Universitat Rovira i Virgili (URV) y Universitat Oberta de Catalunya (UOC)
Master in Computational and Mathematical Engineering

FINAL MASTER PROJECT

Area: Data Science

Generation of information to support the decision at work risk prevention area

A generalizable Data Science study in the context of Big Data

---

Autor: Alberto Burgos Plaza

Tutor: Dr. Agusti Solanas Gómez

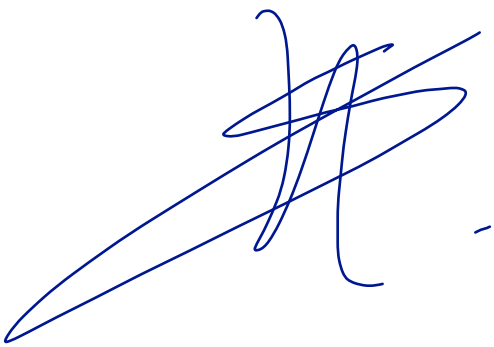
Barcelona, July 11, 2019

Credits/Copyright

This work is subject to a licence of Attribution-NonCommercial-NoDerivs 3.0 of Creative Commons

FINAL PROJECT SHEET

Title:	Generation of information to support the decision at work risk prevention area. A generalizable Data Science study in the context of Big Data.
Autor:	Alberto Burgos Plaza
Tutor:	Dr. Agusti Solanas Gómez
Fecha de entrega (mm/aaaa):	07/2019
Program:	Master in Computational and Mathematical Engineering
Area:	Data Science
Language:	English
Key words	Data Science, Artificial Inteligence, Big Data



Dedicatory / Quote

A mi pareja y familia, por su apoyo incondicional durante el desarrollo de este trabajo y del máster en general.

Acknowledgments

Este trabajo no hubiera sido posible sin la ayuda del director del TFM, Agusti Solanas. Gracias por ayudarme a encaminar este proyecto, a resolver los problemas surgidos y por los ánimos recibidos durante el desarrollo.

Gracias también a los miembros de la empresa Easytech Global por facilitarme los datos necesarios para el desarrollo de este estudio y por resolverme las dudas sobre estos.

Abstract

The field of Data Science has been growing during last years as data has been gaining importance on all scenarios. When Data Science meets Big Data things get complicated. An easy work can be a tedious one since the way to carry out is radically different. Also, time, monetary and compute resources are finite, thus the used strategies to exploit them takes relevance. At this project were selected the most suitable implementations of them in order to carry out a generic Data Science study with this characteristics. Apache Spark, H2O and XGBoost were choosen and compared to other alternatives.

At a second part of this project, the suitability of the selected solutions were proven by a Data Science study based on Labor Risks Prevention. This study tried to answer some questions about the data by an explanatory analysis and/or machine learning models. Some problems were found to answer them, as the lack of data, but finally a way to answer the questions was found in all cases. Due to these problems, the results weren't strong, but a good first approximation of their resolutions.

Keywords: Data Science, Big Data, Labor Risks Prevention

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work tries to explain or infer aspects related to **Labor Risks Prevention** based on the data recorded in *Sabentis Pro* [1] management platform by the mutual insurance company **Colmena Seguros** [2]. *Sabentis Pro* is a labor risks prevention management system developed by **Easy Tech Global** [3] which includes, among other data, accidents, illnesses, worker and company data, risks assessed, risk factors and dangers. These data are mainly managed by the companies where the insured workers work. Colmena Seguros is one of the most important labor mutuals in **Colombia**. It has around 41.000 affiliated companies and 901.000 affiliated workers [4], being the fourth largest labor risk manager in Colombia regarding the volume of companies and workers.

1.1 Context and justification

The labor risks prevention sector includes a series of activities and measures whose purpose is to avoid or reduce the damages derived from work activity such as illnesses, accidents, injuries or pathologies. In order to achieve this, the evaluation of labor risks is used as a main tool, obtaining from it the necessary preventive measures to avoid or reduce these risks.

As in other areas, the labor risks prevention accumulates a large amount of data. These data can be related to different damages suffered by a worker, the risks evaluated and the measures taken, the aspects related to a worker or a company, etc. That is why the explotation of such amount of data can offer to the company a great help in taking decisions, what is usually called **data-driven company**. The exploitation of data can also help to verify if certain actions are helping to meet certain objectives, understand the reason of certain circumstances or obtain

certain relationships that help to predict certain results.

Studies with such amount of data is often a task that requires some complexity since the data itself usually exceeds the computing resources available. For this reason, around the term **big data**, a series of technologies that help to process data of these magnitudes have emerged over the last few years. Nowadays, the specialization obtained in these technologies is such that it is necessary to know which are appropriate for the problem to be solved with the available data.

1.2 Objectives

Once explained the context and justified the work, the proposed objectives are shown:

- Study of the different ways to deal with large amounts of data in R:
  - Study of the approaches and implementations of techniques in R in order to overpass the main memory limitation.
  - Study of the main paradigms and implementations of distributed computation of the tasks done in processes related to data science studies and its integration with R. Mainly the study will be focused on main big data technologies demanded nowadays to data scientists.
  - Selection of most appropriate technologies to accomplish the data analysis in this project and preparation of the work environment with them.
- Data science study about the subject and data related to this project:
  - Study and selection of working frameworks appropriated to the study.
  - Formulation and selection of relevant questions about the data taking into account the point of view of an expert in the field.
  - Extraction of the data from sources and load into the work environment.
  - Iteration over the selected framework steps in order to answer the formulated questions.

1.3 Focus and methodology

This project has the focus of being a generalizable way to introduce data science field in big data. All decisions taken have to go in the direction of a generalizable vision of all data science

solutions, not only regarding R, and with a future vision of them. Also, the solutions selected and methodologies followed needs to popular and suitable for this project.

My personal objective with this project is to be a way to start with my job career in data world and an opportunity to practice the knowledges acquiered in some subject of the Master's Degree, thus a deep study of all topics covered will be carried out.

1.4 Project planning and communication

Once objectives are clear, its needed to carry out a planning of their related tasks over a period of 6 months, where main milestones and the critical path to follow have to be defined. Being a data science work, this has an iterative procedure, therefore, two iterations will be planned as a way to approximate all needed work. These iterations will absorb the possible risks that may appear during the development of the project without affecting too much the general planning.

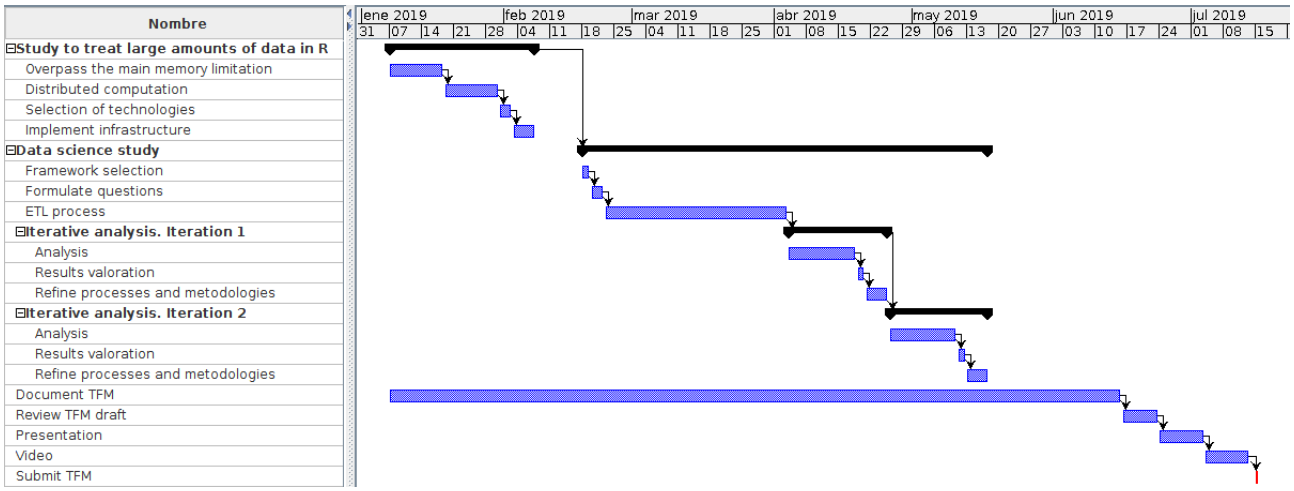


Figure 1.1: Gantt chart of planification

Throughout the development of the work, communication with the director of the TFM has been maintained with certain regularity, informing all time of the fulfillment of the objectives and the problems that have happened. Also, a meeting and different contacts with the data provider, Easy tech global, have been carried out with the aim of obtaining all the necessary information regarding the data and the possible questions to be asked about them.

1.5 Sections summary

The following is a brief explanation of every section of this memory:

1. **Introduction**: This is a comon section of all memory jobs. Here, job context, justification and focus are explained. Also, objectives and its planning are detailed.

2. **Big Data in R**: Here, R limitations are detailed. In order to overcome these limitations, strategies and implementations of them are studied and selected. Finally, the setup of a work enviroment with them is explained.

3. **Data Science study**: This is the biggest part of this document. Here, a brief explanation of the possible work methodologies to follow during the development of the data science study is made, finishing with a selection of them. Afterwards, data source used at this study and its particularities is detailed. Also, the ETL process regarding the source and the work environment is explanied. Finally, an iterative data science study using the methodology selected as a guide is detalied.

4. **Conclusions**: Final conclusions of all work done at this project are detailed here. Also, future work lines related with this project are commented.

Chapter 2

Big Data in R

2.1 R limitations

R is a well used programming environment in data science, mainly due to the abundance of packages related with this field and the interactivity between them, that brings a great flexibility to carry out different works. Also, it's very friendly to non programmer users, a thing that facilitates its use to those data scientist that comes from other fields different than computer science. For this reason, R is at top of used data science languages [5], also at top of demanded skills for data science jobs [6].

Despite these benefits, R has some limitations, specially when big data is present. R by default uses main memory to save all objects during an R session, representing this a limitation when working with datasets bigger than main memory resources. R's base functions loads all the dataset, do copies of it and save all final and intermediate results on main memory. This behavior affects R's base objects like *data.frame*, *tibble* or *matrix*, and their related functions. Another limitation of R is the single thread execution. All R's base functions only use one CPU core of all available, a thing that affects to high demand tasks.

2.2 Strategies to handle large amounts of data

There are some strategies that can be applied to overcome R limits with big data. Below are explained some of them.

2.2.1 External memory algorithm

External memory model (or **disk access model**) is a computational model that consists of a **processor** with an **internal memory** of size M, connected to an **unbounded external memory**. Both, the internal and external memory, are divided in **blocks** of size B (see figure 2.1). One input/output, or memory transfer operation, consists of moving a contiguous amount of these blocks from external to internal memory, being the running time of an algorithm determined by the number of these input/output operations. Using this computational model, an **external memory algorithm** (or **out-of-core algorithm**) [7] can be implemented in order to overpass main memory limitation.

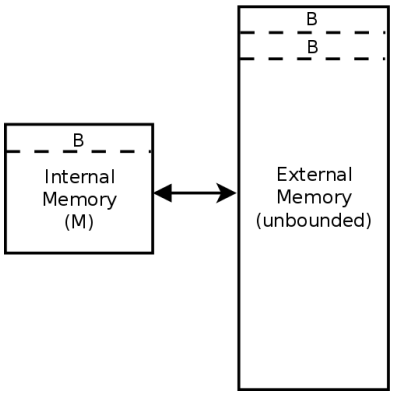


Figure 2.1: External memory model

2.2.2 Online algorithm

Online learning [8] is a way to implement machine learning algorithms which consists in store and proces only one training example at a time sequentially. It assumes an initial predictive model and updates its parameters for future predictions at each step. Since online learning algorithms processes only one training example at a time, its main memory requirements are met by an average system. Nevertheless, due it way to update its weights at learning time, it's very noisy and may go far away from the ideal gradient direction.

2.2.3 Divide and Recombine

Divide and Recombine [9] is a strategy where first, a dataset is divided into subsets, being these small enough to be loaded into memory. The division, depending of the dataset, can be done using random sampling without replacement or by a conditioning-variable. The subsets are

persistent, and can be stored across multiple disks and nodes in a cluster, therefore subsequent actions will be done in a parallel way. Afterwards, an analysis task is applied to each subset, and the obtained results are recombined in order to obtain a statistical result. This recombination is an approximation of the result obtained if data had been processed as a whole. Recombinations can be a numerical or visual result.

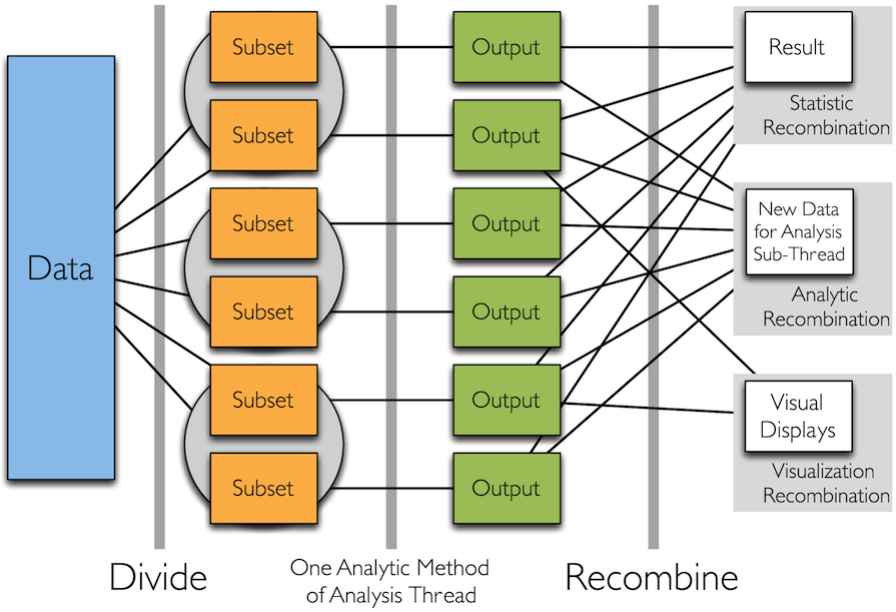


Figure 2.2: Divide and Recombine

Source: https://deltarho.org

A generic storage mechanism to perform this strategy is the *key-value* store. The key-value store consists in pairs where the key is an unique identifier or object that describes the subset, whereas the value contains the data of the subset.

Most important implementation of this strategy is *MapReduce* [10] technique. In MapReduce, three steps, **map**, **shuffle** and **reduce**, are done over every node that stores the divisions. Map step is applied to a local collection of input key-value pairs in parallel, outputting a new set of key-value pairs. The input and output types of the map can be (and often are) different from each other. Afterwards, shuffle step is applied over these new key-value pairs, grouping the results by the keys. Shuffle step implies sort and exchange operations, thus is important to take into account the division of the dataset, specially if these operations are time expensive. Finally, the reduce step is applied in parallel over every key-value pair locally stored.

2.2.4 Data Base Management System

A **Data Base Management System (DBMS)**, is a system that enables users to define, create, maintain and control access to a database, storing all data in files at local storage. This is an easy solution to overcome the main memory limitation since this allows to load into main memory only a subset of the data needed.

2.2.5 Distributed collection

Distributed collections based frameworks are an abstraction to work with a collection of data which provides a processing engine in order to do computations on the collection in a distributed way. These frameworks abstract the users from the distributed nature of underlying data, allowing to do aggregations and transformations over it in an easiest way. It allows the creation of tasks with higher complexity than divide and recombine ones because tasks are not constrained to phases nor their sequence.

2.2.6 Embarassing parallelization

An **embarrassingly parallel** workload or problem is that where little or no effort is needed to separate the problem into a number of parallel tasks. This needs the condition that every data element is independent (or almost) of each other. Tasks as bootstrapping, cross-validation, missings imputation or fitting multiple regression models can be an example of candidates to be parallelized this way.

2.3 Accessible solutions from R

The following implementations are a representation of some of the previous shown strategies. From all the implementations available at the R's ecosystem, the ones explained below are certainly the most popular.

2.3.1 Bigmemory

Bigmemory project [11] offers packages for two purposes. First, it provides a minimalist framework to manage and explore large datasets. As main object, *Bigmemory* uses **big.matrix**,

which are two files, a binary representation of a matrix similar to R's matrix and a file descriptor that holds all metadata (rows, columns, names, etc). Data are kept on disk across sessions and are moved to main memory implicitly. Second, *Bigmemory* project is designed in a way that can be used efficiently as a building block in parallel programming using parallel packages as *foreach*, *snow*, etc.

There are some sister packages that allows to expand the use of big.matrix objects as *biganalytics*, *bigpca* or *bigrf*.

As a disadvantage, *Bigmemory* limits its use to numerical matrices. Also, adding new rows or columns isn't efficient as a whole copy of the matrix is needed.

2.3.2 ff

The *ff* package [12] replaces R's in-RAM storage mechanism with on-disk storage. Data structures are stored on disk but behave as if they were in RAM by transparently mapping only a chunk in main memory. Unlike *bigmemory*, *ff* supports R's atomic data types and R vector types such as factors.

Using **ffdf** class ff can deal with data frames and import/export with CSV files.

ff objects store raw data in binary flat files and complement these with metadata like physical and virtual attributes stored in R session. *ff* objects can be stored and reopened across R sessions and shared by multiple *ff* R objects in a same process or from multiple R processes in order to exploit parallelism.

2.3.3 MonetDBLite

MonetDB [13] is a column store database optimized to allow the management of big data sources thanks to its storage model based on vertical fragmentation, a modern CPU-tuned query execution architecture, automatic and self-tuning indexes, run-time query optimization, and a modular software architecture.

Inside R ecosystem, package *MonetDB.R* [14] acts as an extension to connect to *MonetDB* databases and analyze its data with considerable performance. This connector provides a general-purpose *DBI* driver that enables *MonetDB* support using *dplyr* tools.

As a way to simplify the installation process to R users, package *MonetDBLite* [15] provides a fully embedded version of *MonetDB* database that runs within the R process. This package

also installs the latest version of *MonetDB.R*.

2.3.4 Apache Spark

Apache Spark is an open source multi-node cluster that works as data processing engine. Its API allows developers to carry out machine learning or iterative query workloads. Also, it can perform batch processing which means that it can carry out a group of operations collected over a period of time over a large volumes of data, increasingly this way the efficiency rather than processing each individually. By default, all operations run on main memory due to its in-memory cluster computation capability, decreasing considerably running time.

Spark use a data abstraction called *RDD (Resilient Distributed Dataset)*, an inmutable collection of objects which form the fundamental data structure in *Spark*. Every dataset in *RDD* is logically partitioned across many nodes so it allows a programmer to perform in-memory computations on large clusters in a fault-tolerant manner. In order to facilitate data analysis operations, a higher-level abstraction, *Spark's DataFrame*, allows developers to impose a structure onto a distributed collection of data. *DataFrame* is also inmutable and its data are organized into named columns.

In order to compute complex jobs, *Spark* use *DAG (Directed Acyclic Graph)*, a set of vertices and edges, where vertices represents the *RDDs/DataFrames* and the edges represents the operation to be applied on. Every edge connects from earlier to later in the sequence with no directed cycles. When a task is submited to *Spark*, a new *DAG* is created and a scheduler splits the graph into the stages of the task. *DAG* is a strict generalization of *MapReduce* model but with a better optimization of its operations.

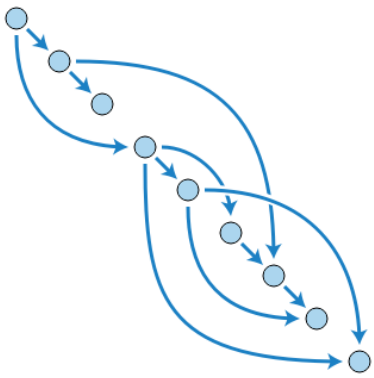


Figure 2.3: Directed Acyclic Graph

Spark is designed in a way that it integrates with other big data tools. This is the case of

Hadoop clusters, where it can run on. Nevertheless, *Spark* can also run independently on its own cluster management system.

A set of machine learning algorithms and feature-related operations is offered by *Spark's MLib*, running them in-memory and in a multi-threaded way. Also, *Spark* provides *Hive* query language style for querying on structured data.

Some high-level APIs are offered to *Scala*, which is the used language for the *Spark* development, *Java*, *Python* and R. In R, we can use *SparkR* implementation package which supports *MLib* models, parallel execution, *Spark* dataframes management, execution of arbitrary *Scala* code and support of different execution modes as *Yarn*, *Mesos*, standalone or *Kubernetes*. Moreover, exists a higher level implementation, *sparklyr*, which uses *SparkR* and provides also support for other R packages as *dplyr*, *DBI* or *broom*. Also provides support to machine learning pipelines, graph processing or connections to *RStudio*.

2.3.5 DeltaRho

DeltaRho project, previously known as *Tessera*, is based on the **Divide an Recombine** approach. This project maintains two D&R solutions, *Trelliscope* and *datadr*.

Trelliscope [16] provides a visualization database system for storing and managing visual artifacts for analysis, as well as a visual recombination system for creating and viewing very large multipanel displays. *Trelliscope* displays can be viewed in an interactive viewer that provides several modes for sorting, filtering, and sampling panels of a display.

On the other hand, *datadr* [17] provides commands that makes the divisions, analytic methods, and recombinations in an easy way. In addition, *datadr* also provides several tools for reading and manipulating data, as well as a collection of division-independent methods that can compute things such as aggregations, quantiles, or summaries across the entire dataset, regardless of how the data are partitioned.

In order to do all D&R tasks, *DeltaRho* uses several backends that performs *MapReduce*. These backends are *R MapReduce*, to compute a multicore operation on local disk, *HDFS* and *Hadoop MapReduce* via *RHIPE*, to compute a scalable task over a *Hadoop* cluster, and *Spark MapReduce* via *SparkR*, to compute scalable task over a *Spark* cluster.

2.3.6 RevoScaleR

The *RevoScaleR* [18] library is part of *Microsoft Machine Learning Services* [19] and *Microsoft R* [20] products. The library includes data transformation and manipulation, visualization, predictions, and statistical analysis functions. It also includes functions for controlling jobs, serializing data and performing common utility tasks. The functions are orientated around three main abstraction concepts that users can specify to process large amount of data that may not fit in memory and also exploit parallel resources to speed up the analysis.

1. Compute context: *RevoScaleR* can run locally or remotely. If runs locally on a standalone *Linux* or *Windows* system, data and operations are local to the machine. *RevoScaleR* can run on *Hadoop* or *Spark* processing frameworks, thus a local compute context means that data and operations are local to the execution environment. On the other side, if runs remotely, the script running on a local *R Client* or *Machine Learning Server* shifts execution to a remote *Machine Learning Server* [21], including *Hadoop* or *Spark* clusters. On *SQL Server*, there are two primary use cases for remote compute context; a call of R functions inside a *T-SQL* script or stored procedures running on *SQL Server*, or a call of *RevoScaleR* functions in an R script that executes on a *SQL Server* machine.

2. Data source: Data source defines where the data comes from. There are various data sources available in *RevoScaleR*, such as text data, in-SQL data, *Spark DataFrame* or a binary dataset implementation called *XDF* [22]. *XDF* is persistent stored columnar storage that is processed by chunks.

3. Analytics: Analytic functions in *RevoScaleR* takes data source object, a compute context and the other parameters needed to build the specific model. In addition to these parameters, one can also specify the level of parallelism, such as the size of the data chunk for each process or the number of processes to build the model.

2.3.7 XGBoost

XGBoost (eXtra Gradient Boosting) [23] is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the *Gradient Boosting* framework as *GBDT* or *GBM*. The library provides a system for use in a range of computing environments, allowing parallelization of tree, distributed computing for use in a cluster, out-of-core computing and cache optimization. Algorithms includes key features as sparse aware implementation with automatic handling of missing data

values, a block structure to support the parallelization of tree construction and a continued training so that one can further boost an already fitted model on new data.

XGBoost dominates structured datasets on classification and regression predictive modeling problems. The evidence is that it is the go-to algorithm for competition winners on **Kaggle** competitive data science platform [24].

In R, *xgboost* [25] package provides the ability to use *XGBoost* library. This package provides parallel computation using *OpenMP*, generally being ten times faster than R's package *gbm*. Allowable input types are R's matrix, local data files or its own class, *xgb.DMatrix*. Also, it supports customized objective and evaluation functions.

2.3.8 H2O

H2O [26] is an open source, distributed in-memory machine learning platform with linear scalability. H2O supports the most widely used statistical and machine learning algorithms including gradient boosted machines, generalized linear models, deep learning or *XGBoost* (used as a backend).

A distributed **key-value** store is used to access and reference data, models, objects, etc, across all machines. The algorithms are implemented on top of *H2O's* distributed *MapReduce* framework and utilize the *Java fork/join* framework for multi-threading. The data are read in parallel, distributed across the cluster and stored in memory in a columnar format in a compressed way. *H2O's* data parser has built-in intelligence to guess the schema of the incoming dataset and supports data ingest from multiple sources in various formats in order to create its own object, an *H2O frame*.

H2O also has *AutoML* functionality which automatically runs through some types of algorithms and their hyperparameters to produce a leaderboard of the models that better fit the data.

From R, package *h2o* [27] acts as an interface of *H2O*, providing the capacity of start an *H2O* cluster and compute inside it all sort of machine learning algorithms.

Another sister product, *Sparkling Water* [28], allows users to combine the fast, scalable machine learning algorithms of *H2O* with the capabilities of *Spark*. With *Sparkling Water*, users can drive computation from R, using *rsparkling* [29] package.

2.3.9 Vowpal Wabbit

The *Vowpal Wabbit* [30] project is a fast **out-of-core learning** system sponsored by **Microsoft Research** and (previously) **Yahoo! Research**. There are several optimization algorithms available, loss functions, supervised (and semi-supervised) learning problems and representation algorithms.

The learning algorithms are pretty fast. They can be effectively applied on learning problems with a sparse terafeature. The parsing of input and learning tasks are done in separate threads, exploiting this way multi-core CPUs. Also, the memory footprint of the program is bounded independently of data. This means that the training set is not loaded into main memory before learning starts. In addition, the size of the set of features is bounded independently of the amount of training data using the hashing trick, which converts feature identities to a weight index via a hash.

The input format for the learning algorithm is flexible. It can be binary, numerical or categorical (via flexible feature-naming and the hash trick). Also, it can deal with missing values/sparse-features. It can have features consisting of free form of text, which is interpreted in a bag-of-words way.

For R, exists a recent package started at **Google Summer of Code 2018** called *rvw* [31], that aims to bring all the *Vowpal Wabbit* functionality to R.

2.3.10 parallel and doParallel

Package *parallel* [32] is build on top of packages *multicore* and *snow*, providing drop-in replacements for most of the functionality of them. In addition, it contains an integrated implementation to handle random number generation. *parallel* package provides parallel implementations of **apply** functions with **mclapply** and **parLapply**. The former uses forking to parallelize the code whereas the later uses sockets.

Another package, *doParallel* [33], is a backend of parallel for the *foreach* package in order to execute foreach loops in parallel.

2.4 Selected solutions and justification

Taking into account the data source and the problem to solve, the selected solutions have been *Apache Spark*, *H2O* and *XGBoost*.

First of all, *Apache Spark* can provide everything needed to work with large volumes of data. Exploratory analysis, feature engineering or machine learning modeling tasks required at this study can be performed with *Spark*. In addition, *Spark* can **scale horizontally** (or **scale-out**). In R, the interaction with *Spark* is very simple through the use of *sparklyr* package and *dplyr* functions, being able to work, with some exceptions, in a same way as with a local R object. In addition, *Spark* provides through the **spark_apply** function a way to execute any function of R on the data in *Spark*, although due to the lower performance it will always be preferable to exploit the possibilities that *Spark* offers. Another important aspect is that *Spark* is on the rise in the use of data science [34] thus it's an important technology to know [35].

Going on with *H2O*, this machine learning library offers great versatility to perform all kinds of studies. In addition, thanks to the *Sparkling Water* project, it perfectly complements *Spark*, boosting it in the modeling phase. Although with *Spark MLib* we have some sort of algorithms, with *H2O* we have a greater variety and a greater capacity of control over these. Moreover, the performance provided by *H2O* algorithms is much higher than the provided by *Spark MLib*, being faster and having a lower memory footprint [36].

Finishing with this selection, we have *XGBoost*, a series of algorithms that often provide the best results when analyzing structured data like the source of this study. In addition, these are fully integrated in both *Spark MLib* and *H2O*.

About the discarded solutions, first of all, I have to say that the Microsoft solution, *RevoScaleR*, has been one of the best candidates to use in the study since it can also be used on *Spark*. Microsoft is strongly committed to facilitate the data science in its *Microsoft SQL Server* product [37] and is providing a series of very interesting tools with a very noticeable performance and facilities when implementing an algorithm in its own database. If we were talking about a final product and not a study, this would be one of the preferred solutions, taking into account that the data source is a *SQL Server* database. However, it's important to take into account that these products requires a *SQL Server* license, so the scope of its use is restricted.

Vowpal Wabbit seems promising to some kinds of problems, as real time sentiment analysis, but it not seems appropiate for this study.

DeltaRho seems a good scalable solution if you don't need to do complex or iterative things. This is not that case.

Bigmemory, *ff*, *parallel* and *doParallel* are well known R packages that works well if your data fits in them and your analysis doesn't go beyond their scope. Also, they don't scale horizontaly.

MonetDB is an easy out-of-core solution, but as the previous solutions it doesn't scale horizontaly.

2.5 Infrastructure implementation

Taking into account the chosen solutions and knowing that *Apache Spark* can support both *H2O* and *XGBoost* over it, for the implementation of the infrastructure on which the analysis will be carried out, it will be necessary to consider *Spark* first.

For the execution of an application in *Spark* it is necessary to define a context, **SparkContext**, in the application's code. The context indicates on which *Spark* cluster the code will be executed, either a local machine or a remote machine. In each cluster there is a manager which manages and allocates the resources to the applications. These resources are used by the executors, which are *Spark* processes that ultimately execute the code on the cluster nodes, interacting them with local data.

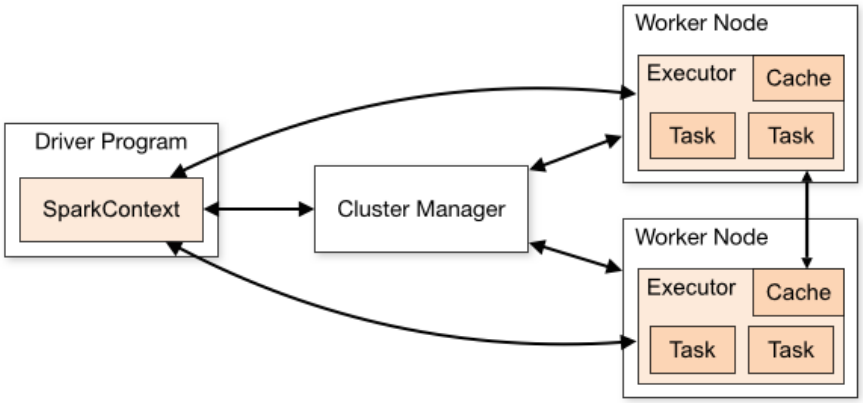


Figure 2.4: SparkContext overview

Source: https://spark.apache.org

As mentioned, a *Spark* cluster must have a resource manager. The possible managers to use are:

- *Standalone*: it is the simplest option, included with *Spark*, which allows managing the resources of the cluster.

