# Hadoop and Hive

*by Igor Baranov*

**Abstract** A dataset was scraped from several websites in Czech Republic and Germany over a period of more than a year. The dataset contains over 3.5 million records and has lots of missing data. The goal of this project was to apply such big data tools as Hadoop and Hive to load and query the data and prepare it to the analysis. During the analysis the questions like what is the most advertised vs sold cars, car maker and car model vere answered.

## Introduction

The first goal of the project was to use apache HIVE as an analytic tool to analyze realistic data. Second goal was to acquare experience working with opened end problems, that are similar to real problems that are faced by data professionals. At the end of this project we should gain sufficient confidence in using Hadoop and Apache Hive, get experience in working with large datasets and be aware of the potential and benefits of analyzing large datasets.

## Environment Preparation

A virtual machine is a software computer that, like a physical computer, runs an operating system and applications. The virtual machine is backed by the physical resources of a host. Every virtual machine has virtual devices that provide the same functionality as physical hardware and have additional benefits in terms of portability, manageability, and security.

Hortonworks Sandbox (Hortonworks Sandbox) is a free version of the Hadoop and Hive installations. Deployment is available in three isolated environments: virtual machine, container or cloud. There are two sandboxes available: Hortonworks Data Platform (HDP) and Hortonworks DataFlow (HDF). We have chosen a CentOS Linux - based Virtual Machine.

## Loading Classified Ads for Cars Data to Hadoop

### Data understanding

The dataset (Classified Ads for Cars) has 16 attributes and 3.5 million instances. The data was scraped from several websites in Czech Republic and Germany over a period of more than a year. The scrapers were tuned slowly over the course of the year and some of the sources were completely unstructured, so as a result the data is dirty, there are missing values and some values are very obviously wrong (e.g. phone numbers scraped as mileage etc.). There are roughly 3,5 Million rows and the following columns:

- maker - normalized all lowercase
- model - normalized all lowercase
- mileage - in KM
- manufacture_year
- engine_displacement - in ccm
- engine_power - in kW
- body_type - almost never present, but I scraped only personal cars, no motorcycles or utility vehicles
- color_slug - also almost never present
- stk_year - year of the last emission control
- transmission - automatic or manual
- door_count
- seat_count
- fuel_type - gasoline, diesel, cng, lpg, electric
- date_created - when the ad was scraped
- date_last_seen - when the ad was last seen. Our policy was to remove all ads older than 60 days
- price_eur - list price converted to EUR

**Loading the Car Ads data to Hadoop HDFS**

To download the dataset requires registering to Kaggle. After downloading it could be copied to the VM. Please note that Hortonworks VM has several users preconfigured in the system. Some of them are administrators, others are regular users. The full list of users with all the access rights, passwords and systems they can access presented in the APPENDIX A of Hortonworks VM tutorial. We have chosen user **maria_dev** for the task. Here are the steps to load Cars Ads data to Hadoop HDFS inside Hortonworks VM:

Login to the Hortonworks VM and create folder for the project using the script below:

```
ssh maria_dev@127.0.0.1 -p 2222
cd used-cars/
exit
```

Open Git bash or Putty console and copy copy the file to the Hortonworks VM:

```
scp -P 2222 classified-ads-for-cars.zip maria_dev@127.0.0.1:/home/maria_dev/used-cars
```

Login to VM, unzip the file and count the number of lines in the created file (should be about 3.5M):

```
ssh maria_dev@127.0.0.1 -p 2222
cd used-cars/
gzip -d --suffix=.zip *.*
wc -l classified-ads-for-cars
```

Copy first line of the file to 'headers' file to use it later for creatin SQL statements:

```
 head -1 classified-ads-for-cars > headers
```

Split the file into 100 chunks and remove headers line from the first file. The number of chunks was taken orbitrary just to make the procedure of loading to HDFS more realistic:

```
mkdir chunks
cd chunks
split --number=l/100 ../classified-ads-for-cars classified-ads-for-cars_
sed -i 1d classified-ads-for-cars_aa
```

Create a directory in HDFS called cars/classified by using the following command:

```
hdfs dfs -mkdir -p  baranov/cars/classified
```

You can now copy the event files you downloaded earlier to the hdfs directory you just created by running the following commands. Those commands for each file will print the name of the file (to see the progress), then load the file to HDFS and then move the processed file to folder **../loaded-files**:

```
mkdir ../loaded-files

for file in *; do echo $file;  \
  hdfs dfs -put $file cars/classified/; \
  mv $file -f ../loaded-files; \
done
```

To check how many unloaded files left, run the following commabd from another(!) Git bash or Putty window:

```
ls events/ | wc -l
```

List files copied to hadoop by running the following command:

```
hdfs dfs -ls cars/classified/
```

After the process of loading is finished, remove chunks:

```
cd ..
rm -r -f chunks
rm -f loaded-files/*
rm -r -f loaded-files
```

