Reproducing Tables 2 and 3 in Zhang Xu and Hua

```
library(tidyverse)
library(alloscore)
```

In this vignette we reproduce the results which are transcribed from tables 2 and 3 in (Zhang, Xu, and Hua 2009) into `zxh_tab2` and `zxh_tab3`:

```
zxh_tab2
#>    Product  v h  c  mu sigma  q_zxh    GIM    Opt
#> 1        1  7 1  4 102  51.0  85.75   0.00   0.00
#> 2        2 12 2  8  73  18.3  62.64   0.00   0.00
#> 3        3 30 4 19 123  30.8 108.90   0.00   0.00
#> 4        4 30 4 17  95  23.8  87.88   0.00   0.00
#> 5        5 40 2 23  62  15.5  58.26   0.00   0.00
#> 6        6 45 5 15 129  43.0 139.89 106.86 106.85
#> 7        7 16 1 10  69  34.5  55.98   0.00   0.00
#> 8        8 21 2 10  83  41.5  80.74  14.02  14.01
#> 9        9 42 3 40 120  30.0  68.96   0.00   0.00
#> 10      10 34 5 20  89  22.3  80.95   0.00   0.00
#> 11      11 20 3 10 115  38.3 108.71  15.58  15.65
#> 12      12 15 5  7  91  30.3  83.32  42.20  42.25
#> 13      13 10 3  4  52  17.3  50.33  34.56  34.60
#> 14      14 20 3 12  76  38.0  61.14   0.00   0.00
#> 15      15 47 2 33  66  16.5  56.66   0.00   0.00
#> 16      16 35 4 21 147  36.8 133.71   0.00   0.00
#> 17      17 22 1 11 104  34.7 102.11  15.23  15.13
zxh_tab3
#>   Product  v h  c x_min x_max Balpha Bbeta  q_zxh    GIM    Opt
#> 1       1  7 1  4   100   300    2.0   1.0 222.47 206.83 207.93
#> 2       2 12 2  7    50   250    1.0   1.2 111.60  95.69  96.73
#> 3       3 30 4 15    75   150    1.0   2.0  93.93  90.10  90.34
#> 4       4 17 3 10    50   200    2.0   2.0 109.79 100.12 100.78
#> 5       5 27 5 15    50   200    2.0   3.0  97.26  90.07  90.55
#> 6       6 10 2  6    73   275    0.8   0.2 239.02 209.35 211.69
```

Column descriptions are in the data set help files.

We begin by converting their "newsvendor" parameters to the `kappa` and `alpha` parameters we use in `alloscore`:

```
(zxh_norm_ex <- zxh_tab2 %>%
     mutate(
     stdize_news_params(ax = c, a_minus = v, a_plus = h),
     .after = c) %>% as_tibble())
#> # A tibble: 17 x 11
#>    Product     v     h     c kappa  alpha    mu sigma q_zxh   GIM   Opt
#>    <chr>   <int> <int> <int> <dbl>  <dbl> <int> <dbl> <dbl> <dbl> <dbl>
#>  1 1           7     1     4     8 0.375    102  51    85.8   0     0
#>  2 2          12     2     8    14 0.286     73  18.3  62.6   0     0
```

```
#>  3 3          30     4    19    34 0.324    123  30.8 109.    0     0
#>  4 4          30     4    17    34 0.382     95  23.8  87.9   0     0
#>  5 5          40     2    23    42 0.405     62  15.5  58.3   0     0
#>  6 6          45     5    15    50 0.6      129  43   140.   107.  107.
#>  7 7          16     1    10    17 0.353     69  34.5  56.0   0     0
#>  8 8          21     2    10    23 0.478     83  41.5  80.7  14.0  14.0
#>  9 9          42     3    40    45 0.0444   120  30    69.0   0     0
#> 10 10         34     5    20    39 0.359     89  22.3  81.0   0     0
#> 11 11         20     3    10    23 0.435    115  38.3 109.   15.6  15.6
#> 12 12         15     5     7    20 0.4       91  30.3  83.3  42.2  42.2
#> 13 13         10     3     4    13 0.462     52  17.3  50.3  34.6  34.6
#> 14 14         20     3    12    23 0.348     76  38    61.1   0     0
#> 15 15         47     2    33    49 0.286     66  16.5  56.7   0     0
#> 16 16         35     4    21    39 0.359    147  36.8 134.    0     0
#> 17 17         22     1    11    23 0.478    104  34.7 102.   15.2  15.1
```