- *Apache Mesos*: a general character manager, which can manage both *Spark* and *Hadoop MapReduce* applications.
- *Hadoop YARN*: is the default manager in *Hadoop* 2.

Another necessary aspect for any *Spark* cluster is the fact that it must run on distributed storage. For this there are numerous supported options such as *Alluxio*, *Hadoop Distributed File System (HDFS)*, *MapR File System (MapR-FS)*, *Cassandra*, *OpenStack Swift*, *Amazon S3*, *Kudu*, *Luster file system*, etc. In addition, as with the resource manager, there is the *Standalone* option, where distributed storage is not used, but rather local storage.

There are payment services that provide the necessary resources for the deployment of a *Spark* cluster in the cloud, such as *Amazon Web Services*, *Google Cloud Platform* or *Microsoft Azure*. This option is preferable than a local cluster since horizontal scaling is easy and immediate whenever necessary.

In the case of this study, the database provided has a size of about 20 GB, which does not require the installation or use of a cluster with large resources. Taking into account that the analysis does not change depending on the cluster on which the code runs, *Standalone* mode will be used and it will run on a local computer.

For the use of *Spark* in R it will be necessary to install the *sparklyr* package, which will allow to establish the context, interact with *Spark*, etc:

```
install.packages ("sparklyr")
```

Since we will use the *Standalone* option, the *Apache Spark* download and installation must be performed. This can be done simply with the *sparklyr* function **spark_install**, which allows to set the *Spark* version to be used, although the *sparklyr* supported versions must be taken into account:

```
library (sparklyr)
spark_install (version = "2.1.0")
```

Now, to initiate any connection with the *Standalone* context it will be necessary to previously execute the following command:

```
sc <- spark_connect(master = "local")
```

Where **sc** will contain the established connection with the local cluster.

Chapter 3

Data Science study

3.1 Methodology selection

To carry out a data science study, it is advisable to follow a methodology in order to have a reliable guide that allows to accomplish the objectives with all the guarantees. Today, the most used are *KDD*, *SEMMA* and *CRISP-DM* [38].

Starting with *KDD (Knowledge Discovery in Database)* [39], the oldest, its defined as a general process of discovery of knowledge from the data, or the extraction of patterns and information from large datasets using machine learning, statistics and database systems. This methodology is mainly focused on the processes related to data mining instead of having a global approach with a project. Its phases are **Selection**, **Preprocessing**, **Transformation**, **Data Mining** and **Evaluation and Interpretation**:

1. **Selection**: Find an objective and identify the data that has to be extracted, looking for the appropriate input attributes and the output information to represent the task.
2. **Preprocessing**: Incomplete data cleaning, noise and lack of data.
3. **Transformation**: Syntactic modifications on the data for the best understanding of the discovered rules when transforming the data from low level to high level, also reducing the execution time of the algorithm.
4. **Data mining**: Exploration and analysis, by automatic or semiautomatic means, of the data obtained in the previous phase in order to discover significant patterns/models and rules. The result of the phase are the patterns/models of that mining.

5. **Evaluation and Interpretation**: evaluation of the patterns and models obtained and the subsequent interpretation to generate the final knowledge.

Sometimes it is usually added an initial **recollection** process, where different data sources are integrated into the same space, a **data warehouse**.

SEMMA (Sample, Explore, Modify, Model, and Assess) [40], is a list of sequential steps developed by **SAS** and defined as a logical organization of the set of functional tools of one of its products, *SAS Enterprise Miner*, to carry out the central tasks of data mining. As with *KDD*, the focus is on the stages of data mining.

CRISP-DM (Cross Industry Standard Process for Data Mining) [41], provides a standardized description of the life cycle of a standard data analysis project, analogous to how it is done in software engineering with development life cycle models of software. The *CRISP-DM* model covers the phases of a project, their respective tasks, and the relationships between these tasks. At this level of description it is not possible to identify all the relationships. Relationships could exist between any task according to the objectives, the context, and the user's interest in the data.

The *CRISP-DM* methodology contemplates the data analysis process as a professional project, thus establishing a much richer context that influences the elaboration of the models. This context takes into account the existence of a client that is not part of the development team, as well as the fact that the project not only does not finish once the ideal model is found (since later a deployment and maintenance is required), but it is related to other projects, and it is necessary to document it exhaustively so that other development teams use the knowledge acquired and work from it.

The life cycle of the data mining project consists of six phases (see figure 3.1): **Business Understanding**, **Data Understanding**, **Data Preparation**, **Modeling**, **Evaluation** and **Deployment**. The sequence of the phases is not rigid: forward and backward movement between different phases is allowed. The result of each phase determines which phase, or which particular task of a phase, should be done later. At figure, the arrows indicate the most important and frequent dependencies. The outer circle in the figure symbolizes the cyclical nature of the data analysis projects. The project does not end once the solution is deployed. The information discovered during the process and the solution deployed can produce new iterations of the model. Subsequent analysis processes will benefit from previous experiences.

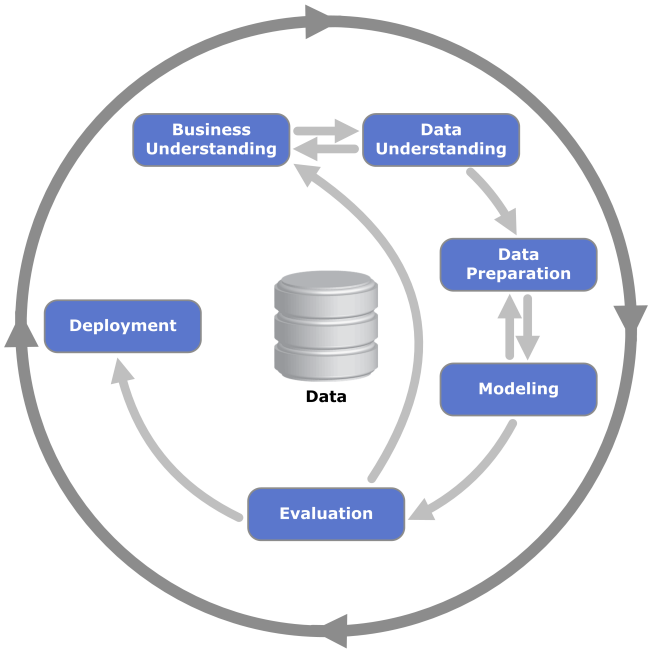


Figure 3.1: Process diagram showing the relationship between the different phases of *CRISP-DM*

Next, we will briefly describe each of the phases:

1. **Business Understanding**: this initial phase focuses on the understanding of project objectives. This knowledge of the data is then converted into the definition of a data mining problem and a preliminary plan designed to achieve the objectives.

2. **Data Understanding**: the data understanding phase begins with the initial data collection and continues with activities that allow you to become familiar with the data, identify quality problems, discover preliminary knowledge about the data, and/or discover interesting subsets to form hypotheses as for the hidden information.

3. **Data Preparation**: the data preparation phase covers all the activities necessary to construct the final data set (the data that will be used in the modeling tools) from the initial raw data. The tasks include the selection of tables, registers and attributes, as well as the transformation and cleaning of data for the tools that they model.

4. **Modeling**: in this phase, the modeling techniques that are relevant to the problem are selected and applied (the more the better), and their parameters are calibrated to optimal values. Typically there are several techniques for the same type of data mining problem. Some techniques have specific requirements on the form of the data. Therefore, almost always in any project it ends up going back to the data preparation phase.

5. **Evaluation**: at this stage in the project, one or several models that seem to reach sufficient quality from the perspective of data analysis have been built. Before proceeding to the final deployment of the model, it is important to evaluate it thoroughly, review the steps executed to create it and compare the model obtained with the business objectives. A key objective is to determine if there is an important business issue that has not been sufficiently considered. At the end of this phase, a decision on the application of the results of the data analysis process should be obtained.

6. **Deployment**: generally, the creation of the model is not the end of the project. Even if the objective of the model is to increase the knowledge of the data, the knowledge obtained will have to be organized and presented so that the client can use it. Depending on the requirements, the development phase can be as simple as the generation of a report or as complex as the periodic and perhaps automated realization of a process of data analysis in the organization.

In 2015, **IBM Corporation** proposed a new methodology methodology called *Analytics Solutions Unified Method for Data Mining/Predictive Analytics (ASUM-DM)* [42] that extends *CRISP-DM*, and is part of the general methodology *ASUM (Analytics Solutions Unified Method)* incorporated in the products and *IBM analytical solutions*.

In addition to the methodologies presented, new specific methodologies for data science projects are emerging recently. These are often a hybrid between data mining and software development methodologies. As examples we have *Microsoft TDSP (Team Data Science Process)* [43], inspired by *CRISP-DM* and *Scrum*, where roles, processes and templates are defined, or *Domino Data Science Lifecycle* [44], which combines *CRISP-DM* and *Agile*.

Being *CRISP-DM* the most used methodology in recent years and covering this the stages of a data science project more generally, this methodology will be used as a guide during the development of the study.

3.2 Data source and formulated questions

The data on which this study is based consists of a database of *Microsoft SQL Server*, specifically I have been provided with a copy of December 2018 whose content has been previously anonymized. This backup has a size of approximately **20 GB**, which is far from being considered as **Big Data**. Anyway, it will be treated as such since it is the perspective that I want the study to have.

The database, whose name is **SABENTISproCOLMENA**, has **589 tables**, each with different attributes. It also has a series of database **views** from which it is possible to obtain certain information regarding the attributes. In this case, it is not necessary to work with all tables since the study scope will be limited to the questions formulated and negotiated with the experts in the matter. These are the following:

1. **What is the relationship between the risk levels assessed and the accident rate of a company?**
2. **How many accidents will a company have in a year depending on its size, the location of its centers and its activity?**
3. **Which workers are more likely to be injured next year (by gender, age range, activity, etc.)**

Due to the considerable number of tables, also the attributes, from which the necessary data must be obtained to answer these questions, a meeting has been held with the relevant members of **Easy tech global** in order to obtain the necessary information of the database structure and their tables relations. From this meeting, a relational diagram of main tables necessary to answer the questions has been extracted:

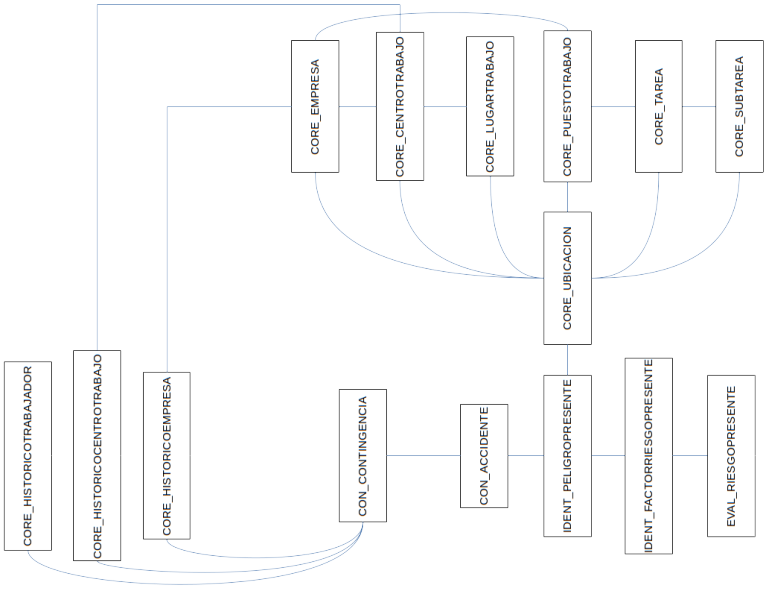


Figure 3.2: Database tables relationship

First, **accidents** are recorded with certain attributes in the table **CON_ACCIDENTE**. A type of accident can be work or common accident, so you have to perform the filtering of the first type. There is a higher level in **CON_CONTINGENCIA** table, where different types of **contingencies**, such as administrative, illness or maternity, are grouped together with accidents. This table is related to the tables that support the different contingencies and with the tables **CORE_HISTORICOEMPRESA**, **CORE_HISTORICOCENTROTRABAJO** and **CORE_HISTORICOTRABAJADOR**, which, for legal reasons, record respectively the **historical** information of the **company**, the **workplace** and the **worker** at the time of the contingency.

In this database, the **locations** are stored as a general type in **CORE_UBICACION** table, and in their specific type in each of the corresponding tables. These types are often interrelated as can be seen in the diagram, where they are also shown from higher to lower specificity in terms of location, going from **CORE_EMPRESA**, a company location, to **CORE_SUBTAREA**, a subtask location. This way, all types of workers are accommodated.

Labor risks assessments are stored mainly in three tables. The first, related to location table, is **IDENT_PELIGROPRESENTE**, which records the **identified dangers**. A danger is a source, situation or act with potential to cause harm in terms of human harm or health deterioration, or a combination of both. These can be a sting, a noise, a lighting, a temperature, a smoke, a posture, electric, etc. On another level, and related to the identified dangers, we have the **risk factors** in table **IDENT_FACTORRIESGOPRESENTE**, which are a combination of the probability of a dangerous event or exposure occurring and the severity of the damage or deterioration of health that the event or exposure may cause. These can be for example intermittent, high voltage, prolonged, overtime, etc. Finally, when an **evaluation** is made, it can be one of three types **INSHT**, **FINE** or **GTC45**. Each of these types is exclusive, so only the values of one type per evaluation will appear. These values are found in the table **EVAL_RIESGOPRESENTE**, which is related to the previous one.

Also, there are some global facts that are important to take into account:

- In some tables, there is a field, **ELIMINADO**, that implies a **logical deletion** from the database. In other words, if an user deletes an observation, it won't be deleted definitely from the database. This way an unwanted deletion can be recovered quickly. Thus, for the analysis we should exclude those observations with a true value at this field.
- All **ID** fields have the **creation timestamp** implicitly at their value. For example, an observation whose ID has a value 20161105172011-3958486-0001-T will have been created at date and time 05/11/2016 17:20:11.

3.3 ETL process

The stage **ETL (Extract, Transform and Load)** [45], is the one that entails the necessary processes for the movement of the data from an origin to the destiny where it will be used, carrying out the pertinent transformations. In this case, we start with the data in the form of a backup file of a *Microsoft SQL Server 2016* database that we want to move to *Apache Spark*. From *Spark* we can obtain the data from different types of data sources [46] such as *CSV*, *JSON*, *Parquet* or a flat text file. As the backup file is an unsupported type, we have the option of first uploading the backup to a running *SQL Server* server and then, through a *JDBC* connection made from *Spark* [47] to the databse obtain the necessary data. A *JDBC (Java Database Connectivity)* [48] connection allows executing standard operations on databases from the *Java* programming language, independently of the operating system from which it is executed or of the database to which it is accessed. To translate these standard operations into a language understandable by *SQL Server*, it is necessary to use a specific *JDBC* driver. In this case, the *Microsoft JDBC Driver 7.2 for SQL Server* [49] driver will be used. Once downloaded and decompressed, it will be referenced from the *Spark* configuration at code before establish the connection with it:

```
config$`sparklyr.shell.driver-class-path` <-
  "sqljdbc_7.2/enu/mssql-jdbc-7.2.1.jre8.jar"
```

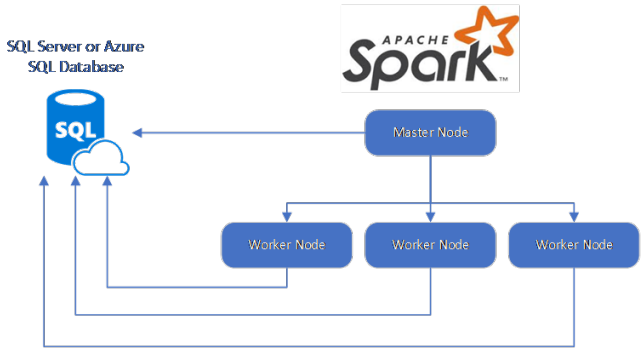


Figure 3.3: *Microsoft SQL Server* and *Spark* connector

Source: https://azure.microsoft.com

Now, once connected to *Spark*, we can read a table from a remote database and load it as *Spark DataFrame*:

```
table_tbl <- spark_read_jdbc(sc, "Spark_DataFrame_name", options = list(
  url = "jdbc:sqlserver://SQL_SERVER_address:1433",
  database = "database_name",
```

```
  user = "database_user_with_sufficient_privileges",
  password = "user_password",
  dbtable = "table_name"))
```

Also, we could use a query to filter the retrieved data. This query will be used as a subquery in the FROM clause.

Since the possible transformations will be made in the **Data Wrangling** stage, we will store the data permanently since it will be possible to read it from the *Spark* cluster in future stages. In order to do this, the *Parquet* [50] format has been chosen, a **columnar storage format** [51] available throughout the *Hadoop* ecosystem:

```
spark_write_parquet(table_tbl,
  path = "path_to_parquet_file")
```

The fact of choosing this format and not others, such as *CSV*, is due to the higher performance and lower use of resources. This is due to the column-oriented storage. Typically, row-oriented formats are more efficient for queries that either must access most of the columns, or only read a fraction of the rows. Column-oriented formats, on the other hand, are usually more efficient for queries that need to read most of the rows, but only have to access a fraction of the columns. Analytical queries typically fall in the latter category, while transactional queries are more often in the first category. Additionally, *CSV* is a text-based format, which can not be parsed as efficiently as a binary format. This makes *CSV* even slower. A typical column-oriented format on the other hand is not only binary, but also allows more efficient compression, which leads to smaller disk usage and faster access [52].

3.4 Iterative analysis

Next, the phases carried out during the development of the analysis will be detailed. Although these phases do not follow a strict sequential order, but are performed within an iterative process in which it is possible to jump from a more advanced phase to a previous one, they will be detailed in sequential order to facilitate their explanation.

The first three phases of the *CRISP-DM* methodology, **Business Understanding**, **Data Understanding** and **Data Preparation** have been carried out together, iterating whenever necessary, while the **modeling** and **evaluation** phases have been developed at the same time. From the latter it has occasionally turned to earlier phases.

3.4.1 Data understanding and data preparation

We will divide this part in the understanding and preparation of three topics: **accident's data**, **historical accident data** and the **identifications and evaluations of risks**. This division is made thinking of, by one hand the volume of data on each, and for another hand on the relationships between their related tables with each of these topics.

All code developed at this section can be found at appendix A.

3.4.1.1 Accident's data

Accident's data are important to answer all 3 questions. We mainly want to obtain when accidents occured as we will need to establish relationships between the number of accidents at a company in a year and some other fields to answer all questions. Also, some other attributes could be important for some questions. Registered accident's information is saved mainly in two tables, **CON_ACCIDENTE** and **CON_CONTINGENCIA**. First, we have to load them into *Spark* from the parquet files that we saved at the **ETL** process. Then, since there are several kinds of accident registered at **CON_ACCIDENTE** but we are interested only on those occurred at work we need to filter them. These are the ones with **FK_CORE_TIPOACCIDENTE** value equal to **20141007182138-3948986-0005-W**. Now, we can join both tables, **CON_ACCIDENTE** and **CON_CONTINGENCIA**, discarding the deleted observations by **ELIMINADO** field. If we check its dimensions, we can see that it has 323850 observations with 89 fields. We can also see a brief resume of its fields using **glimpse**, where it shows that there are missings and some mistypes (truncated output):

```
## Observations: ??
## Variables: 89
## Database: spark_connection
## $ ID                                <chr> "20150520164504-7724446-...
## $ FK_CORE_LUGARTRABAJO              <chr> NA, NA, NA, NA, NA, NA, ...
## $ FK_CORE_PUESTOTRABAJO             <chr> NA, NA, NA, NA, NA, NA, ...
## $ FK_CORE_TRABAJADOR                <chr> "20150520164501-2549301-...
## $ FK_IDENT_FACTORRIESGOPRESENTE     <chr> NA, NA, NA, NA, NA, NA, ...
## $ CARGO                             <chr> "ING DE VENTAS Y SERVICI...
## $ ANTIGUEDADPUESTOTRABAJO           <chr> NA, NA, NA, NA, NA, NA, ...
## $ FECHAINVESTIGACION                <chr> NA, NA, NA, NA, NA, NA, ...
## $ HORASTRABAJOPREVIAS               <chr> "073000", "074500", "040...
```

At this point we can start doing some explorations and required transformations.

***Important key**: For this analysis I used mainly **dplyr** functions as them are more familiar to me than **Spark's Scala API sdf functions**. The code remains clear and the performance is not affected. Thanks to **sparklyr** package, a grand part of dplyr functions are translated to Spark, but some of them are not present. This is the case of **gather** function. In order to overpass this obstacle a function that gather's all passed attributes is defined:*

```
# This function emulates dplyr's gather, converting all
# columns of a tibble to key-value pairs
sdf_gather_all <- function(tbl) {
  gather_cols <- colnames(tbl)
  lapply(gather_cols, function(col_nm){
    tbl %>%
    select(c(col_nm)) %>%
    mutate(key = col_nm) %>%
    rename(value = col_nm)
  }) %>%
  sdf_bind_rows() %>%
  select(c('key', 'value'))
}
```

Logical fields

Starting with **logical** (or boolean) fields, let's see their distribution:

Figure 3.4: Logical attributes distribution

As can be seen at figure 3.4, there are some variables where either all or almost all values are missing. Also, there are some of them that have unique values. These variables must be dropped.

Numerical fields

Now, for **numerical** values, their distribution is:

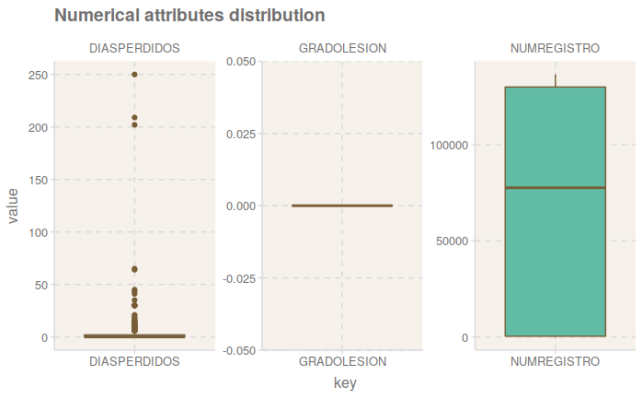


Figure 3.5: Numerical attributes distribution

And some descriptive statistics:

summary	DIASPERDIDOS	GRADOLESION	NUMREGISTRO
count	744	744	323850
mean	2.87	0.0	68586.09
stddev	15.12	0.0	55933.45
min	0	0	1
max	250	0	136590

Table 3.1: Numerical attributes statistics

This reveals that **DIASPERDIDOS** and **GRADOLESION** variables are almost empty. Thus, they must be deleted.

Character fields

For **character** fields, all empty values (**""**) have to be assinged to **NA** in order to count as missings. Afterwards, in order to discard useless variables, we can take the decision of delete those having an amount of missing values over **60%**. This is a simple solution that can work

most of time but it depends of the data. A field can have almost all values missed since the condition to exist is rare but these little cases could be relevant. For a first iteration over data it can be considered a good approximation

Now, if we examine the data we can see that the number of fields has been reduced to 42. Also, most of they are of character type. This is the group where there are mistype fields, thus we have to start to transform them.

***Important key**: Working with dates is not as easy as using R functions like provided by **lubridate** package. Anyway it's preferable to not use **spark_apply** since its performance is worse.*

Character to numerical

Some character fields are in reality numerical. Let's start with **HORASTRABAJOPREVIAS**, a field that should contain the number of hours that a worker has completed of its workday at the moment of the accident. This field is in format HHMMSS, thus we have to get every part separately and apply a conversion to hours:

Listing 3.1: R example

```
accidente <- accidente %>%
  mutate(HORASTRABAJOPREVIAS =
    regexp_replace(HORASTRABAJOPREVIAS, ':', '')) %>%
  mutate(HORASTRABAJOPREVIAS =
    as.numeric(regexp_extract(HORASTRABAJOPREVIAS,
    '([0-9]{2})([0-9]{2})([0-9]{2})', 1)) +
    as.numeric(regexp_extract(HORASTRABAJOPREVIAS,
    '([0-9]{2})([0-9]{2})([0-9]{2})', 2)) / 60 +
    as.numeric(regexp_extract(HORASTRABAJOPREVIAS,
    '([0-9]{2})([0-9]{2})([0-9]{2})', 3)) / 3600)
```

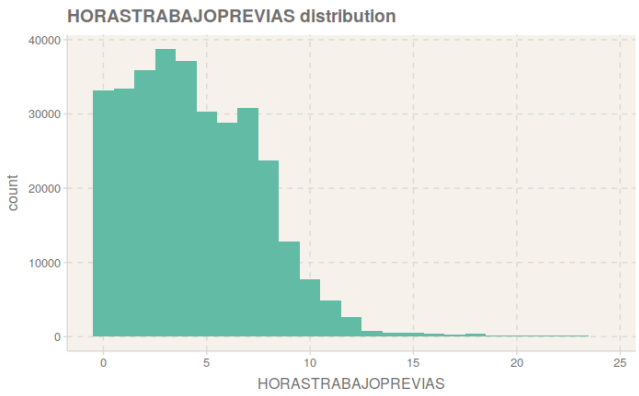


Figure 3.6: HORASTRABAJOPREVIAS distribution

As we can see at figure 3.6, value **0.0** (000000 previously) is used for missing values. Checking the field **DESCRIPCION**, where the accident is detailed, it shows that definitely this value is related to missings. Thus, we must transform all 0.0 values to NA (see figure 3.7):

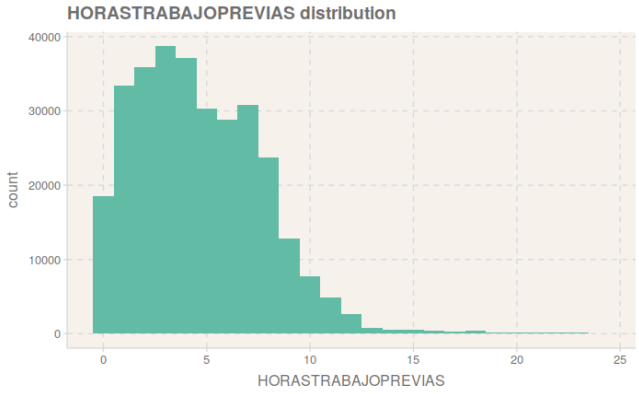


Figure 3.7: HORASTRABAJOPREVIAS distribution (without missings)

Going on with field **SALARIO**, that should contain the salary of the accidented worker, we have the following distribution and statistics once it is converted to numerical type:

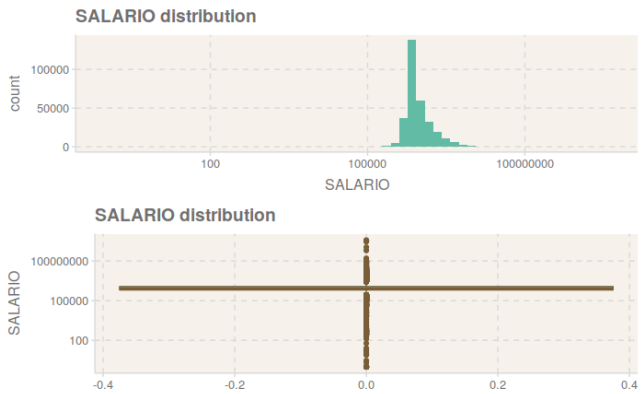


Figure 3.8: SALARIO distribution

summary	SALARIO
count	323703
mean	1189439.69
stddev	1.33E7
min	-1
max	3.82E9

Table 3.2: SALARIO statistics

This shows clearly some outliers at both extremes. Taking into account the minimum salary at Colombia (from 461k at 2008 to 781k at 2018) [53], if we fit the data between 200k, considering also part-time jobs, and 12M, in order to take an upper limit, and assign the rest of values to NA, we have the following distribution:

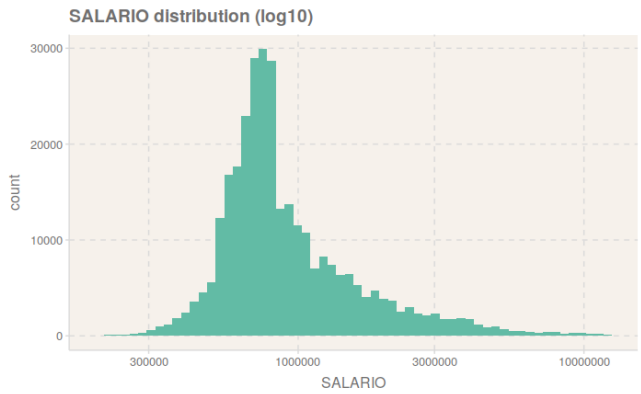


Figure 3.9: SALARIO distribution (without outliers)

SALARIO has been converted in figure 3.9 to base log10 in order to normalize its distribution. Now, its statistics are:

summary	SALARIO
count	312339
mean	1134982.28
stddev	1070067.10
min	200000
max	1.2E7

Table 3.3: SALARIO statistics (without outliers)

Which offers a more realistic salary distribution losing a small part of observations.