## Creating HIVE database

HIVE is available in Hortonworks VM entering **hive** command on the bash command line. Here are the steps to create the HIVE database and to load the Cars Ads dataset into it:

Create folder **hive** in the home directory for the files and results:

```
cd ~
mkdir hive
cd hive
```

Create text file for SQL script that creates the HIVE database and enter the following SQL text:

```
vi create-db.sql

CREATE DATABASE
    IF NOT EXISTS used_cars
    COMMENT 'This is the used cars database'
    With dbproperties ('Created by' = 'baranov','Created on' = 'August-2018');
```

Create text file for SQL script that creates the HIVE table and enter the following SQL text:

```
vi create-table.sql

CREATE EXTERNAL TABLE IF NOT EXISTS used_cars.events (
    maker STRING,
    model STRING,
    mileage INT,
    manufacture_year INT,
    engine_displacement INT,
    engine_power INT,
    body_type STRING,
    color_slug STRING,
    stk_year STRING,
    transmission STRING,
    door_count INT,
    seat_count INT,
    fuel_type STRING,
    date_created TIMESTAMP,
    date_last_seen TIMESTAMP,
    price_eur DECIMAL(13,2)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/maria_dev/cars/classified';

ALTER TABLE used_cars.events
  SET SERDEPROPERTIES ("timestamp.formats"="yyyy-MM-dd HH:mm:ss.SSSSSSZ");
```

Confirm that the HIVE table is created properly. We should see the following output:

```
hive> DESCRIBE used_cars.events;
OK
maker                   string
model                   string
mileage                 int
manufacture_year        int
engine_displacement     int
engine_power            int
body_type               string
color_slug              string
stk_year                string
transmission            string
```

```
door_count            int
seat_count            int
fuel_type             string
date_created          timestamp
date_last_seen        timestamp
price_eur             decimal(13,2)
Time taken: 0.59 seconds, Fetched: 16 row(s)
```

Confirm that the data is loaded properly by requesting number of recors in previously created **events** table. We should see the following output:

```
hive> select count (*) from used_cars.events;
Query ID = maria_dev_20180903015614_32d5f61c-9297-497e-83eb-5f6cbb1e3d6b
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1535776685382_0027)


--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........    SUCCEEDED    20         20        0        0       0       0
Reducer 2 ......    SUCCEEDED     1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 102.01 s
--------------------------------------------------------------------------------
OK
3552912
Time taken: 110.189 seconds, Fetched: 1 row(s)

hive> quit
```

## Analyzing Data

To perform the analysis, certain R libraries were used. The code below was used to load and initialize the library, then loads the data. To pretty-print the tables in this report we used xtable (Dahl, 2016) library.

```
set.seed(42)
library(ggplot2)
library(reshape2)
library(plyr)
library(readr)
library(fpc)
library(data.table)
library(ggplot2)
```

Connecting to USED_CARS database in HIVE running on Hortonworks Sandbox VM

```
options( java.parameters = "-Xmx8g" )
library(rJava)
library(RJDBC)

cp = c("//d:/tools/apache-hive-1.2.2/lib/hive-jdbc-1.2.2-standalone.jar",
       "//d:/tools/hadoop-2.7.7/share/hadoop/common/hadoop-common-2.7.7.jar")
.jinit(classpath=cp)

drv <- JDBC(driverClass = "org.apache.hive.jdbc.HiveDriver",
        classPath = "//d:/tools/apache-hive-1.2.2/lib/hive-jdbc-1.2.2-standalone.jar",
            identifier.quote="`")

conn <- dbConnect(drv, "jdbc:hive2://127.0.0.1:10000/used_cars", "maria_dev", "maria_dev")
```

First we are getting the EVENTS table description. Full description of the attributes presented on Classified Ads for Cars home page.

```
descr <- dbGetQuery(conn, "describe events")
```

|    | col_name            | data_type     | comment |
|----|---------------------|---------------|---------|
| 1  | maker               | string        |         |
| 2  | model               | string        |         |
| 3  | mileage             | int           |         |
| 4  | manufacture_year    | int           |         |
| 5  | engine_displacement | int           |         |
| 6  | engine_power        | int           |         |
| 7  | body_type           | string        |         |
| 8  | color_slug          | string        |         |
| 9  | stk_year            | string        |         |
| 10 | transmission        | string        |         |
| 11 | door_count          | int           |         |
| 12 | seat_count          | int           |         |
| 13 | fuel_type           | string        |         |
| 14 | date_created        | timestamp     |         |
| 15 | date_last_seen      | timestamp     |         |
| 16 | price_eur           | decimal(13,2) |         |

Code below selecting first rows for EVENTS table and saves them to 'cars' dataframe. Note that we rename cars columns for better presentation. Also note that it looks that the dataset has some missing values.

