Trabajo de Fin de Máster

Desarrollo de métodos para la extracción y análisis de datos de fuentes públicas: estudio de la plataforma Airbnb

Development of methods for the extraction and analysis of data from public sources: Airbnb platform research

Sergio Martín Santana

La Laguna, 2 de julio de 2019

D. **Marcos Colebrook Santamaría**, con N.I.F. 43.787.808-V profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y Sistemas de la Universidad de La Laguna

D. **Carlos J. Pérez González**, con N.I.F. 45.452.719-G profesor Ayudante Doctor adscrito al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna

C E R T I F I C A N

Que la presente memoria titulada:

"Desarrollo de métodos para la extracción y análisis de datos de fuentes públicas: estudio de la plataforma Airbnb"

ha sido realizada bajo su dirección por D. **Sergio Martín Santana**, con N.I.F. 54.110.144-E.

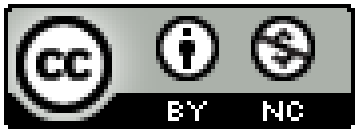
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 2 de Julio de 2019.

Agradecimientos

Dedicado a todas aquellas personas
que de una manera u otra
lo han hecho posible,

gracias.

Licencia



Resumen

El objetivo de este trabajo ha sido realizar un estudio de las diferentes técnicas de Web Scraping, soluciones disponibles y conocer a grandes rasgos algunos aspectos de su marco legal.

Posteriormente, mediante el uso de un proyecto de código abierto, hemos generado un modelo basado en datos de la plataforma abierta AirBnb para el listado y alquiler de alojamientos. En primer lugar, realizamos un proceso de transformación y limpieza de datos, para a continuación ejecutar distintas explotaciones estadísticas.

Palabras clave: *Web Scraping, Turismo, AirBnB, Modelo Predictivo, Explotación de Datos, Análisis Estadíticos.*

Abstract

The objective of this research has been to carry out a study of the different techniques on Web Scraping, the available solutions and to roughly learn some aspects of its legal framework.

Subsequently, using an open source project, we have generated a model based on data from the AirBnb community platform for listing and renting local homes. Firstly, we carried out both a data transformation and a cleaning process, and then we performed different statistical techniques.

Keywords: *Web Scraping, Tourism, AirBnB, Predictive Models, Data Mining, Statistical Analysis.*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1 Introducción

En la actualidad se generan ingentes cantidades de información, por lo que no es de extrañar que existan movimientos paralelos que pretendan apoderarse de esa información para su posterior transformación y visualización.

Es por ello, que surgen técnicas conocidas como *Web Scraping* ("raspado de páginas web"), las cuales consisten en la extracción de datos de una o varias páginas dentro de un sitio web, para su posterior manipulación y análisis. Este no es un concepto inusual, pues es la idea básica en el que se sustenta Google para indexar los sitios web de Internet y presentar resultados.

Esta técnica engloba multitud de conceptos dependiendo de cómo se lleve a cabo esta captura de datos. Los conceptos de *web crawling* o web spider, se refieren a los casos en los que para obtener las páginas deseadas debemos de rastrear sus enlaces web. Es decir, deberemos navegar por el sitio web recursivamente hasta obtener el nivel de profundidad que determinemos. Por otro lado, bajo el término *screen scraping*, mediante el uso de ingeniería inversa se extraen los datos a partir de la capa de presentación.

En cuanto a usos, estas técnicas pueden ser utilizadas en múltiples escenarios diferentes: para realizar optimizaciones web de todo tipo, como SEO "*Search Engine Optimization*" (Posicionamiento en motores de búsqueda) o WPO "*Web Performance Optimization*" (Optimización de rendimiento Web), para extraer datos y analizarlos, para alimentar una base de datos, hacer una migración de un sitio web, para recopilar y ofrecer datos dispersos por varias webs, generar alertas, y algunos otros menos lícitos o desaconsejados como el robo de información de carácter sensible.[1][2]

A nivel práctico este tipo de técnicas son utilizadas por empresas para: monitorización de precios de la competencia, detección de cambios en sitios web, registrar lanzamientos o novedades de un producto, etc.

A la hora de tomar en consideración el uso de estas técnicas en un proyecto de *Web Scraping*, deberíamos de plantearnos tres aspectos fundamentales:

1. **Frecuencia de la Extracción**

Los datos serán capturados de manera puntual, o serán actualizados periódicamente en busca de obtener la información más actualizada posible.

2. **El volumen de datos y los recursos disponibles**

Deberemos plantearnos el volumen de datos, pues no podremos utilizar los mismos procesos y recursos para proyectos de un volumen bajo de datos, como para grandes y ambiciosos proyectos.

3. **La accesibilidad de los datos de origen**

Deberemos de tener en cuenta si nuestros datos están en una o varias páginas y como se encuentran presentados en dicha página/s.

1.2 Web Scraping

1.2.1 Herramientas para *Web Scraping*

Para poder realizar nosotros mismos este tipo de procesos existen multitud de alternativas y modos de ejecutarlos, a continuación, se muestran algunas alternativas interesantes basadas en diferentes tecnologías:

- **Usar un servicio de web scraping**

Podríamos decidir utilizar alguno de los servicios web que nos permiten extraer datos estructurados una página web. Algunos tienen la ventaja de que, si la web de origen se encuentra bien estructurada, podremos extraer los datos de una manera automática, ahorrándonos mucho tiempo, pero en otros hemos de escribir las reglas de extracción.

Como puntos fuertes, este tipo de servicios nos da un buen punto de partida, si no tenemos estudios demasiados complejos. Nos permite realizar un proceso de *scraping* sin necesidad de grandes conocimientos en la materia. Además, muchos de ellos proveen de contramedidas ante sistemas anti-*scraping*.

En contraposición, estamos limitados a las funcionalidades implementadas en el servicio online, así como a sus términos y condiciones sobre la manipulación de datos.[3]

Algunas de estas soluciones son:

80Legs	80legs.com
ScrapingHub	scrapinghub.com/scrapy-cloud
Import.io	www.import.io
DataHut	datahut.co

- **Usar una solución desktop para web *scraping* local**

Otra alternativa sería utilizar herramientas en nuestro sistema local, ya sea usando programas desarrollados con esta finalidad, o bien complementos (*plugins*) para nuestros navegadores web, en las cuales deberemos de implementar reglas para nuestro *crawler* (*bot* de búsqueda). Este tipo de solución es más acertada para casos puntuales, ya que siempre requieren de intervención humana (al menos de manera nativa) y no están orientadas a ejecuciones programadas. También como se puede entender, deberemos contar con algunos conocimientos en la materia, por lo que es una opción a medio plazo.

En cuanto a las ventajas de este tipo de técnicas, primeramente, existen algunas que son de libre acceso (gratis) y nos permiten ejecutarlas sin límite de datos y frecuencia. Al determinar nosotros mismos las reglas, nos permite mayor personalización y libertad de extracción. En cuanto a los puntos flojos, podemos mencionar que suelen ser fácilmente bloqueados por los sitios web de los que queremos obtener la información ya que no cuentan con contramedidas *anti-scraping*, y que tampoco podremos programar la extracción de datos (al menos no de manera nativa) y, finalmente, que deberemos de estar presentes al menos para iniciar el proceso de obtención de datos.

Soluciones basadas en ésta arquitectura:

WebScrapper: webscraper.io
FMiner: www.fminer.com
DataMiner: data-miner.io
Artoo.Js: medialab.github.io/artoo
Scrapper: chrome.google.com/webstore/detail/scraper/mbigbapnjcgaffohmbkdlecaccepngjd

- **Programar el algoritmo de *web scraping***

Utilizando un lenguaje de programación de propósito general, junto con librerías nativas, o mediante el uso de otras librerías de usuarios o *frameworks*, podríamos generar las suficientes líneas de código para generar una aplicación de *Web Scraping* a medida. Esta vía al ser generada desde "cero" por el usuario, se ajusta completamente a la funcionalidad para la que fue diseñada, por lo que aporta mayor flexibilidad e integración con el caso a resolver.

Esto podría verse como una ventaja o desventaja, la tolerancia al cambio, la define el diseñador de la aplicación, así como su alcance. Además de requerir un nivel de conocimientos avanzado. Además de que deberemos de incluir nosotros mismos las operaciones para evitar los bloqueos de las páginas web.

