# MongoDb

Implementation: Vogiatzis George, Zourou Mirsini

We begin our task by loading the required libraries then via the powershell command: dir -Recurse -Name -File > files_list.txt we created a list containing the paths to our data files, and for each file we performed a number of cleansing . We decided to save all listings which didn't contain price info ("Askforprice") as zero values and we also created a new boolean column which gives info on if the listing's price is negotiable. We also decided to convert Price, Mileage, Cubic Capacity and Power to integer values and removed some symbols (km, bhp, € ) via gsub() command. Finally we calculate a new column called Score and add it to our objects and insert them in our local mongodb:

```
library(mongolite)
library(lubridate)

library(httpuv)
library(jsonlite)
library(rmutil)

library(stringr)
library(dplyr)

library(rmarkdown)

#2.1    Add your data to MongoDB.

#read file with paths of json files to a list
my_data <- read.delim("C:\\Users\\George\\Documents\\files_list.txt")

#create a prefix to concat it with the paths
prefix <- "C:\\Users\\George\\Documents\\BIKES\\"

#create a mongo connection object and create an empty collection
m <- mongo("bikeAds", url = "mongodb://localhost")

#variables to be used for adding a Score column to the dataset
ageScore <- 0
mileageScore <- 0
priceScore <- 0
cc_bhpScore <- 0

lower1age <-0
lower2age <-6
```

```
lower3age <-11
upper1age <-5
upper2age <-10

lower1mlg <-0
lower2mlg <-20001
lower3mlg <-50001
upper1mlg <-20000
upper2mlg <-50000

lower1price <-0
lower2price <-2001
lower3price <-5001
upper1price <-2000
upper2price <-5000

group1Factor <-50
group2Factor <-30
group3Factor <-20

lower1cc_bhp <- 0
lower2cc_bhp <-0.051
lower3cc_bhp <-0.11
upper1cc_bhp <-0.05
upper2cc_bhp <-0.1
options(warn=-1)

#for loop where data cleansing, additional manipulation and insert in monogdb
happens
for (row in 1:nrow(my_data))
{
  data <- fromJSON(readLines(paste(prefix , my_data[row,"files_list.txt"],sep
=""), encoding="UTF-8"))
  data$ad_data$Mileage <- gsub(" km", "", data$ad_data$Mileage)
  data$ad_data$Mileage <- gsub(",", "", data$ad_data$Mileage)
  data$ad_data$Mileage <- as.numeric(data$ad_data$Mileage)
  data$ad_data$Price <- gsub('.*€', '', data$ad_data$Price)
  data$ad_data$Price <- gsub("\\.", '', data$ad_data$Price)
  if (str_detect(data$metadata$model , "Negotiable"))
  {
    data$ad_data$Negotiable <- as.logical("TRUE")
  }
  else
  {
    data$ad_data$Negotiable <- as.logical("FALSE")
  }
  data$metadata$model <- gsub(" -.*","",data$metadata$model)
  data$ad_data$`Cubic capacity` <- gsub(" cc.*","",data$ad_data$`Cubic capaci
ty`)
```

```
  data$ad_data$`Cubic capacity` <- gsub(",", "",data$ad_data$`Cubic capacity`
)
  data$ad_data$`Cubic capacity` <- as.numeric(data$ad_data$`Cubic capacity`)
  data$ad_data$Power <- gsub(" bhp.*","",data$ad_data$Power)
  data$ad_data$Power <- as.numeric(data$ad_data$Power)
  if (data$ad_data$Price == 'Askforprice')
  {
    data$ad_data$Price <- "0"
  }
  data$ad_data$Price <- as.numeric(data$ad_data$Price)
  data$ad_data$Registration <- gsub(".*/","", data$ad_data$Registration)
  data$ad_data$Registration <- gsub(" ","", data$ad_data$Registration)
  data$ad_data$Age <- year(Sys.Date()) - as.numeric(data$ad_data$Registration
)

  if(data$ad_data$Age >= lower1age & data$ad_data$Age <= upper1age)
  {
    ageScore <- data$ad_data$Age * group1Factor
  }
  else if (data$ad_data$Age >= lower2age & data$ad_data$Age <= upper2age)
  {
    ageScore <- data$ad_data$Age * group2Factor
  }
  else if (data$ad_data$Age >= lower3age )
  {
    ageScore <- data$ad_data$Age * group3Factor
  }
  else
  {
    ageScore <-0
  }
  if(length(data$ad_data$Mileage) > 0 ){
    if( data$ad_data$Mileage >= lower1mlg & data$ad_data$Mileage <= upper1mlg
)
    {
      mileageScore <- data$ad_data$Mileage * group1Factor
    }
    else if (data$ad_data$Mileage >= lower2mlg & data$ad_data$Mileage <= uppe
r2mlg)
    {
      mileageScore <- data$ad_data$Mileage * group2Factor
    }
    else if (data$ad_data$Mileage >= lower3mlg )
    {
      mileageScore <- data$ad_data$Mileage * group3Factor
    }
    else
    {
      mileageScore <-0
    }
```

```
  }
  else{
    mileageScore <-0
  }

  if(data$ad_data$Price >= lower1price & data$ad_data$Price <= upper1price)
  {
    priceScore <- data$ad_data$Price * group1Factor
  }
  else if (data$ad_data$Price >= lower2price & data$ad_data$Price <= upper2pr
ice)
  {
    priceScore <- data$ad_data$Price * group2Factor
  }
  else if (data$ad_data$Price >= lower3price )
  {
    priceScore <- data$ad_data$Price * group3Factor
  }
  else
  {
    priceScore <-0
  }

  if(length(data$ad_data$`Cubic capacity`) > 0){
    if((data$ad_data$Power/data$ad_data$`Cubic capacity`) >= lower1cc_bhp & (
data$ad_data$Power/data$ad_data$`Cubic capacity`) <= upper1cc_bhp)
    {
      cc_bhpScore <- (data$ad_data$Power/data$ad_data$`Cubic capacity`) * gro
up3Factor
    }
    else if ((data$ad_data$Power/data$ad_data$`Cubic capacity`) >= lower2cc_b
hp & (data$ad_data$Power/data$ad_data$`Cubic capacity`) <= upper2cc_bhp)
    {
      cc_bhpScore <- (data$ad_data$Power/data$ad_data$`Cubic capacity`) * gro
up2Factor
    }
    else if ((data$ad_data$Power/data$ad_data$`Cubic capacity`) >= lower3cc_b
hp )
    {
      cc_bhpScore <-(data$ad_data$Power/data$ad_data$`Cubic capacity`) * grou
p1Factor
    }
    else
    {
      cc_bhpScore <-0
    }
  }
  else{
    cc_bhpScore <-0
  }
```

```
  data$ad_data$Score <- ageScore + mileageScore + priceScore + cc_bhpScore


  data <- toJSON(data, auto_unbox = TRUE)
  m$insert(data)
}
```