The EVENTS table has 3,552,912 records which could be confirmed by running the HQL statement below.

```
dbGetQuery(conn, "select count (*) from events")
```

For the analysis we will extract not more than 30K rows from EVENT table randomly using the HQL statement below. We skip the rows mising the most important attributes:

```
filter <- paste(
      " maker <> ''",
      " AND model <> ''",
      " AND mileage is not NULL",
      " AND manufacture_year is not NULL",
      " AND price_eur is not NULL"
)
count <- dbGetQuery(conn, paste("select count(*) from events", " WHERE", filter))
cars.sample.totalFilered <- count$`_c0`

car.sample.maxSize <- 30000
lim <- car.sample.maxSize/cars.sample.totalFilered

sample_HQL <- paste(
  "select * from events WHERE", filter,
      " AND rand(123) < ", lim,
  " limit ", car.sample.maxSize
)

cars.sample <- dbGetQuery(conn, sample_HQL)
colnames(cars.sample) <- c(
  "Maker", "Model","Mileage","Year", "Disp", "Pwr", "Body", "Color", "Sticker",
  "Trans", "Doors", "Seats", "Fuel", "Listed", "Removed", "Price")

nrow(cars.sample)

#> [1] 29958
```

It is very important to disconnect from the HIVE at the end of the session:
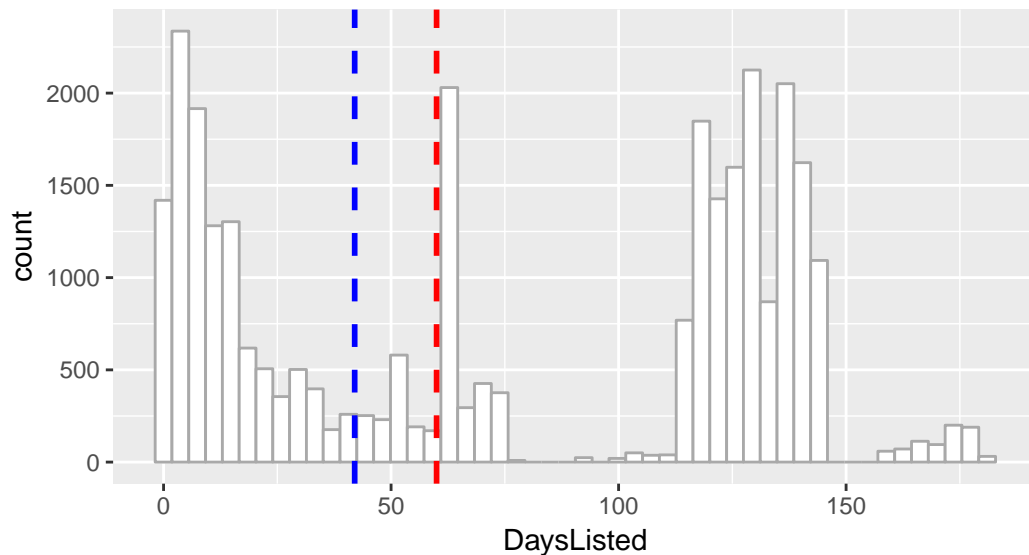
```
dbDisconnect(conn)
```

**Figure 1:** Days Cars Listed

```
#> [1] TRUE
```

Creating additional columns for analysis

```
cars.sample$ListedTS <- strptime(cars.sample$Listed, '%Y-%m-%d %H:%M:%OS')
cars.sample$RemovedTS <- strptime(cars.sample$Removed, '%Y-%m-%d %H:%M:%OS')

cars.sample$Age <- as.integer(ceiling(
  difftime(cars.sample$ListedTS, strptime(cars.sample$Year,'%Y'), units = "days")/365))

cars.sample$DaysListed <- as.integer(ceiling(
  difftime(cars.sample$RemovedTS, cars.sample$ListedTS, units = "days")))
```

How long the cars are usually listed?

```
ggplot(cars.sample, aes(x=DaysListed)) +
  geom_histogram(color="dark grey", fill="white", bins=50) +
  geom_vline(aes(xintercept=42), color="blue", linetype="dashed", size=1) +
  geom_vline(aes(xintercept=60), color="red", linetype="dashed", size=1)
```

Let's consider cars listed less than 42 days (6 weeks) to be sold

```
cars.sample$Sold <- cars.sample$DaysListed <= 42
```

What is the distribition of advertized cars age?

```
ggplot(cars.sample, aes(x=Age)) +
  geom_histogram(color="dark grey", fill="white", binwidth = 1) +
  scale_x_continuous(limits = c(0, 30))+
    geom_vline(aes(xintercept=mean(Age, na.rm=T)),
               color="green", linetype="dashed", size=1)
```

What is the distribition of mileage of the sold cars?

```
ggplot(cars.sample, aes(x=Mileage)) +
  geom_histogram(color="dark grey", fill="white", bins=25) +
  scale_x_continuous(limits = c(0, 250000))+
  scale_y_continuous(limits = c(0, 2200))+
    geom_vline(aes(xintercept=mean(Mileage, na.rm=T)),
               color="green", linetype="dashed", size=1)
```