Next we use the convenience function `add_pdqr_funs` to create list columns of forecast cdfs and quantile functions

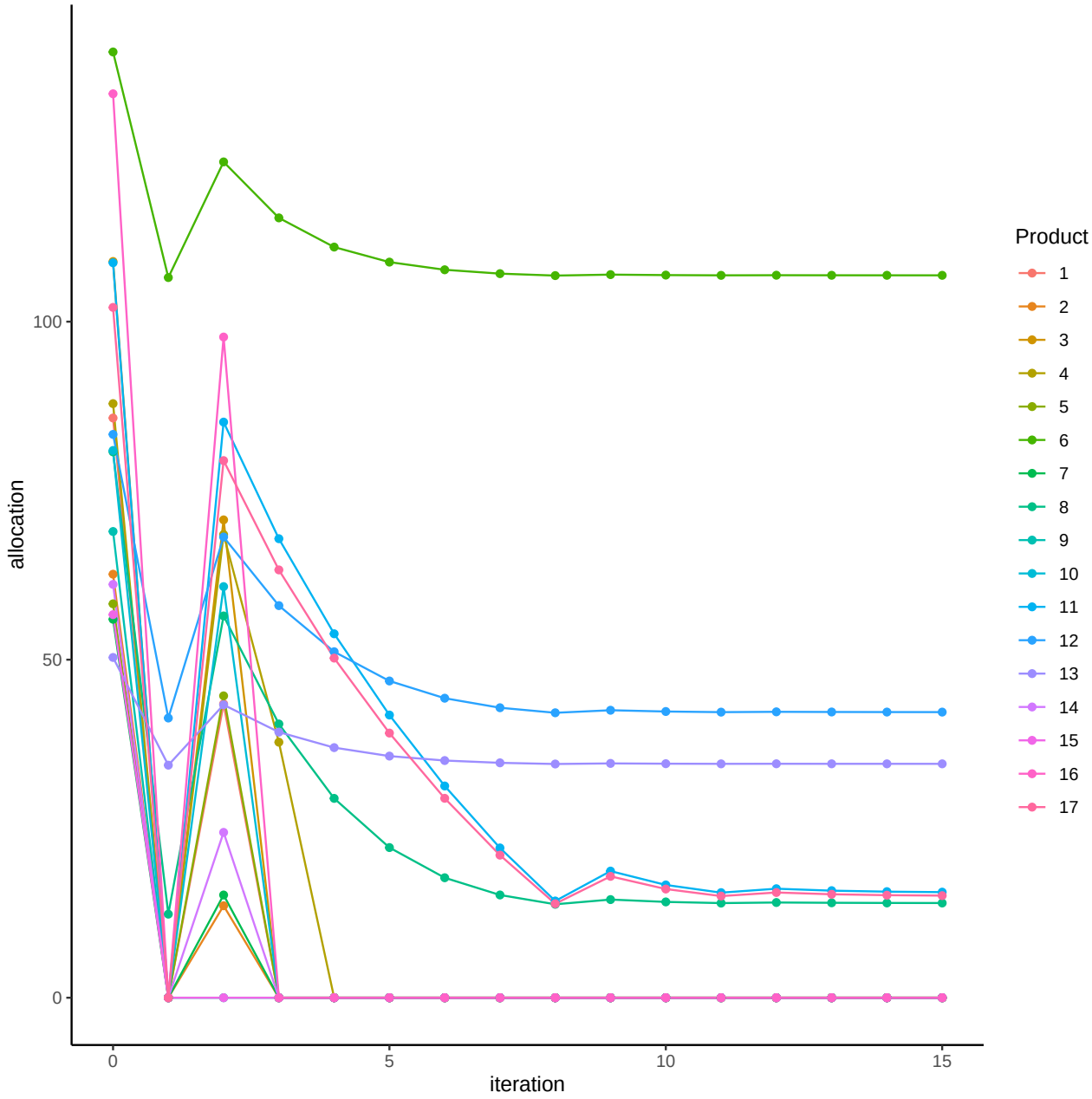
```
zxh_norm_ex_forecasts <- zxh_norm_ex %>%
   rename(mean = mu, sd = sigma) %>%
   add_pdqr_funs(dist = "norm", types = c("p", "q")) %>%
   mutate(q = map2_dbl(Q, alpha, exec), .after = q_zxh)
```