Character to date

Other variables are dates in reality. Starting with **FECHACREACION**, we expect to see the creation date of an accident's observation (see figure 3.10):

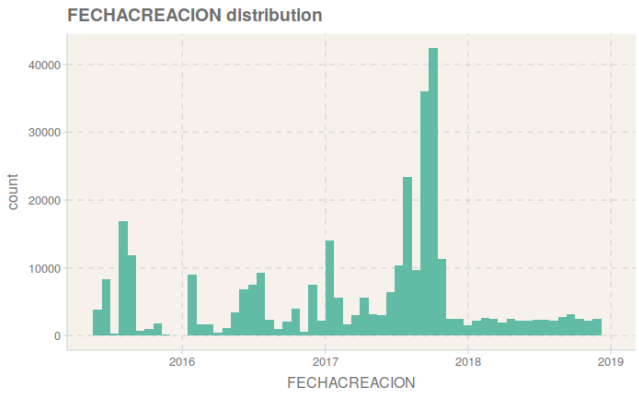


Figure 3.10: FECHACREACION distribution

Nevertheless, this not seems to be a good reference about the real date when accident's occurred since the creations are too much concentrated in certain dates, indicating a possible massive traspassing of information. One expects to have a near uniform distribution or maybe a growing trend due to a grow in number of insured workers. If we check the density of insertions by day we can affirm this hypothesis:

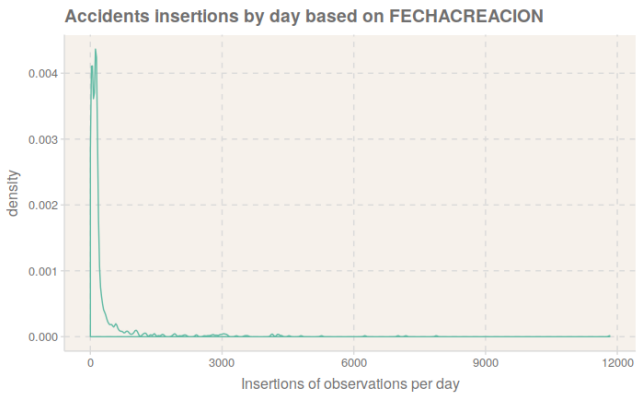


Figure 3.11: Insertions by day based on FECHACREACION

We delete this field and its sister fields **FECHAMODIFICACION** and **FECHA_CREACION**. On another hand, fields **FECHA** and **HORA** from **CON_CONTINGENCIA** should have pressumably the date and hour when the accident happened. If we add and convert them to a new datetime field called **MOMENTOACCIDENTE** we have (see figure 3.12):

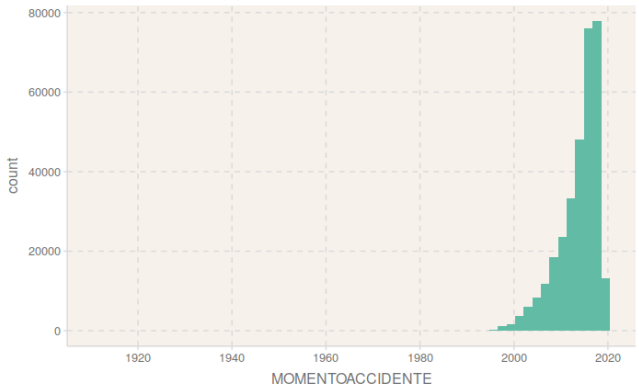


Figure 3.12: MOMENTOACCIDENTE distribution

As we can see, the new field seems a good reference for accident's datetime, but it has some outliers. We can clean them if we see when accidents started to be added:

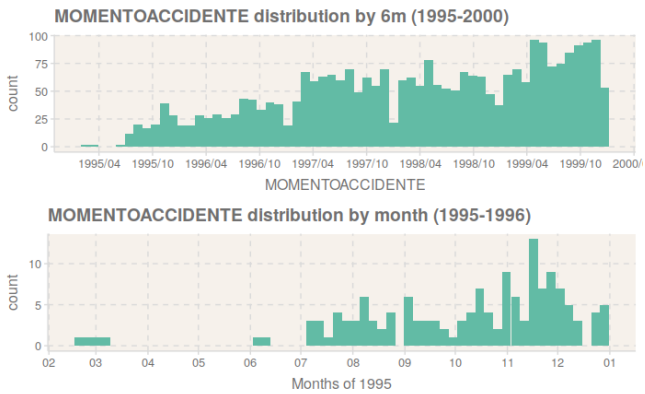


Figure 3.13: MOMENTOACCIDENTE distribution (zoom)

As we see at figure 3.13, it seems that this happened starting July 1995. If we assign previous values to NA:

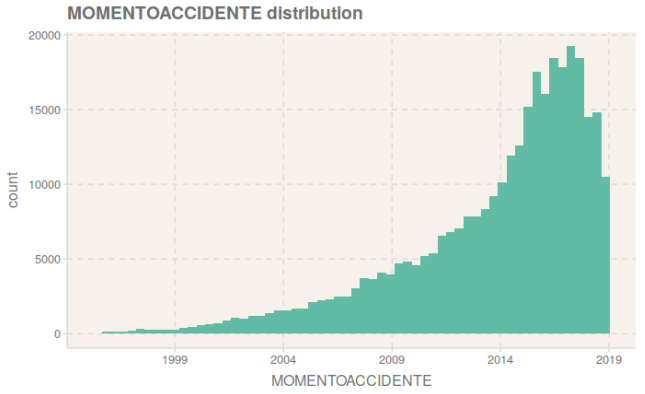


Figure 3.14: MOMENTOACCIDENTE distribution (without outliers)

With figure 3.14 we have a more expected distribution of the moment of accident. Also, if we check another time the density of registered accidents per day we will have something more realistic:

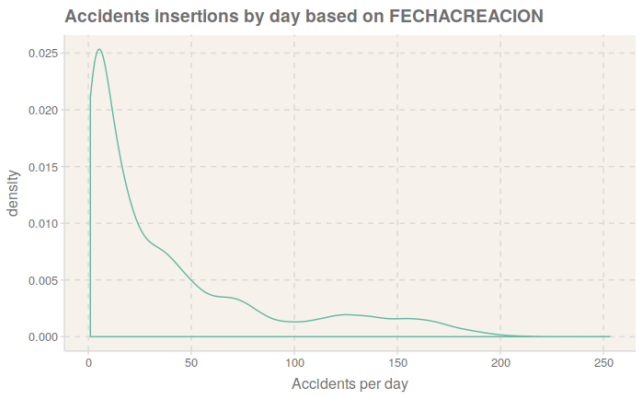


Figure 3.15: Insertions by day based on MOMENTOACCIDENTE

Now, for **FECHAINGRESOEMPRESA**, pressumibly the date when a worker started at a company, it seems that most of observations are erroneous, thus this variable will not be used:

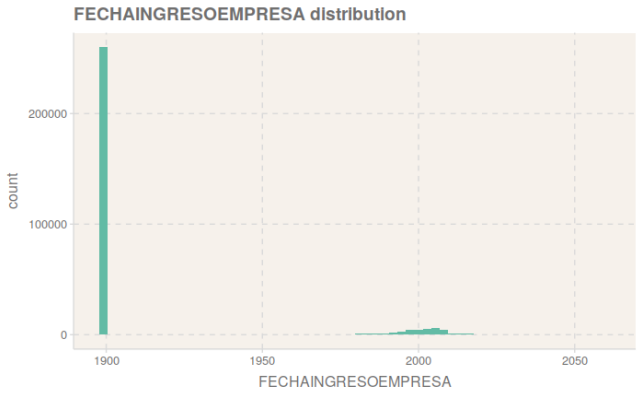


Figure 3.16: FECHAINGRESOEMPRESA distribution

***Important key**: In Spark we can not use categorical data types as factors in R. Character data needs to be treated before Spark's machine learning algorithms. In order to check their sanity, we can check their content.*

Categorical fields

In this case we have only one field, **CARGO**, that seems to be categorical as it should be the worker's position. Let's check how much levels would have at table 3.4:

key	unique
CARGO	31746

Table 3.4: CARGO number of unique values

This field is present in the historical data of an injured worker, therefore, we will use that variable and we delete this one since worker's position could change overtime.

3.4.1.2 Historical accident data

CORE_HISTORICOTRABAJADOR

Historical accident data is also important to answer all questions, specially last two, since we need to infer the number of accidents of a company or a worker in a year based on some fields that mostly we should find at historical accident tables. Historical accident data necessary for answer all questions is saved mainly in two tables, **CORE_HISTORICOTRABAJADOR** and **CORE_HISTORICOEMPRESA**. Their data needs to be wrangled in order to be joined afterwards with the join of **CON_CONTINGENCIA** and **CON_ACCIDENTE**.

Starting with **CORE_HISTORICO_TRABAJADOR**, the historical data of workers affected at an accident, first we have to load it in *Spark*. A first look of this table reveals that its dimensions are 1615792 observations and 50 attributes. Also a resume of its fields is (truncated output):

```
## Observations: ??
## Variables: 50
## Database: spark_connection
## $ ID                 <chr> "20150618231654-1430624-0137-W", "2015061823...
## $ NOMBRE             <chr> "776DE14C", "72457DE9", "11C69437", "D2FA7D2...
## $ APELLIDOS          <chr> "320497D5", "02501A6F", "F9E5B8AB", "35F8551...
## $ IDENTIFICACION     <chr> "7D010AFC", "DEED8B88", "B42A7C17", "6FDBD9B...
## $ DIRECCION          <chr> "691556FF", "F5A978E1", "7B73A4F6", "2EA7844...
## $ PROVINCIA          <chr> "Bogota D. C.", "Bogota D. C.", "Bogota D. C...
## $ POBLACION          <chr> "BOGOTA D.C.                                ", ...
## $ TELF1              <chr> "AF818C7A", "E0FADACC", "A07C1FC5", "E660B3E...
## $ FAX                <chr> "7E6C1C97", "D466C40B", "AE3A4DDF", "4247BDE...
## $ EMAIL1             <chr> "DACAB336", "9013DEFB", "AF64E3B7", "47B6512...
## $ EMAIL2             <chr> "BD981812", "15B08C57", "541C8F31", "E824FE5...
## $ FECHA_NACIMIENTO   <chr> "19870129", "19640320", "19710221", "1986042...
```

Also, we can check that all rows are duplicated. This is a fact that we can see in other tables of this database. We delete it inmediatly. Moreover, there are more than 800k observations but some of them have been deleted from database. This is known by the logical field **ELIMINADO** used before. Although these observations should be deleted, if we take it as deleted we will have an issue at joining time with accidents table since we will have only 799 observations:

```
historicotrabajador %>%
  filter(!ELIMINADO) %>%
  inner_join(accidente,
             by = c("ID" = "FK_CORE_HISTORICOTRABAJADOR")) %>%
  sdf_nrow()

historicotrabajador %>%
  inner_join(accidente,
             by = c("ID" = "FK_CORE_HISTORICOTRABAJADOR")) %>%
  sdf_nrow()

## [1] 799
## [1] 323850
```

The **ELIMINADO** field was designed to recover quickly an observation deleted by an user, thus this could be a mistake. In order to overpass this problem we won't take into account this field as observations are already filtered with **ELIMINADO** at **CON_CONTINGENCIA**. Next, we can use semijoin in order to select those observations that corresponds to non deleted accidents. A **SEMI JOIN** acts as a filter of those observations in a table that share an attribute value with an attribute of another table. We use it in order to select only those historical records of workers that exists in our cleaned accident's dataset.

Character fields

The attributes are majorly character fields but some of these are in reality categorical, numerical or dates, thus they will be transformed. First of all, as with accident's observations, empty character values (**""**) will be assigned as **NA**. Afterwards, following the same consideration as before, all fields with an amount of missing values over **60%** will be dropped.

Now, if we drop all attributes that have been anonymized and check the rest of character type we have the following fields:

```
## Observations: ??
## Variables: 16
## Database: spark_connection
## $ ID                  <chr> "20150623163112-8842233-0993-W", "2015062316...
## $ PROVINCIA           <chr> "Bogota D. C.", "Antioquia", "Bogota D. C.",...
## $ POBLACION           <chr> "BOGOTA D.C.                                ", ...
## $ FECHA_NACIMIENTO    <chr> "19531023", "19880307", "19800216", "1969120...
```

```
## $ CARGO                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ JORNADA              <chr> "Diurna", "Diurna", "Diurna", "Turnos", "Tur...
## $ EPS                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ ARP                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ AFP                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ FK_CORE_TRABAJADOR   <chr> "20150623163109-1855248-0227-W", "2015062316...
## $ TIPOIDENTIFICACION   <chr> "CEDULA DE CIUDADANIA", "CEDULA DE CIUDADANI...
## $ FECHAINGRESOEMPRESA <chr> "19531023", "19880307", "19800216", "1969120...
## $ RANGOANTIGUEDAD      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ RANGOEDAD            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ ESTADOTRABAJADOR     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ FECHARETIRADO        <chr> "20160609", NA, "20160502", "20160502", "201...
```

Character to categorical

Some variables seems to be categorical, let's check how much levels would have:

key	unique
ARP	1
ESTADOTRABAJADOR	2
RANGOANTIGUEDAD	4
RANGOEDAD	5
TIPOIDENTIFICACION	8
JORNADA	10
AFP	29
PROVINCIA	33
EPS	93
POBLACION	455
CARGO	31064

Table 3.5: Number of unique values at categorical fields

Checking table 3.5 we can confirm that all are categorical variables, not an arbitrary text value. If we check the distribution of them, starting with **RANGOEDAD**:

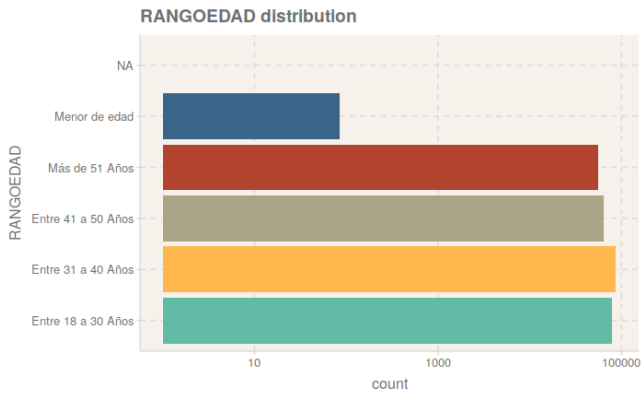


Figure 3.17: RANGOEDAD distribution (log10 scale)

We can see that, at **RANGOEDAD**, where worker's age is assigned to a range of years, there are some underage workers. This is not an error since at Colombia one can work having between 15 to 17 years old [54]. Moreover, this field can be useful since the birth date field, at **FECHA_NACIMIENTO** may not be well registered.

Following with **TIPOIDENTIFICACION**, we can see that most of values are of class **CEDULA DE CIUDADANIA**, an identification for national residents adults [55].

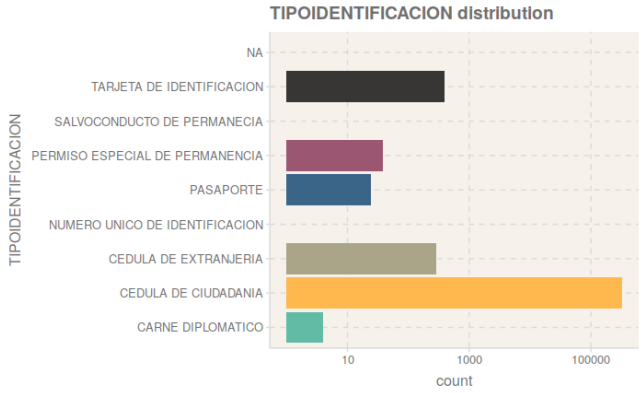


Figure 3.18: TIPOIDENTIFICACION distribution (log10 scale)

The same document, but for foreigners, is the **CEDULA DE EXTRANJERIA**. Also, there is the **TARJETA DE IDENTIFICACION**, an identification for underage national residents. There are other minor classes, as **NUMERO UNICO DE IDENTIFICACION** [56], but in resume, all could be classified between national or foreigners. This could be a new feature (see figure 3.19):

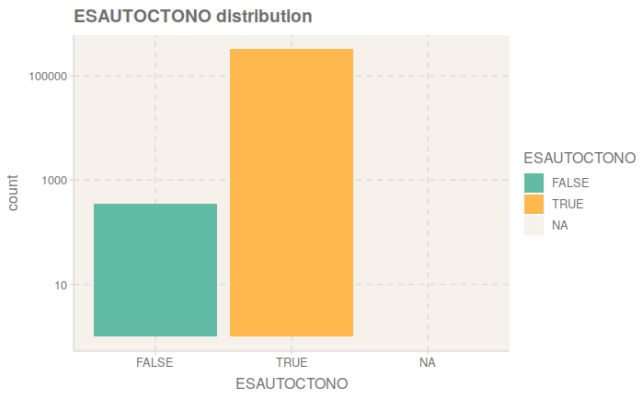


Figure 3.19: ESAUTOCTONO distribution

Now, with **JORNADA**, there are some classes with an explanatory name, like **Turnos**, **Nocturna**, **Mixto** and **Diurna**, kinds of workshifts. There is also a special case, **Sin definir**, that should be NA. Also, there are another five classes, **0-4**, whose representation is unknown but them seems to be the same classes as before:

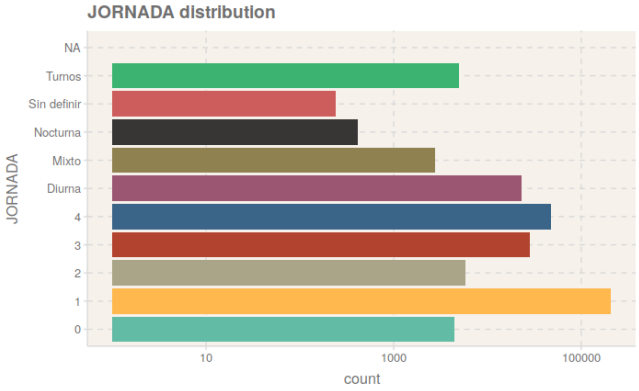


Figure 3.20: JORNADA distribution

At **PROVINCIA**, the province of worker is defined (see figure 3.21):

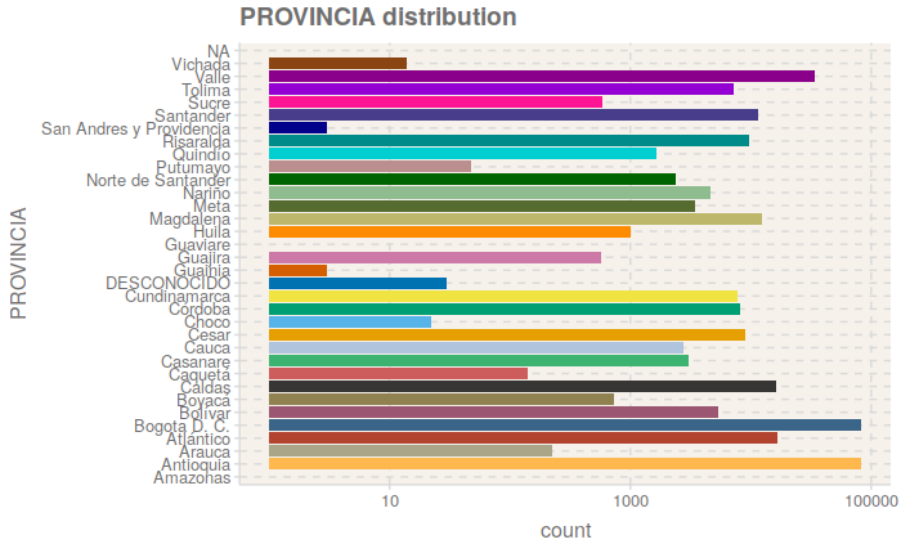


Figure 3.21: PROVINCIA distribution

There is an error with two classes; **VALLE** really exists as **Valle**, and **ANTIOQUIA** really exists as **Antioquia**. Also, there are two classes, **DESCONOCIDO** and **Por establecer**, that must be NA.

At **RANGOANTIGUEDAD**, it seems that almost all accidented workers have been for 5 or less years at their companies, the rest of ranges only sum approximately 30 accidents. Maybe this is due to a lack of data.

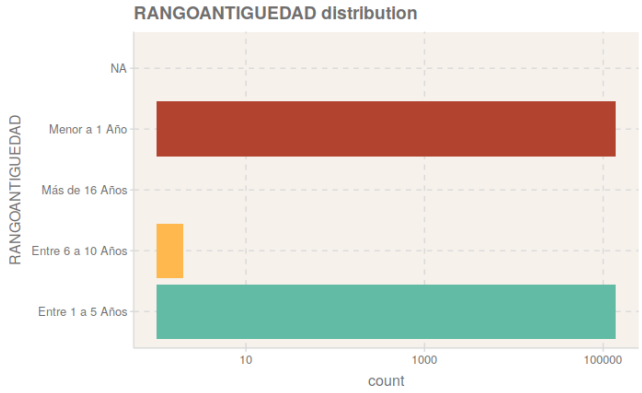


Figure 3.22: RANGOANTIGUEDAD distribution

ARP shows which **Administrator of Profesional Risks** did the registration of the accident, but there is only one, **COLMENA SEGUROS**. This field can be deleted. The same applies to **AFP (Administrator of Pension and Severance Funds)**.

With **ESTADOTRABAJADOR**, there are only few values with value **Retirado** (see figure 3.23):

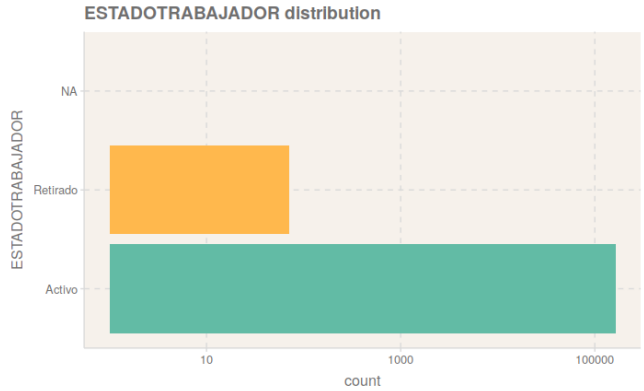


Figure 3.23: ESTADOTRABAJADOR distribution (log10 scale)

Since this variable provides poor content, it will be deleted. Exists a logical variable **RETIRADO** and a date variable **FECHARETIRADO** that seems to be related and could have better information:

If we check the rest of categorical fields, those that have a big amount of classes, starting with **POBLACION** we can check if there are errors with classes values. First, we select all distinct classes of **POBLACION**, collect them and create a new variable, **POBLACION_simplified**, this is the class value uppercased and without any symbol. We use collect here in order to use advanced string functions as **str_replace_all**. The collected amount of data is very small.

```
poblacion_and_poblacion_simple <- historicotrabajador %>%
  select(POBLACION) %>%
  distinct(POBLACION) %>%
  collect() %>%
  mutate(POBLACION_simplified =
    str_replace_all(iconv(toupper(POBLACION),
                    to="ASCII//TRANSLIT"),
                    "[^[:alnum:]]",
                    ""))

poblacion_and_poblacion_simple %>%
  group_by(POBLACION_simplified) %>%
  summarise(n = n()) %>%
```

```
  filter(n > 1) %>%
  pull(POBLACION_simplified)

# [1] "BOGOTADC"            "SANPEDRO"            "SANTAFEDEANTIOQUIA"
```

As we can see, there are some bad entered class names. To solve it, we have to copy the new R dataframe created with the class and its simplification to *Spark* and then left join it with **historicotrabajador** to finally change the class name with the simplification (see result at figure 3.24).

***Important key**: We cannot use join operations with different sources as R and Spark.*

```
paps_tbl <- sdf_copy_to(sc, poblacion_and_poblacion_simple)

historicotrabajador <- historicotrabajador %>%
  left_join(paps_tbl, by = "POBLACION") %>%
  mutate(POBLACION = POBLACION_simplified) %>%
  select(-POBLACION_simplified)
```

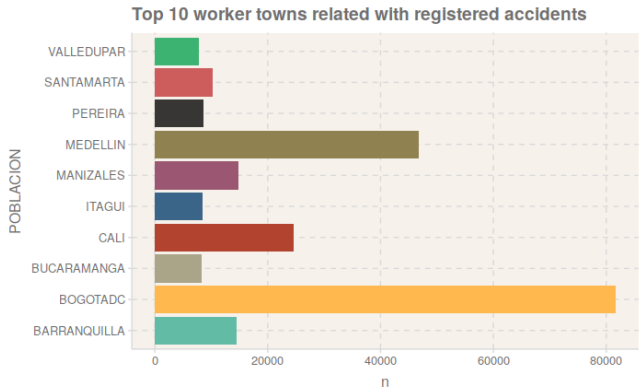


Figure 3.24: Top 10 worker's POBLACION distribution

Also, classes **CIUDADDESCONOCIDA**, **NOSUMINISTRADAA** and **PORESTABLECER** are alias of missing values.

Now, with **CARGO** we can follow the same procedure and see that there are a lot of errors. We can clean it a little bit using same procedure.

Finally, we have **EPS**, an identical case than **ARP** and **AFP**. Thus, we can delete it.

Character to date

There are other fields that are dates in reality, these are **FECHA_NACIMIENTO**, **FECHAINGRESOEMPRESA** and **FECHARETIRADO**. Starting with **FECHA_NACIMIENTO**, the day of birth of a worker, we can use it to impute all possible missing values in field **EDAD**, which contains the age of a worker and has 31323 observations without value. Let's see its relationship with **RANGOEDAD**:

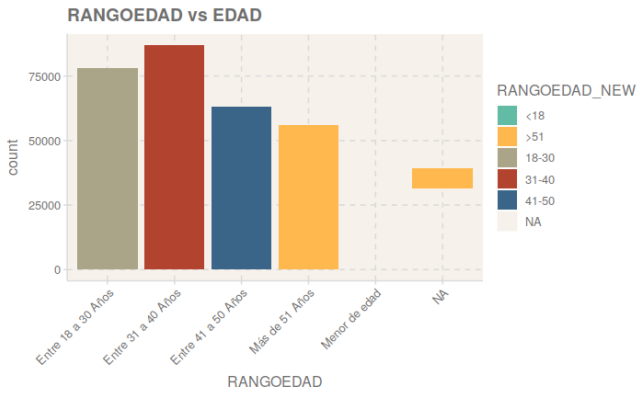


Figure 3.25: RANGOEDAD vs EDAD

As can be seen in figure 3.25, **EDAD** is comparable to **RANGOEDAD**. Now, applying transformations on **FECHA_NACIMIENTO**:

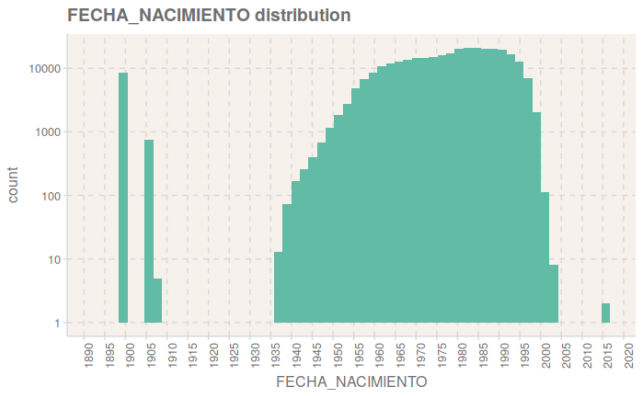


Figure 3.26: FECHA_NACIMIENTO distribution

It seems that there are some errors before 1935 and after 2005. If we remove them we will have a better distribution (see figure 3.27):

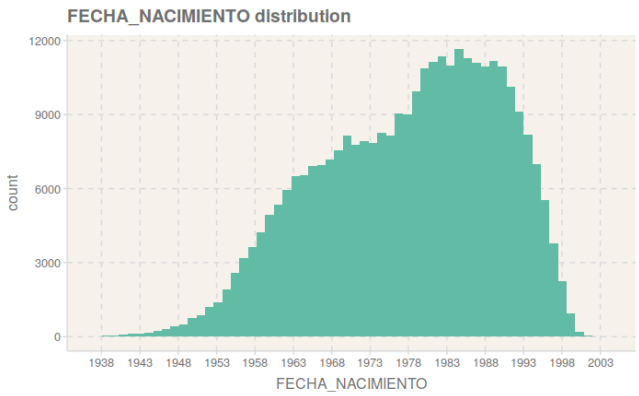


Figure 3.27: FECHA_NACIMIENTO distribution (without outliers)

Now, if we generate the age of workers at accident's time using the difference between the birth's and accident's year and compare it to the current age (see figure 3.28):

```
accidente_year <- accidente %>%
  select(FK_CORE_HISTORICOTRABAJADOR, MOMENTOACCIDENTE) %>%
  mutate(YEAR = year(MOMENTOACCIDENTE)) %>%
  select(-MOMENTOACCIDENTE)

historicotrabajador %>%
  left_join(accidente_year,
            by = c("ID" = "FK_CORE_HISTORICOTRABAJADOR")) %>%
  mutate(EDAD2 = YEAR - as.numeric(YEAR(FECHA_NACIMIENTO))) %>%
  mutate(DIFF_EDAD = abs(EDAD - EDAD2)) %>%
  group_by(DIFF_EDAD) %>%
  summarise(n = n()) %>%
  top_n(10) %>%
  ggplot(aes(x = DIFF_EDAD, y = n)) +
  geom_col() +
  ggtitle("EDAD at accident using FECHA_NACIMIENTO")
```

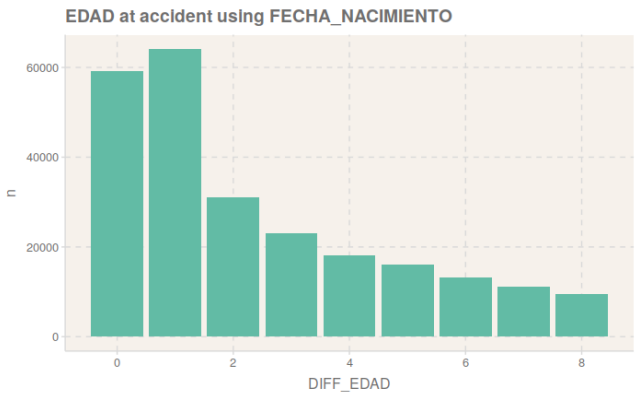


Figure 3.28: EDAD at accident using FECHA_NACIMIENTO

We can see that, although not perfect, it give us an opportunity to do an age imputation that minimizes the number of missings of **EDAD** to 1530. Also, if we compare it again with the current range of years we can see this reduction:

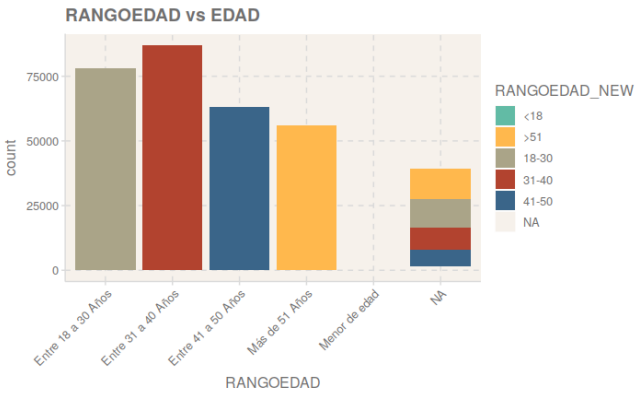


Figure 3.29: RANGOEDAD vs EDAD (after imputation)

Going on with **FECHAINGRESOEMPRESA**, the same field that was dropped at accidents table due to the amount of erroneous values, it has the following distribution (see figure 3.30):

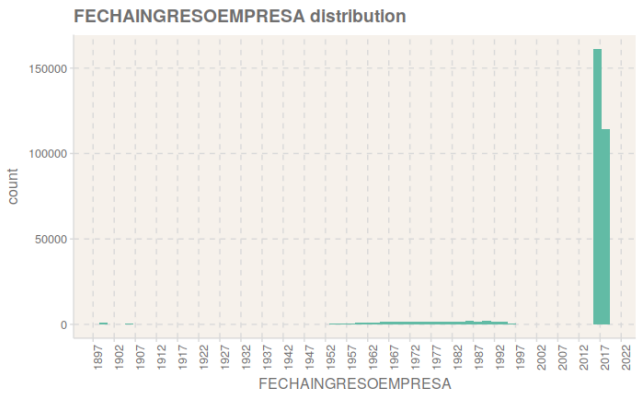


Figure 3.30: FECHAINGRESOEMPRESA distribution

We can see that same issue happens, thus we can't check the quality of values at **RANGOANTIGUEDAD**. This variable has to be deleted.

Logical fields

Let's to see their distribution.

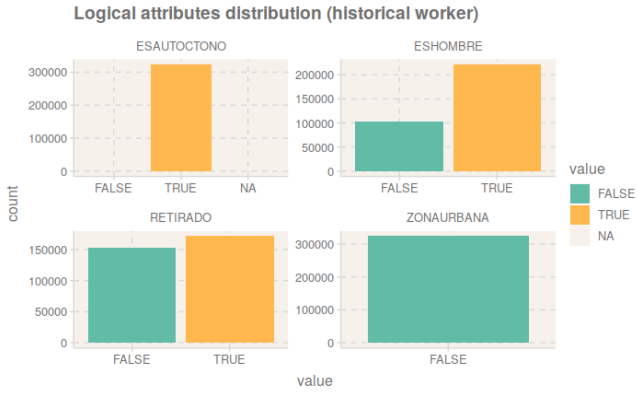


Figure 3.31: Logical attributes distribution

Where we can see the following:

- At **ESHOMBRE**, it seems that mens are accidented twice than womens. In order to affirm this, it should be studied in relation with the amount of men and women at the accident time.
- At **RETIRADO**, there are fairly the same amount of TRUE and FALSE. We should go deeper in this field and its relation with **ESTADOTRABAJADOR**.
- At **ZONAURBANA**, it seems that there is only one class, thus this field will be deleted.