What is the most advertized vs sold car maker?

```
require(forcats)
```
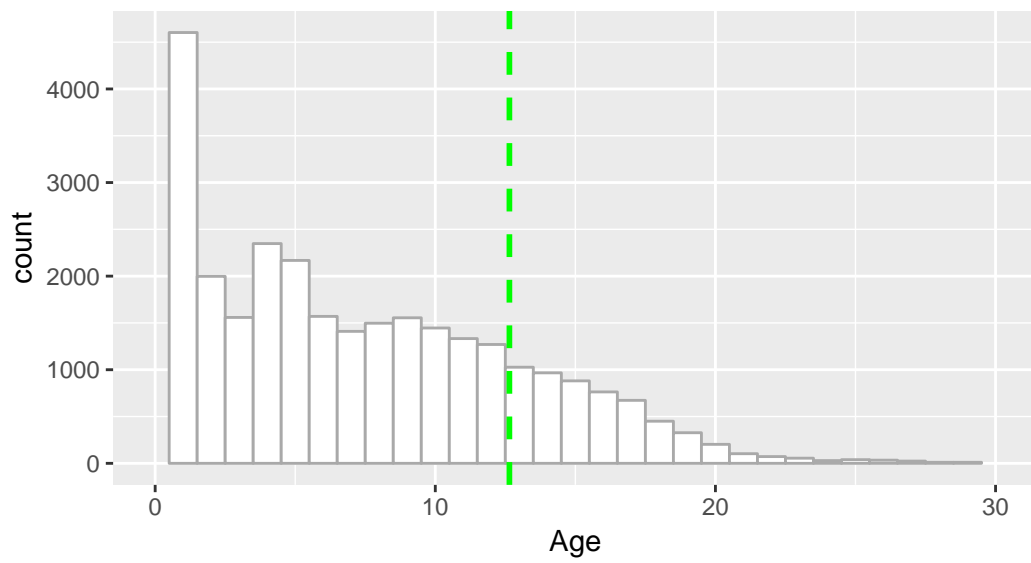
**Figure 2:** Number of Ads by Maker
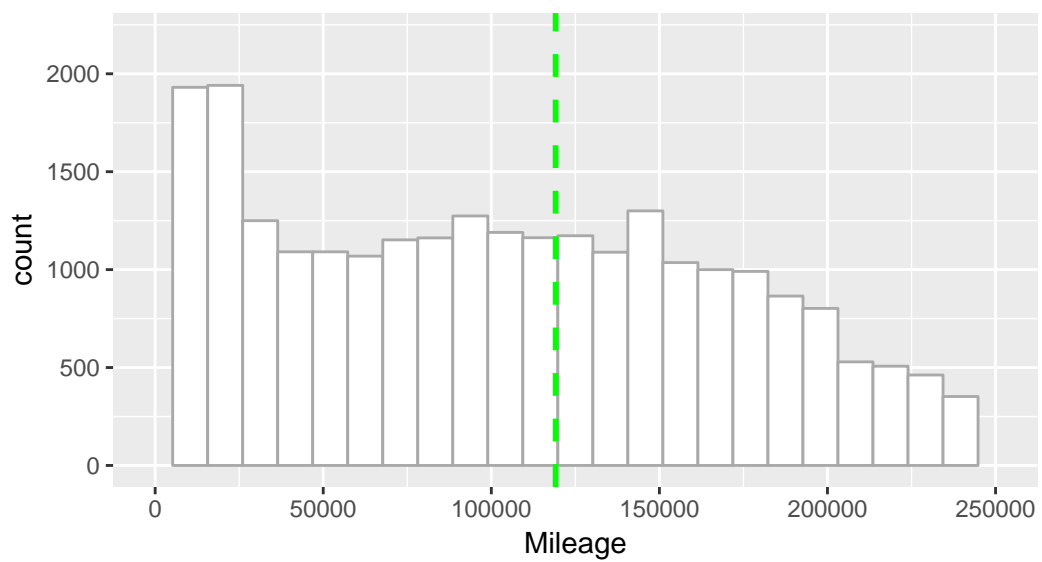