We now allocate with `alloscore::allocate` for the constraint of `K=2500` used for this example by ZXH and compare with their results:

```
allo_norm_ex <- allocate(zxh_norm_ex_forecasts, w = "c", K = 2500, target_names = "Product")
(zxh_norm_ex_allo <- full_join(zxh_norm_ex_forecasts, allo_norm_ex$xdf[[1]], by = "Product") %>%
  relocate(Opt, x))
#> # A tibble: 17 x 16
#>      Opt     x Product     v     h     c kappa  alpha  mean    sd q_zxh     q   GIM dist  F
#>    <dbl> <dbl> <chr>   <int> <int> <int> <dbl>  <dbl> <int> <dbl> <dbl> <dbl> <dbl> <chr> <list>
#>  1   0     0   1           7     1     4     8 0.375    102  51    85.8  85.7   0   norm  <prrr_fn_>
#>  2   0     0   2          12     2     8    14 0.286     73  18.3  62.6  62.6   0   norm  <prrr_fn_>
#>  3   0     0   3          30     4    19    34 0.324    123  30.8 109.  109.    0   norm  <prrr_fn_>
#>  4   0     0   4          30     4    17    34 0.382     95  23.8  87.9  87.9   0   norm  <prrr_fn_>
#>  5   0     0   5          40     2    23    42 0.405     62  15.5  58.3  58.3   0   norm  <prrr_fn_>
#>  6 107.  107.  6          45     5    15    50 0.6      129  43   140.  140.  107.  norm  <prrr_fn_>
#>  7   0     0   7          16     1    10    17 0.353     69  34.5  56.0  56.0   0   norm  <prrr_fn_>
#>  8  14.0  14.0 8          21     2    10    23 0.478     83  41.5  80.7  80.7  14.0 norm  <prrr_fn_>
#>  9   0     0   9          42     3    40    45 0.0444   120  30    69.0  69.0   0   norm  <prrr_fn_>
#> 10   0     0   10         34     5    20    39 0.359     89  22.3  81.0  80.9   0   norm  <prrr_fn_>
#> 11  15.6  15.6 11         20     3    10    23 0.435    115  38.3 109.  109.   15.6 norm  <prrr_fn_>
#> 12  42.2  42.2 12         15     5     7    20 0.4       91  30.3  83.3  83.3  42.2 norm  <prrr_fn_>
#> 13  34.6  34.6 13         10     3     4    13 0.462     52  17.3  50.3  50.3  34.6 norm  <prrr_fn_>
#> 14   0     0   14         20     3    12    23 0.348     76  38    61.1  61.1   0   norm  <prrr_fn_>
#> 15   0     0   15         47     2    33    49 0.286     66  16.5  56.7  56.7   0   norm  <prrr_fn_>
#> 16   0     0   16         35     4    21    39 0.359    147  36.8 134.  134.    0   norm  <prrr_fn_>
#> 17  15.1  15.1 17         22     1    11    23 0.478    104  34.7 102.  102.   15.2 norm  <prrr_fn_>
#> # i 1 more variable: Q <list>
```