Numerical fields

As numerical fields, we have **EDAD** and **ANTIGUEDAD**. Starting with **EDAD**, previously updated:

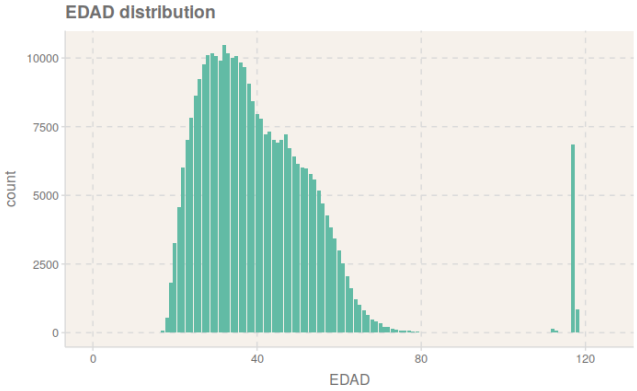


Figure 3.32: EDAD distribution

We can see some errors with accidented workers being more than 100 years and some being 0. This records will be assigned to NA. After cleaning them, going on with **ANTIGUEDAD** distribution:

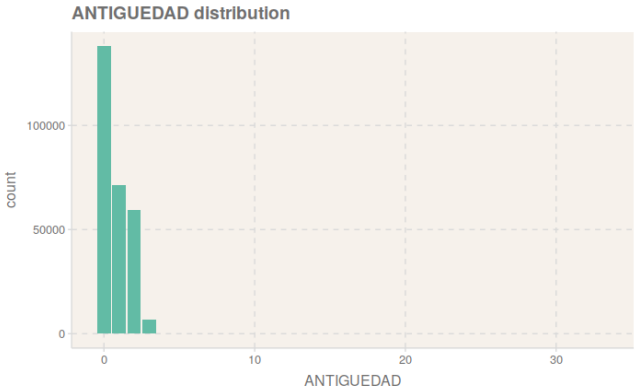


Figure 3.33: ANTIGUEDAD distribution

It seems that it can be comparable with **RANGOANTIGUEDAD** as we did before with **EDAD** and **RANGOEDAD** (see figure 3.34):

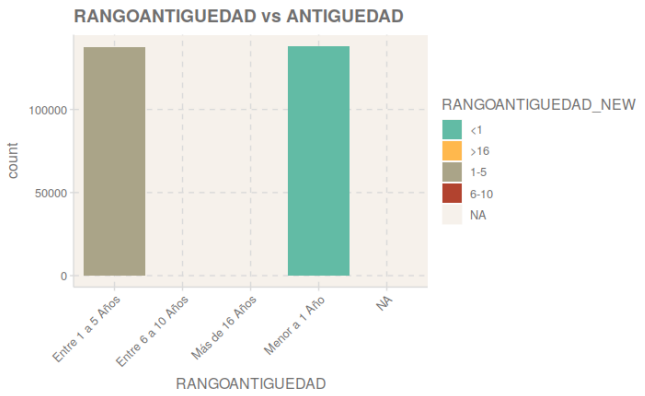


Figure 3.34: RANGOANTIGUEDAD vs ANTIGUEDAD distribution

We can see another time that almost all accidents happened to workers that had been working for less than 5 years at the company.

CORE_HISTORICOEMPRESA

After cleaning **CORE_HISTORICOTRABAJADOR**, we can go on with **CORE_HISTORICOEMPRESA** in order to check those attributes related to the company at the accident time. After loading it into *Spark* and remove all duplicated rows, we can see that there are 809179 observations with 18 attributes.

```
## Observations: ??
## Variables: 18
## Database: spark_connection
## $ ID                 <chr> "20150617231705-2589192-0788-W", "20150617231...
## $ NOMBRE             <chr> "8A9699E3", "6FC2783F", "A1FD1A9D", "218845FC...
## $ CIF                <chr> "CA84E2DA", "D7D9D5D1", "C23ED6D8", "B987CB28...
## $ DIRECCION          <chr> "7C56F09D", "1C9F5521", "DC1B8628", "F05996A5...
## $ CP                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ POBLACION          <chr> "SABANETA", "SABANETA", "SABANETA", "SABANETA...
## $ PROVINCIA          <chr> "Antioquia", "Antioquia", "Antioquia", "Antio...
## $ TELF1              <chr> "AD894BAF", "7807B1B2", "D3CF3A54", "9202F955...
## $ FAX                <chr> "79866916", "BD572E6B", "36158CC0", "01E8CEE0...
## $ EMAIL1             <chr> "16C0126C", "FD7D349A", "B551636B", "EA30C5FC...
## $ CNAE               <chr> "EMPRESAS DEDICADAS A LA FABRICACIÓN DE PRODU...
## $ MUTUA              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ TIPOVINCULACION    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ TIPOIDENTIFICACION <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
```

```
## $ ZONAURBANA          <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL...
## $ FK_CORE_EMPRESA     <chr> "20150617231638-5037045-0537-W", "20150617231...
## $ ELIMINADO           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL...
## $ FECHABORRADO        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
```

Regarding **ELIMINADO** field, we can take the same decision as with **CORE_HISTORICOTRABAJADOR**, thus we use a **SEMI JOIN** in order to filter those observations that are related with an accident in our cleaned accident dataset.

Character fields

Starting with the cleansing of fields, with character type we can do the same done at this study, this is transform all (””) values to **NA** and drop those fields whose amount of NA values are over **60%**. Also, we can delete all anonymized fields.

Character to categorical

Going on with the transformations, lets check how much levels would have those fields that seems to be nominal:

key	unique
PROVINCIA	27
POBLACION	290
CNAE	757

Table 3.6: Number of unique values at categorical fields

At **PROVINCIA**, the province of the company is defined. It seems that all values are correct:

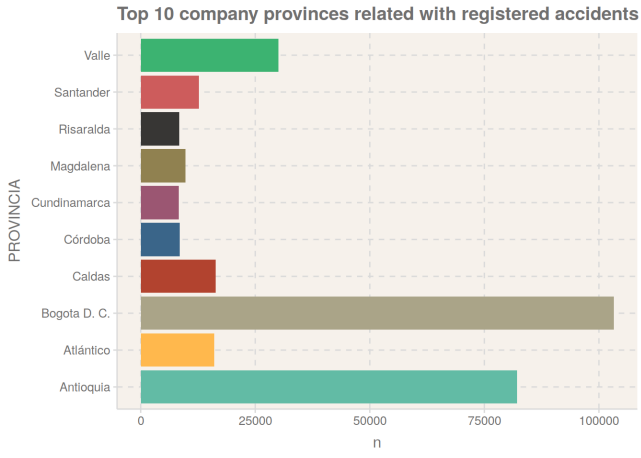


Figure 3.35: Top 10 companies PROVINCIA distribution

If we check the rest of categorical fields, those that have a big amount of classes, we can follow the same procedure used at **CORE_HISTORICOTRABAJADOR**. Starting with **POBLACION**, we can see that there are some bad entered classes names:

```
## [1] "BOGOTADC"
```

We can solve it and check, for simplicity, the 10 most relevant classes:

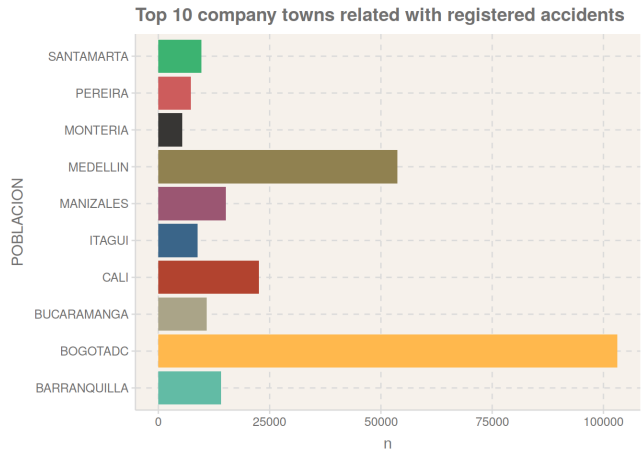


Figure 3.36: Top 10 companies POBLACION distribution

Once the field **POBLACION** is cleaned, we can continue with **CNAE**, which provides the company activity. If we check the amount of missings it seems that we have almost all records with a value, but some of them have errors, like finishing with a comma. In order to solve it, we have to collect the minimum data to R, transform it, upload then to *Spark* as a new tbl and join it with historicoempresa:

```
cnae <- historicoempresa %>%
  select(ID, CNAE) %>%
  collect()

cnae <- cnae %>%
  mutate(CNAE = str_replace(CNAE, ", *$", ""))

cnae_tbl <- sdf_copy_to(sc, cnae, overwrite = TRUE)

historicoempresa <- historicoempresa %>%
  select(-CNAE) %>%
  inner_join(cnae_tbl, by = "ID")
```

We could reduce the number of categories by grouping them into a more general categories.

Logical fields

There is only one logical attribute, **ZONAURBANA**:

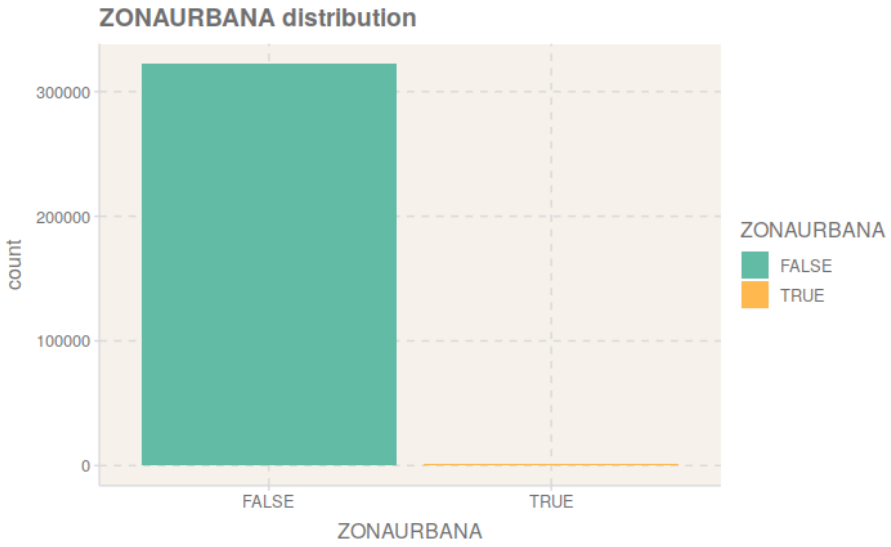


Figure 3.37: ZONAURBANA distribution

Where it seems that almost all values are FALSE. This doesn't make sense if **ZONAURBANA** refers to if the accident happened at an urban zone. By the moment, we don't remove this field, but it seems doesn't be a good predictor.

3.4.1.3 Identifications and evaluations of risks

All tasks done at this section are only useful to answer first question. The necessary tables from which we have to extract data value are **EVAL_RIESGOPRESENTE**, **IDENT_FACTORRIESGOPRESENTE** and **IDENT_PELIGROPRESENTE**. Starting with the **evaluations of risks**, these are registered at **EVAL_RIESGOPRESENTE**, thus we have to check its sanity. First, we need to load it into Spark. This table has all rows duplicated, thus we remove them. A first look of this table reveals that it has 1592644 observations and 41 attributes. Some of them are:

```
## Observations: ??
## Variables: 41
## Database: spark_connection
## $ ID                                    <chr> "20150223163610-683551...
```

```
## $ FK_IDENT_FACTORRIESGOPRESENTE          <chr> "20150223162716-435575...
## $ FK_CORE_RIESGO                         <chr> "1442489-2006020619223...
## $ ELIMINAR                               <lgl> FALSE, FALSE, FALSE, F...
## $ FK_EVAL_INSHT_PROBABILIDAD             <chr> NA, NA, NA, NA, NA, NA...
## $ FK_EVAL_INSHT_CONSECUENCIA             <chr> NA, NA, NA, NA, NA, NA...
## $ FK_EVAL_INSHT_NIVELRIESGO              <chr> NA, NA, NA, NA, NA, NA...
## $ FK_EVAL_FINE_CONSECUENCIA              <chr> NA, NA, NA, NA, NA, NA...
## $ FK_EVAL_FINE_EXPOSICION                <chr> NA, NA, NA, NA, NA, NA...
## $ FK_EVAL_FINE_PROBABILIDAD              <chr> NA, NA, NA, NA, NA, NA...
## $ FK_EVAL_FINE_NIVELRIESGO               <chr> NA, NA, NA, NA, NA, NA...
## $ ELIMINADO                              <lgl> FALSE, FALSE, FALSE, N...
```

Where, from knowledge acquired at a reunion with database creator, we know that **ESTAELIMINADO** reveals if the risk is currently solved, but we don't have the date when this ocurred. In addition, we have the date of the observation creation thanks to the ID field format. Field **FECHA_CREACION** should contain this information but it has too missing values, thus we mutate it with the date extracted from the ID:

```
evalriesgopresente <- evalriesgopresente %>%
  mutate(FECHA_CREACION = from_utc_timestamp(timestamp(unix_timestamp(
    regexp_extract(ID, '([0-9]{8})', 1), 'yyyyMMdd')), 'UTC'))
```

There are 3 types of risk measure methodologies: **GTC45**, **INSHT** and **FINE**. They are excluent, thus only one of them is used to eval a risk. The measured level of every methodology is an ordinal value inside a range of values that goes from less to more urgency. The levels depends of the methodology used and can be found at tables **EVAL_*METHODOLOGY*_NIVELRIESGO**. On **EVAL_RIESGOPRESENTE**, field **FK_EVAL_*METHODOLODY*_NIVELRIESGO** points to the level ID that corresponds to the measure. Now, we can load these tables where levels of measured risks are registered and check their contents:

EVAL_INSHT_NIVELRIESGO

```
## Observations: ??
## Variables: 6
## Database: spark_connection
## $ ID                <chr> "1", "2", "3", "4", "5"
## $ CODIGO            <chr> "TR", "TO", "MO", "IM", "IN"
```

```
## $ ALIAS              <chr> "TRIVIAL", "TOLERABLE", "MODERADO", "IMPORTAN...
## $ ELIMINADO          <lgl> FALSE, FALSE, FALSE, FALSE, FALSE
## $ FECHABORRADO       <chr> NA, NA, NA, NA, NA
## $ DIAS_PLANIFICACION <int> 365, 365, 90, 30, 30
```

EVAL_FINE_NIVELRIESGO

```
## Observations: ??
## Variables: 8
## Database: spark_connection
## $ ID                 <chr> "1", "2", "3"
## $ CODIGO             <chr> "URG", "PRIO", "PROG"
## $ ALIAS              <chr> "URGENTE", "PRIORITARIO", "PROGRAMABLE"
## $ PUNTUACION_MINIMA  <dbl> 200, 90, 0
## $ PUNTUACION_MAXIMA  <dbl> 10000, 200, 90
## $ ELIMINADO          <lgl> FALSE, FALSE, FALSE
## $ FECHABORRADO       <chr> NA, NA, NA
## $ DIAS_PLANIFICACION <int> 30, 90, 365
```

EVAL_GTC45_NIVELRIESGO

```
## Observations: ??
## Variables: 9
## Database: spark_connection
## $ ID                 <chr> "01", "02", "03", "04"
## $ CODIGO             <chr> "I", "II", "III", "IV"
## $ ALIAS              <chr> "No Aceptable", "No Aceptable o Aceptable con...
## $ PUNTUACIONMAXIMA   <int> 4000, 500, 120, 20
## $ PUNTUACIONMINIMA   <int> 600, 150, 40, 20
## $ ELIMINADO          <lgl> FALSE, FALSE, FALSE, FALSE
## $ FECHABORRADO       <chr> NA, NA, NA, NA
## $ DIAS_PLANIFICACION <int> 0, 365, NA, NA
## $ NOMBRE_PYME        <chr> "No Aceptable", "Aceptable con control especí...
```

As we can see at **ALIAS** field, the range of levels are different for each methodology, so we can not compare them. In addition, it seems that some levels have a number of days among which it is acceptable to solve the risk. This can be seen at **DIAS_PLANIFICACION**. We can add the evaluated level to **EVAL_RIESGOPRESENTE** adding **ALIAS** field at each observation. Also, we can assign easily the methodology used and check their distribution:

```
evalriesgopresente <- evalriesgopresente %>%
  mutate(METODOLOGIA_UTILIZADA = case_when(
    !is.na(FK_EVAL_FINE_NIVELRIESGO) ~ "FINE",
    !is.na(FK_EVAL_INSHT_NIVELRIESGO) ~ "INSHT",
    !is.na(FK_EVAL_GTC45_NIVELRIESGO) ~ "GTC45",
    TRUE ~ NA))
```

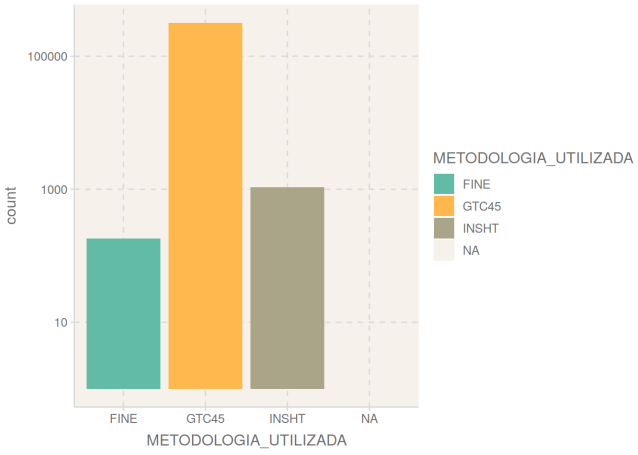


Figure 3.38: METODOLOGIAUTILIZADA distribution (log10 scale)

As we see at figure 3.38, almost all measures are done using **GTC45**. Also, there are some missing values that must be removed. Now, we can add some information about de kind of risk evaluated. In order to do this, first, we need to load tables **CORE_RIESGO** and **CORE_TIPORIESGO**:

CORE_RIESGO

```
## Observations: ??
## Variables: 7
## Database: spark_connection
## $ ID                <chr> "0033559-20060206194825-0001-W", "0033559-201...
## $ CODIGO            <chr> "208", "219", "130", "131", "127", "205", "21...
## $ ALIAS             <chr> "Exposición a vibraciones", "Enfermedades rel...
## $ FK_CORE_PROTOCOLO <chr> "9680321-20060612144511-0001-W", NA, NA, NA, ...
## $ FK_CORE_TIPORIESGO <chr> "3137026-20060206190206-0001-W", "3137026-200...
## $ ELIMINADO         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL...
## $ FECHABORRADO      <chr> NA, NA, NA, NA, NA, NA, "", NA, NA, NA, "", "...
```

CORE_TIPORIESGO

```
## Observations: ??
## Variables: 5
## Database: spark_connection
## $ ID           <chr> "3137026-20060206190206-0001-W", "9415390-200602061...
## $ CODIGO       <chr> "200", "100", "TRPMRM"
## $ ALIAS        <chr> "Enfermedades Laborales", "Accidentes de Trabajo AT...
## $ ELIMINADO    <lgl> FALSE, FALSE, TRUE
## $ FECHABORRADO <chr> NA, NA, NA
```

If we check them, at **CORE_RIESGO** there are 58 types of risk such as explosions, exposure to asbestos or excessive effort that in turn can generate, either occupational diseases or accidents at work. We need to filter only those related to work accidents. This can be checked at **CORE_TIPORIESGO** and are that ones with value **9415390-20060206190136-0001-W** at field **FK_CORE_TIPORIESGO** on **CORE_RIESGO**. After that, we will have only 38 types of risk related to work accidents. Now, we could add risk type to the evaluations using field **ALIAS** of **CORE_RIESGO**.

Once identified all evaluated risks, if we want to study the relationships between measured risk levels and accidents at a company we need to add the **company ID** to the evaluations. In order to do it, will be necessary to go on with the related tables till obtain the location ID at **IDENT_PELIGROPRESENTE**. After this, we will need to obtain for all locations which company pertains them. We can start loading tables **IDENT_FACTORRIESGOPRESENTE** and **IDENT_PELIGROPRESENTE** and checking their contents. This tables also have all records duplicated. After clean them, if we examine their contents:

IDENT_FACTORRIESGOPRESENTE

```
## Observations: ??
## Variables: 8
## Database: spark_connection
## $ ID                        <chr> "20150223135626-9978512-0953-W", "20150...
## $ FK_IDENT_PELIGROPRESENTE <chr> "20150223135625-8553566-0359-W", "20150...
## $ FK_CORE_FACTORRIESGO     <chr> "6.02.02", "6.03.02", "4.06.05", "1.07....
## $ CODIGO                   <chr> "6.02.02", "6.03.02", "4.06.05", "1.07....
## $ NOMBREPERSONALIZADO      <chr> "Baja tensión", "Medios de almacenamien...
## $ ELIMINADO                <lgl> FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, ...
## $ FECHABORRADO             <chr> "", "20170203", "20170203", "20170203",...
## $ DESCRIPCION              <chr> "Descripción", "Descripción", "Descripc...
```

IDENT_PELIGROPRESENTE

```
## Observations: ??
## Variables: 10
## Database: spark_connection
## $ ID                              <chr> "20150219213500-8684705-0790-W", ...
## $ FK_CORE_PELIGRO                 <chr> "1.01", "1.03", "7.05", "5.04", "...
## $ FK_IDENT_PELIGROPRESENTE_PADRE <chr> NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ FK_CORE_UBICACION               <chr> "20141120140210-3069698-0624-W", ...
## $ FK_CORE_SUBPELIGROACTIVIDAD     <chr> NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ CODIGO                          <chr> "1.01", "1.03", "7.05", "5.04", "...
## $ NOMBREPERSONALIZADO             <chr> "Virus", "Hongos", "Derrumbe", "M...
## $ ESOBLIGATORIO                   <lgl> FALSE, FALSE, FALSE, FALSE, FALSE...
## $ ELIMINADO                       <lgl> TRUE, FALSE, TRUE, FALSE, FALSE, ...
## $ FECHABORRADO                    <chr> "20150309", "", "20170203", "", "...
```

We can see that, **IDENT_FACTORRIESGOPRESENTE** has a valuable information about the factor of risk at **NOMBREPERSONALIZADO**, however with a simple check of its contents, it seems to be an editable value subject to errors. The good news are that every observation is related to table **CORE_FACTORRIESGO**, where the information that we are searching for is well inserted. In order to solve this, first, we need to load table **CORE_FACTORRIESGO**:

```
## Observations: ??
## Variables: 8
## Database: spark_connection
## $ ID                          <chr> "1.01.00", "1.02.00", "1.03.00", "1....
## $ CODIGO                      <chr> "1.01.00", "1.02.00", "1.03.00", "1....
## $ ALIAS                       <chr> "Sin especificar", "Sin especificar"...
## $ FK_CORE_PELIGRO             <chr> "1.01", "1.02", "1.03", "1.04", "1.0...
## $ FK_CORE_SUBPELIGROACTIVIDAD <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ ELIMINADO                   <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, F...
## $ FECHABORRADO                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ DEFECTOSELECCIONADO         <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, ...
```

Where there are 146 factors of risk related to illness or accidents like high voltage, impact, heat or noise identified at **ALIAS** field. We can add this information to **IDENT_FACTORRIESGOPRESENTE** and drop the unneeded fields.

With **IDENT_PELIGROPRESENTE** we can act in a same way. In this case, we have to load first table **CORE_PELIGRO**:

```
## Observations: ??
## Variables: 8
## Database: spark_connection
## $ ID                     <chr> "1.01", "1.02", "1.03", "1.04", "1.05", "...
## $ FK_CORE_PELIGROFAMILIA <chr> "1.01", "1.02", "1.03", "1.04", "1.05", "...
## $ ALIAS                  <chr> "Virus", "Bacterias", "Hongos", "Ricketsi...
## $ FK_CORE_TIPOPELIGRO    <chr> "0000001-20051109190600-0002-W", "0000001...
## $ CODIGO                 <chr> "1.01", "1.02", "1.03", "1.04", "1.05", "...
## $ FK_CORE_PELIGRO_PADRE  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ ELIMINADO              <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,...
## $ FECHABORRADO           <chr> "", "", NA, NA, NA, NA, NA, NA, NA, NA, N...
```

Where there are 47 dangers related to factors of risk like electrical or mechanical. We can add this information to **IDENT_PELIGROPRESENTE** and drop the unneeded fields.

As a last step of cleaning, we can also review the classes on risks, factors of risk and hazards, and see if exists any issues. Starting from risks, there is a class designated to other kinds of accidents, **"Otros accidentes"**, which for this analysis can be considered as a missing value. Now, for risk factors there are two classes designated to other risks and to unspecified risks, **"Otros"** and **"Sin especificar"** respectively. We can assume for this analysis that these cases could be the same. Finally, with dangers we can see that there are two classes being the same, **"Terremoto"** and **"Sismo"**, that refers to earthquakes.

At this point, we can obtain the location ID, **FK_CORE_UBICACION**, from **IDENT_PELIGROPRESENTE**. This field points to **CORE_UBICACION** table. Once we have the location ID we need to obtain the company ID, or **FK_CORE_EMPRESA**. This can be done taking it from every type of location, thus some joins will be needed. First, we have to load all type of location tables, and **CORE_UBICACION**. All these tables also will have duplicated rows. Now, doing some join operations with every table depending of their relationships, we can obtain all company ID for every location. If we check the results, almost all locations have a company ID, the ones that don't have will be deleted.

After that, if we join together the evaluations and the locations we will have the company ID for every evaluation of a risk.

3.4.2 Data modeling and evaluation

The following are the explanations of the resolution of questions.

All code developed at this section can be found at appendix B.

3.4.2.1 Question 1

"What is the relationship between the evaluated risk levels and the accidentability of a company?"

This question needs to be answered doing an exploratory analysis and looking for the possible relationships. In order to do it, we need to get the evaluated risks by company per year. First, we have to load the **risk evaluations** and **ubications** wrangled at previous phases. Afterwards, we can join them by **FK_CORE_UBICACION** in order to have every evaluated risk related to its company. Afterwards, there are about 145 observations without **FK_CORE_EMPRESA** value. We have to drop them and also fields **ID** and **FK_CORE_UBICACION** since they are not necessary. Also, there are some observations without either a risk or a risk factor.

Used methodologies and levels

As we saw during previous phases, almost only **GTC45** methodology is used. Then, if we check the distribution of levels assigned to evalutations using this methodology:

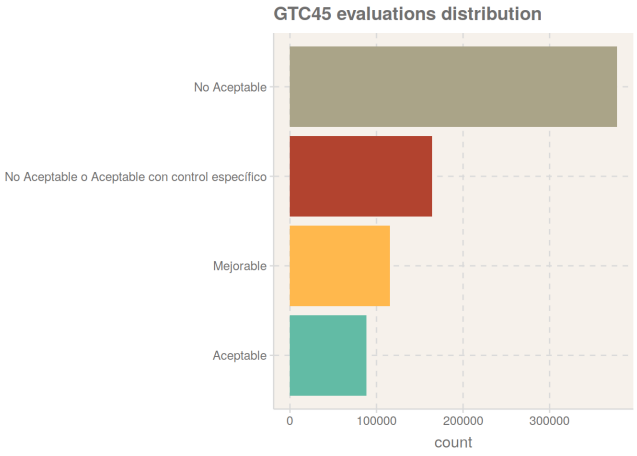


Figure 3.39: GT45 levels distribution

We can see that more than a half part of the evaluations corresponds to **No Aceptable**, this are risks that must be solved as soon as possible. Now, if we check when evaluations

happened we can see that year value goes from 2015 to 2018. If we check their distribution we can see that most of the evaluations were carried out from the end of 2017 until the end of 2018 (see figure 3.40):

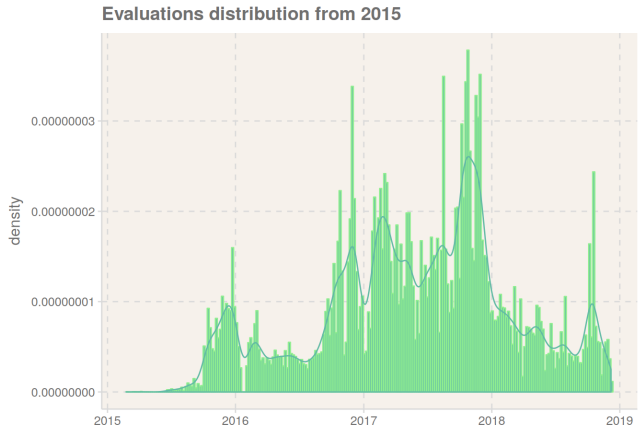


Figure 3.40: Risk evaluations over years

If we take only **GTC45** methodology and verify the distribution of its levels:

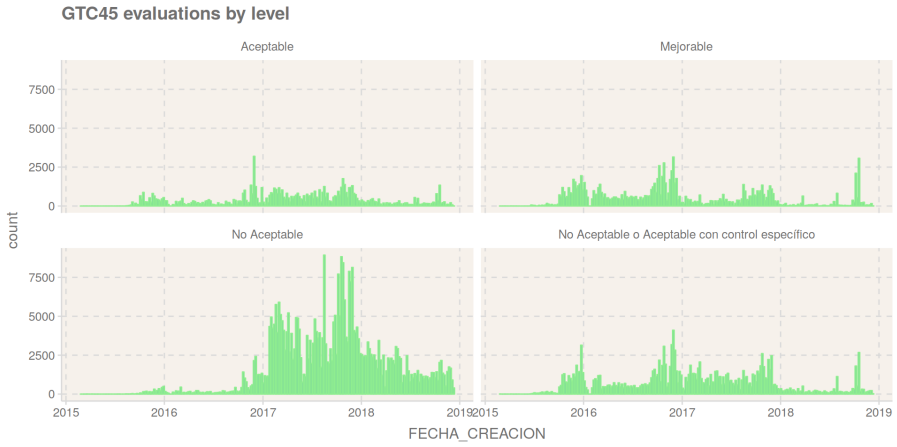


Figure 3.41: GT45 levels distribution over years

We can see that most relevant changes occurs at level **No Aceptable**. This level indicates that the urgency is maximum, so there may be an accident if the risk is not solved. In addition, it doesn't allow a margin of time for its resolution.

