Assignment 9: Auction

Kohei Kawaguchi

2019/1/29

Simulate data

We simulate bid data from a second- and first-price sealed bid auctions.

First, we draw bid data from a second-price sealed bid auctions. Suppose that for each auction $t = 1, \cdots, T$, there are $i = 1, \cdots, n_t$ potential bidders. At each auction, an auctioneer allocates one item and sets the reserve price at r_t. When the signal for bidder i in auction t is x_{it}, her expected value of the item is x_{it}. A signal x_{it} is drawn from an i.i.d. beta distribution $B(\alpha, \beta)$. The distribution of the reserve prices is $B(\gamma, \delta)$. n_t is drawn from a Poisson distribution with mean λ. An equilibrium strategy is such that a bidder participates and bids $\beta(x) = x$ if $x \geq r_t$ and does not participate otherwise.

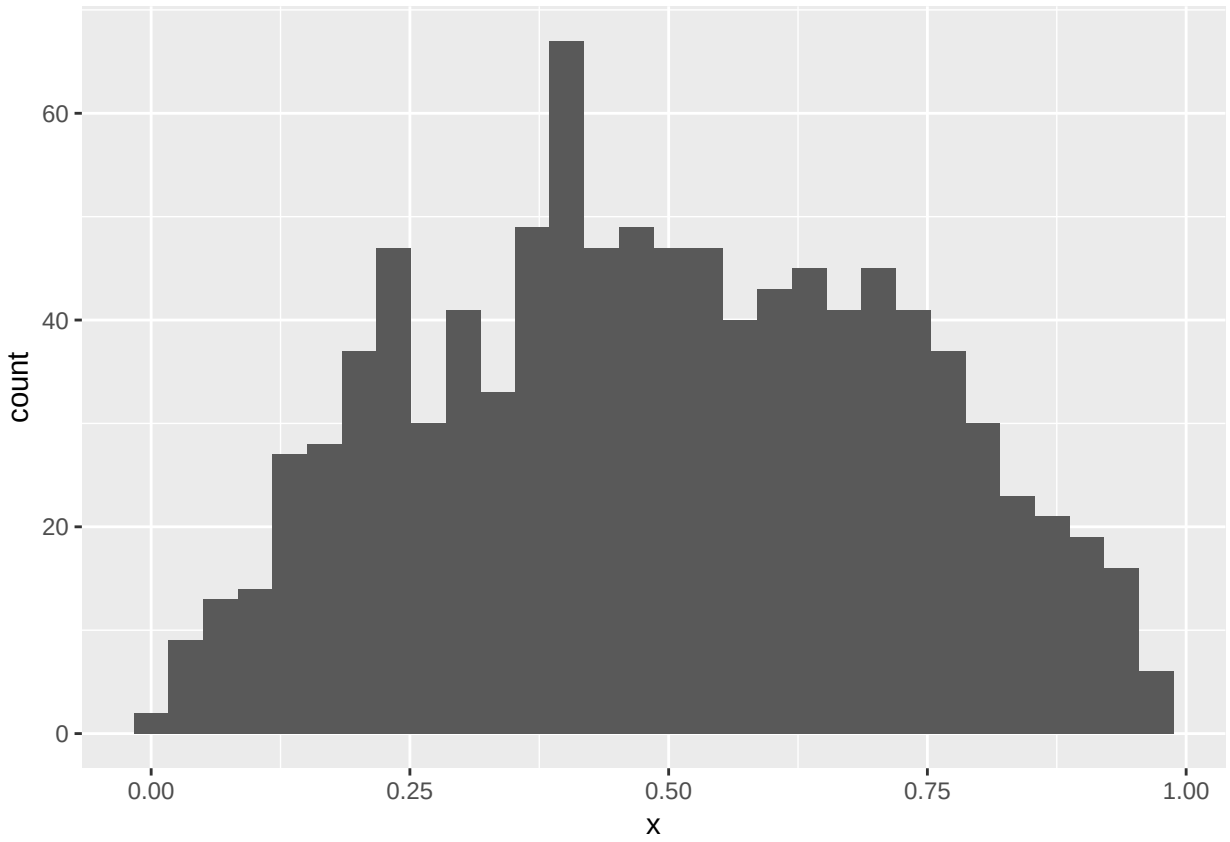
1. Set the constants and parameters as follows:

```
# set seed
set.seed(1)
# number of auctions
T <- 100
# parameter of value distribution
alpha <- 2
beta <- 2
# parameters of reserve price distribution
gamma <- 1
delta <- 3
# prameters of number of potential bidders
lambda <- 10
```