**Figure 3:** Mileage distribution

| | Maker | Model | Year | Mileage | Age | DaysListed | Sold | Price |
|---|---|---|---|---|---|---|---|---|
| 112 | skoda | octavia | 2002 | 147550.00 | 14 | 75 | FALSE | 4067.36 |
| 357 | hyundai | i40 | 2012 | 93183.00 | 4 | 70 | FALSE | 13323.46 |
| 388 | ford | galaxy | 1998 | 296006.00 | 18 | 68 | FALSE | 1220.95 |
| 938 | seat | altea | 2012 | 225000.00 | 4 | 61 | FALSE | 9807.55 |
| 4237 | lancia | musa | 2008 | 134814.00 | 8 | 4 | TRUE | 6500.96 |
| 4427 | lancia | y | 1999 | 212696.00 | 17 | 57 | FALSE | 555.03 |
| 4836 | ford | fiesta | 2014 | 14884.00 | 2 | 1 | TRUE | 11884.38 |
| 5563 | mini | cooper | 2011 | 61000.00 | 5 | 31 | TRUE | 13806.62 |
| 8409 | mitsubishi | pajero | 2001 | 126000.00 | 15 | 166 | FALSE | 5400.59 |
| 10346 | ford | focus | 2014 | 15000.00 | 2 | 142 | FALSE | 16600.00 |
| 12780 | toyota | prius | 2011 | 18500.00 | 5 | 136 | FALSE | 16770.00 |
| 13862 | audi | a1 | 2014 | 15401.00 | 2 | 134 | FALSE | 20990.00 |
| 13934 | volkswagen | golf | 2000 | 142000.00 | 16 | 134 | FALSE | 2300.00 |
| 15495 | volvo | v70 | 2007 | 155750.00 | 9 | 130 | FALSE | 8800.00 |
| 16321 | volkswagen | golf | 2015 | 2500.00 | 1 | 128 | FALSE | 18770.00 |
| 16364 | volkswagen | caddy | 2013 | 43053.00 | 3 | 128 | FALSE | 13990.00 |
| 16438 | citroen | c4-picasso | 2008 | 113100.00 | 8 | 128 | FALSE | 6200.07 |
| 16557 | ford | c-max | 2006 | 139000.00 | 10 | 128 | FALSE | 4899.00 |
| 16812 | renault | captur | 2015 | 23402.00 | 1 | 127 | FALSE | 15925.91 |
| 19277 | smart | fortwo | 2013 | 33000.00 | 3 | 121 | FALSE | 5798.15 |
| 20701 | mazda | 2 | 2016 | 1000.00 | 0 | 118 | FALSE | 21890.00 |
| 21764 | renault | twingo | 2015 | 9904.00 | 1 | 116 | FALSE | 11900.00 |
| 22167 | lexus | 300 | 2016 | 0.00 | 0 | 115 | FALSE | 29854.77 |
| 22393 | opel | meriva | 2008 | 139776.00 | 8 | 103 | FALSE | 3663.95 |
| 23127 | ford | mondeo | 2016 | 8.00 | 0 | 61 | FALSE | 11465.58 |
| 23610 | skoda | fabia | 2009 | 96000.00 | 7 | 61 | FALSE | 1295.34 |
| 24085 | hyundai | getz | 2004 | 108897.00 | 13 | 61 | FALSE | 1295.34 |
| 27484 | volkswagen | passat | 2010 | 400000.00 | 7 | 41 | TRUE | 1295.34 |
| 27943 | peugeot | 407 | 2008 | 160150.00 | 9 | 33 | TRUE | 1295.34 |
| 28172 | peugeot | 307 | 2002 | 131800.00 | 15 | 4 | TRUE | 1295.34 |

**Table 1:** Rows from Sample Car Ads Dataset

```
#> Loading required package: forcats

total <- nrow(cars.sample)
ggplot(cars.sample, aes(fct_rev(fct_infreq(Maker)), fill=Sold)) +
      geom_bar() +
      labs(x="", y="") +
       scale_y_continuous(labels = function(x) sprintf("%.0f%%",x/total*100)) +
      coord_flip()
```

What is the 20 best advertived vs sold car models?

```
require(forcats)
total <- nrow(cars.sample)
cars.sample$Car <- paste(cars.sample$Maker, cars.sample$Model)
betsCarsList <- fct_infreq(cars.sample$Car)
cars.sample.bestCars <- cars.sample[cars.sample$Car %in% levels(betsCarsList)[1:20],]
ggplot(cars.sample.bestCars, aes(fct_rev(fct_infreq(Car)), fill=Sold)) +
      geom_bar() +
      labs(x="", y="") +
      scale_y_continuous(labels = function(x) sprintf("%.0f%%",x/total*100)) +
      coord_flip()
```