Risks

In order to take more information about the types of risks, we can see that there are 37 kinds of them (see figure 3.42):

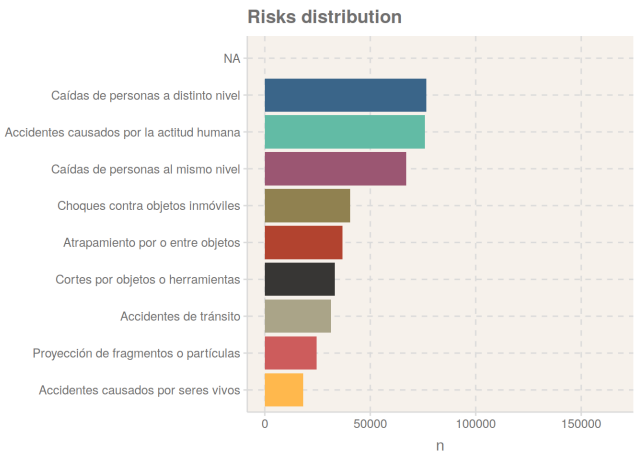


Figure 3.42: Most common types of risk distribution

Where most common are indeterminate and, afterwards, the most representative classes are accidents caused by human activity and falls, either at different levels or at same level. If we check the relationships between the indeterminate class and the factors of risk we can see that a major part of them continues undetermined, but if we do the same again with the dangers we finally have that this kind of risk is mainly related to public dangers and earthquakes. Public dangers are those that evereyone can suffer like theft, kidnapping, attack, sabotage, traffic accident, etc:

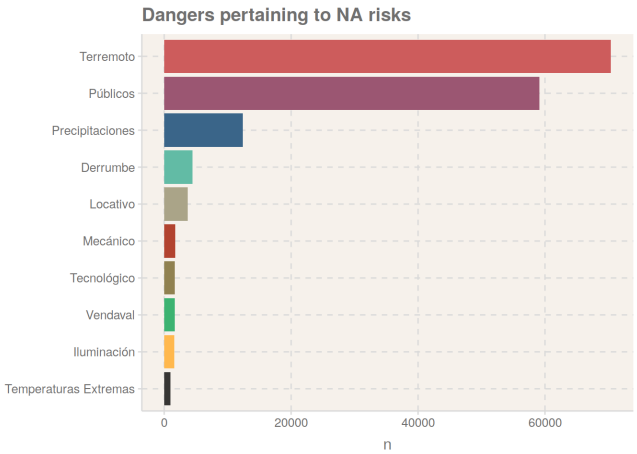


Figure 3.43: Dangers related to NA risks

Factors of risk

Now, if we check the factors of risk, we know that there are 96 factors. Also, the distribution of most common shows that, again, most are indeterminate. Taking into account the rest of the classes, most representative are those related to order and cleanliness, and with sliding or irregular work surfaces (see figure 3.44):

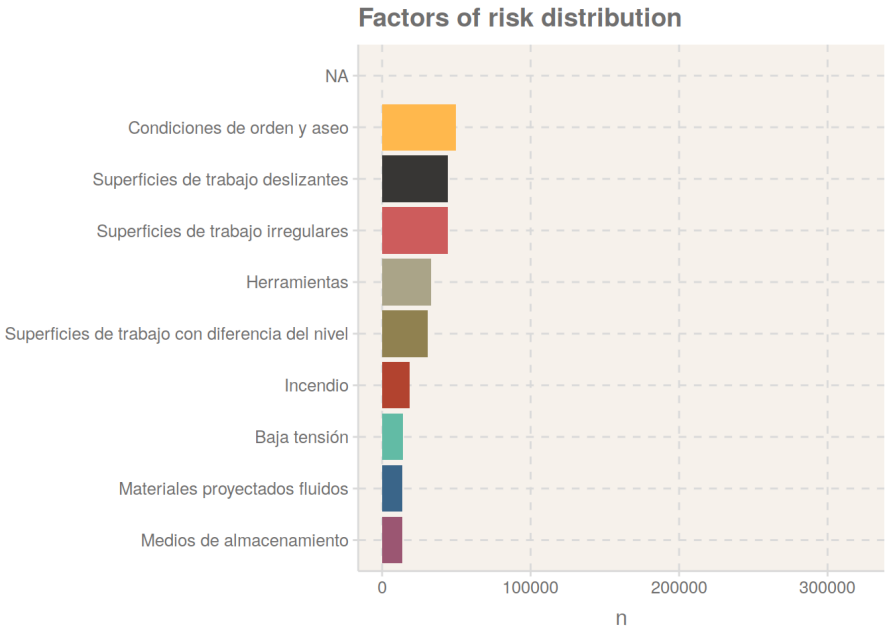


Figure 3.44: Most common types of factor of risk distribution

Dangers

Going on deeper with dangers, we know that there are 45 kinds of them. The distribution of most common is:

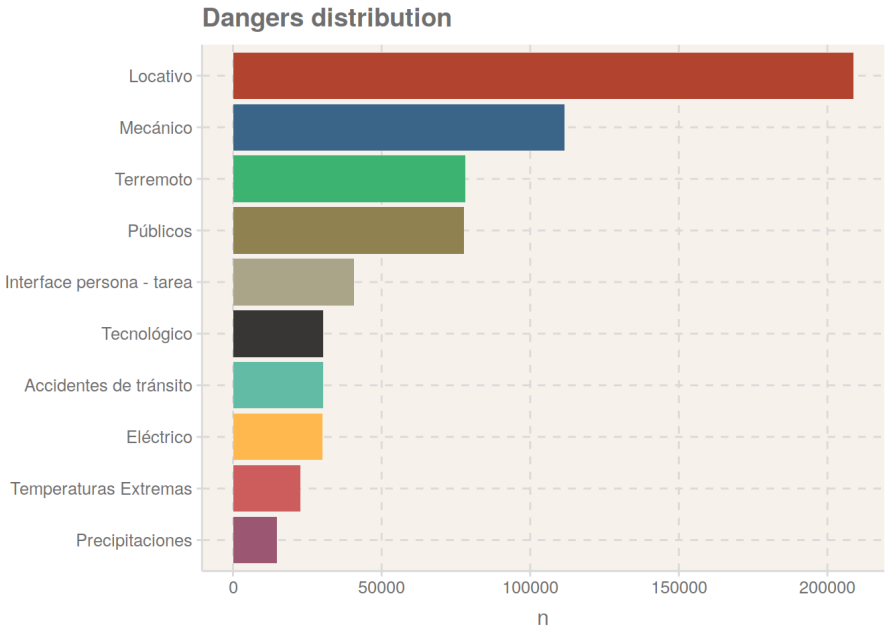


Figure 3.45: Most common types of danger distribution

Where we can see that there are no missings and the most common danger, Locativo, is

related to the place where work is produced, i.e. the geographical or work areas. This risk is important because it's a permanent condition throughout the workday. In addition, mechanical, public or earthquake-related risks are also relevant. If we see those factors related to public danger we can check that these corresponds with the explanation done before about public dangers:

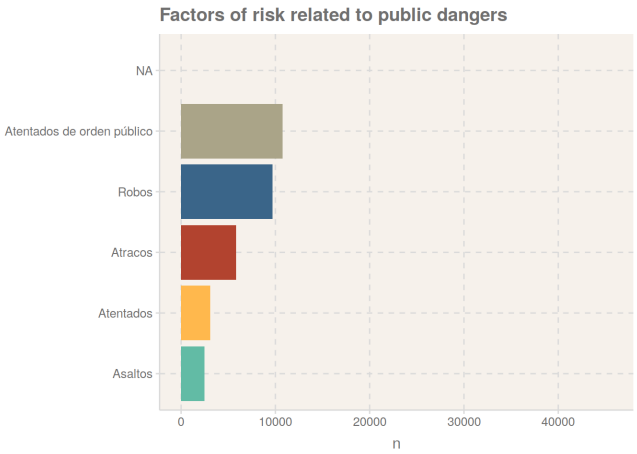


Figure 3.46: Most common types of factors of risk related to public dangers

In addition, we can see the relationships between the evaluated levels of **GTC45** methodology and dangers, and check that, on all levels locative, mechanical and public dangers are present as most important dangers. Also, on **No Aceptable** risk level, earthquakes and interfaces person-task are destacable (see figure 3.47):

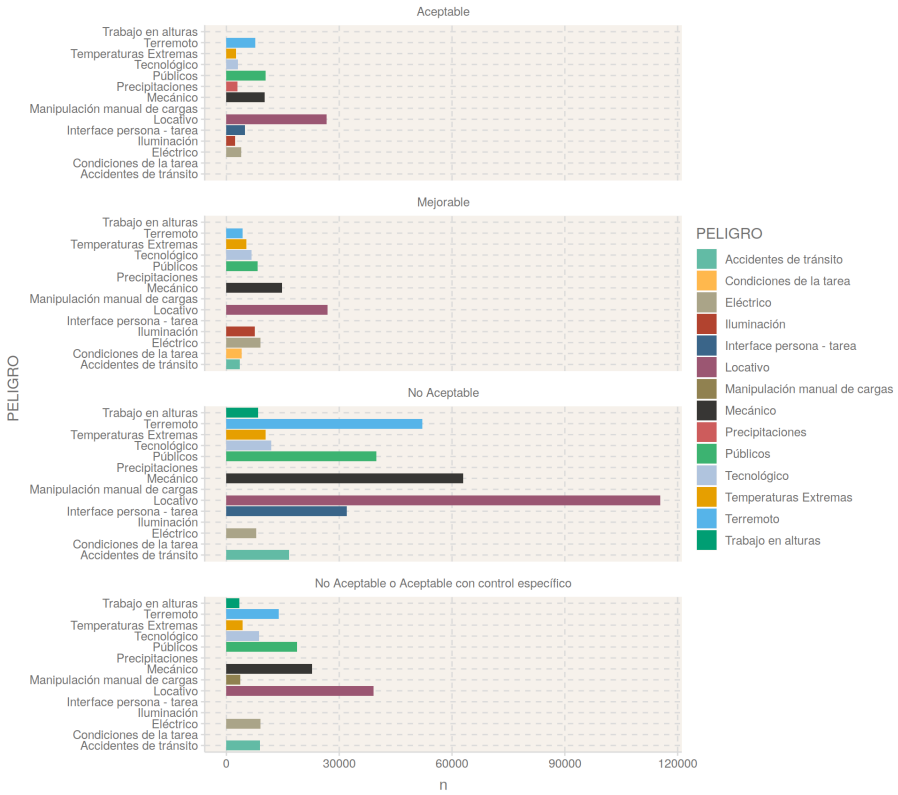


Figure 3.47: Most common types of danger by GTC45 levels

Accidents

Once worked on evaluations, we have to load accidents and historical data about companies in order to relate the accident with a company. After joining them, it seems that all accidents have a company ID.

It is important to know that, for each accident, a present risk factor can be added, if it exists. However, there are only about 400 observations with this condition. We don't know if it happens because the risk factor had not been evaluated once the accident occurred or due to a human error when entering the data. This can be seen with field **FK_IDENT_FACTORRIESGOPRESENTE**.

To be able to compare the accident rate with risk assessments, we need to do it by date. Unfortunately, we do not have the date on which a risk was solved, therefore, we need to establish a time window. If we look at levels, in those that have a certain urgency a window of time is fixed for their solution in attribute **DIAS_PLANIFICACION**. In this case, focusing on **GTC45** methodology, two of its levels have this time window, 0 days for **No Aceptable** and 365 days for **No Aceptable o Aceptable con control específico**. Taking this into account, we should expect a reduction of accidents inside the year just after evaluations. If we

check the years were accidents occured we can see that these goes from 1995 to 2018, but as the evaluations occurred between 2015 and 2019, we must stay between these years to see if there is any change caused by the evaluations. If we check the distribution of accidents and the evaluations calified as **No Aceptable** on these years:

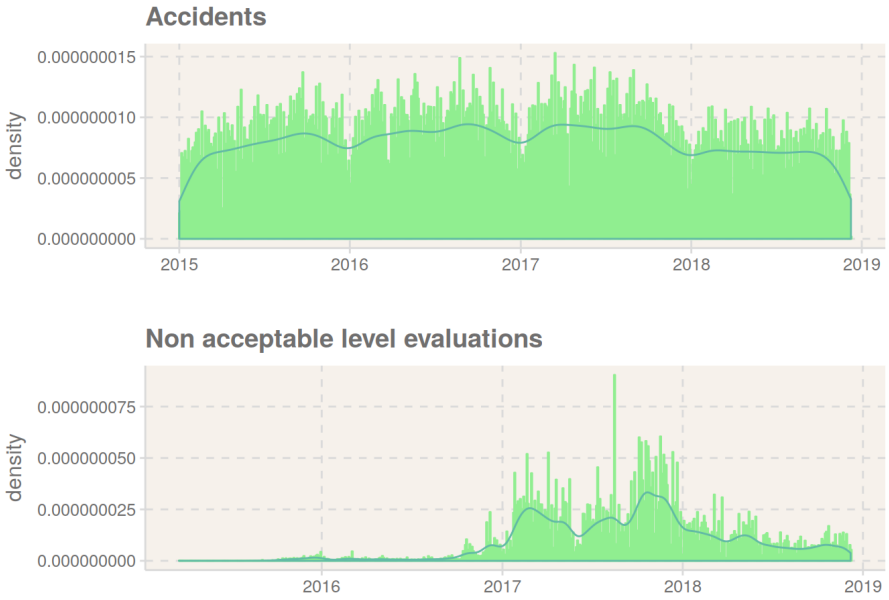


Figure 3.48: Accidents and No Aceptable evaluations from 2015 to 2018

At accidents distribution, we can see a pattern that is repeated every year, where there is a clear reduction in accidents at the end of the year and another reduction, although more subtle, on Easter. We can also see that, since 2015 there has been an upward trend in the number of accidents, while since 2018 these have been drastically reduced. If we compare it with **No Aceptable** evaluations we can think that may exist a relationship between the evaluations made at 2017 and the following reduction of accidents. Also, we can check accidents from 2010 and see that the trend was to increase, changing dramatically from 2014 to 2016, and afterwards obtaining a more leisurely increase until 2017. This increase may happened due to an increase in the companies/workers under control (see figure 3.49):

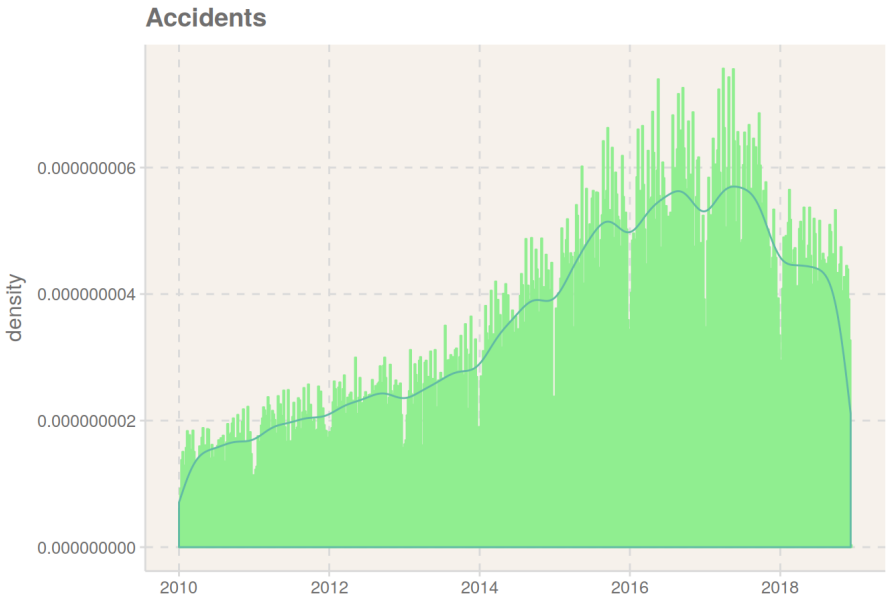


Figure 3.49: Accidents trend from 2010

Finally, if we obtain all **No Aceptable** evaluations present at main dangers and compare them against the number of accidents of companies present at these evaluations:

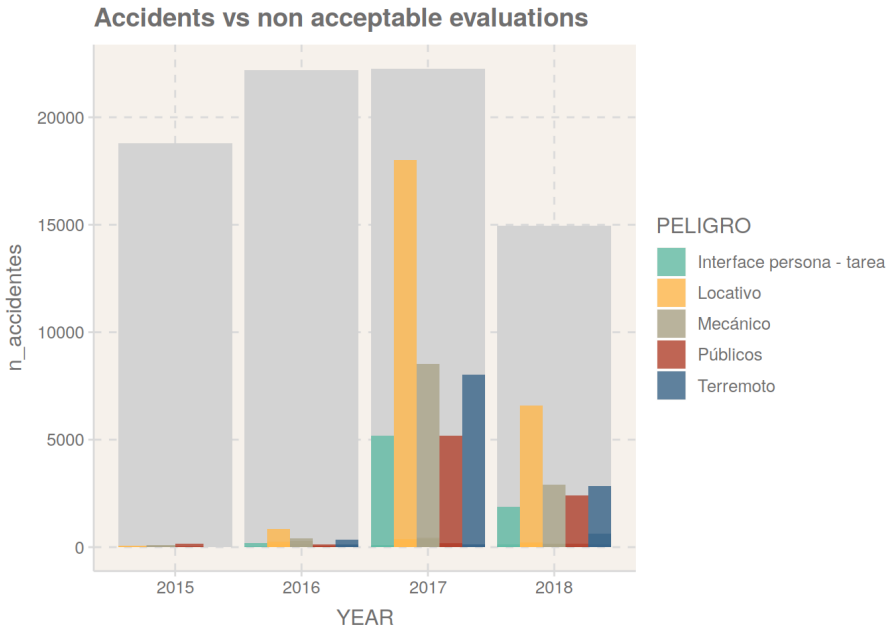


Figure 3.50: Accidents vs No Aceptable evaluations of main dangers

We can again see that may exist an inverse relationship between the number of evaluations made and the consequent number of accidents happened afterwards.

3.4.2.2 Question 2

"How much accidents will have a company in a year depending of its size, the ubication of its centers and its activity?"

Starting to answer the question, we have to load the historical data of the company that is registered when an accident happens. Afterwards, we can see that the information related to the activity of a company, **CNAE** field, is available in all the observations, while the information related to location, village and province, is also available in almost all. Unfortunately we don't have the size of the company when an accident happened. Now, we have to relate this information with the accident. In order to do it, we have to load cleaned accidents dataset and do a join. From the accident itself we can also obtain the location of the attributes **FK_CORE_POBLACION** and **FK_CORE_PROVINCIA**. In reality, this location is the real place where the accident occurred, being able to be a different place than the company. Anyway we must stay using company location since we are looking for a comparision between number of accidents and location of a company, not its different centers. This could be more suitable for a study of accidents by location.

Now, its time to see distribution of the number of accidents by a company in a year:

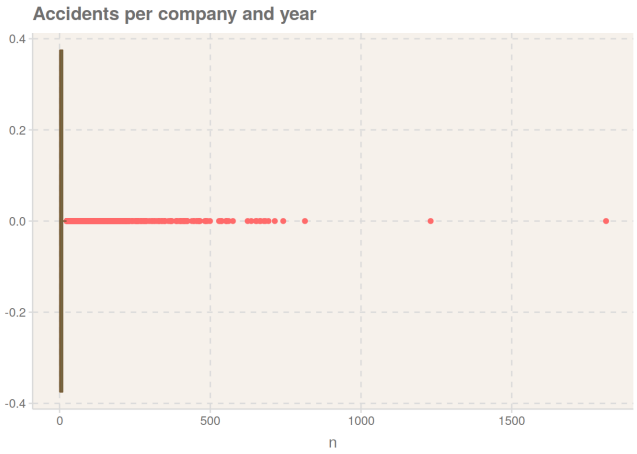


Figure 3.51: Accidents per company and year

This clearly shows that there are some outliers. If we get closer (see figure 3.52):

Figure 3.52: Accidents per company and year (limited by range)

We can see that almost all values goes from 0 to 8 with a low median. Maybe a separation of these different populations should be the best way to do the study, unfortunately we don't have the historical value of the number of workers of a company but we can use the current value, **NUMTRABAJADORES** from **CORE_EMPRESA**, in an approximate way. Then it seems that some companies have 0 workers. As this should be considered as NA, we need to drop these observations. Also, we can create a new attribute, **size**, based on the Colombian definition of a company by it size: <10 for micro, (10-50] for small, (50-200] for medium and >200 for big.

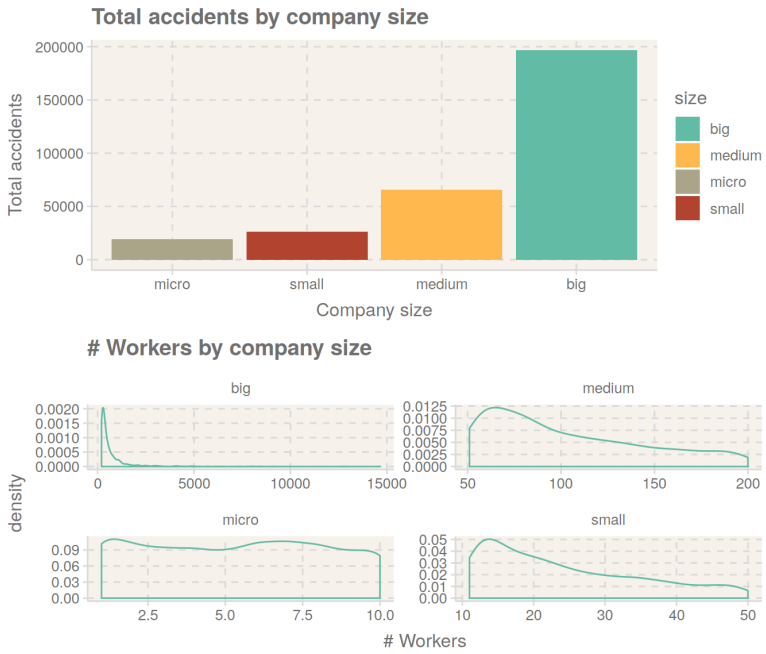


Figure 3.53: Accidents distribution and density of workers by company size

As can be seen at figure 3.53, it seems that almost all accidents have occurred at big companies. This is the expected since bigger the number of workers bigger the number of accidents. Also, we can see that big companies are concentrated below the amount of 1000 workers, medium and small size companies number decrease as their number of workers increases, and the micro companies are near uniformly distributed. Now, if we check the distributions based on the company size:

Figure 3.54: Accidents per company and year by size

We can see that the distribution is a bit more clear, also there are outliers that can be seen in the plot. As these distributions are so skewed we can transform its values to log10 and get a normalized distribution (see figure 3.55):

Figure 3.55: Accidents per company and year by size (log10)

We obtain a more normal distributions for each size. Also, we can check the centrality and variability measures for each size:

```
## # A tibble: 4 x 8
##   size    mean median diff_mean_median   mad skewness    sd kurtosis
##   <chr>  <dbl>  <dbl>            <dbl> <dbl>    <dbl> <dbl>    <dbl>
## 1 big    1.19   1.23          -0.0393  0.365  -0.0989 0.570   -0.168
## 2 medium 0.692  0.699         -0.00724 0.391   0.271  0.505   -0.724
## 3 micro  0.289  0               0.289   0.289   1.84   0.454    3.08
## 4 small  0.278  0               0.278   0.278   1.41   0.356    2.17
```

Which shows that there is a relationship between the size of the company and the number of accidents by year, except for the cases of small and micro companies. These two sizes have almost the same distribution of values. This can be explained due to the fact that a company can grow easier from micro to small than from small to medium or from medium to big, thus the approximation used about the number of workers may be not accurate in these cases. Also, these two distributions are more skewed, probably by a bigger number of outliers. Now, we can remove some outliers taking into account that the data is distributed following a normal-like shape. In order to do it, we can compute the absolute value of the difference between an observation and the median of accidents in a year by size, and dividing it by the median of the absolute deviation. Then we can discard those observations with a value over or equal a threshold:

```
nrows <- nrow(num_accidents_company_year_m)
initial_nrows <- nrow(num_accidents_company_year_m)
threshold <- 7.5

repeat {
  outliers <- num_accidents_company_year_m %>%
    left_join(select(measures_per_size, size, median, mad, kurtosis),
              by = "size") %>%
    filter(abs(n_log10 - median)/mad >= threshold) %>%
    pull(FK_CORE_EMPRESA)

        accident_company <- accident_company %>%
          filter(!(FK_CORE_EMPRESA %in% outliers))

        num_accidents_company_year_m <- num_accidents_company_year_m %>%
          filter(!(FK_CORE_EMPRESA %in% outliers))

        measures_per_size <- num_accidents_company_year_m %>%
          group_by(size) %>%
          summarise(mean = mean(n_log10),
          median = median(n_log10),
          diff_mean_median = mean - median,
          mad = median(abs(n_log10 - mean(n_log10))),
          skewness = e1071::skewness(n_log10),
          sd = sd(n_log10),
          kurtosis = e1071::kurtosis(n_log10, type = 1))

        if (nrows == nrow(num_accidents_company_year_m)) {
          break
        }

    nrows <- nrow(num_accidents_company_year_m)
}
```

Using a threshold of 7.5 we have removed the 0.4% of observations, considered as outliers. All deletions have been done at micro and small sizes, reducing a little bit the skewness and the kurtosis of the distributions.

Another thing that we can do in order to gain more information about the relationship between

company size and accidents by year is to obtain a relative value, this is the number of accidents dividied by the number of workers in a year:

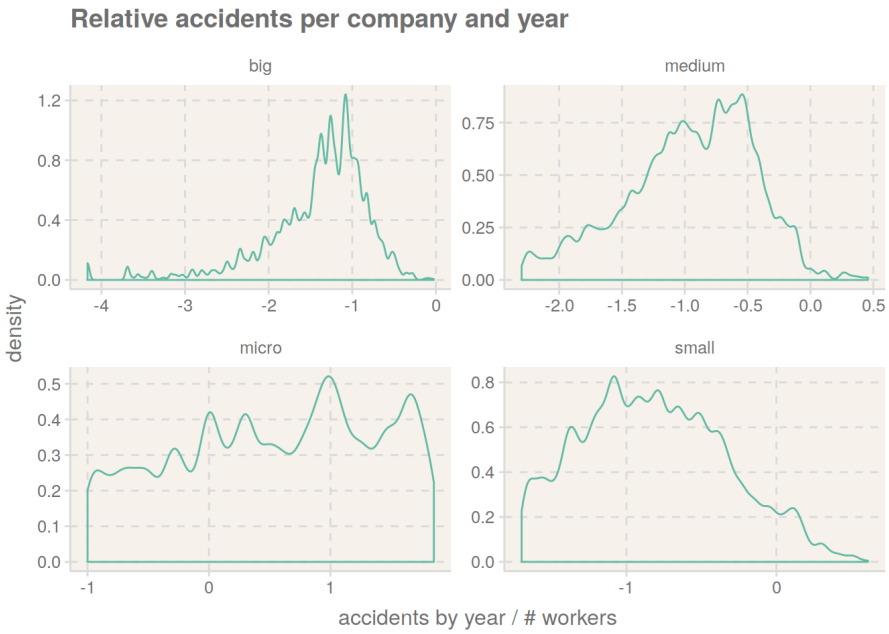


Figure 3.56: Relative accidents per company and year

The results have been converted to log10 in order to obtain a normal-shape. It is expected, as the graphs shows at figure 3.56, that the relationship between accidents per year and number of workers increases as the company size decreases, nevertheless is not expected, for example, that a company with a small amount of workers has a much bigger amount of accidents in a year. In this case, we can analyze, for example, the accidents by year of those companies where number of accidents in a year reaches at some point an amount equal to five times the number of workers. We found about 477 companies, all peartaining to micro size group, thus they may pertain in reality to another size class since the number of workers may be erroneous. We need to drop all these observations.

With this information we can check the correlation between the company size and the amount of accidents in a year using the **Pearson's coefficient**:

```
##
##  Pearson's product-moment correlation
##
## data:  num_accidents_company_year_m$n_log10 and log10(num_accidents_company_year_m
## t = 100, df = 20000, p-value <0.0000000000000002
## alternative hypothesis: true correlation is not equal to 0
```

```
## 95 percent confidence interval:
##  0.65 0.67
## sample estimates:
##  cor
## 0.66
```

This shows that with a 95% of confidence interval that the correlation is 0.66. Also, the distribution shows a spurious correlation.