Soluciones basadas en ésta arquitectura:

Scrapy: scrapy.org/
CasperJS: casperjs.org/
Guzzle: docs.guzzlephp.org/en/stable/

1.2.2 Problemática relacionada con el uso del *Web Scraping*

Como hemos comentado en apartados anteriores, las propias páginas web se protegen de estas técnicas de web scraping, tanto más cuanto mayor valor tenga o determinen sus propietarios que esta pueda tener para ellos, ya que la propiedad puede ser susceptible de opinión.

Entre los métodos más recurrentes a nivel tecnológico se encuentran: el uso de sesiones, uso de *Javascript* o bloqueos por uso.

Si nos encontramos con una web protegida por usuario, nos indicará la necesidad de pasar por alguna página inicial que nos haga generar una sesión de usuario y validar el acceso a los datos que se encuentran en la página.

Otra operativa sería el uso de *Javascript* o peticiones Ajax, en las que necesitaremos de herramientas tipo browser *headless* o complementos de navegador.

También podríamos encontrar bloqueos de información por uso no humano, detectable por medio del número de páginas solicitadas por minuto, no solicitar todos los elementos de la página *html*, navegador web no identificado, identificador "*User-Agent*" sospechoso, entre otros. Ante este tipo de comportamientos la página puede presentar bloqueos de IP o solicitudes de *CAPTCHA*. [4]

Otras prácticas utilizadas por las páginas web no contempladas en los párrafos anteriormente serían:

- Ofuscar o esconder datos en imágenes o códigos CSS.
- Modificación de los *tags*, para evitar que se realicen procesos recurrentes.
- Inclusión de balizas (información falsa) para detectar orígenes y operativa de procesos de *scraping*.

1.2.3 Marco Legal del Web Scraping

Este es un apartado que cabe destacar en el campo de estudio en el que nos encontramos. Para algunos es un delito de robo, para otros no es más no lo ven de esa forma.

Un caso mediático a nivel internacional fue el sucedido entre la importante red social orientada a las empresas, negocios y el empleo, **LinkedIn** y una empresa del campo del *Data Science* denominada **hiQ Labs**. La información

que hiQ proporciona a sus clientes incluye los datos extraídos de los perfiles públicos de LinkedIn de sus empleados. Por esto LinkedIn, denunció a hiQ Labs a mediados de 2017, aunque a finales del mismo año un juez desestimó la denuncia y LinkedIn tuvo que levantar el bloqueo que había impuesto en contra de *hiQ Labs*.

En nuestro país también ha habido casos por estas prácticas y es que, en 2012, a raíz de la sentencia del Tribunal Supremo de 9 de octubre, nº 572/2012, de RyanAir vs. Atrápalo, en la cual se determinó que, en ese caso, el *web scraping* llevado a cabo por Atrápalo era legal.

Sin entrar en demasiado detalle, pasaremos a señalar los diferentes puntos a la hora de determinar la legalidad del *web scraping* y el posterior uso de estos datos, basándonos en el marco legal español.

Vulneración de los derechos de propiedad intelectual de los titulares de la página web, en aquellos casos en los que, además de otros requisitos recogidos en el Real Decreto Legislativo 1/1996, de 12 de abril.

Cumplirán este postulado aquellas bases de datos donde se demuestre la estructura original objeto del *web scraping* convirtiéndola en obra intelectualmente protegida.

En caso de no poderse probar dicha originalidad también se aceptaría aquellos casos en que la creación de la misma haya supuesto una inversión sustancial por parte de su fabricante, ya que podría ser susceptible de ser protegida a través del derecho *sui generis* (Título VIII de la Ley de Propiedad Intelectual).

Conducta de competencia desleal, siempre que la finalidad del estudio sea susceptible de ser considerada una imitación o plagio, ofreciendo servicios similares a los prestados por el *website* objeto de *scraping*, creando en el usuario final confusión o el aprovechamiento indebido de la reputación o esfuerzo ajeno.

Violación de los términos legales y condiciones de uso establecidos por los titulares de la página objeto de *scraping*, una vez sean aceptados por los usuarios que naveguen por la página web y tengan acceso a la información contenida en la misma.

Incumplimiento de la normativa en protección de datos y la vulneración de los derechos de los titulares de los datos personales, en Ley Orgánica 15/1999, de 13 de diciembre de Protección de Datos de Carácter Personal (en adelante, LOPD), se exige que se cuente con el consentimiento del titular de los datos a la hora de proceder el almacenamiento y tratamiento de los mismos.

Además, define también que el propietario debe estar informado de las finalidades del mencionado tratamiento, ya que en caso contrario podríamos

enfrentarnos a una sanción grave en los términos establecidos por el artículo 44.3 B, de la LOPD.

Tampoco deberemos obviar que como define la AEPD (Agencia Española de Protección de Datos), las páginas web no podrán ser consideradas en ningún caso fuentes accesibles al público. Dado que no figuran en el listado del artículo 7 del Real Decreto 1720/2007. En conclusión, postula que, aunque los datos se encuentren publicados en *websites* de pública, esto no significa que puedan ser utilizados de forma indiscriminada para cualquier finalidad, yendo contra los intereses del propietario original y sin contar con el consentimiento previo de los titulares de los datos.[5][6]

Capítulo 2.
Extracción y análisis de datos turísticos Airbnb

En el capítulo anterior se ha introducido el concepto del *Web Scraping,* en el nuevo capítulo nos centraremos en detallar el Objetivo de este estudio.

2.1 Introducción

En este TFM nos hemos decantado por utilizar una aplicación de terceros desarrollada por el usuario TomsLee, dado que AirBnB no cuenta con un API nativa para desarrolladores. Dicho proyecto está basado en una API en Python3 disponible en github.com/tomslee/airbnb-data-collection. Esta API, después de configurar algunos aspectos de la conexión, nos permite capturar de la web de AirBnb [9] los resultados de habitaciones siguiendo varios criterios de búsqueda, lo cuales podremos seleccionar.

Entre otras razones, nos hemos decidido por esta aplicación porque es de código abierto y gratuito, y está basada en tecnologías emergentes en el campo del *Web Scraping*. Por otra parte, cuenta con un script de despliegue en Docker, por lo que facilita la instalación y configuración de la herramienta. De otra manera deberíamos de instalar nosotros previamente la arquitectura de DB (Postgres) y Un host con Python3, y las librerías necesarias para su ejecución. Al usar Docker, junto con un Docker Compose, este proceso se simplifica enormemente, y es el sistema Docker el encargado de instalar y configurarlo todo y además conectar ambos aspectos entre sí.

2.2 Objetivo

El objetivo de este proyecto es aplicar técnicas de *scraping* para capturar información de la plataforma Airbnb. Para ello, realizaremos un estudio de herramientas en el campo del *WebScraping*.

Se creará un modelo de predicción, utilizando viviendas vacacionales del área de Tenerife. Para darle valor al modelo, agregaremos información a los datos capturados por la API utilizando servicios online, para incluir información de la localización, aspecto en el que flaquea la API de origen. También se realizará una transformación de los datos en el proceso de transformación para su correcta visualización.

2.3 Arquitectura y Herramientas Utilizadas

2.3.1 Aplicaciones

Para hacer uso de esta aplicación, deberemos de tener previamente instalados diversos servicios.

- Python 3.4 o posterior
- PostgreSQL 9.5 o posterior
- RStudio (En nuestro caso hemos hecho uso del Stack de herramientas presente en DockerForScience[8])

Por otro lado, se nos presenta una opción basada en Docker por parte del usuario creador. Con esta versión simplemente deberemos tener instalado en nuestro sistema Docker, Docker Compose y conexión a Internet. Utilizando el script proporcionado por el usuario, se encargará de dar de alta y configurar las conexiones entre los diferentes sistemas que lo integran. Para hacer uso de Docker debemos contar con un sistema Linux (x64) o Windows 10 (x64).