What is the best 20 advertised vs sold cars?

```
require(forcats)
total <- nrow(cars.sample)
cars.sample$Car1 <- paste(cars.sample$Maker, cars.sample$Model, cars.sample$Year)
betsCarsList <- fct_infreq(cars.sample$Car1)
```
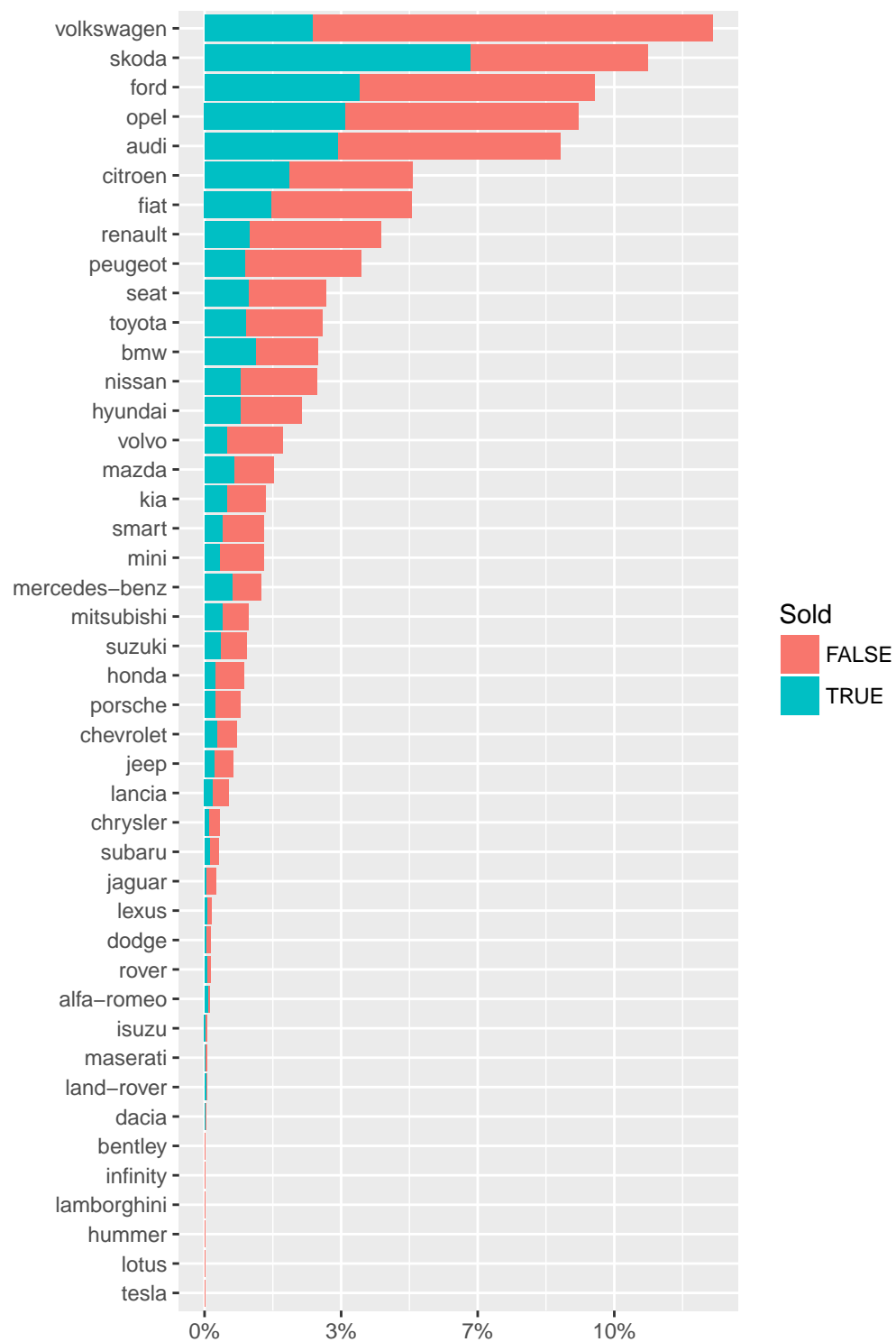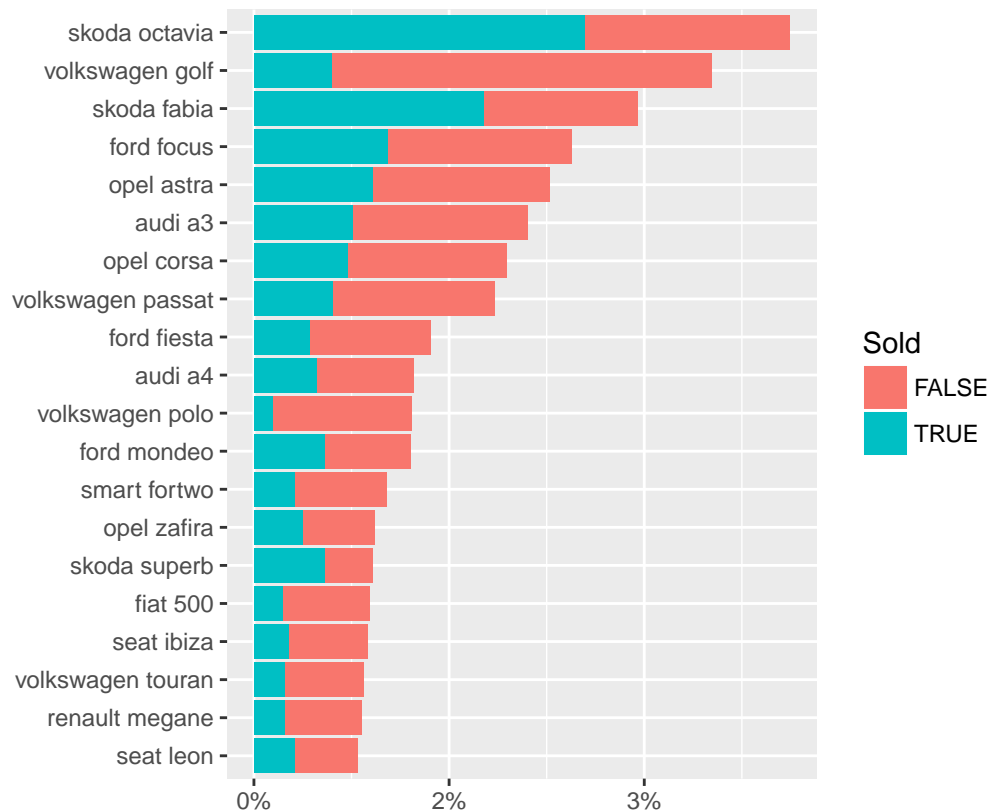
**Figure 4:** Number of Ads by Maker

