices3, aes(x = 1 - G_int, y = prices_int) , alpha = 0.3) +
  ylim(c(0, 20))

gridExtra::grid.arrange(qp3i, qp4i, nrow = 1)
```



The integrated price estimate seems to have a steeper start, which could be a desirable feature. The demand curves between 3 and 4 auction bidding appear to be similar.

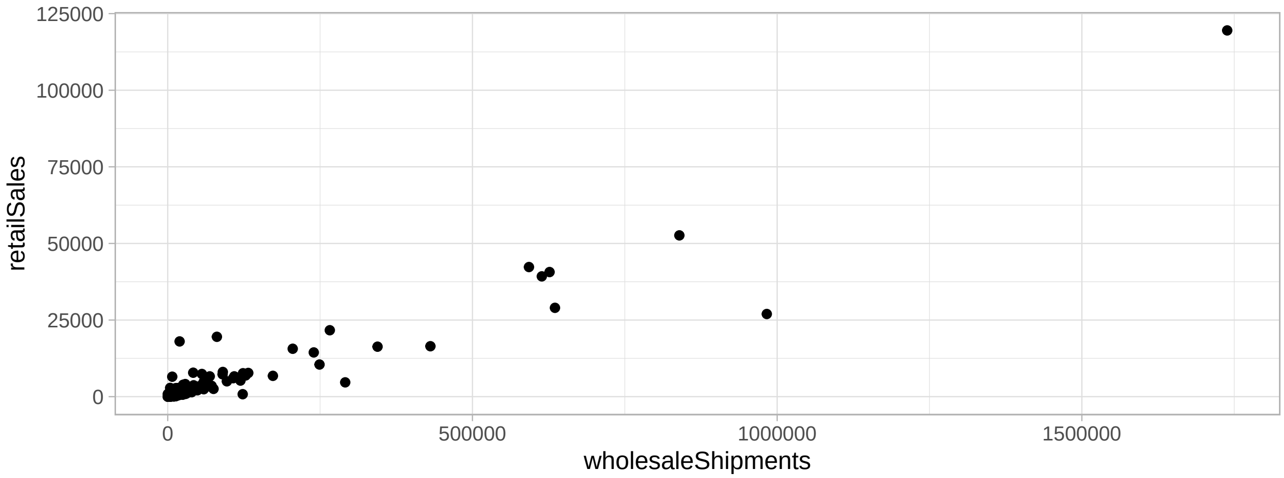
```
# Two-sample Kolmogorov-Smirnov test
ks.test(prices3$prices, prices4$prices)

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  prices3$prices and prices4$prices
## D = 0.48086, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

The test statistic compares the distance between the two distributions: $D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)|$. The null-hypothesis assumes that both samples come from the same distribution. The data suggest that this is very unlikely and we reject this hypothesis ($p < 0.001$).

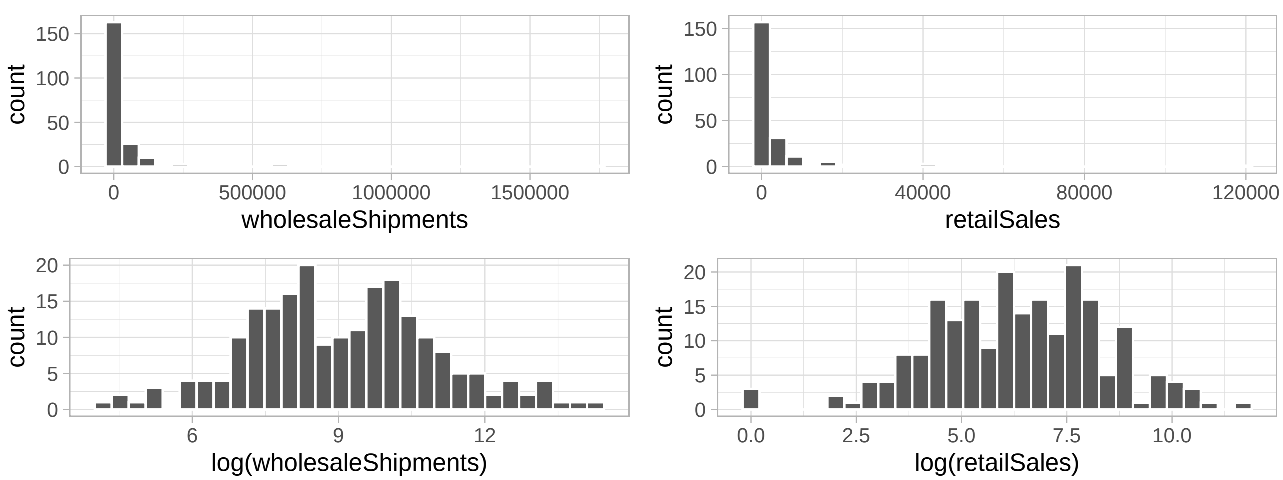
Part 2: Nonparametric Regression

```
d <- read_csv("nonparametric_regression.csv")
```

