```
> library(tidyverse)
> library(dslabs)
>
> suppressWarnings(set.seed(1986, sample.kind="Rounding"))
> n <- round(2^rnorm(1000, 8, 1))
>
> suppressWarnings(set.seed(1, sample.kind="Rounding"))
> mu <- round(80 + 2*rt(1000, 5))
> range(mu)
[1] 67 94
> schools <- data.frame(id = paste("PS",1:1000),
+                       size = n,
+                       quality = mu,
+                       rank = rank(-mu))
>
> schools %>% top_n(10, quality) %>% arrange(desc(quality))
       id size quality rank
1  PS 191 1036      94  1.0
2  PS 567  121      93  2.0
3   PS 95  235      91  3.0
4  PS 430   61      90  4.0
5  PS 343   78      89  5.0
6  PS 981  293      88  6.0
7  PS 558  196      87  7.0
8   PS 79  105      86 13.5
9  PS 113  653      86 13.5
10 PS 163  300      86 13.5
11 PS 266 2369      86 13.5
12 PS 400  550      86 13.5
13 PS 451  217      86 13.5
14 PS 477  341      86 13.5
15 PS 484  967      86 13.5
16 PS 561  723      86 13.5
17 PS 563  828      86 13.5
18 PS 865  586      86 13.5
19 PS 963  208      86 13.5
>
> suppressWarnings(set.seed(1, sample.kind="Rounding"))
> mu <- round(80 + 2*rt(1000, 5))
>
> scores <- sapply(1:nrow(schools), function(i){
+       scores <- rnorm(schools$size[i], schools$quality[i], 30)
+       scores
+ })
> schools <- schools %>% mutate(score = sapply(scores, mean))
>
> dim(schools)
[1] 1000    5
> head(schools)
    id size quality  rank    score
1 PS 1  248      79 696.0 82.20965
2 PS 2  311      79 696.0 82.76347
3 PS 3  305      81 330.5 80.10790
4 PS 4  131      79 696.0 74.37327
5 PS 5  360      80 516.5 79.13786
6 PS 6  158      81 330.5 80.35375
> tail(schools)
          id size quality  rank    score
995   PS 995  327      80 516.5 79.50093
996   PS 996  177      81 330.5 80.85696
997   PS 997  293      80 516.5 76.44958
998   PS 998   90      80 516.5 85.47052
999   PS 999  234      80 516.5 81.68251
1000 PS 1000  654      82 194.5 81.23708
>
```

```
>
>
> schools %>% arrange(desc(score)) %>% select(id, size, score) %>% top_n(10)
Selecting by score
       id size    score
1  PS 567  121 95.84170
2  PS 191 1036 93.54249
3  PS 330  162 90.99615
4  PS 701   83 90.50055
5  PS 591  213 89.74194
6  PS 205  172 89.28585
7  PS 574  199 89.19625
8  PS 963  208 89.00446
9  PS 430   61 88.72107
10 PS 756  245 87.95731
>
>
>
> median(schools$size)
[1] 261
> schools %>% top_n(10, score) %>% .$size %>% median()
[1] 185.5
>
>
>
> schools %>% top_n(-10, score) %>% .$size %>% median()
[1] 219
>
>
>
> schools %>% filter(rank <= 10)
      id size quality rank    score
1  PS 95  235      91    3 84.80231
2 PS 191 1036      94    1 93.54249
3 PS 343   78      89    5 87.08177
4 PS 430   61      90    4 88.72107
5 PS 558  196      87    7 85.36071
6 PS 567  121      93    2 95.84170
7 PS 981  293      88    6 87.40875
> schools %>% ggplot(aes(size, score)) + geom_point(col = 'gray') + geom_point(data = filter(schools, rank
 <= 10), aes(size, quality), color = 'red')
>
>
>
> length(scores)
[1] 1000
>  # EACH ENTRY IN LIST 'SCORES' HAS SCORES SCORED BY THE NUMBER OF STUDENTS THAT SCHOOL HAS
> scores[1:2]
[[1]]