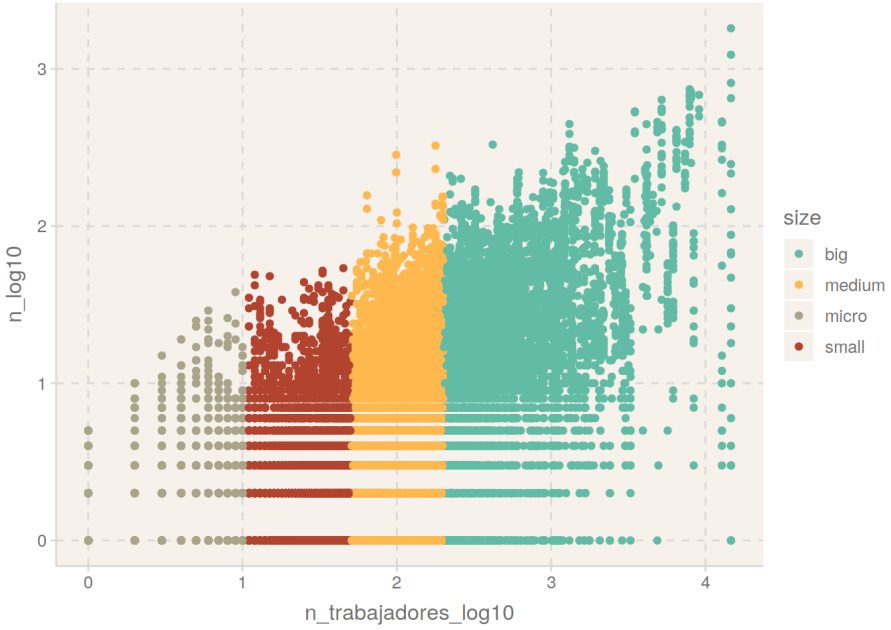


Figure 3.57: Correlation between number of workers and number of accidents by company

Now, once studied the impact of the size of a company over the number of accidents in a year, we can check if there is a relationship between this number and the location of the company. After deletion of a little bit of observations that had missing values at province or village fields we can check the distribution over every province (see figure 3.58):

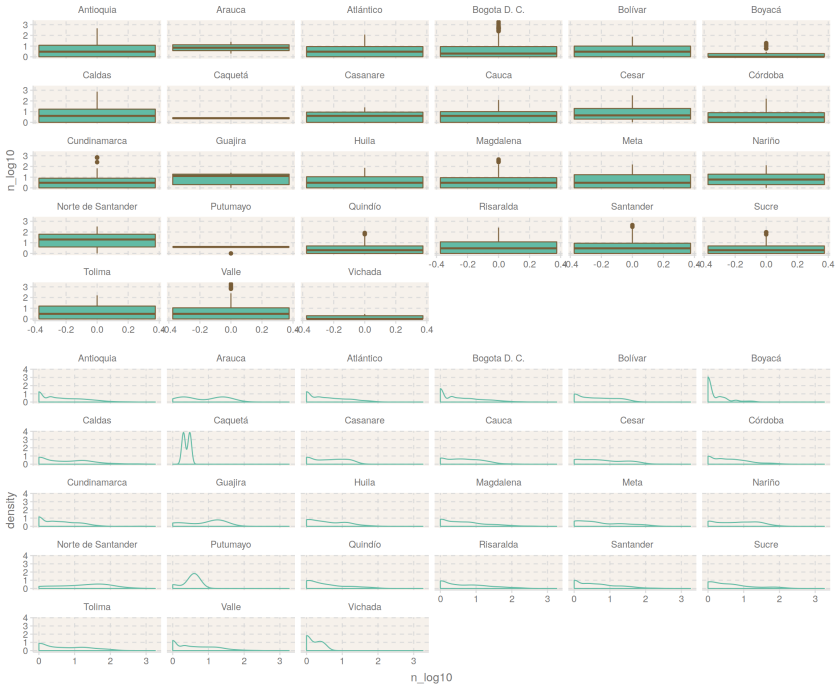


Figure 3.58: Yearly accidents by company province

This shows differences between provinces that could be due to several factors associated to every zone. Also we can check if company size and province have some relationship:

Figure 3.59: Yearly accidents by company province and size

Where we can see at figure 3.59 that, at some places, the expected distribution of accidents by size seen before is not followed. This is the case, for example, of Casanare or Norte de Santander. Also, some provinces don't have some kinds of company sizes, like Boyacá or Vichada. These particularities can be decisive predicting the number of accidents. If we want a measure this difference between provinces we can do a **one-way ANOVA test**:

```
##                Df Sum Sq Mean Sq F value               Pr(>F)
## PROVINCIA      26     86    3.31    9.59 <0.0000000000000002 ***
## Residuals   23481   8109    0.35
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Which shows that there is a significant difference between the average number of accidents by company provinces.

With the **company activity**, we need to clean their names in order to group them. After that, we will see that most of activities have 10 or less companies with an accident registered. This could produce some bias and alter the results of relationships and predictions as this amounts are not representative. If we compare the activities with highest number of accidents in a year these are those related with justice, minerals extraction, manufactures and agriculture like palm oil or banana production, whereas those with relative number of accidents by number of workers are those related with metal foundries, coffee production and some manufactures. If we analyze the companies related to these top accidented activities:

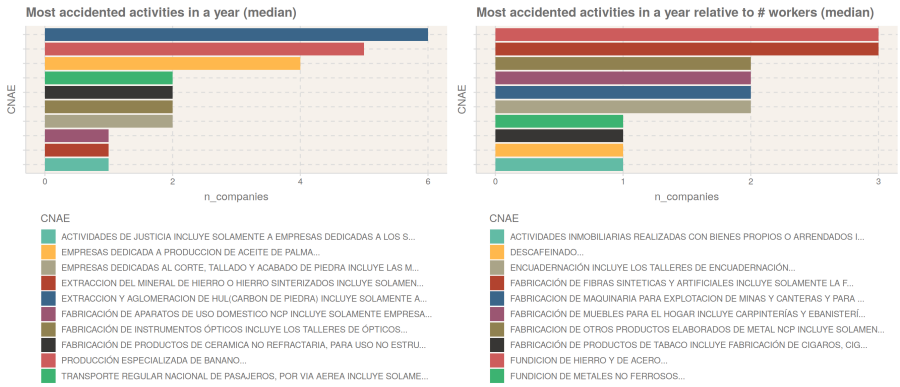


Figure 3.60: Most accidented activities in a year

We can see that, for absolute or relative numbers of median accidents by year, the top activities are represented by a little number of companies. Also, we can divide the analysis by company size and see the possible relationship with activities (see figure 3.61):

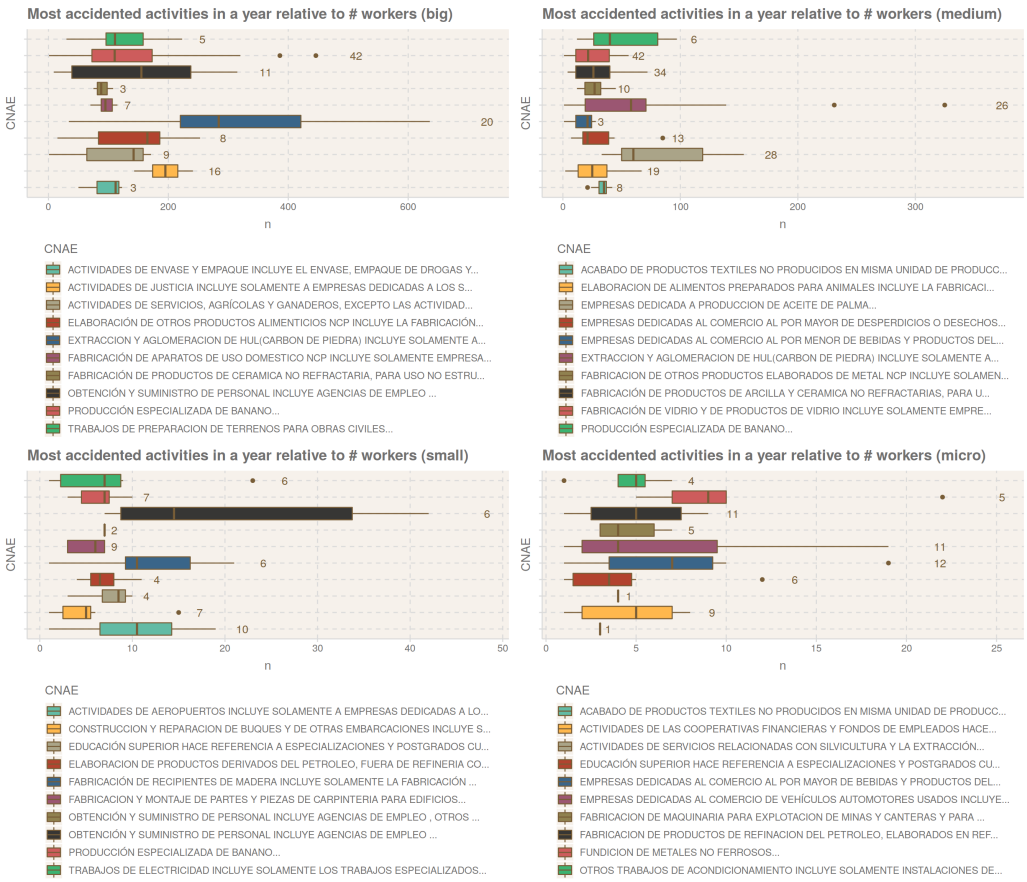


Figure 3.61: Most accidented activities in a year by size

This shows that exist some activities with only registered accidents of one company in one year that have a high median, as its shown at micro size companies. Also, most of activities have a little number of observations whereas only a few of them have an important amount to represent a better distribution of accidents in a year. This is the case of banana production, coal extraction, glass, clay and metal manufactures.

Modeling

In order to infer the number of accidents that a company will have in a year we need to create a model with the predictors seen before. We will start using *Spark's MLib*. First of all, we need to partition data into training and testing sets. Also, we have to **one-hot-encode** province and activity since *Spark's MLib* algorithms don't do this by themselves.

```
data_tbl <- copy_to(sc, num_accidents_company_year_m, "num_accidents_company
data_tbl <- data_tbl %>%
  mutate(n_log10 = log10(n)) %>%
  ft_string_indexer("PROVINCIA", "PROVINCIA_idx") %>%
  ft_one_hot_encoder("PROVINCIA_idx", "PROVINCIA_ohe") %>%
```

```
  ft_string_indexer("CNAE", "CNAE_idx") %>%
  ft_one_hot_encoder("CNAE_idx", "CNAE_ohe")

partitions <- sdf_partition(data_tbl, training = 0.8, testing = 0.2)
```

Checking most common algorithms used at regression of structured data, it seems that ***GBM*** has the best performance. It shows an **error of 2 accidents** on test set and the **variability of data is explained by the model in a 78%**.

Modeling with *H2O* results faster than same algorithms at *Spark's MLib*. Also, *H2O* algorithms do themselves the needed transformations on categorical variables. After checking algorithms as *Random Forest*, *GBM*, *GLM*, *XGBoost*, *Neural Networks* or an *stacked ensemble* of all of them, *XGBoost* seems to have the best results. We have covered some hyperparameters using a grid search:

```
training[,"PROVINCIA"] <- as.factor(training[,"PROVINCIA"])
training[,"CNAE"] <- as.factor(training[,"CNAE"])

nfolds <- 10

xgboost_params <- list(learn_rate = c(0.1, 0.2, 0.3),
                       max_depth = c(20, 25, 30),
                       sample_rate = c(0.4, 0.6, 0.8, 1),
                       ntrees = c(30, 40, 50))

xgboost_grid1 <- h2o.grid("xgboost",
                          x = c("NUMTRABAJADORES", "PROVINCIA", "CNAE"),
                          y = "n_log10",
                          grid_id = "xgboost_grid1",
                          training_frame = training,
                          hyper_params = xgboost_params,
                          distribution = "gaussian",
                          nfolds = nfolds,
                          fold_assignment = "Modulo",
                          keep_cross_validation_predictions = TRUE)
```

Results are pretty similar to those obtained with *GBM* on *Spark's MLib*. Also, the **most important variable** to infer the results is the **number of workers**. Some **activities** and **provinces** contribute a little bit in the improvement of the precision.

Chapter 4

Conclusions

4.1 Work conclusions

As the project is divided in two main objectives I am going to detail them separately.

Regarding of the first objective, the study of the different ways to deal with large amounts of data in R, has been satisfactory accomplished. A big amount of work of study and trial has been done with the consequent acquirement of knowledge. Has been checked that, despite R's limitations, it can be boosted on big data areas thanks to the studied extensions. Using the selected solutions, one can carry out all kinds of data science studies dealing with large amounts of data, moving the R's limits far away. This fulfill one of the main personal objectives for this study since its focus was to be generalizable to other big data problems.

About the data science study, all subtasks have been carried out except for the fact that, due to time restrictions, modeling phase of question 3 has been discarded. Nevertheless, this question was well worked at previous phases as can be seen at this memory. Some issues have happened during the study development. Most important was the fact that the database fields were in a bad state, needing too much time for cleansing them. The database seems to have underlying operations as massive data traspassings or database updates that have produced too much outliers or bad insertions. Also, regarding answering the questions, asked by the same professionals that made the database, there was a lack of information needed to answer them. This is the case of the historical number of workers of a company registered at the accident's time or the moment when a risk was solved. These issues were solved using different strategies as can be seen at this memory. Anyway, I could show a relationship for the first question and a suitable model for the second. A better solution would require time resources that are out of the scope of this project.

4.2 Future work

One of the reasons to choose this project was the possibility to combine it at work. At work, a data science project with scopes bigger than this study, was in an advanced state of acceptation. Nevertheless, finally the project was canceled, thus there is no future work to do with it. Anyways, I currently use all knowledges acquired with this project in other ones and I plann to going on exploring the selected solutions used at this study and beign up to date with their changes.

Appendix A

Data understanding and data preparation RMarkdown script

````
---
title: "COLMENA SEGUROS Data Wrangling"
author: "Alberto Burgos"
date: "2/27/2019"
output:
  html_document:
    df_print: paged
  pdf_document: default
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)
```

## COLMENA SEGUROS EDA

# Environment setup

## Necessary libraries and options

```{r, results='hide', message=FALSE}
library(dplyr)
library(sparklyr)
library(ggplot2)
````
````

```
library(lubridate)
library(purrr)
library(tidyverse)
library(reshape2)

options(digits = 2, scipen = 999)
```

```
## Spark connection

```{r, results='hide', message=FALSE}
config <- spark_config()
config$`sparklyr.shell.driver-memory` <- "6G"
config$`sparklyr.shell.executor-memory` <- "4G"
config$`sparklyr.shell.executor-cores` <- "4"
config$`spark.yarn.executor.memoryOverhead` <- "1g"
sc <- spark_connect(master = "local", config = config)
```

Variables and auxiliary functions

```{r}
pc_na_threshold <- 60

sdf_gather_all <- function(tbl, gather_cols){

  gather_cols <- colnames(tbl)

  lapply(gather_cols, function(col_nm){
    tbl %>%
      select(c(col_nm)) %>%
      mutate(key = col_nm) %>%
      rename(value = col_nm)
  }) %>%
    sdf_bind_rows() %>%
    select(c('key', 'value'))
}
```
```

```
## Tables importation

```{r, message=FALSE, warning=FALSE}
setwd("~/")
accidente <- spark_read_parquet(sc, "accidente", "parquet/con_accidente")
contingencia <- spark_read_parquet(sc, "contingencia",
  "parquet/con_contingencia")
```

A first look

First of all, its needed to select only the kind of accidents occurred
at work:

```{r}
accidente <- accidente %>%
  filter(FK_CORE_TIPOACCIDENTE == "20141007182138-3948986-0005-W") %>%
  select(-FK_CORE_TIPOACCIDENTE)
```

Now, joining with the table CON_CONTINGENCIA, and discarding the deleted
observations:

```{r echo=FALSE, message=FALSE, warning=FALSE}
accidente <- accidente %>%
  inner_join(contingencia, by.x = FK_CON_CONTINGENCIA, by.y = ID,
             all = TRUE) %>%
  filter(!ELIMINADO) %>%
  select(-c(ELIMINADO, FECHABORRADO))
```

Lets to see a glimpse of every table:

```{r}
glimpse(accidente)
```

Checkpoint:
```{r message=FALSE, warning=FALSE, include=FALSE}
```
```

```
sdf_persist(accidente)
spark_write_parquet(accidente, path = "~/parquet/accidente",
  mode = "overwrite")
db_drop_table(sc, "accidente")
db_drop_table(sc, "contingencia")
```
```
```{r message=FALSE, warning=FALSE, include=FALSE}
accidente <- spark_read_parquet(sc, "accidente", "~/parquet/accidente")
```
```

Logical variables

Lets to see their distribution:

```
```{r, fig.height=12, fig.width=16, warning=FALSE, message=FALSE}
accidente %>%
  select_if(~is.logical(.)) %>%
  sdf_sample(fraction = 0.2, replacement = FALSE) %>%
  collect() %>%
  gather(key, value) %>%
  ggplot(aes(x = value, fill = value)) +
    geom_bar() +
    facet_wrap(~key, scales = "free")
```
```

As can be seen, there are some variables that either all or almost all values are missing, or have unique values. These variables must be dropped:

```
```{r}
accidente_c1 <- accidente %>%
  select(-c("ACTOINISEGURO", "ASISTENCIAMEDICA", "BOTIQUIN",
            "CAUSAMUERTE", "CONPERDIDADIAS", "ENSUPUESTO",
            "ENTRAYECTO", "LLEVADOHOSPITAL", "MUTUA",
            "OTROSCOSTES", "PERDIDAHORASACCIDENTADO",
            "PERDIDASHORAS", "PERDIDASMATERIALES",
            "TIPOACCIDENTE"))
```
```

```
## Numerical variables

Now, for numerical values, their distribution is:

```{r}
accidente_c1 %>%
  select_if(~is.numeric(.)) %>%
  sdf_sample(fraction = 0.2, replacement = FALSE) %>%
  collect() %>%
  gather(key, value) %>%
  ggplot(aes(x = key, y = value)) +
    geom_boxplot() +
    facet_wrap(~key, scales = "free")
```

```{r, message=FALSE}
accidente_c1 %>%
  select_if(~is.numeric(.)) %>%
  sdf_describe()
```

This denotes that these variables are almost empty:

```{r}
accidente_c2 <- accidente_c1 %>%
  select(-c("DIASPERDIDOS", "GRADOLESION"))
```

Character variables

For character variables, all empty values (""), are assigned to NA:

```{r}
accidente_c3 <- accidente_c2 %>%
  mutate_if(is.character, list(~if_else(. == "", NA, .)))
```

Now, all ones with an amount of NA values > 60% are dropped:
```

```{r, message=FALSE, warning=FALSE}
pc_na_threshold <- 60

sdf_gather_all <- function(tbl, gather_cols){

  gather_cols <- colnames(tbl)

  lapply(gather_cols, function(col_nm){
    tbl %>%
      select(c(col_nm)) %>%
      mutate(key = col_nm) %>%
      rename(value = col_nm)
  }) %>%
    sdf_bind_rows() %>%
    select(c('key', 'value'))
}

na_per_variable <- accidente_c3 %>%
  select_if(~is.character(.)) %>%
  sdf_gather_all() %>%
  group_by(key) %>%
  summarise(num_na = sum(if_else(is.na(value), 1, 0)),
    pc_na = sum(if_else(is.na(value), 1, 0)) / n() * 100) %>%
  rename(variable = key) %>%
  collect()

char_vars_to_delete <- na_per_variable %>%
  filter(pc_na > pc_na_threshold) %>%
  arrange(desc(num_na)) %>%
  select(variable) %>%
  pull()

accidente_c3 <- accidente_c3 %>%
  select(-char_vars_to_delete)
```

If we check glimpse again:

```{r}
```

```
glimpse(accidente_c3)
```

Checkpoint:

```{r message=FALSE, warning=FALSE, include=FALSE}
sdf_persist(accidente_c3)
spark_write_parquet(accidente_c3, path = "~/parquet/accidente_c3",
  mode = "overwrite")
db_drop_table(sc, "accidente")
```
```{r message=FALSE, warning=FALSE, include=FALSE}
accidente_c3 <- spark_read_parquet(sc, "accidente",
  "~/parquet/accidente_c3")
```

There are some character variables that in reality are numeric. Lets start with "HORASTRABAJOPREVIAS", a field that contains the number of hours that a worker has completed of its workday at the moment of the accident:

```{r, warning=FALSE}
accidente_c4 <- accidente_c3 %>%
  mutate(HORASTRABAJOPREVIAS = regexp_replace(HORASTRABAJOPREVIAS,
                                              ':', '')) %>%
  mutate(HORASTRABAJOPREVIAS =
   as.numeric(regexp_extract(HORASTRABAJOPREVIAS,
     '([0-9]{2})([0-9]{2})([0-9]{2})', 1)) +
   as.numeric(regexp_extract(HORASTRABAJOPREVIAS,
     '([0-9]{2})([0-9]{2})([0-9]{2})', 2)) / 60 +
   as.numeric(regexp_extract(HORASTRABAJOPREVIAS,
     '([0-9]{2})([0-9]{2})([0-9]{2})', 3)) / 3600)

ggplot(accidente_c4, aes(x = HORASTRABAJOPREVIAS)) +
  geom_histogram(binwidth = 1)
```

It seems that value 0 is imputed to missings. Checking the field "DESCRIPCION", where the accident is related, indicates that definitely this value is related to missings.

```r
accidente_c4 <- accidente_c4 %>%
  mutate(HORASTRABAJOPREVIAS = if_else(HORASTRABAJOPREVIAS == 0.0, NA,
    HORASTRABAJOPREVIAS))

ggplot(accidente_c4, aes(x = HORASTRABAJOPREVIAS)) +
  geom_histogram(binwidth = 1)
```

With table "SALARIO", that contains the salary of the accidented worker:

```r
accidente_c4 <- accidente_c4 %>%
  mutate(SALARIO = as.numeric(SALARIO))

accidente_c4 %>%
  select(SALARIO) %>%
  ggplot(aes(x = SALARIO)) +
    geom_histogram(bins = 60) +
    scale_x_continuous(trans = "log10")

accidente_c4 %>%
  select(SALARIO) %>%
  ggplot(aes(y = SALARIO)) +
    geom_boxplot() +
    scale_y_continuous(trans = "log10")

accidente_c4 <- accidente_c4 %>%
  mutate(SALARIO = if_else(SALARIO < 100000 | SALARIO > 10000000, NA,
    SALARIO))

accidente_c4 %>%
  select(SALARIO) %>%
  ggplot(aes(x = SALARIO)) +
    geom_histogram(bins = 60) +
    scale_x_continuous(trans = "log10")

accidente_c4 %>%
```

```
  select(SALARIO) %>%
  sdf_describe()
```

Other variables are dates in reality. Starting with FECHACREACION, we expect to see the creation date of the accident observation:

```{r}
accidente_c5 <- accidente_c4 %>%
  mutate(FECHACREACION = from_utc_timestamp(timestamp(unix_timestamp(
    FECHACREACION, 'yyyyMMddHHmmss')), 'UTC'))

accidente_c5 %>%
  ggplot(aes(x = FECHACREACION)) +
    geom_histogram(bins = 60)
```

Perhaps, this seems not to be a good reference about the real date when the accident occurred as the creations are too much concentrated in certain dates, indicating a possible massive traspassing of information. One expects to have a near uniform distribution. If we check the density of insertions by day we can affirm this hypothesis:

```{r}
accidente_c5 %>%
  select(FECHACREACION) %>%
  mutate(YEAR = year(FECHACREACION),
         MONTH = month(FECHACREACION),
         DAY = day(FECHACREACION)) %>%
  group_by(YEAR, MONTH, DAY) %>%
  summarise(n = n()) %>%
  ggplot(aes(x = n)) +
    geom_density() +
    xlab("Insertions of observations per day")

accidente_c5 <- accidente_c5 %>%
  select(-c(FECHACREACION, FECHAMODIFICACION))
```

Variables FECHA and HORA, from CON_CONTINGENCIA, have pressumably the date

```
and hour when the accident occurs. If we add them and convert it to
datetime we have:

```{r}
accidente_c5 <- accidente_c5 %>%
  mutate(MOMENTOACCIDENTE = from_utc_timestamp(timestamp(unix_timestamp(
    paste(FECHA, HORA, sep = ''), 'yyyyMMddHHmmss')), 'UTC')) %>%
  select(-c(FECHA, HORA))

accidente_c5 %>%
  ggplot(aes(x = MOMENTOACCIDENTE)) +
    geom_histogram(bins = 60)
```

As we can see, the new variable, MOMENTOACCIDENTE, has some outliers.
We can clean it if we know where the accidents started to be added:

```{r}
accidente_c5 %>%
  filter(MOMENTOACCIDENTE > '1995-01-01' &
    MOMENTOACCIDENTE < '2000-01-01') %>%
  ggplot(aes(x = MOMENTOACCIDENTE)) +
    geom_histogram(bins = 60) +
    scale_x_datetime(date_breaks = "6 months", date_labels = "%Y/%m")

accidente_c5 %>%
  filter(MOMENTOACCIDENTE > '1995-01-01' &
    MOMENTOACCIDENTE < '1996-01-01') %>%
  ggplot(aes(x = MOMENTOACCIDENTE)) +
    geom_histogram(bins = 60) +
    scale_x_datetime(date_breaks = "1 months", date_labels = "%m") +
    xlab("Months of 1995")
```

It seems that it occurs starting July 1995. If we assign the rest to NA:

```{r}
accidente_c5 <- accidente_c5 %>%
  mutate(MOMENTOACCIDENTE = if_else(MOMENTOACCIDENTE < '1995-07-01', NA,
```
```

```
    MOMENTOACCIDENTE))

accidente_c5 %>%
  ggplot(aes(x = MOMENTOACCIDENTE)) +
    geom_histogram(bins = 60) +
    scale_x_datetime(date_breaks = "5 years", date_labels = "%Y")
```

If we check the density of accidents we have some more realistic:

```{r}
accidente_c5 %>%
  select(MOMENTOACCIDENTE) %>%
  mutate(YEAR = year(MOMENTOACCIDENTE),
         MONTH = month(MOMENTOACCIDENTE),
         DAY = day(MOMENTOACCIDENTE)) %>%
  group_by(YEAR, MONTH, DAY) %>%
  summarise(n = n()) %>%
  ggplot(aes(x = n)) +
    geom_density() +
    xlab("Accidents per day")
```

Now, for FECCHAINGRESOEMPRESA, pressumibly the date when the worker started to work at the company, shows that the vast majority of observations are erroneous:

```{r}
accidente_c5 <- accidente_c5 %>%
  mutate(FECHAINGRESOEMPRESA = from_utc_timestamp(timestamp(unix_timestamp(
    FECHAINGRESOEMPRESA, 'yyyyMMdd')), 'UTC'))

accidente_c5 %>%
  ggplot(aes(x = FECHAINGRESOEMPRESA)) +
    geom_histogram(bins = 60)
```

This variable will not be used:

```
```{r}
accidente_c5 <- accidente_c5 %>%
  select(-FECHAINGRESOEMPRESA)
```
```

Checkpoint:

```
```{r message=FALSE, warning=FALSE, include=FALSE}
sdf_persist(accidente_c5)
spark_write_parquet(accidente_c5, path = "~/parquet/accidente_c5",
  mode = "overwrite")
db_drop_table(sc, "accidente")
```
```
```
```{r message=FALSE, warning=FALSE, include=FALSE}
accidente_c5 <- spark_read_parquet(sc, "accidente",
  "~/parquet/accidente_c5")
```
```

```
## Historical accident data

Now, historical data recorded at the moment of the accident at tables
CORE_HISTORICOTRABAJADOR, CORE_HISTORICOCENTROTRABAJO and
CORE_HISTORICOEMPRESA will be wrangled in order to be joined afterwards
with the join of CON_CONTINGENCIA and CON_ACCIDENTE.

### CORE_HISTORICOTRABAJADOR

Starting with the info of the worker affected at the accident, first we
have to load it to Spark:

```
```{r message=FALSE, warning=FALSE}
historicotrabajador <- spark_read_parquet(sc, "historicotrabajador",
  "~/parquet/core_historicotrabajador")
```
```

A first look of this table reveals the following information:

```
```{r}
sdf_nrow(historicotrabajador)
glimpse(historicotrabajador)
```
```
```

```

There are more than 1.6 million of observations but some of them have been deleted. This is known by the logical field ELIMINADO. Although this observations should be deleted, if we take it as deleted we have an issue at joining time with accidents table as we will have only 1500 observations.

```{r}
historicotrabajador %>%
  filter(!ELIMINADO) %>%
  inner_join(accidente_c5,
    by = c("ID" = "FK_CORE_HISTORICOTRABAJADOR")) %>%
  sdf_nrow()
```

The ELIMINADO field was designed in order to recover quickly an observation deleted by an application user, thus this is pressumably a problem produced by users. In order to overpass this problem I dont take into account this field as its already filtered at CON_CONTINGENCIA.

```{r}
historicotrabajador <- historicotrabajador %>%
  select(-c(ELIMINADO, FECHABORRADO))
```

The rest of variables, are majorly character variables but some of these are in reality categorical classes, numbers or dates. Them will be transformed.

Character variables

First of all, as with accident observations, character values "" will be assigned to NA:

```{r}
historicotrabajador_c1 <- historicotrabajador %>%
  mutate_if(is.character, list(~if_else(. == "", NA, .)))
```
```

Now, all ones with an amount of NA values > 60% are dropped:

```{r message=FALSE, warning=FALSE}
na_per_variable <- historicotrabajador_c1 %>%
  select_if(~is.character(.)) %>%
  sdf_gather_all() %>%
  group_by(key) %>%
  summarise(num_na = sum(if_else(is.na(value), 1, 0)),
            pc_na = sum(if_else(is.na(value), 1, 0)) / n() * 100) %>%
  rename(variable = key) %>%
  collect()

char_vars_to_delete <- na_per_variable %>%
  filter(pc_na > pc_na_threshold) %>%
  arrange(desc(num_na)) %>%
  select(variable) %>%
  pull()

historicotrabajador_c2 <- historicotrabajador_c1 %>%
  select(-char_vars_to_delete)
```

Checkpoint:

```{r message=FALSE, warning=FALSE, include=FALSE}
sdf_persist(historicotrabajador_c2)
spark_write_parquet(historicotrabajador_c2,
  path = "~/parquet/historicotrabajador_c2", mode = "overwrite")
db_drop_table(sc, "historicotrabajador")
```

```{r message=FALSE, warning=FALSE, include=FALSE}
historicotrabajador_c2 <- spark_read_parquet(sc, "historicotrabajador",
  "~/parquet/historicotrabajador_c2")
```

If we check now character variables:

```{r}
historicotrabajador_c2 %>%
  select_if(~is.character(.)) %>%
```

```
  glimpse()
```

And drop ones that have been anonymized:

```{r}
historicotrabajador_c3 <- historicotrabajador_c2 %>%
  select(-c(NOMBRE, APELLIDOS, IDENTIFICACION, DIRECCION, TELF1, FAX,
    EMAIL1, EMAIL2, EMPRESA, CENTROTRABAJO))
```

We have the following character variables:

```{r}
historicotrabajador_c3 %>%
  select_if(~is.character(.)) %>%
  glimpse()
```

Character to categorical

In Spark, we can not use categorical data types as factors in the R base, this categorical data is transformed into every Spark machine learning algorithm. One thing we can do is clean the classes of these categorical variables by checking their content. At the time of analysis, we will can transform these variables if necessary.

Some variables seems to be categorical, lets check how much levels would have:

```{r}
historicotrabajador_c3 %>%
  select(c(PROVINCIA, POBLACION, OCUPACION, CARGO, JORNADA, EPS, ARP, AFP,
    TIPOIDENTIFICACION, RANGOANTIGUEDAD, RANGOEDAD, ESTADOTRABAJADOR)) %>%
  sdf_gather_all() %>%
  group_by(key) %>%
  summarise(unique = n_distinct(value)) %>%
  collect() %>%
  arrange(unique)
```

```
```

We can check that all are categorical variables, not an arbitrary text value. If we check the distribution of the smallest:

```{r fig.height=5}
vars_to_factor <- c("PROVINCIA", "JORNADA", "ARP", "AFP",
  "TIPOIDENTIFICACION", "RANGOANTIGUEDAD", "RANGOEDAD", "ESTADOTRABAJADOR")

for (variable in vars_to_factor) {
  print(historicotrabajador_c3 %>%
    select(variable) %>%
    sdf_sample(fraction = 0.2, replacement = FALSE) %>%
    collect() %>%
    gather() %>%
    ggplot(aes(x = value, fill = value)) +
      geom_bar() +
      coord_flip() +
      scale_y_log10() +
      theme(legend.position = "none") +
      ggtitle(variable))
}
```

We can see the following:
* At RANGOEDAD, where worker is binned to a range of years old, there are some under-age workers. This is not an error as at Colombia, one can work having between 15 to 17 years old. This variable can be util if the information of the date field (to be transformed) FECHA_NACIMIENTO is not well inserted.
* At TIPOIDENTIFICACION, we can see that the major values are of class CEDULA DE CIUDADANIA, an identification for national residents adults. The same document but for foreigners is the CEDULA DE EXTRANJERIA. Also, there is the TARJETA DE IDENTIFICACION, an identification for national residents under-age. There are another minor classes, as NUMERO UNICO DE IDENTIFICACION, but in resum all could be classified between national or foreigners. This could be a new feature:

```{r}
```

```
historicotrabajador_c4 <- historicotrabajador_c3 %>%
  mutate(TIPOIDENTIFICACION = if_else(TIPOIDENTIFICACION == "NO INFORMA",
    NA, TIPOIDENTIFICACION))

identificaciones_autoctonas <- c("TARJETA DE IDENTIFICACION",
  "CEDULA DE CIUDADANIA", "NUMERO UNICO DE IDENTIFICACION")

historicotrabajador_c4 <- historicotrabajador_c4 %>%
  mutate(ESAUTOCTONO = if_else(!is.na(TIPOIDENTIFICACION),
    if_else(TIPOIDENTIFICACION %in% identificaciones_autoctonas,
      TRUE, FALSE),
    NA))

historicotrabajador_c4 %>%
  ggplot(aes(x = ESAUTOCTONO, fill = ESAUTOCTONO)) +
    geom_bar() +
    scale_y_log10()
```

```
* At JORNADA, there are some classes with an explanatory name, like Turnos,
Nocturna, Mixto and Diurna, kinds of workshifts. There is a special case,
Sin definir, that possibly has to be NA. Also, there are another five
classes, 0-4, that representation is unknown. We probably see them
at Database views.
* At PROVINCIA, the province of the worker (not of the company) is defined.
There is an error with two classes; VALLE really exists as Valle, and
ANTIOQUIA really exists as Antioquia. Also, there are two classes,
DESCONOCIDO and Por establecer, that must be NA.
```

```{r}
historicotrabajador_c5 <- historicotrabajador_c4 %>%
  mutate(PROVINCIA = if_else(PROVINCIA %in% c("DESCONOCIDO",
    "Por establecer"), NA, PROVINCIA)) %>%
  mutate(PROVINCIA = if_else(PROVINCIA == "VALLE",
    "Valle", PROVINCIA)) %>%
  mutate(PROVINCIA = if_else(PROVINCIA == "ANTIOQUIA",
    "Antioquia", PROVINCIA))
```

```
* At RANGOANTIGUEDAD, it seems that almost all workers realted in accident
have 5 or less years at their companies, the rest of ranges only sum
approximately 30 accidents. Maybe this is due to a lack of data.
* At ARP, shows from which Administrator of Profesional Risks comes the
register of accident, but it comes only from COLMENA SEGUROS. This variable
can be deleted. The same applies to the AFP, the Administrator of Pension
and Severance Funds.