On the next question we just had to count the total number of listings since each listing represents a bike:

```
#2.2 How many bikes are there for sale


#count all records in database
bikesForSale <- m$count('{}')


print(bikesForSale)
```

```
## [1] 29701
```

For the next question we have to calculate the average price of a bike. Before we make the aggregation we have to match first all bikes with a price greater or equal to 100e because all the listings with price less than this usually are for parts and not for actual bikes. This results in a smaller number of listings used in comparison with the previous question.

```
#2.3 What is the average price of a motorcycle (give a number)?
#What is the number of listings that were used in order to calculate this ave
rage (give a number as well)?
#Is the number of listings used the same as the answer in 1.2? Why?

#Aggregation to find average price of all ads with price greater than or equa
l to 100 euros
bikesAvgPrice <- m$aggregate(
  '[
  {"$match": {"ad_data.Price": { "$gte": 100 }}},
  {"$group":{"_id": null, "average":{"$avg":"$ad_data.Price"}}}
  ]'
)

#count the number of listings used
bikesUsedForAverage <-  nrow(m$aggregate('[
  {"$match": {"ad_data.Price": { "$gte": 100 }}}]'))
```

```r
#print the results
print(bikesAvgPrice$average)
```

```
## [1] 3030.624
```

```r
print(bikesUsedForAverage)
```

```
## [1] 28490
```