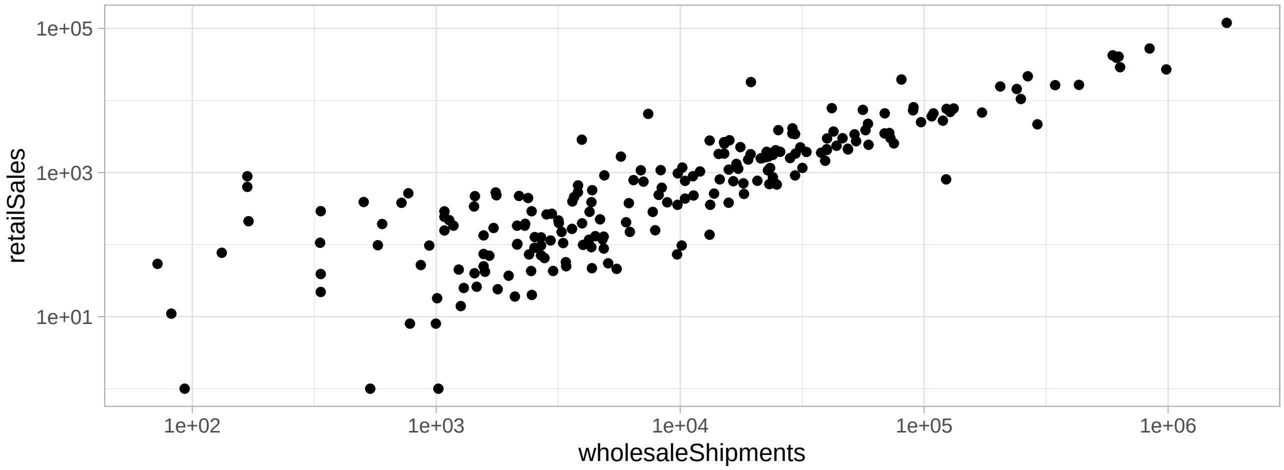
```
ggplot(d) + geom_point(aes(wholesaleShipments, retailSales))
```




```
gridExtra::grid.arrange(
  ggplot(d) + geom_histogram(aes(wholesaleShipments), color = "white"),
  ggplot(d) + geom_histogram(aes(retailSales), color = "white"),
  ggplot(d) + geom_histogram(aes(log(wholesaleShipments)), color = "white"),
  ggplot(d) + geom_histogram(aes(log(retailSales)), color = "white"),
nrow = 2
)
```




```
ggplot(d) + geom_point(aes(wholesaleShipments, retailSales)) +
  scale_x_log10() +
  scale_y_log10()
```




```
m1 <- lm(retailSales ~ wholesaleShipments, data = d)
summary(m1)
```