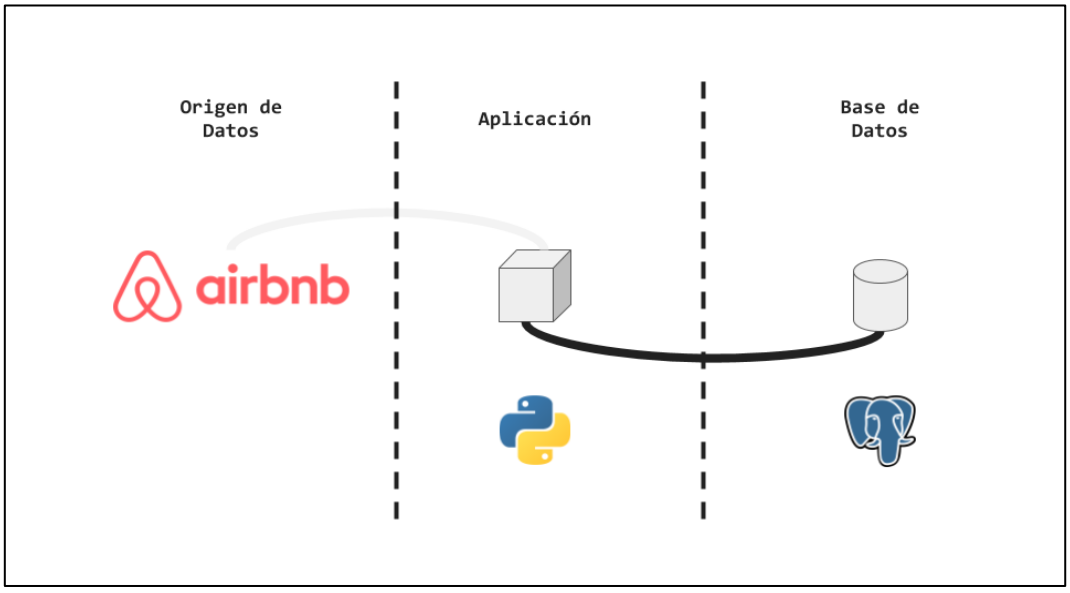


Figura 2.1. Arquitectura sin Docker

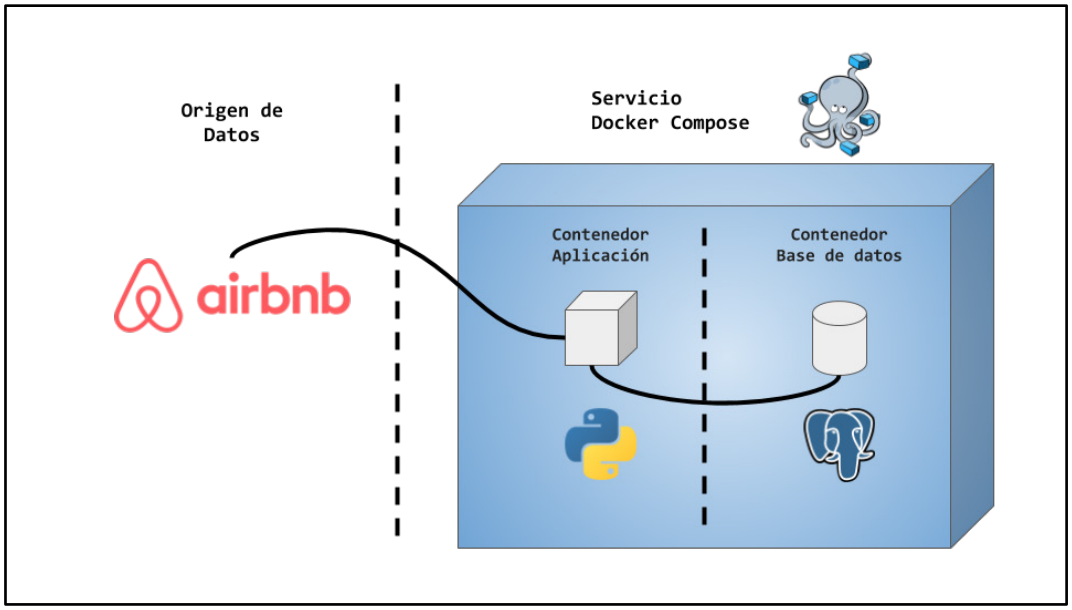


Figura 2.2. Arquitectura con Docker

2.3.2 Sistema

El equipo utilizado para probar este proyecto cuenta con una máquina virtual Oracle VirtualBox 5.2. Esta máquina virtual Basada en Ubuntu 16.04LTS x64, cuenta con 8GB de memoria RAM dedicada, 2 núcleos de CPU intel i7-7700HQ 2.8GHz (3.8 GHz Turbo Boost).

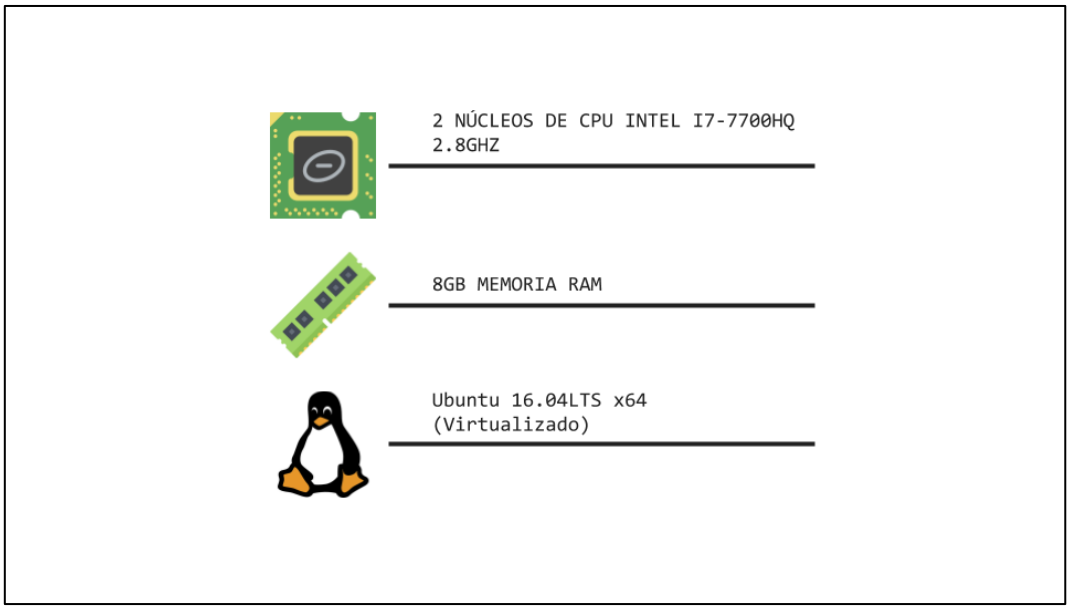


Figura 2.3. Recursos del Sistema

2.4 API: Configuración y Uso

Para la configuración de la aplicación podremos utilizar dos vertientes, mediante el uso de Docker o configurándolo nosotros mismos. Empezaremos

primero, mostrando cómo configurar el servicio sin Docker, para poder conocer cómo funciona y qué partes lo componen. Más tarde, veremos cómo hacer uso de Docker, el cual es mucho más sencillo y automatizado. Posteriormente, indicaremos cómo ejecutar una consulta mediante la aplicación de *scrapping*.

2.4.1 Configuración del sistema sin el uso de Docker

- **Base de datos**

El primer paso a realizar será configurar una base de datos en la que hacer persistentes las consultas a la web. Para ello, utilizaremos PostgreSQL 9.5 o superior, y añadiremos el *bundle* de PostGIS, para poder trabajar con las coordenadas geográficas. La instalación de este servicio podría realizarse de manera nativa o mediante el uso de un contenedor Docker de libre acceso en Docker Hub.

Una vez configurado el sistema de base de datos, crearemos una nueva base de datos, en la que haremos persistentes los resultados de las consultas generadas por la API a la Web. En nuestro proyecto la hemos denominado airbnb. Para gestionar esta base de datos, podremos hacer uso de varias herramientas, mediante línea de comandos encontramos psql (herramienta nativa proporcionada por PostgreSQL que nos permite el uso de todas las funcionalidades del gestor). Su uso es algo complejo por lo que optamos por utilizar herramientas gráficas para la gestión de bases de datos, como son pgAdmin o DBeaver.