  [1]  65.4224739  86.0471260 100.0377331 118.7979495  31.8060655  85.5484891
  [7]  71.3999390 110.5871204 145.6913338  78.7643721  91.8550504 110.6901545
 [13] 124.6510362 114.9876415  17.2328720  44.3011804  69.8774418 135.9738211
 [19]  83.2551961  81.7696982  81.9641238  96.7891381 117.2452727 109.2096943
 [25]   8.1451101  15.8781950  76.4668159 101.6898826  31.5785184 100.2173786
 [31]  47.6203698  86.7742400  78.9494555  43.7334077 131.3388629 100.5482352
 [37] 164.8642824 100.5395844  69.1218439  75.0856140  62.5801555  40.5471910
 [43]  13.8846526  79.0168965 125.3200483  90.6578228  69.5551616 102.7250002
 [49]  84.6575677  96.5302580  94.4497716  73.8929696 110.2360635  88.6345715
 [55]  32.2722238  97.7922338 138.2095025  88.1044978 138.6868977  93.4097815
 [61]  95.3547452  72.0207495  17.3162595  44.5791778  73.5984498  45.9905598
 [67]  31.9325903  77.5497037  61.0632963 151.5201785  82.2874192  92.4985603
 [73]  93.7232179  80.5065861  98.6406786  94.9534805 102.1215583 108.7937700
 [79] 110.3053028  80.7347134  41.1026253  -0.6087895  65.5169482  33.4778387
 [85]  69.4581642  71.7202813  87.7822269 129.1573932 129.9374717 106.5602100
 [91]  85.9694454  91.6662676  77.5640571  31.1946226 137.4737751 114.5999367
 [97] 114.2245044  63.8248250  96.3477479  93.7657820  31.9808629  73.9594091
[103]  90.3281853  91.0929081  60.1750191 135.4797639 134.6664495  99.8890201
[109]  59.7339010  50.8721504  40.0176893  59.3621312  23.8531656  93.9266108
[115]  35.0955624  43.0550149  99.4330909  91.2285845 113.1495224  94.0516897
[121]  76.9887187  92.2016063  33.3411664  82.2886498  84.6713365 102.4312360
[127]  77.6047207  81.8729779  38.9422640  95.4635772  21.8426943  85.1282310
[133]  91.2411696 121.8025312 106.7581232 109.7297265  25.0557364 127.0696564
[139]  65.9369994 158.4136011  58.7875201  43.6355959 100.4065182  78.6369681
[145] 116.7960896  64.6906960  50.7183686 101.8202665  71.5014573 111.3658957
[151]  98.2853270  72.1721604  84.2784857  90.0585943  43.9315167  61.6170240
[157]  87.9201928 134.5734497 103.2014097  93.6456237 106.7950403  92.7492969
[163]  59.4416992 115.4741339  46.4047491  49.7468886  88.3907894 129.7459897
[169]  61.8283743 110.5378931  73.5623567  93.5694111  35.9220154  47.1417683
[175] 135.7781992  79.7347209  81.0356117   1.8275043 115.6089265  74.9890315
[181]  92.6386919  89.8504118  90.7425037  66.9391592 135.5499288  83.0736763
[187]  73.9258267 127.6547630  67.8095394  38.6060792  16.3395778  75.9514497
[193]  61.1681200  72.5232077  81.2578000 144.0077617  98.3621490  73.0800205
[199] 117.2103275  69.7798232  94.6606458 113.8789855  76.2145336  21.9415545
[205]  64.2020140 134.9107716  72.6157301 113.0998006  92.2821835  81.0971344
[211]   0.2235059 100.4924150  60.8616483  67.8824651  62.9899491 133.3655960
[217]  43.1107279  -1.7530312  95.3954007 102.0677923  73.3972660  72.1192073
[223] 127.9055690  14.0598568  46.6667189 101.1734476  90.5982618 117.8919152
[229]  54.8932493  30.9212299 106.9975291 133.1826776  77.3048910 135.5773396
[235] 126.3515029  94.0685387  91.8974264  41.0306341 146.0869699  88.9590513