2. Draw a vector of valuations and reservation prices.

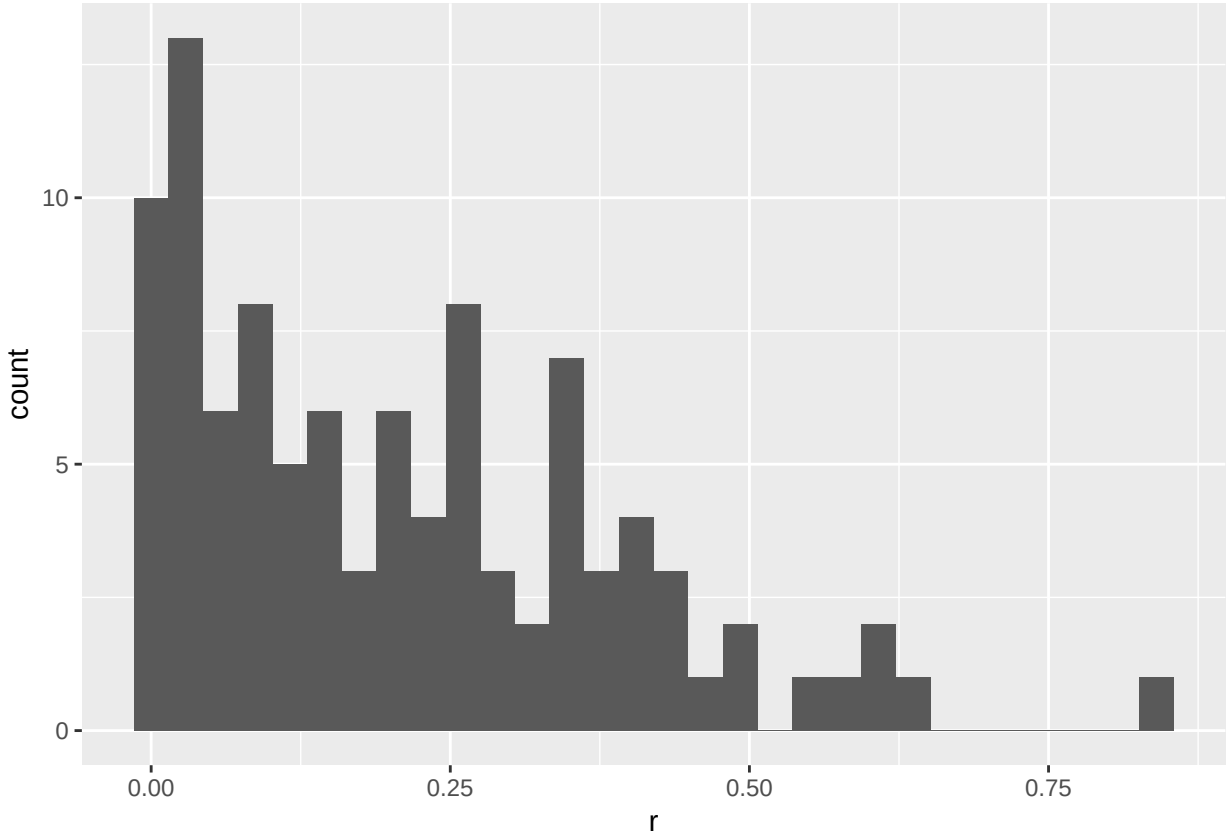
```
# number of bidders
N <- rpois(T, lambda)
# draw valuations
valuation <-
  foreach (tt = 1:T, .combine = "rbind") %do% {
    n_t <- N[tt]
    header <- expand.grid(t = tt, i = 1:n_t)
    return(header)
  }
valuation <- valuation %>%
  tibble::as_tibble() %>%
  dplyr::mutate(x = rbeta(length(i), alpha, beta))
ggplot(valuation, aes(x = x)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# draw reserve prices
reserve <- rbeta(T, gamma, delta)
reserve <- tibble::tibble(t = 1:T, r = reserve)
ggplot(reserve, aes(x = r)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



3. Write a function `compute_winning_bids_second(valuation, reserve)` that returns a winning bid from each second-price auction. It returns nothing for an auction in which no bid was above the reserve price. In the output, `t` refers to the auction index, `m` to the number of actual bidders, `r` to the reserve price, and `w` to the winning bid.