Para crear el esquema de base de datos que posteriormente utilizará el API, el desarrollador proporciona un script SQL, presente en **postgresql/schema_current.sql**. Para generar el nuevo esquema simplemente deberemos de ejecutar este fichero, el cual cuenta con todas las llamadas a funciones para crear y relacionar el esquema de datos. Previamente a la ejecución del script, deberíamos de contar con:

- Servidor de base de datos Postgres+PostGIS (Servidor DB)
- Base de datos airbnb creada en el servidor DB
- En caso de utilizar herramientas gráficas, crear conexión entre el equipo y la base de datos.

Una vez creada y configurada la base de datos, pasaremos a cumplimentar el fichero **root.config**. el cual cuenta con la configuración de la ejecución de la aplicación de Scrapping. En caso de no encontrarlo en la ruta especificada, generamos una copia de **example.config** y la renombramos a **root.config**. Utilizaremos el fichero *root*, debido a que toda la ejecución del proceso se realizará con este usuario; en caso de utilizar otro, deberemos de generar el fichero **<usuario>.config**.

- **Fichero Configuración**

Para completar el fichero de configuración deberíamos de centrarnos en los siguientes aspectos básicos: configuración de la conexión con nuestra base de datos, y la conexión entre el sistema y la plataforma Airbnb.

```
[DATABASE]
# db_host =
# db_port =
# db_name =
# db_user =
# db_password =
```

```
[NETWORK]
# api_key =
# client_session_id =
```

Obtener los datos de conexión no debería de ser complejo, ya que conocemos los datos de realizar la conexión entre la base de datos y la herramienta DBeaver o pgAdmin. Bastará con poner:

- **db_host**: ip de la máquina de base de datos.
- **db_port**: de manera predeterminada postgresql utiliza el 5432.
- **db_name**: nombre que le hemos dado a nuestra base de datos (en nuestro ejemplo airbnb).
- **db_user**: si no hemos generado un nuevo usuario, este será postgres.
- **db_password**: contraseña del usuario.

En el caso del segundo conjunto de datos, los datos específicos de este servicio, necesitaremos realizar algunos pasos antes de obtener dichos valores.

Para obtener el API Key, deberemos utilizar cualquier navegador Web de uso cotidiano, preferiblemente Google Chrome o Mozilla Firerox.

1. Dentro del navegador, navegamos hasta la web de airbnb.
2. Seleccionamos un destino turístico aleatorio, no tiene por qué tener relación con las búsquedas realizadas en el estudio.
3. Activamos ver en mapa.
4. Abrimos la consola de Desarrollador del navegador (**F12**).
5. Seleccionamos la opción de Red (*Network*).
6. En la vista de desarrollador pulsamos Limpiar (*Clear*).
7. Seleccionamos XHR, para filtrar los eventos a este tipo.
8. Movemos el mapa para crear una nueva búsqueda.
9. Buscamos un evento denominado **explore_tabs**, y lo abrimos en una nueva pestaña.
10. Dentro de la URL de la nueva pestaña deberemos de buscar dos elementos:
    - ***"key="*** el cual será nuestra *API key*
    - ***"federated_search_session_id="*** que corresponde al *client_session_id*

2.4.2 Configuración del sistema mediante el uso de Docker

La configuración del sistema mediante el uso de la herramienta Docker y Docker Compose simplifica enormemente el proceso, dado que no tendremos que ir a buscar instalables, ni configurar conexiones entre ellos.

Para instalar y configurar el servicio bastaría con:

- **Prerrequisitos**

  Instalación de docker: Para la instalación de docker podremos hacer uso de la documentación oficial presente en:
  - docs.docker.com/install

  Instalación de Docker Compose: Para la instalación de docker compose podremos hacer uso de la documentación oficial presente en:
  - docs.docker.com/compose/install
- **Instalación**

Nota: *Las rutas relativas presentes en este apartado, suponen que el usuario se encuentra ubicado en la carpeta raíz del proyecto clonado de GitHub.*

```
cd docker
docker compose-up
```

Con estos dos simples comandos, el sistema comenzará el proceso de instalación y configuración autónomamente, sin necesidad de acción humana. Una vez concluido, podremos acceder al contenedor principal del proyecto, usando el comando: **docker exec -it airbnbcollector /bin/bash**

Los scripts de Python del proyecto se encuentran en la ruta **/collector**. Así como una configuración lista para su ejecución bajo **/collector/configs**.

Por otro lado, podremos acceder a la consola de la base de datos utilizando, **docker exec -it airbnbcollector-db /bin/bash**. Aunque podremos optar por simplificar la gestión de la misma usando herramientas externas de tipo gráfico. Para poder conocer la dirección IP del contenedor y crear la conexión con la base de datos ejecutaremos: **docker info airbnbcollector-db**.

2.4.3 Configuraciones previas a la ejecución.

El primer paso necesario para comprobar la correcta ejecución del proceso, será comprobar la conexión entre el servicio principal y la base de datos configurada, para ello haremos uso de: **python3 airbnb.py –dbp**

Para generar la primera consulta, previamente añadimos un área de búsqueda a la base de datos, utilizando el mismo comando anterior, pero con nuevas opciones indicadas: **python3 airbnb.py -asa “<Nombre-Ciudad>”**

(por ejemplo: **python3 airbnb.py -asa "Madrid"**). Al ejecutar este script con estas opciones, veremos reflejado en la base de datos un nuevo registro en la tabla **search_area** con el valor de la ciudad indicada.

Generada el área de búsqueda, esta aplicación nos permite generar búsquedas independientes en un mismo área de búsqueda Por ello, antes de ejecutar la consulta crearemos una instancia o *"survey"* , que identifica a cada una de las búsquedas realizadas sobre una misma área de búsqueda, para generar la nueva *"survey"* indicaremos los siguientes parámetros en la ejecución del script: **python3 airbnb.py -asv "Nombre-Ciudad"** (por ejemplo: python3 airbnb.py -asv "Madrid"). Como comentamos, al ejecutar este comando se creará una nueva entrada en la tabla survey, relacionada al área de búsqueda (Ciudad) requerido.

Con estos pasos ya estaríamos listos para lanzar la consulta de viviendas vacacionales, la ejecución de este aspecto se describe a continuación, junto a los diferentes tipos de búsqueda implementados.

2.4.4 Ejecución de una consulta o búsqueda.

En el proyecto podremos encontrar diversos métodos de búsqueda mediante un algoritmo de vecindad (*neighbourhood*), en el cual se buscarían habitaciones asignadas al mismo vecindario.

Un algoritmo de "cajas" (*bounding boxes*) basado en una "caja" generada por las coordenadas geográficas superior izquierda y derecha, las inferiores de un recinto prefijado, el cual iría creciendo paulatinamente recogiendo los resultados presentes dentro de la misma.

El último tipo de búsqueda que nos permite el proyecto es una búsqueda mediante código postal (*zipcode*), en el cual se buscarían aquellos resultados con un determinado código postal.

Siguiendo las postulaciones del desarrollador podríamos centrarnos como usuarios finales en dos de ellos, bounding**-boxes** y **neighbourhood**. El primero considerado es el más probado, por lo que este estudio se basará en él. En el caso de **neighbourhood** según postula el autor, es el más sencillo de configurar de todos, por lo que podríamos empezar por este. Aunque como también indica, puede darse el caso en que Airbnb no asignase vecindarios a la ciudad que estamos buscando, los resultados puede que estén incompletos.

- **Búsqueda por Vecindad (Neighbourhood)**

Por parte de Airbnb, existe una lista de "vecindarios" para determinadas ciudades, esto es lo que se consultará para buscar habitaciones en la búsqueda. Para ejecutar una búsqueda por vecindad ejecutaremos:

```
python3 airbnb.py -s <survey_id>
python3 airbnb.py -s 1
```

Este proceso puede llevar horas. Dado que, como en prácticamente todas las API, Airbnb nos permite un número limitado de peticiones por espacio de tiempo, devolviendo errores HTTP 503. Para ello, el script controla este error y espera regularmente un espacio de tiempo determinado, de manera predeterminada 1 minuto.