For the next question we calculate the min and max price. For the min price we can safely pick 100 euros as the minimun for the same reason we decribed in the previous question

```r
#2.4    What is the maximum and minimum price of a motorcycle currently avail
able in the market?

#calculate the max price with aggregate()
maxPrice <- m$aggregate(
  '[
   {"$match": {"ad_data.Price": { "$gte": 100 }}},
   {"$group":{"_id": null, "max":{"$max":"$ad_data.Price"}}}
   ]'
)

#print the result
print(maxPrice$max)
```

```
## [1] 89000
```

```r
#calculate the max price with aggregate()
minPrice <- m$aggregate(
  '[
   {"$match": {"ad_data.Price": { "$gte": 100 }}},
   {"$group":{"_id": null, "min":{"$min":"$ad_data.Price"}}}
   ]'
)

#print the result
print(minPrice$min)
```

```
## [1] 100
```

```r
#since we idenitfied as valid the ads with price >100 euros in q.2.3 we can s
ay the minimum price is 100 euros
```

For the next question we calculate the number of listings with a negotiable price. To achieve this we first match all listings with positive price value

and then counting the true occurrences in our Boolean column ‘ad_data.Negotiable’

```
#2.5    How many listings have a price that is identified as negotiable?

#calculate the number of listings matching first the listings with price >0 a
nd then with   aggregate() we count their number
negotiableBikes <- m$aggregate(
  '[
  {"$match": {"ad_data.Negotiable": true}},
  {"$group": { "_id": null, "negCount": { "$sum": 1 }}}
                   ]'
)
#print the result
print(negotiableBikes$negCount)

## [1] 1348
```