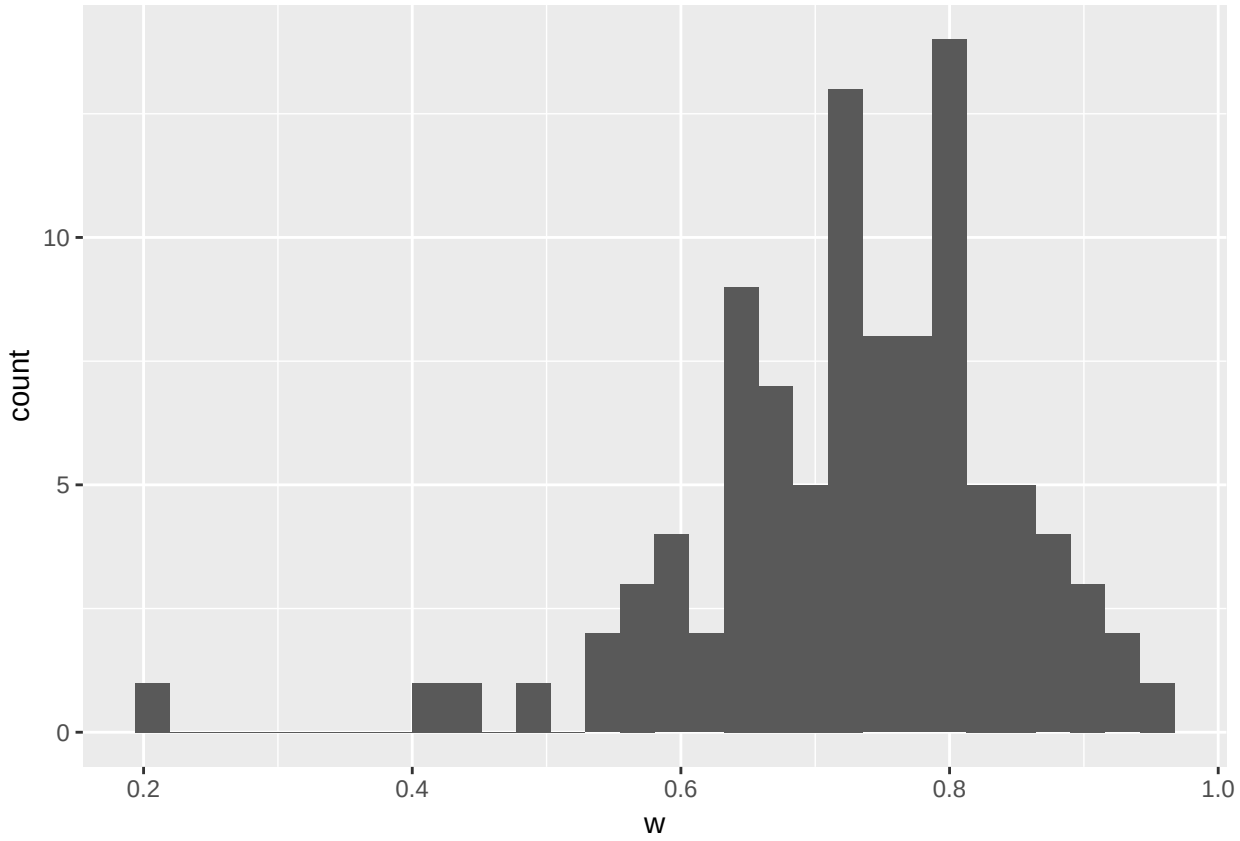
```
compute_winning_bids_second <-
  function(valuation, reserve) {
    df_second_w <- valuation %>%
      dplyr::left_join(reserve, by = "t") %>%
      dplyr::group_by(t) %>%
      # compute the number of potential bidders
      dplyr::mutate(n = length(i)) %>%
      # drop inactive bidders
      dplyr::filter(x >= r) %>%
      # compute the number of actual bidders
      dplyr::mutate(m = length(i)) %>%
      # rank the bids
      dplyr::mutate(y = dplyr::dense_rank(-x)) %>%
      # drop inactive auctions
      dplyr::filter(m >= 1) %>%
      # keep the winning bids
      dplyr::filter(y == 2 | (y == 1 & m == 1)) %>%
      # when there is one bidder, the winnig bid is equal to the reserve price
      dplyr::mutate(x = ifelse(m == 1, r, x)) %>%
      dplyr::ungroup() %>%
      # rename
      dplyr::rename(w = x) %>%
      dplyr::select(t, n, m, r, w)
```