- **Búsqueda por Código Postal (Zip Code)**

Para realizar una búsqueda mediante códigos postales:

```
python3 airbnb.py -sz <zipcode>
python3 airbnb.py -sz 38201
```

Para realizar este tipo de búsqueda, previamente deberemos introducir en una nueva tabla de nuestra base de datos, un conjunto de códigos postales. El esquema de esta nueva tabla sería:

```
CREATE TABLE zipcode (
  zipcode character varying(10) NOT NULL,
  search_area_id integer,
  CONSTRAINT z PRIMARY KEY (zipcode),
  CONSTRAINT zipcode_search_area_id_fkey
    FOREIGN KEY (search_area_id)
    REFERENCES search_area (search_area_id)
)
```

- **Búsqueda por Cajas (Bounding box)**

Para realizar una búsqueda mediante cajas:

```
python3 airbnb.py -sb <survey_id>
python3 airbnb.py -sb 1
```

Este tipo de búsqueda realiza una batida geográfica recursiva, dividiendo un cuadro delimitador de una ciudad en porciones más pequeñas, mientras que se identifiquen nuevas listas. Para que el algoritmo identifique estas cajas, previamente deberemos de modificar el área creado agregando los datos geográficos a la ciudad, ya que la aplicación no lo hace automáticamente, para ello haremos uso de una web que nos proporciona el creador: www.mapdevelopers.com/geocode_bounding_box.php

Buscaremos la ciudad que hemos generado escribiendo en la caja de búsqueda, y modificando tras buscarla los valores de la siguiente consulta por los devueltos:

```
UPDATE search_area
SET bb_n_lat = NN.NNN,
bb_s_lat = NN.NNN,
```

```
bb_e_lng = NN.NNN,
bb_w_lng = NN.NNN
WHERE search_area_id = NNN
```

Dentro del fichero <user>.config podremos modificar los valores de las variables **search_max_pages** y **search_max_rectangle_zoom**, el autor del proyecto postula que basado en sus pruebas personales podríamos tomar como valores óptimos iniciales para nuestra ejecución **search_max_pages = 20** y **search_max_rectangle_zoom = 6**.

Capítulo 3.
Captura y análisis de datos

En las próximas líneas explicaremos el proceso de Obtención, procesamiento y análisis de los datos obtenidos mediante el uso de la herramienta configurada en el apartado anterior.

3.1 Captura de datos

Una vez comencemos la ejecución del proceso de extracción de datos, mediante el comando **python3 airbnb.py -sb <survey_id>**, este comenzará a buscar aquellas habitaciones presentes en el cuadrado las coordenadas especificadas inicialmente, y las irá modificando buscando entre las diferentes cajas colindantes, así como el nivel de zoom nuevas habitaciones. A continuación, se muestra una pequeña entrada del log de la aplicación:

```
INFO    Rectangle calculated: [28.05344, -16.52266, 28.04421, -16.53526]
INFO    Searching rectangle: zoom factor = 6, node = [[1, 1], [1, 0], [1, 0], [0, 0],
[1, 0], [0, 0]]
INFO    Page 01 returned 18 listings
INFO    Page 02 returned 18 listings
INFO    Page 03 returned 18 listings
INFO    Page 04 returned 18 listings
INFO    Page 05 returned 18 listings
INFO    Page 06 returned 18 listings
INFO    Page 07 returned 18 listings
INFO    Page 08 returned 18 listings
INFO    Page 09 returned 15 listings
INFO    Final page of listings for this search
INFO    Results: 9 pages, 72 new rooms
```

En el caso de ejemplo, nos muestra que ha encontrado 9 páginas diferentes en las cuales se hallan 72 habitaciones diferentes.

3.2 Conjunto y Carga de Datos

Una vez finalizada la captura de datos o, pulsando CTRL+C o CMD+C, finalizamos manualmente el proceso. Podremos hacer una consulta SQL de la tabla de habitaciones, en las que tendremos el resultado de nuestro estudio, y la cual filtraremos por la última "*survey*" realizada o tomaremos todas las presentes, en nuestro caso solo tomaremos las habitaciones capturadas con la última "*survey*".

Mediante la siguiente consulta revisaremos los datos obtenidos con la *"survey"* ejecutada referente a cada una de las habitaciones presentes en airbnb extraída.

```
SELECT * FROM public.room
WHERE survey_id = 44
ORDER BY room_id ASC, survey_id ASC
```

En nuestro ejemplo, hemos extraído de la zona de Tenerife un número superior a 7700 registros, los cuales comprenderán nuestro estudio de la herramienta, de forma nativa esta API nos devuelve los siguientes campos de información:

Variables	
• room_id • host_id • room_type • country • city • neighborhood • address • reviews • overall_satisfaction • accommodates • bedrooms • bathrooms • price	• deleted • minstay • last_modified • latitude • longitude • survey_id • location • coworker_hosted • extra_host_languages • name • property_type • currency • rate_type

Tabla 3.1. Variables presentes en el modelo

El siguiente paso será generar un *dataset* con la información de esta base de datos y, con el fin de agregar un identificador único a cada fila, haremos uso de una herramienta de ETL (Extracción, Transformación y Carga) Pentaho Data Integration, la cual cuenta con una licencia libre, su funcionamiento es bastante sencillo y cubre abiertamente las necesidades que necesitamos cumplir.

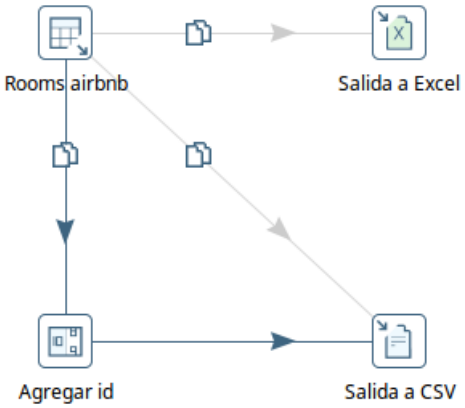


Figura 3.1. Proceso ETL

Ya contamos con un conjunto de datos inicial, con el que poder comenzar a trabajar, pero la información presente en este grupo de datos, es inconsistente y bastante pobre. Con el fin de anonimizar las viviendas, los usuarios no registran todos los datos de localización, por lo que deberemos de tratar toda esta información sin demasiado valor, salvo la latitud y longitud, las cuales utilizaremos para generar información geográfica de manera inversa. Desde una localización, tomaremos información de su entorno.

Para este estudio utilizaremos el lenguaje R, el cual según la documentación oficial se postula como uno de los estándares para el trabajo estadístico y de representación gráfica.

Citando dicho recurso web,

"R es un lenguaje y entorno para computación estadística y gráficos. Es un proyecto GNU que es similar al lenguaje y el entorno S que fue desarrollado en los Laboratorios Bell [...].

R proporciona una amplia variedad de técnicas estadísticas (modelos lineales y no lineales, pruebas estadísticas clásicas, análisis de series de tiempo, clasificación, agrupación, ...) y técnicas gráficas, y es altamente extensible.

Una de las fortalezas de R es la facilidad con la que se pueden producir gráficos de calidad bien diseñados, incluyendo símbolos matemáticos y fórmulas donde sea necesario."[7]

Entendido el método a seguir comenzaremos a trabajar en un script de R, en el cual agregaremos sobre los datos iniciales mayor información referente a la zona geográfica en la que se encuentra la habitación en cuestión, y poder cruzarla con otro tipo de datos interesantes en un futuro.

Lo primero será cargar el script en una instancia de RStudio, y para ello haremos uso del stack data-science: Docker4Science.[8] Desarrollada en colaboración con los profesores Marcos Colebrook, Carlos J. Pérez González y José L. Roda García, y con la participación de Pedro González Yanes. Una vez instalada y configurada la instancia, abriremos nuestro servicio RStudio.

3.3 Análisis de Datos

Antes de comenzar a realizar trabajos sobre los datos, realizaremos un análisis previo y algunas transformaciones genéricas. En el proceso de extracción de información de AirBnb, hemos logrado capturar un número cercano a **46.800 registros de habitaciones**, pero de diferentes zonas en varias consultas independientes.