For the next question we have to calculate the percentage of negotiable bikes per brand. First we group by brand by conditionally calculating the negotiable sum and the total sum and then we add a new column which consists of the negotiable count divided by the total count and then the whole is multiplied by 100

```
#2.6    For each Brand, what percentage of its listings is listed as negotiab
le?

#calculate the percentages with an aggregate pipeline where we group by brand
and we count the total listings per brand
#and the number of negotiable listings per brand then we perform some numeric
operations to have the desired results
negPercentage <- m$aggregate(
  '[
  { "$group": {
    "_id": "$metadata.brand",
    "totalCount": { "$sum": 1 },
    "negotiableCount": {
      "$sum": {
        "$cond": {
          "if":{ "$eq": [ "$ad_data.Negotiable", true ] },
          "then": 1,
          "else": 0
        }
      }
    }
  }},
  { "$addFields": {
```

```
    "negotiablePercentage": {
      "$cond": {
        "if": { "$ne": [ "$negotiableCount", 0 ] },
        "then": {
          "$multiply": [
            { "$divide": [ "$negotiableCount", "$totalCount" ] },
            100
            ]
        },
        "else": 0
      }
    }
  }},
  { "$sort": { "negotiablePercentage": -1 } }
  ]'
)

#print the result
print(negPercentage)
```

```
##                                _id totalCount negotiableCount
## 1                           Jinlun          1               1
## 2                        Bombardier          1               1
## 3                            Qingqi          2               2
## 4                             Fever          1               1
## 5                               Niu          1               1
## 6                              Dias          3               3
## 7                          Amstrong          3               3
## 8                       Regal-Raptor          2               2
## 9                          Apokotos          2               2
## 10                      Buggy Motors          3               3
## 11                           Victory          1               1
## 12                            Kuberg          1               1
## 13                             Odess          2               2
## 14                            Motobi          1               1
## 15                            Jmstar          1               1
## 16                           HighPer          2               2
## 17                       Boom-Trikes          1               1
## 18                               Wsk          1               1
## 19                             E-ATV          2               2
## 20                      Nitro Motors          1               1
## 21                            Joyner          1               1
## 22                            Sherco          1               1
## 23                               Mtg          4               4
## 24                          ZhongYu          1               1
## 25                         Vee Road          1               1
## 26                            Xgjao          8               7
## 27                            Adiva          5               4
## 28                            Kaisar          4               3
## 29                         Lambretta         14              10
```

```
## 30                            Haojin         10                7
## 31                              Znen          6                4
## 32                            Indian          3                2
## 33                            Gemini          6                4
## 34                                AB         13                8
## 35                            Jonway          5                3
## 36                           Baotian         14                8
## 37                               Dkw         16                9
## 38                            Bashan         21               11
## 39                         Super Moto          2                1
## 40                            Bultaco          2                1
## 41                              Nomik          2                1
## 42                              E-Ton          2                1
## 43                                Emw          2                1
## 44                             Skyjet         10                5
## 45                         FB Mondial          4                2
## 46                            JetMoto          4                2
## 47                               Jawa         16                7
## 48                            Zuendapp          7                3
## 49                                Imr         10                4
## 50                         Dirt Motos         15                6
## 51                            Heinkel          5                2
## 52                                MBK          5                2
## 53                             Quadro          5                2
## 54                      Royal Enfield         13                5
## 55                            Xingyue          8                3
## 56  <U+0391><U+03BB><U+03BB><U+03BF>        339              121
## 57                            Cheetah         17                6
## 58                             Rewaco          3                1
## 59                             Access          9                3
## 60                            Boatian          6                2
## 61                           Hercules          3                1
## 62                            Montesa          6                2
## 63                                Bsa         15                5
## 64                                SMC          9                3
## 65                             Lintex          3                1
## 66                                Swm          6                2
## 67                              Bajaj          6                2
## 68                            Motivas          3                1
## 69                            Zundapp         62               20
## 70                              Sachs         62               19
## 71                               Hsun          7                2
## 72                      Access Motor           7                2
## 73                           Jincheng          7                2
## 74                               Beta         40               11
## 75                             Simson         11                3
## 76                                 MZ         11                3
## 77                              Solex         15                4
## 78                              Maico          4                1
## 79                                Lem         12                3
```

```
## 80                          Fuxin          4                1
## 81                        Kinroad          4                1
## 82                      Arctic Cat          8                2
## 83                     Moto Morini          4                1
## 84                           Lifan        104               25
## 85                             Ymc         36                8
## 86                             LML         27                6
## 87                            Puch          9                2
## 88                         Kreidler         83               17
## 89                            Ural          5                1
## 90                            Asus          5                1
## 91                       AGM motors          5                1
## 92                          Norton          5                1
## 93                         Siamoto          5                1
## 94                         Italjet          5                1
## 95                        Zongshen         26                5
## 96                              TM         33                6
## 97                         Polaris         35                6
## 98                           Vmoto          6                1
## 99                         Generic          6                1
## 100                         Dayang         30                5
## 101                          Horex          6                1
## 102                            Vor          6                1
## 103                            Tgb          7                1
## 104                         CFmoto         21                3
## 105                            Kxd          7                1
## 106                        Daytona        393               52
## 107                        Garelli         24                3
## 108                         Loncin         25                3
## 109                        Hyosung         43                5
## 110                       Nipponia          9                1
## 111                       Shineray         28                3
## 112                            CPI         10                1
## 113                            Pgo         20                2
## 114                         Cagiva         32                3
## 115                         CAN-AM         22                2
## 116                            TCB         44                4
## 117                         Keeway        103                9
## 118                        Gas-Gas         35                3
## 119                        Benelli         41                3
## 120                      Mv Agusta         32                2
## 121                          Derbi         86                5
## 122                          Vespa        274               14
## 123                        Jialing         20                1
## 124                         Linhai         48                2
## 125                       Husqvarna        145                6
## 126                          Honda       6190              247
## 127                Harley Davidson        309               12
## 128                         Modenas        261               10
## 129                        Malaguti         55                2
```

```