```
    return(df_second_w)
}
df_second_w <-
  compute_winning_bids_second(valuation, reserve)
df_second_w
```

```
## # A tibble: 99 x 5
##        t     n     m       r     w
##    <int> <int> <int>   <dbl> <dbl>
##  1     1     8     8 0.00486 0.637
##  2     2    10    10 0.254   0.647
##  3     3     7     4 0.352   0.484
##  4     4    11     5 0.462   0.804
##  5     5    14    14 0.0861  0.920
##  6     6    12    12 0.00593 0.942
##  7     7    11    10 0.147   0.810
##  8     8     9     5 0.336   0.724
##  9     9    14     9 0.545   0.880
## 10    10    11    11 0.0470  0.677
## # ... with 89 more rows
```

```
ggplot(df_second_w, aes(x = w)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Next, we simulate bid data from first-price sealed bid auctions. The setting is the same as the second-price

auctions expect for the auction rule. An equilibrium bidding strategy is to participate and bid:

$$\beta(x) = x - \frac{\int_{r_t}^{x} B(t;\alpha,\beta)^{N-1}}{B(x;\alpha,\beta)^{N-1}},$$

if $x \geq r$ and not to participate otherwise when there are more than one active bidder. If $x \geq r$ and there is only one bidder, the bidder bids r_t.

4. Write a function `bid_first(x, r, alpha, beta, n, m)` that returns the equilibrium bid. To integrate a function, use `integrate` function in R. It returns `NA` if $x < r$.

```
bid_first <-
  function(x, r, alpha, beta, n, m) {
    if (x >= r) {
      if (m > 1) {
        f <- function(t) {pbeta(t, alpha, beta)^{n - 1}}
        numerator <- integrate(f, r, x)$value
        denominator <- f(x)
        b <- x - numerator/denominator
      } else {
        b <- r
      }
    } else {
      b <- NA
    }
  return(b)
  }
n <- N[1]
m <- N[1]
x <- valuation[1, "x"] %>% as.numeric(); x
```

```
## [1] 0.3902289
```

```
r <- reserve[1, "r"] %>% as.numeric(); r
```

```
## [1] 0.00485617
```

```
b <- bid_first(x, r, alpha, beta, n, m); b
```

```
## [1] 0.3596625
```

```
b <- bid_first(x, r, alpha, beta, n, 1); b
```

```
## [1] 0.00485617
```

```
x <- r/2; x
```

```
## [1] 0.002428085
```

```
b <- bid_first(x, r, alpha, beta, n, m); b
```