Con el uso de esta Aplicación de Web Scraping hemos podido obtener un total de **28 campos de información** diferentes:

```
> rooms <- read.table("rooms.csv", sep=";",
                      header=TRUE,quote="\"", comment.char = "",
                      allowEscapes=TRUE,stringsAsFactors=FALSE,
                      encoding = "UTF-8")
> str(rooms)
'data.frame':     46772 obs. of  28 variables:
 $ room_id             : chr  "18628" "19864" "24805" "24836" ...
 $ host_id             : chr  "71597" "74966" "101471" "101653" ...
 $ room_type           : chr  "Entire home/apt" "Entire home/apt" "Entire
home/apt" "Entire home/apt" ...
 $ country             : chr  "" "" "" "" ...
 $ city                : chr  "" "" "" "" ...
 $ neighborhood        : chr  "" "" "" "" ...
 $ address             : chr  "" "" "" "" ...
 $ reviews             : int  39 84 2 43 7 101 148 0 125 58 ...
 $ overall_satisfaction: num  4.5 4.5 NA 5 5 4.5 4.5 NA 4.5 5 ...
 $ accommodates        : chr  "2" "2" "3" "4" ...
 $ bedrooms            : num  0 0 0 2 2 1 1 1 1 2 ...
 $ bathrooms           : num  1 1 1 1 2 1 1 0 2 2 ...
 $ price               : num  60 58 66 138 157 30 69 179 35 144 ...
 $ deleted             : chr  "0" "0" "0" "0" ...
 $ minstay             : logi  NA NA NA NA NA NA ...
 $ last_modified       : chr  "2018/06/02 11:40:04.756774000" "2018/06/02
21:04:14.718286000" "2018/06/02 15:54:25.451506000" "2018/06/02
13:28:03.093574000" ...
 $ latitude            : chr  "40.424715" "40.413418" "40.422022" "40.419951"
...
 $ longitude           : num  -3.7 -3.71 -3.7 -3.7 -3.69 ...
 $ survey_id           : chr  "1" "1" "1" "1" ...
 $ location            : chr
"0101000020E61000005AF10D85CF960DC0718FA50F5D364440"
"0101000020E6100000758F6CAE9AA70DC0F3C98AE1EA344440"
"0101000020E6100000691B7FA2B2A10DC0A41820D104364440"
```

```
"0101000020E61000001FD61BB5C2940DC0147651F4C0354440" ...
 $ coworker_hosted     : int  NA NA NA NA NA NA NA NA NA NA ...
 $ extra_host_languages: logi  NA NA NA NA NA NA ...
 $ name                : chr  "Greta Studio Wifi Chueca en Madrid" "PLAZA
MAYOR  (wifi, air conditioning)" "Gran Via Studio Madrid" "Select the Madrid
more \"cool\"." ...
 $ property_type       : logi  NA NA NA NA NA NA ...
 $ currency            : chr  "USD" "USD" "USD" "USD" ...
 $ rate_type           : chr  "nightly" "nightly" "nightly" "nightly" ...
 $ id_row              : int  1 2 3 4 5 6 7 8 9 10 ...
 $ price_categ         : Factor w/ 4 levels "(0,50]","(50,100]",..: 2 2 2 3 3 1
2 3 1 3 ...
```

La primera fase de análisis comprenderá un proceso de "*cleanising*" o limpieza de datos. En primer lugar, modificaremos el formato de los valores numéricos y eliminando espacios de la variable **room_type**:

```
rooms$bedrooms<-as.numeric(gsub(",",".",rooms$bedrooms))
rooms$bathrooms<-as.numeric(gsub(",",".",rooms$bathrooms))
rooms$price<-as.numeric(gsub(",",".",rooms$price))
rooms$overall_satisfaction<-
as.numeric(gsub(",",".",rooms$overall_satisfaction))
rooms$room_type<-trimws(rooms$room_type)
```

Una vez realizada la primera de las fases de limpieza de datos, analizaremos los diferentes valores que toman las variables de datos que creemos más interesantes para el estudio.

Desde el punto de vista del tipo de habitación, podríamos determinar que la gran mayoría de las habitaciones de la muestra son del tipo ***Entire home/apt***.

```
> table(rooms$room_type)
      5     Entire home/apt   Private room      Shared room    USD
7     1     37020             9172              445            127
```

Con la satisfacción general, podemos conocer como es el tipo de votaciones que se realizan y también determinar que en los lugares estudiados este tipo de servicios gozan de una gran popularidad.

```
> table(rooms$overall_satisfaction)

    1   1.5     2   2.5     3   3.5     4   4.5     5
    1     3     8    26    58   356  1837 11459 15610
```

Con el estudio de la variable *reviews* vemos como la gran mayoría de las habitaciones se concentran en los valores más bajos. Esto podría traducirse en que son poco visitadas o muy nuevas, dado que la plataforma

requiere de una valoración (*review*) de cada huésped una vez concluye la estancia.

```
> table(rooms$reviews)

0     1     2     3     4     5     6     7     8     9    10    11
10243 4134  2879  2033  1817  1495  1274  1148  976   941  833   765
[...]
69    70    71    72    73    74    75    76    77    78    79    80
78    73    63    64    63    62    63    68    83    64    43    67
[...]
184   185   186   187   188   189   190   191   192   193   194   195
9     9     6     12    7     5     8     9     13    9     10    10
[...]
438   439   440   448   450   451   467   472   473   482   506   513
1     1     1     1     1     1     1     1     1     1     1     1
```

En la categoría de precios, vemos la diversidad de precios presentes en la muestra original de datos. Existe una habitación desde los 7 €/noche hasta los 2.254 €/noche.

```
> table(rooms$price)

7    9   10   11   12   13   14   15   16   17   18   19   20   21   22
1    7   24   96   97   102  103  134  163  74   289  298  237  342  432
[...]
116  117  118  119  120  121  122  123  124  125  126  127  128  129  130
153  171  212  126  23   341  40   131  24   38   19   91   49   199  31
[...]
1406 1411 1447 1471 1488 1529 1564 1695 1757 1764 1817 1866 1924 1948
2    2    1    2    1    1    1    1    1    1    1    1    1    1
2254
1
```

Para homogeneizar esta información, crearemos una nueva variable para agrupar habitaciones con precios similares. De esta manera podremos dividir las habitaciones en grupos de precios. Además, nos permite ver mucho más intuitivamente cuál es la moda entre los distintos grupos de precios de nuestro *dataframe*, en este caso son aquellos alquileres desde los 50 hasta los 100 €/noche, por lo que podríamos decir que el enfoque de esta aplicación es ofertar estancias de bajo coste.

```
cutpoints <- c(0, 50, 100, 200, Inf)
rooms$price_categ <- cut(rooms$price, breaks = cutpoints)
table(rooms$price_categ)

   (0,50]  (50,100] (100,200] (200,Inf]
    16836     20603      7190      2008
```

Posteriormente quitaremos algunos registros nulos que podrían “ensuciar” el estudio, y filtramos el estudio para solo aquellas 3 búsquedas más actuales sobre la zona de Santa Cruz de Tenerife. Para ello:

```
sapply(rooms, function(x) sum(is.na(x))/length(x))*100  # missing values in
variables
sum(is.na(rooms$host_id) | is.na(rooms$room_id))        # without id
sum(rooms$survey_id %in% c("2","3","4","5","7","44"))   # Canarias
# View(subset(rooms,survey_id=="5" ))  #S/C. Tenerife
# View(subset(rooms,survey_id=="7" ))  #S/C. Tenerife
# View(subset(rooms,survey_id=="44" ))  #S/C. Tenerife
tmp_rooms <- subset(rooms,!(is.na(rooms["host_id"]) | is.na(rooms["room_id"])))
c_rooms <- tmp_rooms[tmp_rooms$room_type!="5,0",]
c_rooms <-
tmp_rooms[tmp_rooms$survey_id=="7"|tmp_rooms$survey_id=="5"|tmp_rooms$survey_id
=="44",]
```

Ahora tenemos en **c_rooms** el listado de aquellas habitaciones con los filtros y formatos adecuados. Ahora añadiremos un nuevo *dataframe*, para guardar nueva información geográfica, la cual no nos proporciona Airbnb.

```
#### geo
data <- data.frame(room_id=as.double(),
                   host_id=as.double(),
                   town=character(),
                   stringsAsFactors=FALSE)
library(dplyr)
c_rooms_to_lookup<- c_rooms %>%
  group_by(room_id,latitude,longitude) %>%
  summarize() %>% arrange(room_id) %>% as.data.frame()

subset(c_rooms_to_lookup,room_id==35651)

library(jsonlite)

for(i in <-inicio->:<-final->){
  #url<-"https://maps.googleapis.com/maps/api/geocode/json?latlng=28.0305421,-
16.6126199&key=AIzaSyB7U6q3LB_i4XntKoEoKf05be_9W0Ozigc"
  url<-
paste("https://nominatim.openstreetmap.org/reverse.php?format=json&lat=",c_room
s$latitude[i],"&lon=",c_rooms$longitude[i],"&zoom=18",sep="")

  data_json<-fromJSON(url)
  data<-rbind(data,data.frame(room_id=c_rooms_to_lookup$room_id[i],
                              town=ifelse(!is.null(data_json$address$city),
data_json$address$city,
                                          ifelse(!is.null(data_json$address$tow
n), data_json$address$town,
                                                 data_json$address$village
                                          ))
  ))
}
```

```
View(data)
```

Para este apartado hacemos uso de una API externa gratuita proporcionada por Open Street Maps, aunque con limitaciones por uso. Por ello el proceso de carga de Territorios, se ha realizado en diversos días, para no colapsar el volumen completo de peticiones al servicio.