## 130                        Husaberg          30                1
## 131                        Kawasaki        1953               65
## 132                          Gilera         590               18
## 133                            Aeon          33                1
## 134                          Yamaha        5529              156
## 135                         Skyteam          44                1
## 136                          Suzuki        2365               52
## 137                             KTM         966               21
## 138                             Bmw        1394               28
## 139                       Moto Guzzi          50                1
## 140                          Peugeot         306                6
## 141                             Sym        1090               20
## 142                           Kymco        1148               21
## 143                          Aprilia         892               16
## 144                          Triumph         335                6
## 145                          Piaggio        2685               45
## 146                           Ducati         373                6
## 147                           Daelim          69                1
## 148                            Adler           1                0
## 149                           Genata           2                0
## 150                          Barossa           1                0
## 151                            Ariel           2                0
## 152                              Aie           4                0
## 153                            Vedim           1                0
## 154                            Eagle           3                0
## 155                             Goes           1                0
## 156                           Harlow           1                0
## 157                            Semog           1                0
## 158                          Enfield           4                0
## 159                          Mobster           2                0
## 160                               KL           1                0
## 161                             Nova           2                0
## 162                          Beeline           1                0
## 163                          Jianshe          10                0
## 164                       Chang Jiang           1                0
## 165                          G-force           1                0
## 166                              AJP           2                0
## 167                         Hartford           1                0
## 168                            Seckam           3                0
## 169                          Mikilon           1                0
## 170                           Morini           3                0
## 171                       Euromotors           6                0
## 172                            Geely           1                0
## 173                            Dinli           6                0
## 174                          Evomoto           5                0
## 175                           Polini           2                0
## 176                 New Force Motor           1                0
## 177                           Bimota           2                0
## 178          <U+03A7>-<U+039C>otors          15                0
## 179                              AMS           3                0
```

```
## 180                              Fym          1                0
## 181                              Ccm          6                0
## 182                           Sokudo          4                0
## 183                              Nsu          9                0
## 184                          Lingben          1                0
## 185                  Shandong Liangzi          1                0
## 186                          Laverda          2                0
## 187                            Buell         17                0
## 188                           Unilli          1                0
## 189                          Brixton         18                0
## 190                            Masai          2                0
## 191                              Emb          1                0
## 192                           Cectek          2                0
## 193                            Gamax          1                0
## 194                        Shan Yang          1                0
## 195                          Xinling          2                0
## 196                           Subaru          1                0
## 197                         Victoria          1                0
## 198                             Adly          7                0
##     negotiablePercentage
## 1              100.000000
## 2              100.000000
## 3              100.000000
## 4              100.000000
## 5              100.000000
## 6              100.000000
## 7              100.000000
## 8              100.000000
## 9              100.000000
## 10             100.000000
## 11             100.000000
## 12             100.000000
## 13             100.000000
## 14             100.000000
## 15             100.000000
## 16             100.000000
## 17             100.000000
## 18             100.000000
## 19             100.000000
## 20             100.000000
## 21             100.000000
## 22             100.000000
## 23             100.000000
## 24             100.000000
## 25             100.000000
## 26              87.500000
## 27              80.000000
## 28              75.000000
## 29              71.428571
## 30              70.000000
```

```
## 31             66.666667
## 32             66.666667
## 33             66.666667
## 34             61.538462
## 35             60.000000
## 36             57.142857
## 37             56.250000
## 38             52.380952
## 39             50.000000
## 40             50.000000
## 41             50.000000
## 42             50.000000
## 43             50.000000
## 44             50.000000
## 45             50.000000
## 46             50.000000
## 47             43.750000
## 48             42.857143
## 49             40.000000
## 50             40.000000
## 51             40.000000
## 52             40.000000
## 53             40.000000
## 54             38.461538
## 55             37.500000
## 56             35.693215
## 57             35.294118
## 58             33.333333
## 59             33.333333
## 60             33.333333
## 61             33.333333
## 62             33.333333
## 63             33.333333
## 64             33.333333
## 65             33.333333
## 66             33.333333
## 67             33.333333
## 68             33.333333
## 69             32.258065
## 70             30.645161
## 71             28.571429
## 72             28.571429
## 73             28.571429
## 74             27.500000
## 75             27.272727
## 76             27.272727
## 77             26.666667
## 78             25.000000
## 79             25.000000
## 80             25.000000
```

```
## 81             25.000000
## 82             25.000000
## 83             25.000000
## 84             24.038462
## 85             22.222222
## 86             22.222222
## 87             22.222222
## 88             20.481928
## 89             20.000000
## 90             20.000000
## 91             20.000000
## 92             20.000000
## 93             20.000000
## 94             20.000000
## 95             19.230769
## 96             18.181818
## 97             17.142857
## 98             16.666667
## 99             16.666667
## 100            16.666667
## 101            16.666667
## 102            16.666667
## 103            14.285714
## 104            14.285714
## 105            14.285714
## 106            13.231552
## 107            12.500000
## 108            12.000000
## 109            11.627907
## 110            11.111111
## 111            10.714286
## 112            10.000000
## 113            10.000000
## 114             9.375000
## 115             9.090909
## 116             9.090909
## 117             8.737864
## 118             8.571429
## 119             7.317073
## 120             6.250000
## 121             5.813953
## 122             5.109489
## 123             5.000000
## 124             4.166667
## 125             4.137931
## 126             3.990307
## 127             3.883495
## 128             3.831418
## 129             3.636364
## 130             3.333333
```

```
## 131             3.328213
## 132             3.050847
## 133             3.030303
## 134             2.821487
## 135             2.272727
## 136             2.198732
## 137             2.173913
## 138             2.008608
## 139             2.000000
## 140             1.960784
## 141             1.834862
## 142             1.829268
## 143             1.793722
## 144             1.791045
## 145             1.675978
## 146             1.608579
## 147             1.449275
## 148             0.000000
## 149             0.000000
## 150             0.000000
## 151             0.000000
## 152             0.000000
## 153             0.000000
## 154             0.000000
## 155             0.000000
## 156             0.000000
## 157             0.000000
## 158             0.000000
## 159             0.000000
## 160             0.000000
## 161             0.000000
## 162             0.000000
## 163             0.000000
## 164             0.000000
## 165             0.000000
## 166             0.000000
## 167             0.000000
## 168             0.000000
## 169             0.000000
## 170             0.000000
## 171             0.000000
## 172             0.000000
## 173             0.000000
## 174             0.000000
## 175             0.000000
## 176             0.000000
## 177             0.000000
## 178             0.000000
## 179             0.000000
## 180             0.000000
```

```
## 181             0.000000
## 182             0.000000
## 183             0.000000
## 184             0.000000
## 185             0.000000
## 186             0.000000
## 187             0.000000
## 188             0.000000
## 189             0.000000
## 190             0.000000
## 191             0.000000
## 192             0.000000
## 193             0.000000
## 194             0.000000
## 195             0.000000
## 196             0.000000
## 197             0.000000
## 198             0.000000
```