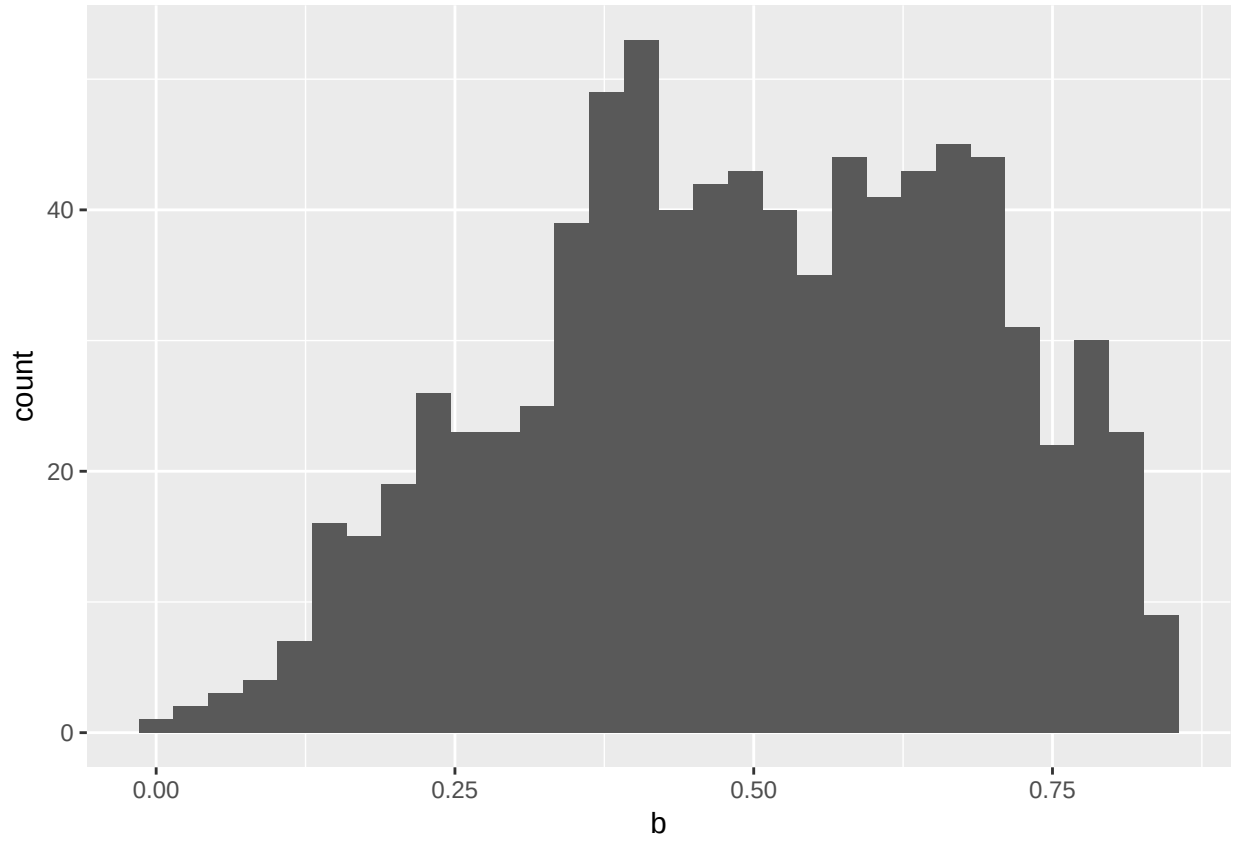
```
## [1] NA
```

5. Write a function `compute_bids_first(valuation, reserve, alpha, beta)` that returns bids from each first-price auctions. It returns nothing for an auction in which no bid was above the reserve price.

```
compute_bids_first <-
  function(valuation, reserve, alpha, beta) {
    df_first <-
      valuation %>%
      dplyr::left_join(reserve, by = "t") %>%
```

```
      dplyr::group_by(t) %>%
      # number of potential bidders
      dplyr::mutate(n = length(i)) %>%
      # drop inactive bidders
      dplyr::filter(x >= r) %>%
      # number of active bidders
      dplyr::mutate(m = length(i)) %>%
      # drop inactive auctions
      dplyr::filter(m >= 1) %>%