**Figure 5:** 40 Best Car Models

```
cars.sample.bestCars <- cars.sample[cars.sample$Car1 %in% levels(betsCarsList)[1:20],]
ggplot(cars.sample.bestCars, aes(fct_rev(fct_infreq(Car1)), fill=Sold)) +
      geom_bar() +
      labs(x="", y="") +
      scale_y_continuous(labels = function(x) sprintf("%.2f%%",x/total*100)) +
      coord_flip()
```

What is the distribution of car prices in the ads for the cars that were not sold?

```
ggplot(cars.sample[!(cars.sample$Sold),], aes(x=Price)) +
  geom_histogram(color="dark grey", fill="white", bins=40) +
  scale_x_continuous(limits = c(0, 40000)) +
    geom_vline(aes(xintercept=mean(Price, na.rm=T)), color="red", linetype="dashed", size=1)
```

What is the distribution of car prices of the cars that were sold?

```
ggplot(cars.sample[cars.sample$Sold,], aes(x=Price)) +
  geom_histogram(color="dark grey", fill="white", bins=40) +
  scale_x_continuous(limits = c(0, 40000)) +
    geom_vline(aes(xintercept=mean(Price, na.rm=T)), color="red", linetype="dashed", size=1)
```

## Conclusion

The project was a success.

## Bibliography

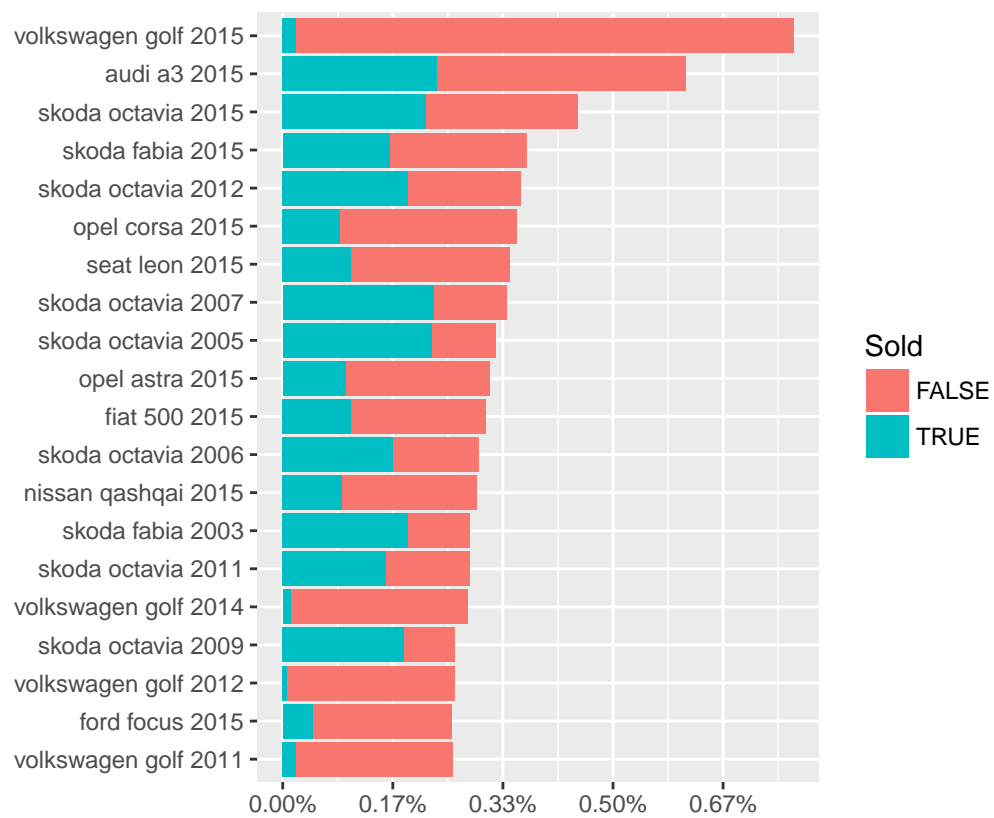Classified Ads for Cars. Classified ads for cars | kaggle. URL https://www.kaggle.com/mirosval/
    personal-cars-classifieds/home. [p1]
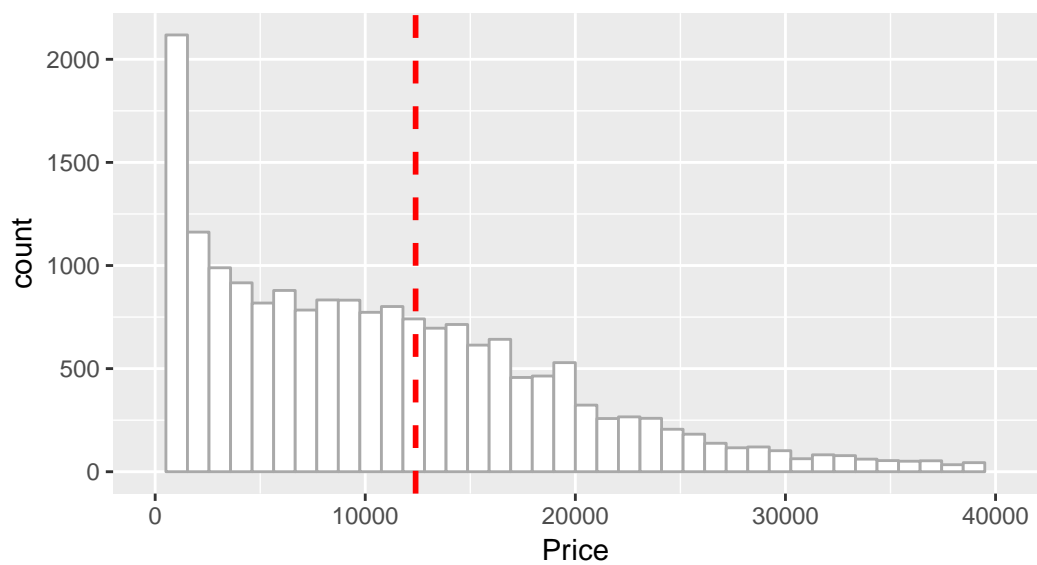
**Figure 6:** 20 Best Cars