For the next question we have to identify *motorcycle brand with the highest average price. First we filter out every bike which costs lees than 100e then we group by brand and average price, we sort the result and keep the first line.*

```
#2.7    What is the motorcycle brand with the highest average price?

#calculate the result with aggregate() by matching all listings with price >1
00 euros then group by brand
#and calculate the avg price and then sort in descending order and limit the
result by one
bikesAvgHighestPrice <- m$aggregate(
  '[
  {"$match": {"ad_data.Price": { "$gte": 100 }}},
  {"$group":{"_id": "$metadata.brand", "average":{"$avg":"$ad_data.Price"}}},
  { "$sort": { "average": -1}},
  {"$limit": 1}
  ]'
)

#print the result
print(bikesAvgHighestPrice)

##     _id average
## 1 Semog   15600
```

For the next question we act like in the previous, grouping by model and average mileage and then sorting by average age and mileage descending. We do calculate the avg mileage also to use it as a criteria in case of draws. If two models share the same avgAge then we first choose the one with less miles.

```
#2.8    What are the TOP 10 models with the highest average age? (Round age b
y one decimal number)

#in the same manner with the above question
top10highest <- m$aggregate(
  '[
              {"$match": {"ad_data.Mileage": { "$gt": 0 }}},
                   {"$group":{"_id": "$metadata.model","avgMileage": {"$avg
":"$ad_data.Mileage"}, "avgAGE":{"$avg":"$ad_data.Age"}}},
              {"$sort": {"avgAGE":-1, "avgMileage":1}},
              {"$limit": 10}
                          ]'
)

top10highest <- select(top10highest, 1, 3)

#print the result
print(top10highest)
```

```
##                                                                    _id avgAGE
## 1  <U+0391><U+03BB><U+03BB><U+03BF>  henderson indian replica '31     88
## 2                                                              R 12     85
## 3                                                    Norton   '35     84
## 4          <U+0391><U+03BB><U+03BB><U+03BF>  MATCHLESS G3 350 '35     84
## 5                                                 Norton  H16 '36     83
## 6               <U+0391><U+03BB><U+03BB><U+03BF>  Matsoules '38     81
## 7            <U+0391><U+03BB><U+03BB><U+03BF>  Matchless G3/L '39     80
## 8                <U+0391><U+03BB><U+03BB><U+03BF>  NEW HUDSON '39     80
## 9                                        Bsa  M20 ARMY MOTO! '39     80
## 10                                           Bsa   '39     80
```

For the next question we first match all listings containing ‘ABS’ keyword in ‘extras’ field and then counting the number of occurrences.

```
#2.9    How many bikes have "ABS" as an extra?

#matching the listings with ABS and then counting the total number of listing
s
abs <- m$aggregate('[
  {"$match": {"extras": "ABS"}},
  {"$group": {"_id": null, "ABSCount": {"$sum": 1}}}
```

```
                    ]')

#print the result
print(abs$ABSCount)

## [1] 4025
```