      dplyr::ungroup() %>%
      # draw bids
      dplyr::rowwise() %>%
      dplyr::mutate(b = bid_first(x, r, alpha, beta, n, m)) %>%
      dplyr::ungroup()
    return(df_first)
  }
df_first <- compute_bids_first(valuation, reserve, alpha, beta)
ggplot(df_first, aes(x = b)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



6. Write a function `compute_winning_bids_first(valuation, reserve, alpha, beta)` that returns only the winning bids from each first-price auction. It will call `compute_bids_first` inside the function.

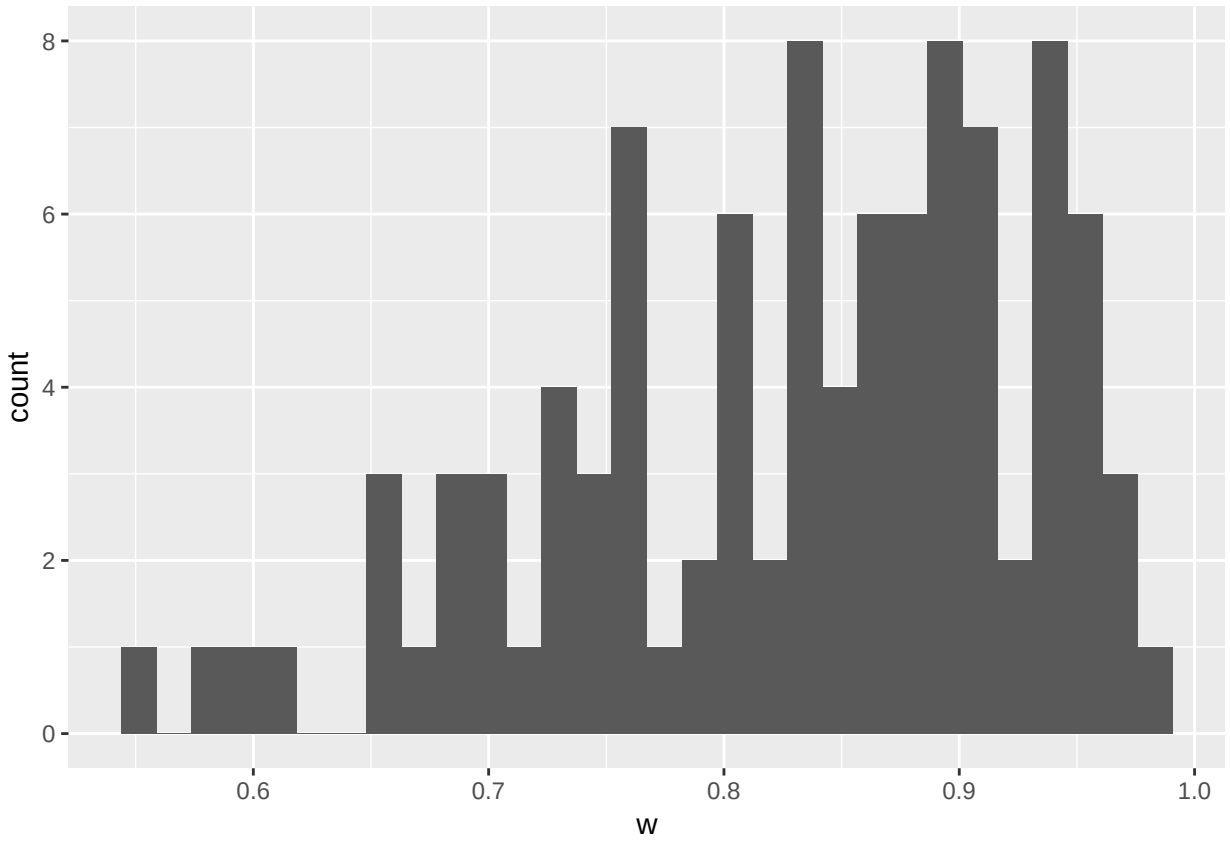
```
compute_winning_bids_first <-
  function(valuation, reserve, alpha, beta) {
    # compute bids
    df_first <- compute_bids_first(valuation, reserve, alpha, beta)
```