[241]  74.8215284  56.9621852  -4.3277829  69.3143812  47.9324058  80.6425795
[247] 125.3105476 107.9333876

[[2]]
  [1]  24.030915  82.988840  47.997939  27.151384  44.203869  37.196692
  [7]  48.942057 119.451758  25.724908  90.111241  80.258818  75.017485
 [13]  78.581739  68.018821  75.844782 128.807331 125.652329  92.296148
 [19]  65.839998 115.296054  36.312220  77.233070  85.112089 101.290319
 [25]  67.669032 150.360405  52.362041  47.917713 150.033593  95.834621
 [31]  65.621798  50.967505  93.385047  86.109416  38.817928  75.777129
 [37]  96.331694  92.621706  43.290315 138.196297  46.718876  84.818535
 [43] 111.581212  81.361778 123.105501 160.429934 112.490491 108.662743
 [49]  48.222180 129.832777  77.684262  91.267730  82.456284  73.707904
 [55] 141.451489 140.471986  53.659335  66.322460  78.998712  42.268330
 [61]  97.484347  96.783673 119.668138  53.861248 109.136034 106.515463
 [67]  74.601313  41.160152  71.393058 129.008520 100.669405  96.817169
 [73]  -7.550538  79.483344  97.441268  44.944474  64.718463  88.471142
 [79]  83.916742 107.729751  75.067110  77.597136 111.476917  80.682828
 [85]  40.551742  52.156556 102.202740 102.021227 100.029220  95.166565
 [91] 112.939900  65.848714 121.069525  38.844863 101.920761  67.134797
 [97] 100.200219  75.515844  64.558063  80.465197  88.339090  78.867394
[103]  73.135140  52.298600 114.933584  80.397250  86.851537 119.190858
[109] 130.045036 125.538859  69.330815  98.864264 128.191985  39.452608
[115]  38.051776  82.700731  61.239089 144.334114 116.593741  -1.299746
[121]  29.020545 104.274321  90.565190 143.154151  85.383974 134.215680
[127]  20.134028  47.016939  83.984696  64.340073  54.881675  43.498735
[133]  62.828059  32.287905  58.848295 106.407263  55.982900 106.608119
[139]  88.875820  90.928451  14.563923  21.554503  83.252849 131.523277
[145] 107.368308  75.587089 117.140039  74.944049  78.708321 124.893175
[151]  43.332410  74.629405  53.053862  62.685737  30.191410  82.632131
[157] 108.433156  76.387658  70.078455  97.407605 119.485918 108.633119
[163] 102.998078 121.401493  49.329297  77.103810 107.263286  61.166701
[169] 100.515459  81.553874 110.054675  23.586552  88.942075 109.672846
[175]  95.535762 122.226199  76.773430  75.605368 104.694395  73.432477
[181] 121.841555 141.453023  61.975297  46.367307 118.630306 100.544856
[187] 100.585132  50.047053  83.993763 110.810301  68.013278  89.193224
[193]   9.508364 114.650437  37.289919  31.414449 118.088015 101.104545
[199]  61.866715  66.388576 105.877581 111.721133  77.439145  58.897605
[205]  67.013463  92.016703  73.374583  49.188851  51.327694  78.322405
[211]  78.677804  63.992891  73.549850  84.686243  94.089669 129.217584
[217] 155.416845  94.664465  52.227396 100.849378  44.232309 123.798513
```

```
[223]  53.552503  88.230458 136.990672 100.076301  79.489292  98.244056
[229] 128.656569  53.210936  97.521520 119.075097  62.979737  94.851185
[235]  76.789053  82.283785 112.005426  74.392660  74.312263  67.408806
[241]  53.595415  13.812196  57.462814 129.993483  76.924202  40.654335
[247]  61.737768  37.310665  33.922225 106.937612  70.266128  99.315094