WitH `allocate`'s default convergence parameters `eps_lam = 1e-4` and `eps_K = .01` it took us 15 iterations to find our allocations, the last few of which maybe weren't necessary:

```
allo_norm_ex$xs[[1]] %>% rename(`0` = qs) %>%
  tidyr::pivot_longer(
    cols = -Product,
    names_to = "iteration",
    values_to = "allocation") %>%
  mutate(
    iteration = as.numeric(iteration),
    Product = as.factor(as.integer(Product))
    ) %>%
  ggplot(aes(x = iteration, y = allocation, color = Product)) +
  geom_line() + geom_point() + theme_classic()
```



Now we compute the components of the objective function which are the expected pinball losses. (`alloscore` could also be used to do this... will add in next draft.)

```
(Zs_norm <- zxh_norm_ex_allo %>% mutate(
  expected_gpl_loss = pmap(., function(F, kappa, alpha, mu, c, ...) {
    exp_gpl_loss_fun(
      F = F,
      kappa = kappa,
      alpha = alpha,
      offset = c*mu)
    })
  ) %>% mutate(
  Z_Opt = map2_dbl(expected_gpl_loss, Opt, exec),
  Z_allo = map2_dbl(expected_gpl_loss, x, exec)
) %>% select(Product, c, Opt, x, Z_Opt, Z_allo))
#> # A tibble: 17 x 6
#>    Product     c   Opt     x Z_Opt Z_allo
#>    <chr>   <int> <dbl> <dbl> <dbl>  <dbl>
#>  1 1           4   0     0    309.   309.
#>  2 2           8   0     0    292.   292.
#>  3 3          19   0     0   1353.  1353.
#>  4 4          17   0     0   1235.  1235.
#>  5 5          23   0     0   1054.  1054.
#>  6 6          15 107.  107.  1080.  1080.
#>  7 7          10   0     0    419.   419.
#>  8 8          10  14.0  14.0  778.   778.
#>  9 9          40   0     0    240.   240.
#> 10 10         20   0     0   1246.  1246.
#> 11 11         10  15.6  15.6  995.   995.
#> 12 12          7  42.2  42.2  404.   404.
#> 13 13          4  34.6  34.6  123.   123.
#> 14 14         12   0     0    615.   615.
#> 15 15         33   0     0    924.   924.
#> 16 16         21   0     0   2058.  2058.
#> 17 17         11  15.1  15.1  979.   979.
```

And compare our objective function values:

```
Zs_norm %>% summarise(
  Z_Opt = sum(Z_Opt),
  Z_allo = sum(Z_allo),
  cost_zxh = sum(c*Opt),
  cost_allo= sum(c*x)
  ) %>% tidyr::pivot_longer(everything()) %>%
  as.data.frame()
#>        name     value
#> 1     Z_Opt 14104.178
#> 2    Z_allo 14104.832
#> 3  cost_zxh  2499.930
#> 4 cost_allo  2499.269
```

Now repeat the process for their second data set which uses beta distributions with supports bounded away from 0:

```
zxh_tab3
#>   Product  v h  c x_min x_max Balpha Bbeta  q_zxh    GIM    Opt
#> 1       1  7 1  4   100   300    2.0   1.0 222.47 206.83 207.93
#> 2       2 12 2  7    50   250    1.0   1.2 111.60  95.69  96.73
```

```
#> 3       3 30 4 15    75   150    1.0   2.0  93.93  90.10  90.34
#> 4       4 17 3 10    50   200    2.0   2.0 109.79 100.12 100.78
#> 5       5 27 5 15    50   200    2.0   3.0  97.26  90.07  90.55
#> 6       6 10 2  6    73   275    0.8   0.2 239.02 209.35 211.69
```

Reparametrize:

```
(zxh_beta_ex <- zxh_tab3 %>% mutate(
     stdize_news_params(ax = c, a_minus = v, a_plus = h),
     .after = c))
#>   Product  v h  c kappa     alpha x_min x_max Balpha Bbeta  q_zxh    GIM    Opt
#> 1       1  7 1  4     8 0.3750000   100   300    2.0   1.0 222.47 206.83 207.93
#> 2       2 12 2  7    14 0.3571429    50   250    1.0   1.2 111.60  95.69  96.73
#> 3       3 30 4 15    34 0.4411765    75   150    1.0   2.0  93.93  90.10  90.34
#> 4       4 17 3 10    20 0.3500000    50   200    2.0   2.0 109.79 100.12 100.78
#> 5       5 27 5 15    32 0.3750000    50   200    2.0   3.0  97.26  90.07  90.55
#> 6       6 10 2  6    12 0.3333333    73   275    0.8   0.2 239.02 209.35 211.69
```