For the next question we first match all listings containing ‘ABS’ and ‘Led l ights’ keywords and then grouping by the whole result (id : null) and calcula ting the average mileage

```
#2.10   What is the average Mileage of bikes that have “ABS” AND “Led lights” as an extra?


#in the same manner with the above question
absLed <- m$aggregate(
  '[
  {"$match": { "$and": [ {"extras": "ABS"}, {"extras":"Led lights"}]}},
  {"$group":{"_id": null, "absLedAvg":{"$avg": "$ad_data.Mileage"}}}
                    ]'
)

#print the result
print(absLed$absLedAvg)

## [1] 30125.7
```

For the next question we first group all listings  by category and color with the count of each color per category, then we sort by category and count in d escending order. After this stage of the aggregation pipeline we feed the res ult in new documents via  project and finally we slice the desired results to contain only 3 results per category.

```
#2.11   What are the TOP 3 colors per bike category?

#calculate the result with aggregate() by grouping by category and color and calculating the number of occurences per color
#then sorting by count in descending order and grouping again this time pushi ng the results in in new documents and projecting the elements we want
#by slicing the top 3 colors
top3colors <- m$aggregate(
```

```
  '[
           {"$group":
              {"_id": {"category": "$ad_data.Category","color": "$ad_data.Col
or"},"count": {"$sum": 1}}},
           {"$sort": {"_id.category": -1,"count": -1}},
           {"$group":
              {"_id":"$_id.category","topColors":{"$push":"$_id.color"}}},
           {"$project": { "category": "$_id.category", "top3Colors": { "$slic
e": [ "$topColors", 3 ] } } }
                       ]'
)

print(top3colors)
```

```
##                         _id                                    top3Colors
## 1         Bike - Cafe Racer                 Black, Black (Metallic), Red
## 2            Bike - Chopper Black, Black (Metallic), Bordeaux (Metallic)
## 3             Bike - Custom                 Black, Black (Metallic), Red
## 4     Bike - Four Wheel-ATV                            Red, Black, White
## 5          Bike - Motocross                          Red, Orange, Green
## 6             Bike - On/Off              Black, Black (Metallic), White
## 7              Bike - Other                Black, Black (Metallic), Red
## 8        Bike - Three Wheel                            Black, White, Red
## 9      Bike - Sport Touring   Black (Metallic), Black, Silver (Metallic)
## 10       Bike - Super Sport                Black, Black (Metallic), Red
## 11 Bike - Mobility scooter     Red, Black (Metallic), Silver (Metallic)
## 12   Bike - Roller/Scooter              Black, Black (Metallic), White
## 13             Bike - Trial                White, Red, Black (Metallic)
## 14 Bike - UTV Side by Side                          Black, Blue, White
## 15       Bike - Street Bike                Black, Black (Metallic), Red
## 16        Bike - Mini..Moto                           Red, Black, White
## 17         Bike - Underbone               Black, Blue, Black (Metallic)
## 18             Bike - Naked    Black, Black (Metallic), White (Metallic)
## 19      Bike - <U+0392>uggy                            Black, Red, Blue
## 20      Bike - Super Motard             Black, Orange, Black (Metallic)
## 21             Bike - Moped                           Red, Black, Blue
## 22     Bike - <U+0395>nduro                          White, Orange, Red
```

For the next question we had to calculate a new column called Score which represents the score of each bike based on some factors we assumed where vital for a bike's listing quality. So we decided to define some ranges for the values of Age, Price, Milage and Power/Cubic Capacity, and depending on each listing's value we give a score per value multiplied by the corresponding factor.

The factors we chose where 3 values: 50, 30, 20. For example a bike in the first range of Age (0-5 years) will have a factor of 50 and a bike in the range of 6-10 will have a factor of value 30 etc.

Finally we add the scores of each field to calculate the listing's total score then we match our target badget (let's say in this case we wanted to buy a bike which costs up to 10000 euros) and we sort by score and limit the result to contain the top 100 bikes in terms of score. In that way we have a flexible tool to give us best deal sets depending on our needs. We could also define desired age or mileage also.

```
#2.12   Identify a set of ads that you consider "Best Deals".

#we match our target badget then sort by score and limit the result to 100 li
stings
bestDeals <- m$aggregate(
  '[
  {"$match": { "$and": [ {"ad_data.Price":  { "$gt": 0 }}, {"ad_data.Price":
{ "$lte": 10000 }}]}},
  {"$sort": {"ad_data.Score": -1}},
  {"$limit": 100}
     ]'
)
#view the result
View(bestDeals)
```