[253]  60.196888  35.725670  61.286453  55.232596  51.043901  78.469103
[259] 108.791884  80.306951  18.294071 135.845374 114.409556 151.424904
[265]  71.780105 141.138682  85.981422  67.795950  53.999631  35.097758
[271] 110.071578 117.786241 113.319536 100.156257  87.635348  86.728994
[277] 138.849587  65.631870  58.708181  91.348168  61.394457  40.768972
[283]  52.093133  94.079804 136.000893 125.765016  45.833683  84.315960
[289] 139.957400 121.271838 107.493794  85.291279 115.694704 111.611589
[295]  73.195265  55.134997  70.307279  35.918490 105.320687  75.843556
[301]   8.959485  46.088044  90.585803 109.627574  90.255046  44.773151
[307]  56.449046 170.935717 122.104139  88.676662  75.775453

> overall <- mean(sapply(scores, mean))
> overall
[1] 80.00523
> identical(overall, mean(schools$score))
[1] TRUE
>
> alpha <- 25
> score_reg <- sapply(scores, function(x) {
+ overall + sum(x - overall)/(length(x) + alpha)
+ })
> schools %>% mutate(score_reg = score_reg) %>% top_n(10, score_reg) %>% arrange(desc(score_reg))
       id size quality  rank    score score_reg
1  PS 191 1036      94   1.0 93.54249  93.22352
2  PS 567  121      93   2.0 95.84170  93.12997
3  PS 330  162      84  53.5 90.99615  89.52677
4  PS 591  213      83 104.5 89.74194  88.71918
5  PS 574  199      84  53.5 89.19625  88.17047
6  PS 205  172      85  28.5 89.28585  88.10811
7  PS 701   83      83 104.5 90.50055  88.07108
8  PS 963  208      86  13.5 89.00446  88.03888
9  PS 756  245      83 104.5 87.95731  87.22101
10 PS 561  723      86  13.5 87.39858  87.15148
>
>
>
> # PLOT SCORE - SIZE (QUESTION 4) VS SCORE_REG - SIZE (QUESTION 5)
> schools %>% ggplot(aes(size, score)) + geom_point(col = 'black') + geom_point(data = schools %>% mutate(
score_reg = score_reg), aes(size, score_reg), color = 'red')
>
>
>
> alphas <- seq(10,250)
> rmse <- sapply(alphas, function(alpha){
+ score_reg <- sapply(scores, function(x) overall+sum(x-overall)/(length(x)+alpha))
+ sqrt(mean((score_reg - schools$quality)^2))
+ })
> plot(alphas, rmse, col = 'gray')
> alphas[which.min(rmse)]
[1] 135
>
>
>
> alpha <- alphas[which.min(rmse)]
> score_reg <- sapply(scores, function(x)
+ overall+sum(x-overall)/(length(x)+alpha)
+ )
> schools %>% mutate(score_reg = score_reg) %>% top_n(10, score_reg) %>% arrange(desc(score_reg))
       id size quality  rank    score score_reg
1  PS 191 1036      94   1.0 93.54249  91.98183
```

```
2  PS 567  121      93   2.0 95.84170  87.49043
3  PS 561  723      86  13.5 87.39858  86.23529
4  PS 330  162      84  53.5 90.99615  86.00028
5  PS 591  213      83 104.5 89.74194  85.96477
6  PS 400  550      86  13.5 87.38269  85.92873
7  PS 865  586      86  13.5 87.17508  85.83260
8  PS 266 2369      86  13.5 85.98176  85.65954
9  PS 563  828      86  13.5 86.45072  85.54714
10 PS 574  199      84  53.5 89.19625  85.48132
>
>
>
> alphas <- seq(10,250)
> rmse <- sapply(alphas, function(alpha){
+ score_reg <- sapply(scores, function(x) sum(x)/(length(x)+alpha))
+ sqrt(mean((score_reg - schools$quality)^2))
+ })
> plot(alphas, rmse)
> alphas[which.min(rmse)]
[1] 10
>
```