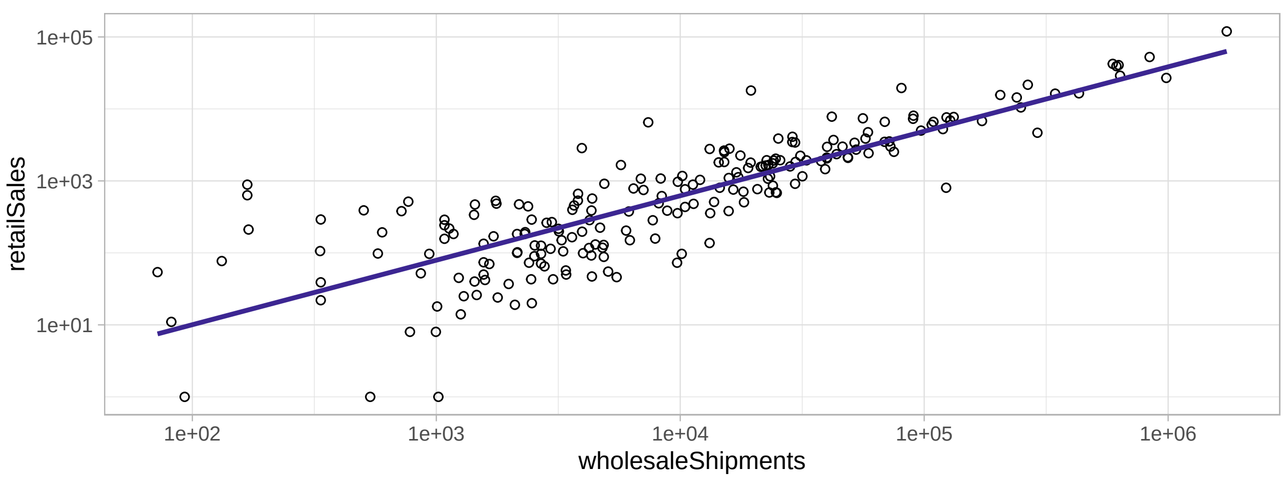
```
##
## Call:
## lm(formula = retailSales ~ wholesaleShipments, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -30611.3   -371.8   -195.1    165.7  17832.0
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        2.102e+02  2.442e+02   0.861     0.39
## wholesaleShipments 5.837e-02  1.342e-03  43.506   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3399 on 212 degrees of freedom
## Multiple R-squared:  0.8993, Adjusted R-squared:  0.8988
## F-statistic:  1893 on 1 and 212 DF,  p-value: < 2.2e-16
```

```
m2 <- lm(log(retailSales) ~ log(wholesaleShipments), data = d)
summary(m2)
```

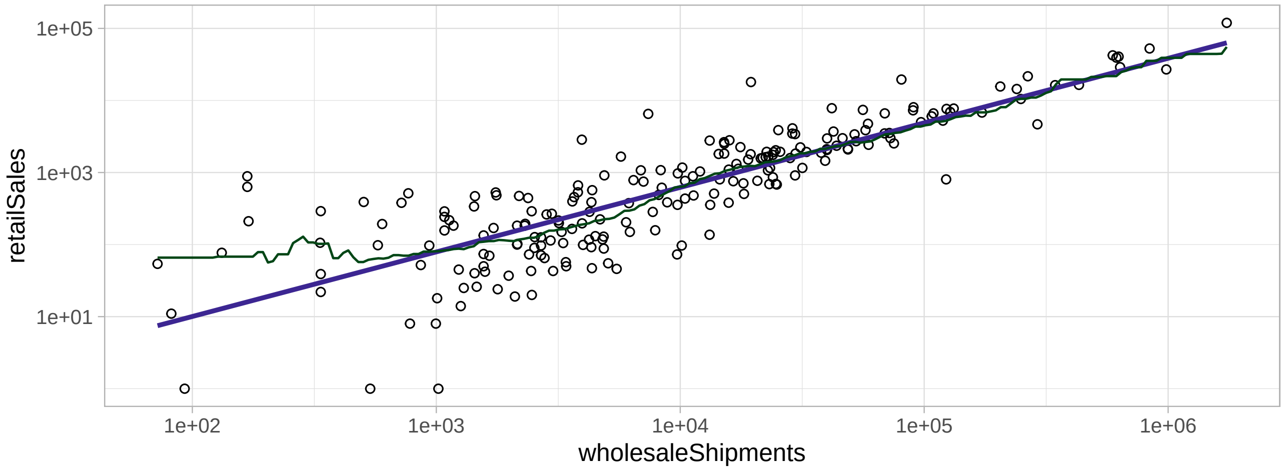
```
##
## Call:
## lm(formula = log(retailSales) ~ log(wholesaleShipments), data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -4.3880 -0.5657  0.1003  0.5810  4.0166
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -1.81750    0.36926  -4.922 1.72e-06 ***
## log(wholesaleShipments)  0.89577    0.03973  22.544  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.114 on 212 degrees of freedom
## Multiple R-squared:  0.7057, Adjusted R-squared:  0.7043
## F-statistic: 508.2 on 1 and 212 DF,  p-value: < 2.2e-16
```