```
    # keep only winning bids
    df_first_w <- df_first %>%
      dplyr::group_by(t) %>%
      # keep the winner
      dplyr::mutate(y = dplyr::dense_rank(-x)) %>%
      dplyr::filter(y == 1) %>%
      dplyr::ungroup() %>%
      dplyr::rename(w = x) %>%
      dplyr::select(t, n, m, r, w)
    return(df_first_w)
  }
df_first_w <-
  compute_winning_bids_first(valuation, reserve, alpha, beta)
df_first_w
```

```
## # A tibble: 99 x 5
##        t     n     m       r     w
##    <int> <int> <int>   <dbl> <dbl>
##  1     1     8     8 0.00486 0.837
##  2     2    10    10 0.254   0.695
##  3     3     7     4 0.352   0.586
##  4     4    11     5 0.462   0.943
##  5     5    14    14 0.0861  0.954
##  6     6    12    12 0.00593 0.961
##  7     7    11    10 0.147   0.860
##  8     8     9     5 0.336   0.856
##  9     9    14     9 0.545   0.973
## 10    10    11    11 0.0470  0.911
## # ... with 89 more rows
```

```
ggplot(df_first_w, aes(x = w)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Problems

1. The data set `data_ps7_second.csv` contains a series of winning bids of 100 sealed-bid second price auctions. In each round, there were 10 bidders. Assume that the bidders play a dominant strategy equilibrium in each round and assume that the private value of bidder i in auction t is drawn from a beta distribution of shape parameters $(\alpha, \beta), B(\alpha, \beta)$, and that they are identical and independent across bidders and auctions. Estimate parameters (α, β) following the steps below:

    1. Estimate the empirical distribution of winning bids and let $\hat{F}_w$ denote it. Plot the histogram.
    2. Pick up 10 points on the support of winning bids, say, 0, 10, $\cdots$, 90, 100 percentiles and let $\{w_j\}_{j=0}^{100}$ denote it.
    3. Write a function to compute the predicted distribution of winning bids:

    $$\varphi(B(w;\alpha,\beta),N) = N(N-1)\int_0^{B(w;\alpha,\beta)} u^{N-2}(1-u)du, \tag{1}$$

    at $\{w_j\}_{j=0}^{100}$. To numerically compute the integral of a function, you can use `integrate` function build in R.
    4. Write a function that computes the objective function such that:

    $$\frac{1}{10}\sum_{j=0}^{100}[\hat{F}_w(w_j) - \varphi(B(w_j;\alpha,\beta),N)]^2, \tag{2}$$

    as a function of model parameters (α, β).
    5. Then, find paremters (α, β) that minimize the objective function.

2. The data set `data_ps7_first.csv` contains a series of bids of 100 sealed-bid first price auctions. In each round, there were 10 bidders. Suppose that the equilibrium strategy of bidders are characterized as in the lecture slide. Estimate the distribution of bidder's private values nonparametrically by following the steps below:

    1. Compute the kernel density and kernel cumulative distribution functions of bids, f_b and F_b. For this, you can use `kde` and `kcde` functions in package `ks`. Let $\hat{f}_b$ and $\hat{F}_b$ denote the estimated functions.
    2. For each bid, compute the values $\hat{f}_b(b_i^t)$ and $\hat{F}_b(b_i^t)$.
    3. Use the first order condition:

    $$v_i^t = b_i^t + \frac{\hat{F}_b(b_i^t)}{(N-1)\hat{f}_b(b_i^t)}, \tag{3}$$

    to compute the implied bidder's values.
    4. Plot the empirical cumulative distribution of the implied bidder's values.