**Figure 7:** Prices asked for the cars not sold
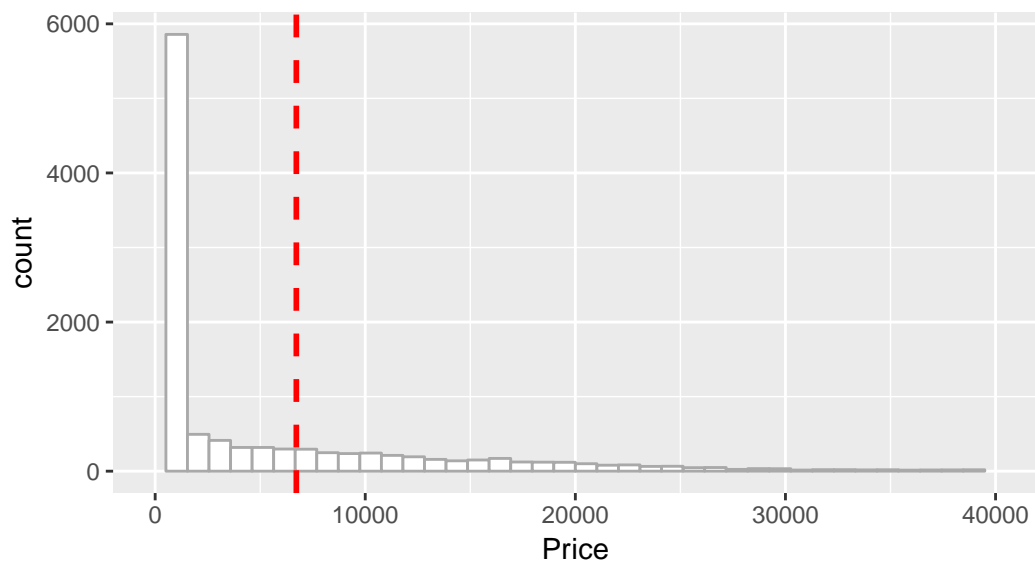
**Figure 8:** Prices asked for the sold cars

D. B. Dahl. *xtable: Export Tables to LaTeX or HTML,* 2016. URL https://CRAN.R-project.org/package=xtable. R package version 1.8-2. [p4]

Hortonworks Sandbox. Sandbox deployment and install guide. URL https://hortonworks.com/tutorial/sandbox-deployment-and-install-guide/. [p1]

## Note from the Authors

This file was generated using *The R Journal* style article template, additional information on how to prepare articles for submission is here - Instructions for Authors. The article itself is an executable R Markdown file that could be downloaded from Github with all the necessary artifacts.

*Igor Baranov*
*York University School of Continuing Studies*

https://learn.continue.yorku.ca/user/profile.php?id=21219