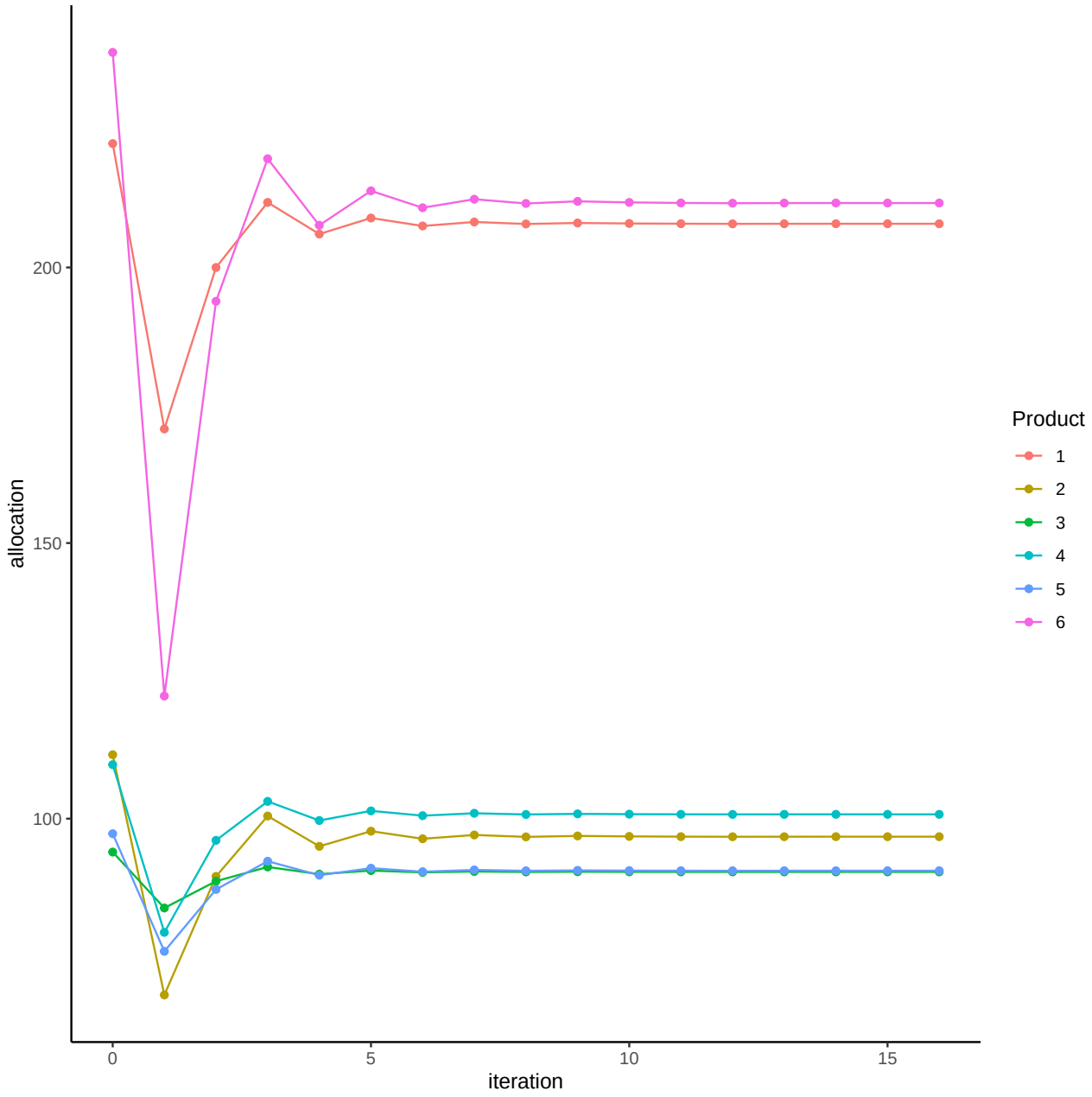
Create distributions and allocate:

```
zxh_beta_ex_forecasts <- zxh_beta_ex %>%
   rename(shape1 = Balpha, shape2 = Bbeta) %>%
   add_pdqr_funs(
     types = c("p", "q"),
     dist = "beta",
     trans = function(x, x_min, x_max) {(x-x_min)/(x_max-x_min)},
     trans_inv = function(q, x_min, x_max) {x_min + (x_max-x_min)*q}) %>%
   mutate(q = map2_dbl(Q, alpha, exec), .after = q_zxh)

allo_beta_ex <- allocate(zxh_beta_ex_forecasts, w = "c", K = 6500, target_names = "Product")
(zxh_beta_ex_allo <- full_join(zxh_beta_ex_forecasts, allo_beta_ex$xdf[[1]], by = "Product") %>%
  relocate(Opt, x))
#> # A tibble: 6 x 20
#>     Opt     x Product     v     h     c kappa alpha x_min x_max shape1 shape2 q_zxh     q   GIM
#>   <dbl> <dbl> <chr>   <int> <int> <int> <dbl> <dbl> <int> <int>  <dbl>  <dbl> <dbl> <dbl> <dbl>
#> 1 208.  208.  1           7     1     4     8 0.375   100   300    2      1   222.  222.  207.
#> 2  96.7  96.7 2          12     2     7    14 0.357    50   250    1      1.2 112.  112.   95.7
#> 3  90.3  90.3 3          30     4    15    34 0.441    75   150    1      2    93.9  93.9  90.1
#> 4 101.  101.  4          17     3    10    20 0.35     50   200    2      2   110.  110.  100.
#> 5  90.6  90.5 5          27     5    15    32 0.375    50   200    2      3    97.3  97.3  90.1
#> 6 212.  212.  6          10     2     6    12 0.333    73   275    0.8    0.2 239.  239.  209.
#> # i 5 more variables: dist <chr>, trans <list>, trans_inv <list>, F <list>, Q <list>
```