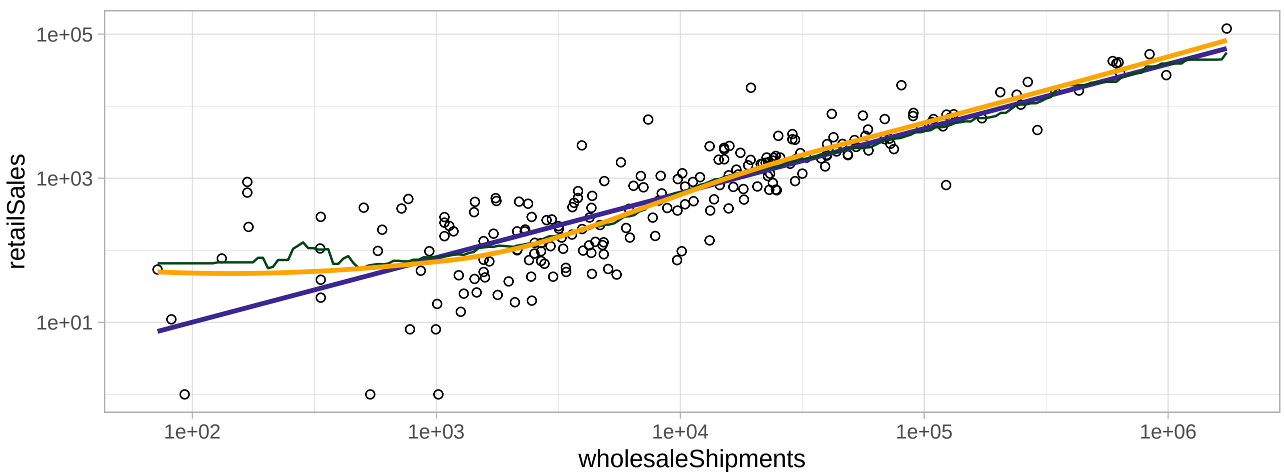
```
# Add kernel regression line to plot
kp <- ggplot(d, aes(wholesaleShipments, retailSales)) +
  geom_point(shape = 1) +
  geom_smooth(method = "lm", se = FALSE, color = epa4[1]) +
  scale_x_log10() +
  scale_y_log10()

kp
```




```
ks <- ksmooth(log(d$wholesaleShipments), log(d$retailSales),
              bandwidth = 2)
ks <- tibble(x = ks$x, y = ks$y)

kp + geom_line(data = ks, aes(exp(x), exp(y)), color = rec4[1])

# Add loess regression line to plot
kp +
```