```{r}
historicotrabajador_c6 <- historicotrabajador_c5 %>%
  select(-c(ARP, AFP))
```

* At ESTADOTRABAJADOR, the content of this variable should be consulted
as range of years of these workers don't correspond with a retired worker:

```{r}
historicotrabajador_c6 %>%
  filter(ESTADOTRABAJADOR == "Retirado") %>%
  ggplot(aes(x = RANGOEDAD, fill = RANGOEDAD)) +
    geom_bar()
```

Also, exists logical variable RETIRADO and date variable FECHARETIRADO
(deleted due to high number of missings).

If we check the rest of categorical variables, those that have a big amount
of classes, starting with POBLACION we can check if there are errors with
the classes names. For it, first we select all distinct classes of
POBLACION, collect them and create a new variable, POBLACION_simplified,
that is the class value uppercased and without any symbol. We use collect
here in order to use advanced string functions as str_replace_all.

```{r}
poblacion_and_poblacion_simple <- historicotrabajador_c6 %>%
  select(POBLACION) %>%
  distinct(POBLACION) %>%
  collect() %>%
  mutate(POBLACION_simplified = str_replace_all(
```
```

```
    iconv(toupper(POBLACION), to="ASCII//TRANSLIT"), "[^[:alnum:]]", ""))

poblacion_and_poblacion_simple %>%
  filter(POBLACION_simplified %in% (poblacion_and_poblacion_simple %>%
                                      group_by(POBLACION_simplified) %>%
                                      summarise(n = n()) %>%
                                      filter(n > 1) %>%
                                      pull(POBLACION_simplified))) %>%
  arrange(POBLACION)
```

As we can see, there are some bad entered classes names. To solve it, we have to copy the new R dataframe created with the class and its simplification to Spark and then use a left join of historicotrabajador_c6 with it to finally change the class name with the simplification. We cannot use join operations with different sources (R and Spark).

```{r}
paps_tbl <- sdf_copy_to(sc, poblacion_and_poblacion_simple)

historicotrabajador_c7 <- historicotrabajador_c6 %>%
  left_join(paps_tbl, by = "POBLACION") %>%
  mutate(POBLACION = POBLACION_simplified) %>%
  select(-POBLACION_simplified)

historicotrabajador_c7 %>%
  group_by(POBLACION) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  ggplot(aes(x = POBLACION, y = n, fill = POBLACION)) +
    geom_col() +
    theme(axis.text.x = element_text(angle = 90), legend.position = "none")
```

Also, there are some classes that are in reality missings:

```{r}
historicotrabajador_c7 <- historicotrabajador_c7 %>%
```

```
  mutate(POBLACION = if_else(POBLACION %in% c("CIUDADDESCONOCIDA",
    "NOSUMINISTRADAA", "PORESTABLECER"),
                               NA,
                               POBLACION))
```

We can do the same with OCUPACION:

```{r}
ocupacion_and_ocupacion_simple <- historicotrabajador_c7 %>%
  select(OCUPACION) %>%
  distinct(OCUPACION) %>%
  collect() %>%
  mutate(OCUPACION_simplified = str_replace_all(iconv(toupper(OCUPACION),
    to="ASCII//TRANSLIT"), "[^[:alnum:]]", ""))

ocupacion_and_ocupacion_simple %>%
  filter(OCUPACION_simplified %in% (ocupacion_and_ocupacion_simple %>%
                                      group_by(OCUPACION_simplified) %>%
                                      summarise(n = n()) %>%
                                      filter(n > 1) %>%
                                      pull(OCUPACION_simplified))) %>%
  arrange(OCUPACION)
```

As it's only one record, we maintain the original title:

```{r}
oaos_tbl <- sdf_copy_to(sc, ocupacion_and_ocupacion_simple)

ocupacion_to_change <-
  "COORDINADORESYSUPERVISORESDEPRODUCCION
  YOPERACIONESENINSTALACIONMANTENIMIENTOYREPARACION"
ocupacion_replacement <- "Coordinadores y supervisores de produccion
y operaciones en instalacion, mantenimiento y reparacion"

historicotrabajador_c8 <- historicotrabajador_c7 %>%
  left_join(oaos_tbl, by = "OCUPACION") %>%
  mutate(OCUPACION = if_else(OCUPACION_simplified == ocupacion_to_change,
```

```
    ocupacion_replacement, OCUPACION)) %>%
  select(-OCUPACION_simplified)
```

Nevertheless, this variable have a big amount of missings and some relevant classes are not descriptive:

```{r}
historicotrabajador_c8 %>%
  group_by(OCUPACION) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10)
```

Now, with CARGO we have:

```{r}
cargo_and_cargo_simple <- historicotrabajador_c8 %>%
  select(CARGO) %>%
  distinct(CARGO) %>%
  collect() %>%
  mutate(CARGO_simplified = str_replace_all(iconv(toupper(CARGO),
    to="ASCII//TRANSLIT"), "[^[:alnum:]]", ""))

cargo_and_cargo_simple %>%
  filter(CARGO_simplified %in% (cargo_and_cargo_simple %>%
                                  group_by(CARGO_simplified) %>%
                                  summarise(n = n()) %>%
                                  filter(n > 1) %>%
                                  pull(CARGO_simplified))) %>%
  arrange(CARGO)
```

There are a lot of errors, we can clean it a little bit:

```{r}
cacs_tbl <- sdf_copy_to(sc, cargo_and_cargo_simple)
```

```
historicotrabajador_c9 <- historicotrabajador_c8 %>%
  left_join(cacs_tbl, by = "CARGO") %>%
  mutate(CARGO = CARGO_simplified) %>%
  select(-CARGO_simplified)
```

This is the same case as before, if both variables worth it we will need to spend hours cleaning them. One thing that we could do at analysis is to take only the most important classes.

Finally, we have EPS, a case identical to ARP and AFP. We can delete it:

```{r}
historicotrabajador_c9 <- historicotrabajador_c9 %>%
  select(-EPS)
```

Checkpoint

```{r message=FALSE, warning=FALSE, include=FALSE}
sdf_persist(historicotrabajador_c9)
spark_write_parquet(historicotrabajador_c9,
  path = "~/parquet/historicotrabajador_c9", mode = "overwrite")
db_drop_table(sc, "historicotrabajador")
```

```{r message=FALSE, warning=FALSE, include=FALSE}
historicotrabajador_c9 <- spark_read_parquet(sc, "historicotrabajador",
  "~/parquet/historicotrabajador_c9")
```

Character to date

There are other variables that are dates, these are FECHA_NACIMIENTO and FECHAINGRESOEMPRESA:

Logical variables

Numerical variables

CORE_HISTORICOCENTROTRABAJO

Now, with the info of the work center were the accident occured, first we have to load it to Spark:

```{r message=FALSE, warning=FALSE}
historicocentrotrabajo <- spark_read_parquet(sc, "historicocentrotrabajo",
  "~/parquet/core_historicocentrotrabajo")
```

A first look of this table reveals the following information:

```{r}
sdf_nrow(historicocentrotrabajo)
glimpse(historicocentrotrabajo)
```

CORE_HISTORICOEMPRESA

Now, with the info of the company were the accident occured, first we have to load it to Spark:

```{r message=FALSE, warning=FALSE}
historicoempresa <- spark_read_parquet(sc, "historicoempresa",
  "~/parquet/core_historicoempresa")
```

A first look of this table reveals the following information:

```{r}
sdf_nrow(historicoempresa)
glimpse(historicoempresa)
```

Desconexion

```{r desconexion_spark, message=FALSE, warning=FALSE}
spark_disconnect(sc)
```

Appendix B

Data modeling and evaluation RMarkdown script

```
---
title: "COLMENA SEGUROS"
author: "Alberto Burgos"
date: "27/4/2019"
output:
  html_document: default
  pdf_document: default
---

```{r include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

Environment setup

Libraries and global options

Load of libraries and setup of global options:

```{r message=TRUE, warning=TRUE}
library(dplyr)
library(h2o)
library(rsparkling)
library(sparklyr)
```
```

```
library(ggplot2)
library(gridExtra)
library(ggthemr)
library(tidyverse)

options(digits = 2, scipen = 999)
ggthemr('light')
set_swatch(c("#785d37", "#62bba5", "#ffb84d", "#aaa488", "#b2432f",
  "#3a6589", "#9b5672", "#908150", "#373634", "#cd5c5c",
  "#3cb371", "#b0c4de", "#E69F00", "#56B4E9", "#009E73",
  "#F0E442", "#0072B2", "#D55E00", "#CC79A7"))
```

```
## Spark connection

Configuration and start Spark standalone connection:

```{r message=FALSE, warning=FALSE}
config <- spark_config()
config$`sparklyr.shell.driver-memory` <- "8G"
config$`sparklyr.shell.executor-memory` <- "6G"
config$`spark.yarn.executor.memoryOverhead` <- "512"
sc <- spark_connect(master = "local", config = config)
```

Variables and auxiliary functions

Definition of global variables and auxiliary functions:

```{r}
# Maximum tolerable percentage of missing values
pc_na_threshold <- 60

# This function emulates dplyr's gather, converting all
# columns of a tibble to key-value pairs
sdf_gather_all <- function(tbl) {
  gather_cols <- colnames(tbl)
  lapply(gather_cols, function(col_nm){
    tbl %>%
```
```

```
      select(c(col_nm)) %>%
      mutate(key = col_nm) %>%
      rename(value = col_nm)
  }) %>%
    sdf_bind_rows() %>%
    select(c('key', 'value'))
}
```

```
# Analysis

## Question 1

"**What is the relationship between the evaluated risk levels and the
accidentability of a company?**"

This first question needs to be answered doing an exploratory analysis
and comparing these relationships.

### Evaluations

#### Explorations

In order to start with it, we need to get on one hand the evaluated risks
by company. To do this, we have to load the risk evaluations and
ubications wrangled at ETL process:

```{r message=FALSE, warning=FALSE}
evalriesgopresente <- spark_read_parquet(sc, "evalriesgopresente",
  "~/parquet/evalriesgopresente_c12")
coreubicacion <- spark_read_parquet(sc, "coreubicacion",
  "~/parquet/coreubicacion_c7")
```

Now, we can join them by **FK_CORE_UBICACION** in order to have the risk
evaluation related to the company:

```{r}
evaluated_risk_company <- evalriesgopresente %>%
```
```

```
  left_join(coreubicacion %>% select(ID, FK_CORE_EMPRESA),
            by = c("FK_CORE_UBICACION" = "ID"))

sdf_describe(evaluated_risk_company)
```

There are about 145 observations without **FK_CORE_EMPRESA** value. We have to drop them and also the fields **ID** and **FK_CORE_UBICACION** in order to get a cleaner dataset:

```{r}
evaluated_risk_company_c1 <- evaluated_risk_company %>%
  filter(!is.na(FK_CORE_EMPRESA)) %>%
  select(-c(ID, FK_CORE_UBICACION))
```

Also, there are some observations without a risk or a risk factor. We dont drop them but we need to take this into account for the analysis.

Used methodologies and levels

If we check the use of methodologies, we can see that almost only **GTC45** is used:

```{r}
evaluated_risk_company_c1 %>%
  ggplot(aes(x = METODOLOGIA_UTILIZADA, fill = METODOLOGIA_UTILIZADA)) +
    geom_bar() +
    theme(legend.position = "none") +
    ggtitle("Distribution of methodologies")
```

And if we check the distribution of levels assigned to the evalutations using this methodology:

```{r}
evaluated_risk_company_c1 %>%
  filter(METODOLOGIA_UTILIZADA == "GTC45") %>%
  ggplot(aes(x = reorder(NIVEL_EVALUADO, NIVEL_EVALUADO,
```

```
    function(x) length(x)), fill = NIVEL_EVALUADO)) +
    geom_bar() +
    coord_flip() +
    theme(axis.title.y = element_blank(),
          legend.position = "none") +
    ggtitle("GTC45 evaluations distribution")
```

We can see that almost a half part of the evaluations corresponds to risks that must be solved as soon as possible.

Now, if we check the years were evaluations occured we can see that these goes from 2015 to 2018:

```{r}
evaluated_risk_company_c1 %>%
  mutate(YEAR = year(FECHA_CREACION)) %>%
  select(YEAR) %>%
  sdf_describe()
```

Also, if we check their distribution:

```{r}
evaluated_risk_company_c1 %>%
  ggplot(aes(x = FECHA_CREACION)) +
    geom_histogram(aes(y = ..density..),
                   color = "lightgreen",
                   bins = 240) +
    geom_density() +
    xlab("") +
    ggtitle("Evaluations distribution from 2015")
```

We can see that most of the evaluations were carried out from the end of 2017 until the end of 2018.

If we take the **GTC45** methodology and verify the distribution of its levels:

```
```{r fig.width=10}
evaluated_risk_company_c1 %>%
  filter(METODOLOGIA_UTILIZADA == "GTC45") %>%
  ggplot(aes(x = FECHA_CREACION)) +
    geom_histogram(color = "lightgreen",
                   bins = 240) +
    ggtitle("GTC45 evaluations by level") +
    facet_wrap(~ NIVEL_EVALUADO)
```

We can see that the most relevant changes occur at the **"No Aceptable"**
level. This level indicates that the urgency is maximum, so there may
be a potential accident if the risk is not resolved. In addition, it
does not leave a margin of time for its resolution.

Risks

In order to take more information about the types of risks, we can see
that there are 37 kinds of them:

```
```{r}
evaluated_risk_company_c1 %>%
  distinct(RIESGO) %>%
  sdf_describe()
```

And the distribution of the most common of them:

```
```{r message=FALSE}
evaluated_risk_company_c1 %>%
  group_by(RIESGO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  ggplot(aes(x = reorder(RIESGO, n), y = n, fill = RIESGO)) +
    geom_col() +
    coord_flip() +
    theme(axis.title.y = element_blank(),
```
```

```
          legend.position = "none") +
    ggtitle("Risks distribution")
```

This shows that most of the risks are indeterminate and the most representative classes are accidents caused by human activity and falls, either at different levels or at the same level. If we check the relationships between the indeterminate class and the factors of risk we can see that a major part of them continues undetermined:

```{r message=FALSE}
evaluated_risk_company_c1 %>%
  filter(is.na(RIESGO)) %>%
  group_by(FACTORRIESGO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  ggplot(aes(x = reorder(FACTORRIESGO, n), y = n, fill = FACTORRIESGO)) +
    geom_col() +
    coord_flip() +
    theme(axis.title.y = element_blank(),
          legend.position = "none") +
    ggtitle("Factor of risks pertaining to NA risks")
```

Also, if we do the same again but with the dangers we finally have that this risk is mainly related to public dangers and earthquakes. Public dangers are those that evereyone can suffer like theft, kidnapping, attack, sabotage, traffic accident, etc:

```{r message=FALSE}
evaluated_risk_company_c1 %>%
  filter(is.na(RIESGO)) %>%
  group_by(PELIGRO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  ggplot(aes(x = reorder(PELIGRO, n), y = n, fill = PELIGRO)) +
```

```
    geom_col() +
    coord_flip() +
    theme(axis.title.y = element_blank(),
          legend.position = "none") +
    ggtitle("Dangers pertaining to NA risks")
```

Now, if we check the distribution of the main risks over all years:

```{r fig.height=8, fig.width=10, message=FALSE}
top10_RIESGO <- evaluated_risk_company_c1 %>%
  group_by(RIESGO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  pull(RIESGO)

evaluated_risk_company_c1 %>%
  filter(RIESGO %in% top10_RIESGO) %>%
  ggplot(aes(x = FECHA_CREACION)) +
    geom_histogram(color = "lightgreen",
                   bins = 240) +
    facet_wrap(~ RIESGO, ncol = 1)
```

We see that major part of evaluations of top risks were made between 2017 and 2018.

Factors of risk

Now, if we check the factors of risk, we know that there are 96 factors:

```{r}
evaluated_risk_company_c1 %>%
  filter(!is.na(FACTORRIESGO)) %>%
  distinct(FACTORRIESGO) %>%
  sdf_describe()
```

And the distribution of the most common of them:

```{r message=FALSE}
evaluated_risk_company_c1 %>%
  group_by(FACTORRIESGO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  ggplot(aes(x = reorder(FACTORRIESGO, n), y = n, fill = FACTORRIESGO)) +
    geom_col() +
    coord_flip() +
    theme(axis.title.y = element_blank(),
          legend.position = "none") +
    ggtitle("Factors of risk distribution")
```

Again we see that most are indeterminate. Taking into account the rest of the classes, the most representative are those related to order and cleanliness, and with sliding or irregular work surfaces.

Now, if we check the distribution of the main factors of risk over all years:

```{r fig.height=8, fig.width=10, message=FALSE}
top10_FACTORRIESGO <- evaluated_risk_company_c1 %>%
  group_by(FACTORRIESGO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  pull(FACTORRIESGO)

evaluated_risk_company_c1 %>%
  filter(FACTORRIESGO %in% top10_FACTORRIESGO) %>%
  ggplot(aes(x = FECHA_CREACION)) +
    geom_histogram(color = "lightgreen",
                   bins = 240) +
    facet_wrap(~ FACTORRIESGO, ncol = 1)
```

Again, major part of evaluations of top risk factors were made between 2017 and 2018.

Dangers

Going on deeper with dangers, we know that there are 45 kinds of them:

```{r}
evaluated_risk_company_c1 %>%
  distinct(PELIGRO) %>%
  sdf_describe()
```

And the distribution of the most common of them:

```{r message=FALSE}
evaluated_risk_company_c1 %>%
  group_by(PELIGRO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  ggplot(aes(x = reorder(PELIGRO, n), y = n, fill = PELIGRO)) +
    geom_col() +
    coord_flip() +
    theme(axis.title.y = element_blank(),
          legend.position = "none") +
    ggtitle("Dangers distribution")
```

Where we can see that the most common danger is related to the place where the work is produced, ie the geographical or work areas. This risk is important because it is a permanent condition throughout the workday. In addition, mechanical, public or earthquake-related risks are also relevant. If we see those factors related to public danger:

```{r message=FALSE}
evaluated_risk_company_c1 %>%
  filter(PELIGRO == "Publicos") %>%
  group_by(FACTORRIESGO) %>%
```

```
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  ggplot(aes(x = reorder(FACTORRIESGO, n), y = n, fill = FACTORRIESGO)) +
    geom_col() +
    coord_flip() +
    theme(axis.title.y = element_blank(),
          legend.position = "none") +
    ggtitle("Factors of risk related to public dangers")
```

We see that this corresponds with the explanation done before.

Now, if we check the distribution of the main dangers over all years:

```{r fig.height=8, fig.width=10, message=FALSE}
top10_PELIGRO <- evaluated_risk_company_c1 %>%
  group_by(PELIGRO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  pull(PELIGRO)

evaluated_risk_company_c1 %>%
  filter(PELIGRO %in% top10_PELIGRO) %>%
  ggplot(aes(x = FECHA_CREACION)) +
    geom_histogram(color = "lightgreen",
                   bins = 240) +
    facet_wrap(~ PELIGRO, ncol = 1)
```

Again, we see the same behavior.

Relationships

We can see the relationships between the evaluated levels and the dangers:

```{r fig.height=9, fig.width=10, message=FALSE}
evaluated_risk_company_c1 %>%
```

```
  filter(METODOLOGIA_UTILIZADA == "GTC45") %>%
  group_by(NIVEL_EVALUADO, PELIGRO) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  top_n(10) %>%
  ggplot(aes(x = PELIGRO, y = n, fill = PELIGRO)) +
    geom_col() +
    coord_flip() +
    facet_wrap(~ NIVEL_EVALUADO, ncol = 1)
```

And check that, on all levels locative, mechanical and public dangers are present as most important dangers. Also, on non acceptable risk level, earthquakes and interfaces person-task are destacable.

Accidents

Explorations

By the other hand, we have to load the accidents and their historical data about company in order to relate the accident with a company:

```{r message=FALSE, warning=FALSE}
historicoempresa <- spark_read_parquet(sc, "historicoempresa",
  "~/parquet/historicoempresa_c5")
accidente <- spark_read_parquet(sc, "accidente",
  "~/parquet/accidente_c17")
```

After joining them, it seems that all accident have a company id:

```{r}
accident_company <- accidente %>%
  left_join(historicoempresa %>% select(ID, FK_CORE_EMPRESA, CNAE),
            by = c("FK_CORE_HISTORICOEMPRESA" = "ID"))

sdf_describe(accident_company %>% select(ID, FK_CORE_EMPRESA, CNAE))
```

It is important to know that, for each accident, a present risk factor can be added, if it exists. However, there are only about 400 observations with this condition. We do not know if it is because the risk factor had not been evaluated once the accident occurred or due to human error when entering the data. This can be seen with variable **FK_IDENT_FACTORRIESGOPRESENTE**:

```{r message=TRUE, warning=FALSE, include=TRUE}
accidente_orig <- spark_read_parquet(sc, "accidente_orig",
  "~/parquet/accidente")
sdf_describe(accidente_orig %>% select(ID,
  FK_IDENT_FACTORRIESGOPRESENTE))
```

Now, if we check the years were accidents occured we can see that these goes from 1995 to 2018:

```{r}
accident_company %>%
  mutate(YEAR = year(MOMENTOACCIDENTE)) %>%
  select(YEAR) %>%
  sdf_describe()
```

To be able to compare the accident rate with risk assessments, we need to do it by date. Unfortunately, we do not have the date on which a risk is resolved, therefore, we need to establish a time window. If we look at the levels, in those that have a certain urgency a window of time is fixed for its solution in the attribute DIAS_PLANIFICACION. In this case, focusing on the GTC45 methodology, two of its levels have this time window, 0 days for non-acceptable risks and 365 days for those risks that are not acceptable or acceptable with controls. Taking this into account, we should expect a reduction of accidents in a year just after evaluations.

As the evaluations occurred between 2015 and 2019, we must stay between these years to see if there is any change caused by the evaluations. If we check the distribution of accidents on these years:

```{r}
g1 <- accident_company %>%
  filter(MOMENTOACCIDENTE >= '2015') %>%
  ggplot(aes(x = MOMENTOACCIDENTE)) +
    geom_histogram(aes(y = ..density..),
                   color = "lightgreen",
                   bins = 600) +
    geom_density() +
    xlab("") +
    ggtitle("Accidents")

g2 <- evaluated_risk_company_c1 %>%
  filter(NIVEL_EVALUADO == "No Aceptable") %>%
  ggplot(aes(x = FECHA_CREACION)) +
    geom_histogram(aes(y = ..density..),
                   color = "lightgreen",
                   bins = 600) +
    geom_density() +
    xlab("") +
    ggtitle("Non acceptable level evaluations")

grid.arrange(g1, g2, nrow = 2)
```

We can see a pattern that is repeated every year, where there is a clear reduction in accidents at the end of the year and another reduction, although more subtle, on Easter. We can also see that, since 2015 there has been an upward trend in the number of accidents, while since 2018 these have been drastically reduced. If we compare it with non acceptable evaluations we can think that may exist a relationship between the evaluations made at 2017 and the following reduction of accidents.

Also, we can check the trend from 2010:

```{r}
accident_company %>%
  filter(MOMENTOACCIDENTE >= '2010') %>%
  ggplot(aes(x = MOMENTOACCIDENTE)) +
    geom_histogram(aes(y = ..density..),
```

```
                    color = "lightgreen",
                    bins = 600) +
    geom_density() +
    xlab("") +
    ggtitle("Accidents")
```

We see that the trend is to increase from 2010, changing dramatically from 2014 to 2016, and thereafter obtaining a more leisurely increase until 2017. This increase may be due to an increase in companies under control.

Relationships

If we obtain all non acceptable evaluations at main dangers and then compare these numbers against the number of accidents of companies present at these evaluations:

```{r}
accident_company_year_20152018 <- accident_company %>%
  filter(MOMENTOACCIDENTE >= "2015") %>%
  group_by(FK_CORE_EMPRESA, YEAR = year(MOMENTOACCIDENTE)) %>%
  summarise(n_accidentes = n())

non_acceptable_evaluations <- evaluated_risk_company_c1 %>%
  filter(METODOLOGIA_UTILIZADA == "GTC45" &
           NIVEL_EVALUADO == "No Aceptable" &
           PELIGRO %in% c("Terremoto", "Locativo",
             "Interface persona - tarea", "Publicos", "Mecanico")) %>%
  group_by(FK_CORE_EMPRESA, PELIGRO, YEAR = year(FECHA_CREACION)) %>%
  summarise(n_evaluaciones = n())

companies_non_acceptable_evaluations <- non_acceptable_evaluations %>%
  select(FK_CORE_EMPRESA) %>%
  distinct() %>%
  pull(FK_CORE_EMPRESA)

accident_company_year_20152018 %>%
  filter(FK_CORE_EMPRESA %in% companies_non_acceptable_evaluations) %>%
```

```
  ungroup() %>%
  select(-FK_CORE_EMPRESA) %>%
  collect() %>%
  group_by(YEAR) %>%
  summarise(n_accidentes = sum(n_accidentes)) %>%
  ggplot() +
    geom_col(aes(x = YEAR, y = n_accidentes), fill = "lightgray") +
    geom_col(data = non_acceptable_evaluations,
             aes(x = YEAR, y = n_evaluaciones, fill = PELIGRO),
             position = "dodge",
             alpha = 0.8) +
    ggtitle("Accidents vs non acceptable evaluations")
```

We can, again, see that may exist an inverse relationship between the number of evaluations made and the consequent number of accidents happened afterwards.

```
## Question 2
"**How much accidents will have a company in a year depending of its
size, the ubication of its centers and its activity?**"

### Accidents

#### Explorations
```

Starting to answer the question, we have to start by loading the historical data of the company that is registered when an accident occurs:

```
```{r message=FALSE, warning=FALSE}
historicoempresa <- spark_read_parquet(sc, "historicoempresa",
  "~/parquet/historicoempresa_c5")

sdf_describe(historicoempresa)
```
```

As we can see, the information related to the activity of the company, CNAE field, is available in all the registers, while the information related to the location, the population and the province, is also available

in almost all. Unfortunately we do not have the size of the company at the time of the accident.

Now we have to relate this information with the accident. In order to do it, first we have to load registered accidents:

```{r}
accidente <- spark_read_parquet(sc, "accidente", "~/parquet/accidente_c17")
```

After joining them, it seems that all accidents have a company id:

```{r}
accident_company <- accidente %>%
  left_join(select(historicoempresa, ID, POBLACION, PROVINCIA, F
    K_CORE_EMPRESA, CNAE),
            by = c("FK_CORE_HISTORICOEMPRESA" = "ID"))

sdf_describe(accident_company %>% select(ID, FK_CORE_EMPRESA))
```

Also, there are some accidents without the moment with it ocurred:

```{r}
accident_company_c1 <- accident_company %>%
  filter(!is.na(MOMENTOACCIDENTE))
```

Now, its time to see distribution of the number of accidents by a company in a year:

```{r}
num_accidents_company_year <- accident_company_c2 %>%
  group_by(FK_CORE_EMPRESA, YEAR = year(MOMENTOACCIDENTE)) %>%
  summarise(n = n())

num_accidents_company_year %>%
  ggplot(aes(y = n)) +
    geom_boxplot(outlier.colour = "indianred1") +
```

```
    coord_flip() +
    ggtitle("Accidents per company and year")
```

This clearly show that there are some outliers. We will study them first in a coarse grain manner. First, zomming in the distribution:

```{r}
g1 <- num_accidents_company_year %>%
  filter(n <= 500) %>%
  ggplot(aes(y = n)) +
    geom_boxplot(outlier.colour = "indianred1") +
    coord_flip() +
    ggtitle("Accidents per company and year. Range 0 to 500.")

g2 <- num_accidents_company_year %>%
  filter(n <= 50) %>%
  ggplot(aes(y = n)) +
    geom_boxplot(outlier.colour = "indianred1") +
    coord_flip() +
    ggtitle("Accidents per company and year. Range 0 to 50.")

grid.arrange(g1, g2, nrow = 2)
```

We can see that almost all values goes from 0 to 8 with a low median. This can be the difference between a small, medium or large company. Maybe a separation of these different populations should be the best way to do the study, unfortunately we dont have the historical value of the number of workers of a company but we can use the current value in an approximate way:

```{r}
empresa <- spark_read_parquet(sc, "empresa", "~/parquet/core_empresa")

accident_company_c3 <- accident_company_c2 %>%
  left_join(select(empresa, ID, NUMTRABAJADORES) %>% distinct(),
            by = c("FK_CORE_EMPRESA" = "ID"))
```

```
sdf_describe(accident_company_c3 %>% select(NUMTRABAJADORES))
```

It seems that some companies have 0 as number of workers. As this should be considered as NA, we need to drop these observations. Also, we can create a new attribute based on the Colombian definition of a company by it size:

```{r fig.height=6, warning=FALSE}
accident_company_c4 <- accident_company_c3 %>%
  filter(NUMTRABAJADORES != 0) %>%
  mutate(size = case_when(
    NUMTRABAJADORES <= 10 ~ "micro",
    NUMTRABAJADORES > 10 & NUMTRABAJADORES <= 50 ~ "small",
    NUMTRABAJADORES > 50 & NUMTRABAJADORES <= 200 ~ "medium",
    NUMTRABAJADORES > 200 ~ "big"))

g1 <- accident_company_c4 %>%
  group_by(size) %>%
  summarise(n_total = n()) %>%
  ggplot(aes(x = reorder(size, n_total, FUN = max), y = n_total,
    fill = size)) +
    geom_col() +
    ggtitle("Total accidents by company size") +
    xlab("Company size") +
    ylab("Total accidents")

g2 <- accident_company_c4 %>%
  select(FK_CORE_EMPRESA, size, NUMTRABAJADORES) %>%
  distinct() %>%
  ggplot(aes(x = NUMTRABAJADORES)) +
    geom_density() +
    ggtitle("# Workers by company size") +
    xlab("# Workers") +
    facet_wrap(~size, scales = "free", ncol = 2)

grid.arrange(g1, g2, nrow = 2)
```

It seems that almost all accidents have occurred at big companies. This is the expected. Also, we can see that big companies are concentrated below the amount of 1000 workers, medium and small size companies number decrease as their number of workers increases, and the micro companies size is near uniformly distributed.

Now, if we check the distributions based on the company size:

```
```{r fig.height=6}
num_accidents_company_year <- accident_company_c4 %>%
  group_by(FK_CORE_EMPRESA, YEAR = year(MOMENTOACCIDENTE)) %>%
  summarise(n = n()) %>%
  left_join(select(accident_company_c4, FK_CORE_EMPRESA, size) %>%
    distinct(),
            by = "FK_CORE_EMPRESA")

g1 <- num_accidents_company_year %>%
  ggplot(aes(y = n)) +
    geom_boxplot(outlier.colour = "indianred1") +
    xlab("Accidents by year") +
    coord_flip() +
    ggtitle("Accidents per company and year (per size)") +
    facet_wrap(~size, scales = "free", ncol = 2)

g2 <- num_accidents_company_year %>%
  ggplot(aes(x = n)) +
    geom_density() +
    xlab("Accidents by year") +
    ggtitle("Accidents per company and year (per size)") +
    facet_wrap(~size, scales = "free", ncol = 2)

grid.arrange(g1, g2, nrow = 2)
```
```

We can see that the distribution is a bit more clear, also there are outliers that can be seen in the plot. As these distributions are so skewed we can transform its values to log10 and get a normalized distribution:

```{r fig.height=6}
num_accidents_company_year <- num_accidents_company_year %>%
  mutate(n_log10 = log10(n))

g1 <- num_accidents_company_year %>%
  ggplot(aes(y = n_log10)) +
    geom_boxplot(outlier.colour = "indianred1") +
    xlab("log10(accidents by year)") +
    coord_flip() +
    ggtitle("Accidents per company and year (per size)") +
    facet_wrap(~size, scales = "free", ncol = 2)

g2 <- num_accidents_company_year %>%
  ggplot(aes(x = n_log10)) +
    geom_density() +
    xlab("log10(accidents by year)") +
    ggtitle("Accidents per company and year (per size)") +
    facet_wrap(~size, scales = "free", ncol = 2)

grid.arrange(g1, g2, nrow = 2)
```

We obtain a more normal distributions for each size. Also, we can check the centrality and variability measures for each size:

```{r}
num_accidents_company_year_m <- num_accidents_company_year %>% collect()

(measures_per_size <- num_accidents_company_year_m %>%
  group_by(size) %>%
  summarise(mean = mean(n_log10),
            median = median(n_log10),
            diff_mean_median = mean - median,
            mad = median(abs(n_log10 - mean(n_log10))),
            skewness = e1071::skewness(n_log10),
            sd = sd(n_log10),
            kurtosis = e1071::kurtosis(n_log10, type = 1)))
```

Which shows that there is a relationship between the size of the company and the number of accidents by year, except for the cases of small and micro companies. These two sizes have almost the same distribution of values. This can be explained due to the fact that a company can grow easier from micro to small than small to medium or from medium to big, thus the approximation used about the number of workers not accurate in this cases. Also, these two distributions are more skewed, probably by a bigger number of outliers.

Now, we can remove some outliers taking into account that the data is distributed following a normal-like shape. In order to do it, we can calculate the absolute value of the difference between an observation and the median of accidents in a year by size, and dividing it by the median of the absolute deviation. Then we can discard those observations with a value over or equal a threshold:

```{r}
nrows <- nrow(num_accidents_company_year_m)
initial_nrows <- nrow(num_accidents_company_year_m)
threshold <- 7.5
accident_company_c5 <- accident_company_c4

repeat {
  outliers <- num_accidents_company_year_m %>%
    left_join(select(measures_per_size, size, median, mad, kurtosis),
              by = "size") %>%
    filter(abs(n_log10 - median)/mad >= threshold) %>%
    pull(FK_CORE_EMPRESA)

  accident_company_c5 <- accident_company_c5 %>%
    filter(!(FK_CORE_EMPRESA %in% outliers))

  num_accidents_company_year_m <- num_accidents_company_year_m %>%
    filter(!(FK_CORE_EMPRESA %in% outliers))

  measures_per_size <- num_accidents_company_year_m %>%
    group_by(size) %>%
    summarise(mean = mean(n_log10),
              median = median(n_log10),
```

```
              diff_mean_median = mean - median,
              mad = median(abs(n_log10 - mean(n_log10))),
              skewness = e1071::skewness(n_log10),
              sd = sd(n_log10),
              kurtosis = e1071::kurtosis(n_log10, type = 1))

  if (nrows == nrow(num_accidents_company_year_m)) {
    break
  }

  nrows <- nrow(num_accidents_company_year_m)
}

(sdf_nrow(accident_company_c4) - sdf_nrow(accident_company_c5)) /
sdf_nrow(accident_company_c4) * 100
(initial_nrows - nrow(num_accidents_company_year_m)) /
initial_nrows * 100

measures_per_size
```

Using a threshold of 7.5 we have removed the 0.4% of observations, considered as outliers. All deletions have been done at micro and small sizes, reducing the skewness and the kurtosis of the distributions. We can also check it graphically:

```
```{r fig.height=6}
num_accidents_company_year <- accident_company_c5 %>%
  group_by(FK_CORE_EMPRESA, YEAR = year(MOMENTOACCIDENTE)) %>%
  summarise(n = n()) %>%
  left_join(select(accident_company_c5, FK_CORE_EMPRESA, size) %>%
    distinct(),
            by = "FK_CORE_EMPRESA") %>%
  mutate(n_log10 = log10(n))

g1 <- num_accidents_company_year %>%
  ggplot(aes(y = n_log10)) +
    geom_boxplot(outlier.colour = "indianred1") +
    xlab("log10(accidents by year)") +
```
```

```
    coord_flip() +
    ggtitle("Accidents per company and year (per size)") +
    facet_wrap(~size, scales = "free", ncol = 2)

g2 <- num_accidents_company_year %>%
  ggplot(aes(x = n_log10)) +
    geom_density() +
    xlab("log10(accidents by year)") +
    ggtitle("Accidents per company and year (per size)") +
    facet_wrap(~size, scales = "free", ncol = 2)

grid.arrange(g1, g2, nrow = 2)
```

Another thing that we can do in order to gain more information about the relationship between company size and accidents by year is to obtain a relative value.

```{r}
num_accidents_company_year <- num_accidents_company_year %>%
  left_join(select(accident_company_c5, FK_CORE_EMPRESA, NUMTRABAJADORES),
            by = "FK_CORE_EMPRESA") %>%
  mutate(n_relative = n/NUMTRABAJADORES,
         n_relative_log10 = log10(n_relative))

num_accidents_company_year %>%
  ggplot(aes(x = n_relative_log10)) +
    geom_density() +
    xlab("accidents by year / # workers") +
    ggtitle("Relative accidents per company and year") +
    facet_wrap(~size, scales = "free", ncol = 2)
```

The results have been converted to log10 in order to obtain a normal-shape. It is expected, as the graphs show, that the relation between accidents per year and number of workers increases as the company size decrease, nevertheless is not expected, for example, that a company with a small amount of workers have a much bigger amount of accidents in a year. We can analyze, for example, the accidents by year of those companies where number

of accidents reaches at some year an amount equal to five times the number of workers:

```{r fig.height=8}
companies_high_relative_accidents <- num_accidents_company_year %>%
  ungroup() %>%
  inner_join((num_accidents_company_year %>%
                ungroup() %>%
                filter(n_relative > 5) %>%
                select(FK_CORE_EMPRESA, n_relative) %>%
                arrange(desc(n_relative)) %>%
                select(FK_CORE_EMPRESA)),
             by = "FK_CORE_EMPRESA") %>%
  distinct()

companies_high_relative_accidents %>%
  ggplot(aes(x = n_relative)) +
    geom_density() +
    geom_vline(xintercept = 5, color = "red") +
    xlab("accidents by year / # workers") +
    ggtitle("Accidents > 5*workers") +
    facet_wrap(~NUMTRABAJADORES, scales = "free", ncol = 2)
```

In this case, all companies pertain to micro size group. This 477 companies have a big problem with this amount of accidents in a year or maybe in reality pertain to another size class, thus the number of workers may be erroneous. As the explanation seems to be like the second option we need to drop all the observations of these companies:

```{r}
accident_company_c6 <- accident_company_c5 %>%
  anti_join(companies_high_relative_accidents, by = "FK_CORE_EMPRESA")

num_accidents_company_year <- accident_company_c6 %>%
  group_by(FK_CORE_EMPRESA, YEAR = year(MOMENTOACCIDENTE)) %>%
  summarise(n = n()) %>%
  left_join(select(accident_company_c5, FK_CORE_EMPRESA, NUMTRABAJADORES,
    size),
```

```
            by = "FK_CORE_EMPRESA") %>%
  distinct() %>%
  mutate(n_log10 = log10(n),
         n_relative = n/NUMTRABAJADORES,
         n_relative_log10 = log10(n_relative))

num_accidents_company_year %>%
  filter(size == "micro") %>%
  ggplot(aes(x = n_relative_log10)) +
    geom_density() +
    xlab("accidents by year / # workers") +
    ggtitle("Relative accidents per company and year (micro)")
```

With this information we can check the Pearson coefficient between the compa
amount of accidents in a year. We have to convert before the number of worke

```{r}
num_accidents_company_year_m <- num_accidents_company_year %>% collect()

cor.test(num_accidents_company_year_m$n_log10,
         log10(num_accidents_company_year_m$NUMTRABAJADORES))

num_accidents_company_year_m %>%
  mutate(n_trabajadores_log10 = log10(NUMTRABAJADORES)) %>%
  ggplot(aes(x = n_trabajadores_log10, y = n_log10)) +
    geom_point(aes(color = size))
```

This shows that with a 95% of CI that the correlation is 0.66. Also, the distribution shows a spurious correlation.

Now, once studied the impact of the size of a company over the number of accidents in a year, we can check if there is a relationship between this number and the location of the company, afterwards we can do the same but with the location of the accident:

```{r}
accident_company_c6 %>%
```

```
  select(ID, PROVINCIA, POBLACION) %>%
  sdf_describe()

accident_company_c7 <- accident_company_c6 %>%
  filter(!is.na(PROVINCIA)) %>%
  filter(!is.na(POBLACION))
```

We have deleted a little bit of observations that had missing values at province or village fields.

Checking the distributions over every province we have:

```{r fig.height=10, fig.width=12}
num_accidents_company_year <- accident_company_c7 %>%
  group_by(FK_CORE_EMPRESA, YEAR = year(MOMENTOACCIDENTE)) %>%
  summarise(n = n()) %>%
  left_join(select(accident_company_c7, FK_CORE_EMPRESA, NUMTRABAJADORES,
    PROVINCIA, POBLACION, size),
            by = "FK_CORE_EMPRESA") %>%
  distinct() %>%
  mutate(n_log10 = log10(n),
         n_relative = n/NUMTRABAJADORES,
         n_relative_log10 = log10(n_relative),
         n_trabajadores_log10 = log10(NUMTRABAJADORES))

g1 <- num_accidents_company_year %>%
  ggplot(aes(y = n_log10)) +
    geom_boxplot() +
    facet_wrap(~PROVINCIA)

g2 <- num_accidents_company_year %>%
  ggplot(aes(x = n_log10)) +
    geom_density() +
    facet_wrap(~PROVINCIA)

grid.arrange(g1, g2, ncol = 1)
```

This shows differences between provinces that could be due to several factors associated to every zone. Also we can check if size company and province have some relationship:

```
```{r fig.height=10, fig.width=12, warning=FALSE}
num_accidents_company_year %>%
  ggplot(aes(y = n_log10, fill = size, x = reorder(size, size,
    function(x) if (x == "micro"){ 2 }else{ 1 }))) +
    geom_boxplot() +
    xlab("") +
    facet_wrap(~PROVINCIA)
```
```

Where we can see that, at some places, the expected distribution of accidents by size seen before is not followed. This is the case for example of Casanare or Norte de Santander. Also, some provinces doesnt have some kinds of company sizes, like Boyaca or Vichada. This particularities can be decisive predicting the number of accidents.

If we want a measure of this difference between provinces we can do a one-way ANOVA test:

```
```{r message=FALSE, warning=FALSE}
num_accidents_company_year_m <- num_accidents_company_year %>%
  ungroup() %>%
  select(n_log10, PROVINCIA) %>%
  collect() %>%
  mutate(PROVINCIA = as.factor(PROVINCIA))

res.aov <- aov(n_log10 ~ PROVINCIA, data = num_accidents_company_year_m)
summary(res.aov)
```
```

Which shows that there is a significat difference between the average number of accidents of provinces.

With the company activity, we first need to clean their names in order to group them:

```r
accident_company_c8 <- accident_company_c7 %>%
  filter(CNAE != "NA")

num_accidents_company_year_m <- accident_company_c8 %>%
  group_by(FK_CORE_EMPRESA, YEAR = year(MOMENTOACCIDENTE)) %>%
  summarise(n = n()) %>%
  left_join(select(accident_company_c8, FK_CORE_EMPRESA, NUMTRABAJADORES,
    PROVINCIA, POBLACION, size, CNAE),
            by = "FK_CORE_EMPRESA") %>%
  distinct() %>%
  ungroup() %>%
  collect() %>%
  mutate(n_relative = n/NUMTRABAJADORES) %>%
  mutate(CNAE = str_remove(CNAE, "EMPRESAS DEDICADAS A ")) %>%
  mutate(CNAE = str_remove(CNAE, "LA ")) %>%
  mutate(CNAE = substr(CNAE, 1, 70)) %>%
  mutate(CNAE = gsub(",$", "", CNAE)) %>%
  mutate(CNAE = gsub(", $", "", CNAE)) %>%
  mutate(CNAE = paste(CNAE, "...", sep = ''))

num_accidents_company_year_m %>%
  group_by(CNAE) %>%
  summarise(n_companies = n_distinct(FK_CORE_EMPRESA)) %>%
  ggplot(aes(x = n_companies)) +
    geom_density() +
    scale_x_log10()

num_accidents_company_year_m %>%
  group_by(CNAE) %>%
  summarise(n_companies = n_distinct(FK_CORE_EMPRESA)) %>%
  group_by(n_companies) %>%
  summarise(n_activities = n()) %>%
  arrange(desc(n_activities)) %>%
  head(10)
```

As can be seen, most of activities have 10 or less companies with an accident registered. This could produce some bias and alter the results

```
of relationships and predictions as this amounts are not representative.

```{r fig.width=14, warning=FALSE}
num_accidents_company_year_m2 <- num_accidents_company_year_m %>%
  group_by(CNAE) %>%
  summarise(n_mean = mean(n),
            n_median = median(n),
            n_relative_mean = mean(n_relative),
            n_relative_median = median(n_relative))

g1 <- num_accidents_company_year_m2 %>%
  arrange(desc(n_mean)) %>%
  head(10) %>%
  ggplot(aes(x = reorder(CNAE, n_mean), y = n_mean,
    fill = CNAE)) +
    geom_col() +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year (mean)")

g2 <- num_accidents_company_year_m2 %>%
  arrange(desc(n_median)) %>%
  head(10) %>%
  ggplot(aes(x = reorder(CNAE, n_median), y = n_median,
    fill = CNAE)) +
    geom_col() +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year (median)")

g3 <- num_accidents_company_year_m2 %>%
  arrange(desc(n_relative_mean)) %>%
  head(10) %>%
```
```

```
  ggplot(aes(x = reorder(CNAE, n_relative_mean), y = n_relative_mean,
    fill = CNAE)) +
    geom_col() +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year relative to
      # workers (mean)")

g4 <- num_accidents_company_year_m2 %>%
  arrange(desc(n_relative_median)) %>%
  head(10) %>%
  ggplot(aes(x = reorder(CNAE, n_relative_median), y = n_relative_median,
    fill = CNAE)) +
    geom_col() +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year relative to
      # workers (median)")

grid.arrange(g1, g2, g3, g4, ncol = 2, nrow = 2)
```

As can be seen, the activities with highest number of accidents in a year are those related with justice, minerals extraction, manufactures and agriculture like palm oil or banana production whereas those with relative number of accidents are related with metal foundries, coffee production and some manufactures. If we analyze the companies related to these top accidented activities:

```
```{r fig.height=6, fig.width=14, message=FALSE, warning=FALSE}
g1 <- num_accidents_company_year_m %>%
  inner_join(num_accidents_company_year_m2 %>%
               arrange(desc(n_median)) %>%
```
```

```
                head(10) %>%
                select(CNAE),
              by = "CNAE") %>%
  group_by(CNAE) %>%
  summarise(n_companies = n_distinct(FK_CORE_EMPRESA)) %>%
  ggplot(aes(x = reorder(CNAE, n_companies), y = n_companies,
    fill = CNAE)) +
    geom_col() +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year (median)")

g2 <- num_accidents_company_year_m %>%
  inner_join(num_accidents_company_year_m2 %>%
                arrange(desc(n_relative_median)) %>%
                head(10) %>%
                select(CNAE),
              by = "CNAE") %>%
  group_by(CNAE) %>%
  summarise(n_companies = n_distinct(FK_CORE_EMPRESA)) %>%
  ggplot(aes(x = reorder(CNAE, n_companies), y = n_companies,
    fill = CNAE)) +
    geom_col() +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year relative to
      # workers (median)")

grid.arrange(g1, g2, ncol = 2)
```

We can see that, for absolut or relative numbers of median accidents by year, the top activities are represented by a little number of companies.

Also, we can divide the analysis by company size and see the possible relationship with activities:

```
```{r fig.height=12, fig.width=14, message=FALSE, warning=FALSE}
give.n <- function(x){
  return(c(y = max(x)*1.15, label = length(x)))
}

num_accidents_company_year_m2 <- num_accidents_company_year_m %>%
  group_by(size, CNAE) %>%
  summarise(n_mean = mean(n),
            n_median = median(n),
            n_relative_mean = mean(n_relative),
            n_relative_median = median(n_relative))

g1 <- num_accidents_company_year_m %>%
  filter(size == "big") %>%
  inner_join(num_accidents_company_year_m2 %>%
               filter(size == "big") %>%
               arrange(desc(n_median)) %>%
               head(10) %>%
               select(CNAE),
             by = "CNAE") %>%
  ggplot(aes(x = CNAE, y = n, fill = CNAE)) +
    geom_boxplot() +
    stat_summary(fun.data = give.n, geom = "text", fun.y = median) +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year relative to
      # workers (big)")

g2 <- num_accidents_company_year_m %>%
  filter(size == "medium") %>%
  inner_join(num_accidents_company_year_m2 %>%
               filter(size == "medium") %>%
```
```

```
                arrange(desc(n_median)) %>%
                head(10) %>%
                select(CNAE),
              by = "CNAE") %>%
  ggplot(aes(x = CNAE, y = n, fill = CNAE)) +
    geom_boxplot() +
    stat_summary(fun.data = give.n, geom = "text", fun.y = median) +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year relative to
      # workers (medium)")

g3 <- num_accidents_company_year_m %>%
  filter(size == "small") %>%
  inner_join(num_accidents_company_year_m2 %>%
                filter(size == "small") %>%
                arrange(desc(n_median)) %>%
                head(10) %>%
                select(CNAE),
              by = "CNAE") %>%
  ggplot(aes(x = CNAE, y = n, fill = CNAE)) +
    geom_boxplot() +
    stat_summary(fun.data = give.n, geom = "text", fun.y = median) +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year relative to
      # workers (small)")

g4 <- num_accidents_company_year_m %>%
  filter(size == "micro") %>%
  inner_join(num_accidents_company_year_m2 %>%
                filter(size == "micro") %>%
                arrange(desc(n_median)) %>%
```

```
                head(10) %>%
                select(CNAE),
              by = "CNAE") %>%
  ggplot(aes(x = CNAE, y = n, fill = CNAE)) +
    geom_boxplot() +
    stat_summary(fun.data = give.n, geom = "text", fun.y = median) +
    xlab("CNAE") +
    coord_flip() +
    theme(legend.position = "bottom",
          legend.direction = "vertical",
          axis.text.y = element_blank()) +
    ggtitle("Most accidented activities in a year relative to
      # workers (micro)")

grid.arrange(g1, g2, g3, g4, ncol = 2)
```

This shows that exist some activities with only registered accidents of one company in one year that have a high median, as its shown at micro size companies. Also, most of activities have a little number of observations whereas only a few of them have an important amount to represent a better distribution of accidents in a year. This is the case of banana production, coal extraction, glass, clay and metal manufactures.

Modeling

Partitioning and feature encoding

First of all, we need to partition data into training and testing sets. Also we have to one-hot-encode province and activity as Spark MLib algorithms doesn do this by itself.

```{r warning=FALSE}
data_tbl <- copy_to(sc, num_accidents_company_year_m,
  "num_accidents_company_year", overwrite = TRUE)
data_tbl <- data_tbl %>%
  mutate(n_log10 = log10(n)) %>%
  ft_string_indexer("PROVINCIA", "PROVINCIA_idx") %>%
```

```
  ft_one_hot_encoder("PROVINCIA_idx", "PROVINCIA_ohe") %>%
  ft_string_indexer("CNAE", "CNAE_idx") %>%
  ft_one_hot_encoder("CNAE_idx", "CNAE_ohe")

partitions <- sdf_partition(data_tbl, training = 0.8, testing = 0.2)
```

```
##### Modeling with Spark MLib

Checking some algorithms it seems that GBM has the best performance.

```{r}
pipeline <- ml_pipeline(sc) %>%
  ft_r_formula(n_log10 ~ NUMTRABAJADORES + PROVINCIA_ohe + CNAE_ohe) %>%
  ml_gbt_regressor(max_iter = 100)

grid <- list(gbt_regressor = list(
  max_depth = c(30)))

cv <- ml_cross_validator(
  sc, estimator = pipeline, estimator_param_maps = grid,
  evaluator = ml_regression_evaluator(sc),
  num_folds = 10,
  parallelism = 5)

cv_model <- ml_fit(cv, partitions$training)

pred_test <- ml_predict(cv_model, partitions$testing)
pred_train <- ml_predict(cv_model, partitions$training)
print(paste("RMSE train:", ml_regression_evaluator(pred_train,
  label_col = "n_log10", metric_name = "rmse")))
print(paste("RMSE test:", ml_regression_evaluator(pred_test,
  label_col = "n_log10", metric_name = "rmse")))
print(paste("R2:", ml_regression_evaluator(pred_train,
  label_col = "n_log10", metric_name = "r2")))
```

```
This shows an error of ~2 accidents on test set and that the
variability of the data is explained by the model in a ~78%.
```
```

```
##### Modeling with H2O

Modeling with H2O results extremely fast compared to same algorithms
at Spark MLib. Also, H2O algorithms do the needed transformation
on categorical variables.

After checking algorithms as random forest, GBM, GLM, XGBoost,
neural networks or an stacked ensemble of all of them, XGBoost
seems to have the best results. We have covered some hyperparameters
using a grid search:

'''{r}
training <- as_h2o_frame(sc, partitions$training,
  strict_version_check = FALSE)
test <- as_h2o_frame(sc, partitions$testing,
  strict_version_check = FALSE)

training["PROVINCIA"] <- as.factor(training["PROVINCIA"])
training["POBLACION"] <- as.factor(training["POBLACION"])
training["CNAE"] <- as.factor(training["CNAE"])

nfolds <- 10

xgboost_params <- list(learn_rate = c(0.1, 0.2, 0.3),
                       max_depth = c(20, 25, 30),
                       sample_rate = c(0.4, 0.6, 0.8, 1),
                       ntrees = c(30, 40, 50))

xgboost_grid1 <- h2o.grid("xgboost", x = c("NUMTRABAJADORES", "PROVINCIA",
  "CNAE"),
                          y = "n_log10",
                          grid_id = "xgboost_grid1",
                          training_frame = training,
                          hyper_params = xgboost_params,
                          distribution = "gaussian",
                          nfolds = nfolds,
                          fold_assignment = "Modulo",
                          keep_cross_validation_predictions = TRUE)
```

```
xgboost_grid1perf <- h2o.getGrid(grid_id = "xgboost_grid1",
                                 sort_by = "RMSE")

aml_model <- h2o.automl(x = c("NUMTRABAJADORES", "PROVINCIA", "CNAE"),
                        y = "n_log10",
                        training_frame = training,
                        nfolds = nfolds)

best_model <- h2o.getModel(xgboost_grid1perf@model_ids[[1]])

summary(best_model)

h2o.performance(model = best_model, newdata = test)
```

```{r}
aml_model@leaderboard
aml_best_model <- aml_model@leader
summary(aml_best_model)
perf <- h2o.performance(model = aml_best_model, newdata = test)
h2o.r2(perf)
perf
```

Results are pretty similar to those obtained with GBM and Spark MLib. Also, the most important variable to infer the results is the number of workers. Some activities and provinces contribute a little bit in the improvement of the precision.

Bibliography

[1] Sabentis pro. http://easytechglobal.com/soluciones/#sabentis1. Accessed: 2019-06-21.

[2] Colmena seguros. https://www.colmenaseguros.com/Paginas/default.aspx. Accessed: 2019-06-21.

[3] Easy tech global. https://en.wikipedia.org/wiki/Online_algorithm. Accessed: 2019-04-17.

[4] Fondo de riesgos laborales, afiliados, consolidado estadísticas accidentes y enfermedades laborales - 2018. http://www.fondoriesgoslaborales.gov.co/wp-content/uploads/2019/02/AfiliadosATEL-DIC-2018-18-ene-2019.xlsx. Accessed: 2019-06-21.

[5] The popularity of data science software. https://r4stats.com/articles/popularity/. Accessed: 2019-02-11.

[6] The most in demand skills for data scientists. https://towardsdatascience.com/the-most-in-demand-skills-for-data-scientists-4a4a8db896db. Accessed: 2019-02-11.

[7] External memory algorithm. https://en.wikipedia.org/wiki/External_memory_algorithm. Accessed: 2019-02-11.

[8] Wikipedia, online algorithm. https://en.wikipedia.org/wiki/Online_algorithm. Accessed: 2019-02-14.

[9] Hafen ryan. 18 feb 2016, divide and recombine: Approach for detailed analysis and visualization of large complex data from: Handbook of big data, routledge. https://www.routledgehandbooks.com/doi/10.1201/b19567-6. Accessed: 2018-08-15.

[10] Mapreduce. https://en.wikipedia.org/wiki/MapReduce. Accessed: 2019-02-15.

[11] Bigmemory. https://cran.r-project.org/web/packages/bigmemory/index.html. Accessed: 2019-03-02.

[12] ff. https://cran.r-project.org/web/packages/ff/ff.pdf. Accessed: 2019-03-02.

[13] monetdb. https://www.monetdb.org/Home. Accessed: 2019-03-14.

[14] monetdbr. https://cran.r-project.org/web/packages/MonetDB.R/index.html. Accessed: 2019-03-14.

[15] monetdblite. https://github.com/MonetDB/MonetDBLite-R. Accessed: 2019-03-14.

[16] trelliscope - tutorial. http://deltarho.org/docs-trelliscope/. Accessed: 2019-02-11.

[17] datadr - divide and recombine in r. http://deltarho.org/docs-datadr/. Accessed: 2019-02-15.

[18] Revoscaler. https://docs.microsoft.com/en-us/machine-learning-server/r-reference/revoscaler/revoscaler. Accessed: 2019-04-25.

[19] Microsoft machine learning services. https://docs.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/data-science-and-machine-learning. Accessed: 2019-04-25.

[20] Microsoft r. https://mran.microsoft.com/open. Accessed: 2019-04-25.

[21] Microsoft machine learning server. https://docs.microsoft.com/en-us/machine-learning-server/what-is-machine-learning-server. Accessed: 2019-04-25.

[22] Xdf. https://docs.microsoft.com/en-us/machine-learning-server/r/concept-what-is-xdf. Accessed: 2019-04-25.

[23] Xgboost. https://xgboost.ai/. Accessed: 2019-03-15.

[24] Kaggle competitions and xgboost. https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions. Accessed: 2019-03-15.

[25] xgboost. https://cran.r-project.org/web/packages/xgboost/. Accessed: 2019-03-15.

[26] H2o. https://www.h2o.ai/products/h2o/. Accessed: 2019-04-19.

[27] h2o. https://cran.r-project.org/web/packages/h2o/. Accessed: 2019-04-19.

[28] Sparkling water. https://www.h2o.ai/products/h2o-sparkling-water/. Accessed: 2019-04-19.

[29] H2o. https://cran.r-project.org/web/packages/rsparkling/index.html. Accessed: 2019-04-19.

[30] Vowpal wabbit. https://github.com/VowpalWabbit/vowpal_wabbit/wiki. Accessed: 2019-04-26.

[31] rvw. https://github.com/rvw-org/rvw. Accessed: 2019-04-26.

[32] parallel. https://stat.ethz.ch/R-manual/R-devel/library/parallel/html/parallel-package.html. Accessed: 2019-05-02.

[33] doparallel. https://cran.r-project.org/web/packages/doParallel/. Accessed: 2019-05-02.

[34] Kdnuggets, 2019 poll. https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html. Accessed: 2019-05-10.

[35] r4stats, ds job report 2019. http://r4stats.com/2019/05/28/data-science-jobs-report-2019-python-way-up-tensorflow-growing-rapidly-r-use-dou Accessed: 2019-05-10.

[36] benchm-ml. https://github.com/szilard/benchm-ml/blob/master/README.md. Accessed: 2019-05-10.

[37] Sql server 2019 preview. https://cloudblogs.microsoft.com/sqlserver/2018/09/24/sql-server-2019-preview-combines-sql-server-and-apache-spark-to-create-a-unified Accessed: 2019-02-10.

[38] Kdnuggets, top methodology analytics data mining and data science projects. https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html. Accessed: 2019-05-24.

[39] From data mining to knowledge discovery in databases. https://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf. Accessed: 2019-05-24.

[40] Sas, data mining and semma. http://support.sas.com/documentation/cdl/en/emcs/66392/HTML/default/viewer.htm#n0pejm83csbja4n1xueveo2uoujy.htm. Accessed: 2019-05-24.

[41] Wikipedia, crisp-dm. https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining. Accessed: 2019-05-24.

[42] Ibm asum-dm. https://web.archive.org/web/20160308065035/https://developer.ibm.com/predictiveanalytics/2015/10/16/have-you-seen-asum-dm/. Accessed: 2019-05-24.

[43] Team data science process. https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle. Accessed: 2019-05-24.

[44] Domino data science lifecycle. https://www.dominodatalab.com/wp-content/uploads/domino-managing-ds.pdf. Accessed: 2019-05-24.

[45] Wikipedia, "extract, transform, load". https://en.wikipedia.org/wiki/Extract,_transform,_load. Accessed: 2019-05-23.

[46] Spark, sql data sources. https://spark.apache.org/docs/latest/sql-data-sources.html. Accessed: 2019-05-23.

[47] Spark, jdbc to other databases". https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html. Accessed: 2019-05-23.

[48] Java jdbc api. https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/. Accessed: 2019-05-23.

[49] Microsoft jdbc driver for sql server. https://docs.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-2017. Accessed: 2019-05-23.

[50] Apache parquet. http://parquet.apache.org/. Accessed: 2019-05-23.

[51] Wikipedia, column-oriented dbms. https://en.wikipedia.org/wiki/Column-oriented_DBMS. Accessed: 2019-05-23.

[52] The design and implementation of modern column-oriented data. http://db.csail.mit.edu/pubs/abadi-column-stores.pdf. Accessed: 2019-05-23.

[53] Salario mínimo colombia, histórico. https://www.salariominimocolombia.net/historico/. Accessed: 2019-05-24.

[54] Portafolio, así puede trabajar legalmente un menor de edad en colombia. https://www.portafolio.co/economia/empleo/asi-puede-trabajar-legalmente-un-menor-de-edad-en-colombia-499730. Accessed: 2019-05-24.

[55] Wikipedia, documento de identidad (colombia). https://es.wikipedia.org/wiki/Documento_de_identidad#Colombia. Accessed: 2019-05-24.

[56] Colconectada, nip y nuip. https://www.colconectada.com/nip-y-nuip/. Accessed: 2019-05-24.