Plot iterations:

```
allo_beta_ex$xs[[1]] %>% rename(`0` = qs) %>%
  tidyr::pivot_longer(
    cols = -Product,
    names_to = "iteration",
    values_to = "allocation") %>%
  mutate(
    iteration = as.numeric(iteration),
    Product = as.factor(as.integer(Product))
    ) %>%
  ggplot(aes(x = iteration, y = allocation, color = Product)) +
  geom_line() + geom_point() + theme_classic()
```




```
Zs_beta <- zxh_beta_ex_allo %>% mutate(
  expected_gpl_loss = pmap(., function(F, kappa, alpha, mu, c, ...) {
    exp_gpl_loss_fun(
      F = F,
      kappa = kappa,
      alpha = alpha,
      offset = c*mu)
    })
  ) %>% mutate(
  Z_Opt = map2_dbl(expected_gpl_loss, Opt, exec),
  Z_allo = map2_dbl(expected_gpl_loss, x, exec)
) %>% select(Product, c, Opt, x, Z_Opt, Z_allo)
```

Compare scores:

```r
Zs_beta %>% summarise(
  Z_Opt = sum(Z_Opt),
  Z_allo = sum(Z_allo),
  cost_zxh = sum(c*Opt),
  cost_allo= sum(c*x)
  ) %>% tidyr::pivot_longer(everything()) %>%
  as.data.frame()
#>        name    value
#> 1     Z_Opt 1646.856
#> 2    Z_allo 1646.868
#> 3  cost_zxh 6500.120
#> 4 cost_allo 6500.045
```

Zhang, Bin, Xiaoyan Xu, and Zhongsheng Hua. 2009. "A Binary Solution Method for the Multi-Product Newsboy Problem with Budget Constraint." *International Journal of Production Economics* 117 (1): 136–41.