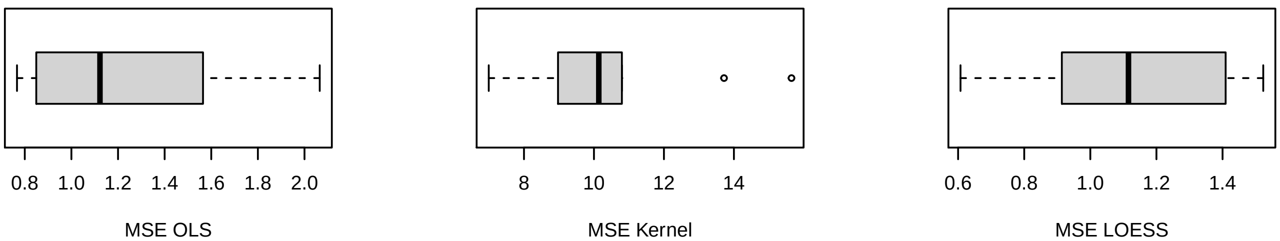

```
geom_line(data = ks, aes(exp(x), exp(y)), color = rec4[1]) +
geom_smooth(method = "loess", color = 'orange', se = F)
```




```
# Create the different models
require(caret)

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
flds <- createFolds(d$prod_id, k = 10, list = TRUE, returnTrain = FALSE)

ols_m <- map(flds, function(x) {
  train <- filter(d, !(prod_id %in% x))
  lm(log(retailSales) ~ log(wholesaleShipments), train)
  })

kernel_m <- map(flds, function(x) {
  train <- filter(d, !(prod_id %in% x))
  ksmooth(log(train$wholesaleShipments), log(train$retailSales),
          bandwidth = 2)
  })

loess_m <- map(flds, function(x) {
  train <- filter(d, !(prod_id %in% x))
  loess(log(retailSales) ~ log(wholesaleShipments), train)
  })

# Mean Squared Error
mse <- function(x) mean(x^2, na.rm = T)

test_ols <- map_dbl(flds, function(x) {
  test <- filter(d, prod_id %in% x)
  y <- log(test$retailSales)
  f <- names(flds)
  res <- y - unlist(predict(ols_m[f], test))
  mse(res)
  })

test_kernel <- map_dbl(flds, function(x) {
  test <- filter(d, prod_id %in% x)
  train <- filter(d, !(prod_id %in% x))
  y <- log(test$retailSales)
  y_hat <- ksmooth(log(train$wholesaleShipments), log(train$retailSales), bandwidth = 2, x.points = y) %>%
  res <- y - y_hat
  mse(res)
  })

test_loess <- map_dbl(flds, function(x) {
  test <- filter(d, prod_id %in% x)
  y <- log(test$retailSales)
  f <- names(flds)
  res <- y - unlist(predict(loess_m[f], test))
  mse(res)
```

```
  })


par(mfrow = c(1,3))
boxplot(test_ols, horizontal = T, xlab = "MSE OLS")
boxplot(test_kernel, horizontal = T, xlab = "MSE Kernel")
boxplot(test_loess, horizontal = T, xlab = "MSE LOESS")
```




```
mean(test_ols)
## [1] 1.233286
mean(test_kernel)
## [1] 10.32166
mean(test_loess)
## [1] 1.108219
```

The lowest MSE appears to be in the local-linear regression. The kernel function I'm using sometimes produces NAs. Its MSE is way above the other two methods, too. This showcases the bias-variance trade-off in prediction.

```
# Try out different bandwidths
kbws <- seq(0.01, 1,  length.out = 10)

kernel_bws <- map2_dbl(flds, kbws, function(x, bw) {
  test <- filter(d, prod_id %in% x)
  train <- filter(d, !(prod_id %in% x))
  y <- log(test$retailSales)
  y_hat <- ksmooth(log(train$wholesaleShipments), log(train$retailSales), bandwidth = bw, x.points = y) %>
  res <- y - y_hat
  mse(res)
  })

cat("Optimal bandwith based on MSE between 0.1 and 1: bw = ",
    kbws[which.min(kernel_bws)])
## Optimal bandwith based on MSE between 0.1 and 1: bw =  0.34
```