En cada paso lo que obtenemos es el municipio (ciudad) de cada una de las habitaciones, basándonos en las coordenadas X e Y con las que sí que cuentan nuestros datos de entrada del *scraping*.

```
> str(data)
'data.frame':    2758 obs. of  2 variables:
 $ room_id: Factor w/ 2656 levels "10006584","10013405",..: 1 2 3 4 5 6 7 8 9
10 ...
 $ town   : Factor w/ 33 levels "San Juan de la Rambla",..: 1 1 2 2 3 3 4 4 5 2
...
```

Ahora haremos un primer uso de esta nueva dimensión, con la cual sacaremos aquellas 5 ciudades con mayor número de habitaciones, y aquellas con menos:

```
towns<- data %>% group_by(town) %>% summarize() %>% as.data.frame()
rooms_towns <- merge(data,rooms,by.x="room_id",by.y="room_id",all.x=TRUE)
#towns2<-merge(towns,rooms,by.x="room_id",by.y="room_id",all.x=TRUE)
towns_count <- data.frame(table(rooms_towns$town))
towns_count2 <- towns_count[c(-33,-32),]
###
#View(towns_count2)
top_n(towns_count2,5,towns_count2$Freq) # TOP
                       Var1 Freq
1          Puerto de la Cruz  516
2                      Arona  625
3 San Cristóbal de La Laguna  449
4     Santa Cruz de Tenerife  583
5                      Adeje  840

top_n(towns_count2,-5,towns_count2$Freq) # BOTTOM
                     Var1 Freq
1    Buenavista del Norte   32
2               Los Silos   30
3 La Victoria de Acentejo    7
4                  Fasnia   21
5                Vilaflor   10
```

Una vez limpiados y formateados, estos datos podrán ser procesados para su análisis. El primero de los pasos a tener en cuenta será hacer una representación gráfica de los municipios y la distribución de habitaciones en ellos.

Capítulo 4. Explotación de los datos obtenidos en el estudio

En las secciones de este capítulo, realizaremos diferentes representaciones gráficas: mapas coropléticos, histogramas, etc. Posteriormente generaremos un modelo predictivo y analizaremos cuan acertado es el modelo, dada la estructura de datos actual.

4.1 Mapas Coropléticos: Representación del número de habitaciones por municipio

Para este apartado haremos uso de varias librerías de R:

```
library(sf)
library(tidyverse)
```

Posteriormente, una vez cargadas estas dos librerías, cargaremos los datos de entrada geográficos de los municipios. Directamente de un fichero GeoJson, que cuenta con la información relativa a los municipios de la isla.

```
mapas <- st_read("municipios.geojson")
str(mapas)
Classes 'sf' and 'data.frame':     31 obs. of  9 variables:
 $ municipio    : Factor w/ 31 levels "Adeje","Arafo",..: 17 30 29 7 18 23 6 21
24 15 ...
 $ municipio_cod: Factor w/ 31 levels "38001","38004",..: 14 29 27 19 15 17 6
18 20 13 ...
 $ area_gis     : num  1.02e+08 2.66e+07 3.00e+07 3.93e+07 1.39e+07 ...
 $ nombre       : Factor w/ 31 levels "Adeje","Arafo",..: 17 30 29 7 18 23 6 21
24 15 ...
 $ perim_gis    : num  98082 23783 30172 36243 24233 ...
 $ superficie   : num  10245 2664 3002 3930 1389 ...
 $ id           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ poblacion    : num  153474 11110 23799 17272 8827 ...
 $ geometry     :sfc_MULTIPOLYGON of length 31; first list element: List of 1
  ..$ :List of 2
  .. ..$ : num [1:9576, 1:2] -16.3 -16.3 -16.3 -16.3 -16.3 ...
  .. ..$ : num [1:1904, 1:2] -16.3 -16.3 -16.3 -16.3 -16.3 ...
  ..- attr(*, "class")= chr  "XY" "MULTIPOLYGON" "sfg"
 - attr(*, "sf_column")= chr "geometry"
 - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA
NA NA NA
```

```
  ..- attr(*, "names")= chr  "municipio" "municipio_cod" "area_gis" "nombre"
...
```

Como vemos en el Geojson existen varios campos interesantes, como puede ser la población, por lo que vamos a representar su contenido como primer paso a realizar. Para ello, usamos la librería ggplot.

```
ggplot()+geom_sf(data = mapas)
```

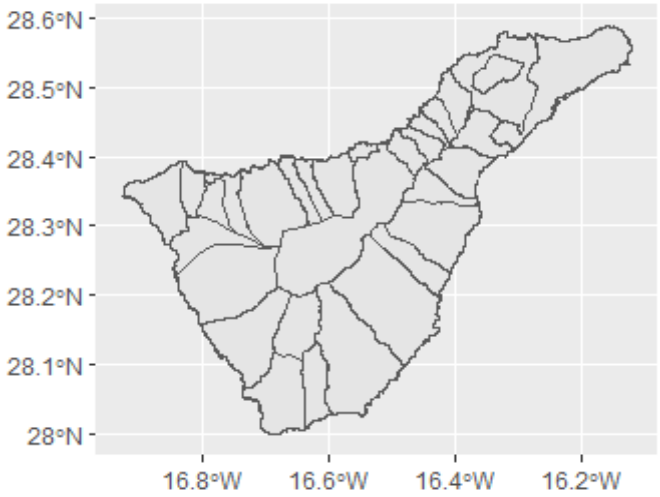


Figura 4.1. Mapa coroplético de la isla de Tenerife inicial

Una vez dibujado el mapa de la isla por municipios, usando los datos geográficos agregados a los datos, utilizaremos dicha información para responder a alguna de las preguntas que nos hemos planteado.

Lo primero que mostraremos es la distribución de habitaciones a lo largo del Territorio estudiado, apareciendo la zona sur de Tenerife (Adeje y Arona) y la zona Metropolitana (Santa Cruz de Tenerife) como municipios con mayor número de casos.

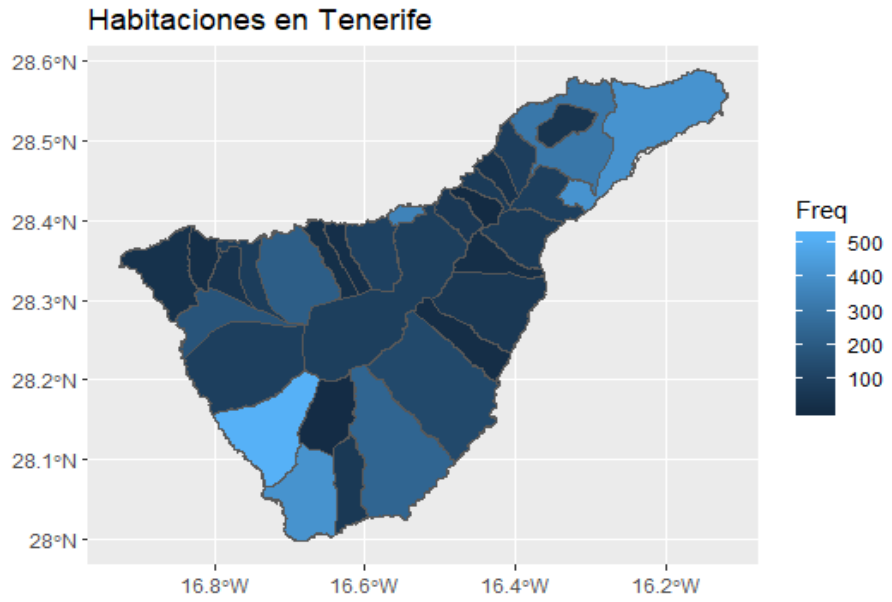


Figura 4.2. Habitaciones por Municipio

Para seleccionar los filtros nos hemos basado en estas dos gráficas:

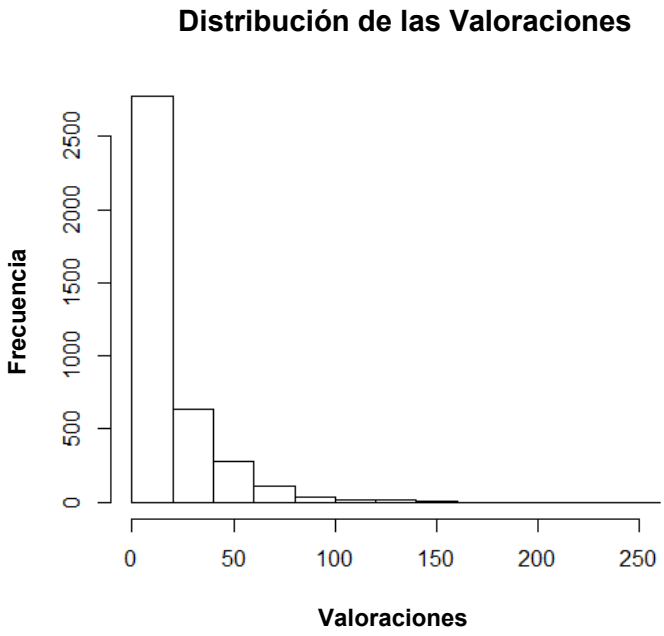


Figura 4.3. Distribución del número de Valoraciones

De este gráfico podemos descubrir que la gran mayoría de las habitaciones cuentan con entre 0 y 50 valoraciones u opiniones, también podríamos visualizarlo utilizando algunas funciones de R:

```
> cutpoints <- c(0, 25, 50, 100, Inf)
> rooms_towns1$reviews.max_categ <- cut(rooms_towns1$reviews.max, breaks =
```

```
cutpoints)
> table(rooms_towns1$reviews.max_categ)

   (0,25]   (25,50]  (50,100] (100,Inf]
     2438       609       256        44
```

Ahora usaremos la variable número de huéspedes para ver la distribución de las habitaciones según este campo.

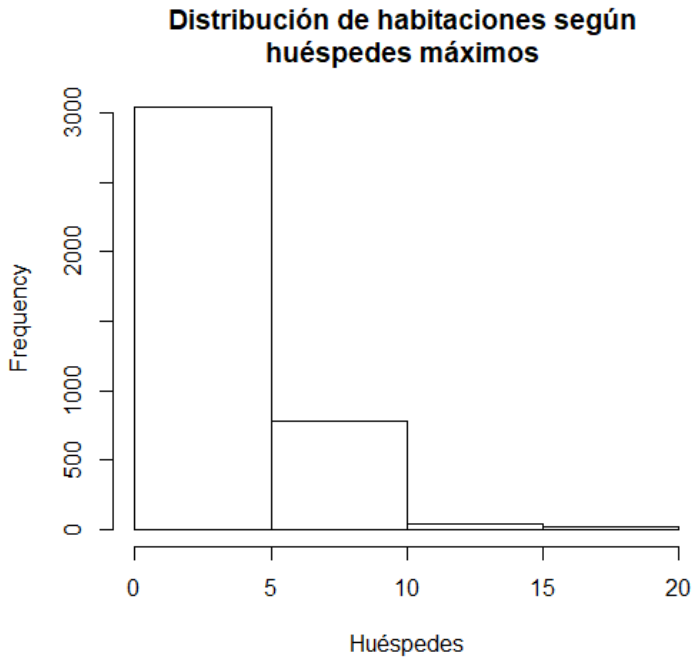


Figura 4.4. Distribución de habitaciones según huéspedes máximos

Para concretar más usaremos directamente los métodos y funciones que nos proporciona nativamente R:

```
> cutpoints <- c(0, 1, 2, 3, 4, Inf)
> rooms_towns1$accomodates.max_categ <-
cut(as.numeric(rooms_towns1$accomodates.max), breaks = cutpoints)
> table(rooms_towns1$accomodates.max_categ)

  (0,1]   (1,2]   (2,3]   (3,4] (4,Inf]
     90     752     511    1348    1180
```

Como vemos la plataforma, al menos para la zona que estudiamos (Tenerife), se centra en hospedajes para grupos de 4 personas en su mayoría.

Una vez comprobado cómo se distribuyen los datos según estas dos categorías, pasaremos a mostrar cómo se distribuyen por municipios de la

isla de Tenerife los alquileres vacacionales de airbnb, según algunos criterios: Número de Huéspedes máximos, Calificaciones, Precio, etc. Los datos serán categorizados en función de un gradiente de color Azul, donde cuanto más claro existirá un mayor número de casos en dicho municipio.

4.1.1 Menos de 50 Opiniones y más de 3 estrellas

Este primer estudio se centra en utilizar menos de 50 reviews, para que aparezcan el mayor número de habitaciones y que su puntuación no se vea afectada por la media que la genera. Como podemos apreciar los municipios con mayor número de viviendas vacacionales son las zonas turísticas del sur de la isla, destacando Adeje como municipio con Mayor densidad. Por otro lado, también aparece un alto número de casos en Santa Cruz de Tenerife.

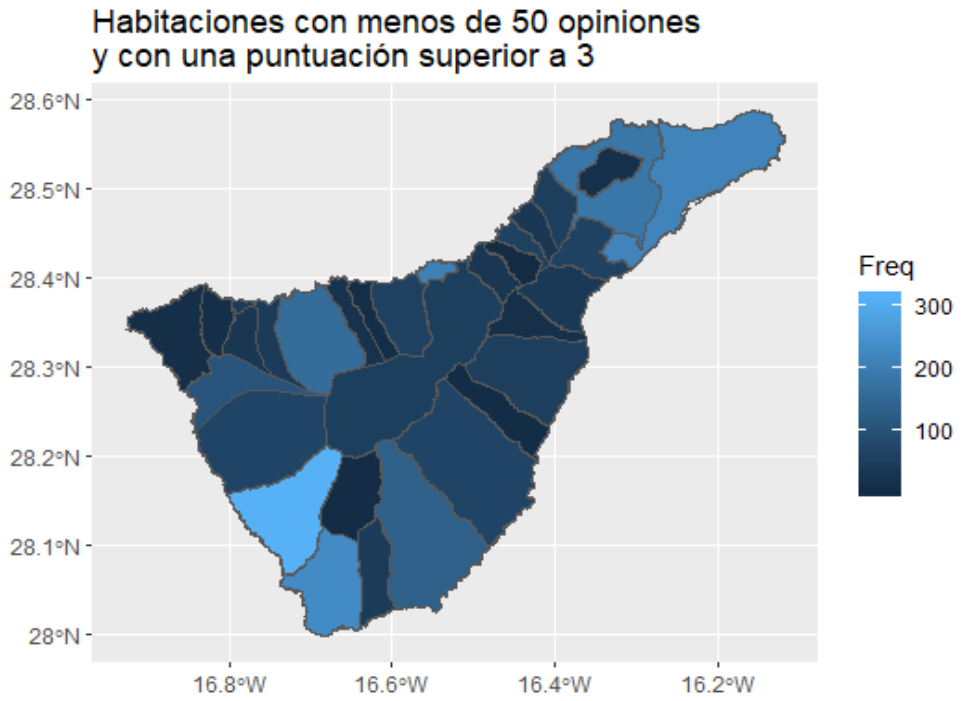


Figura 4.5. Distribución de habitaciones con menos de 50 valoraciones y puntuación media superior a 3

4.1.2 Habitaciones según su precio

- **Habitaciones con precio inferior a 50 €/noche**

Los alquileres Vacacionales más baratos se concentran en los municipios de Adeje mayoritariamente, aunque en segundo lugar tendríamos a los municipios de Arona, San Cristóbal de la Laguna, o Santa Cruz de Tenerife.

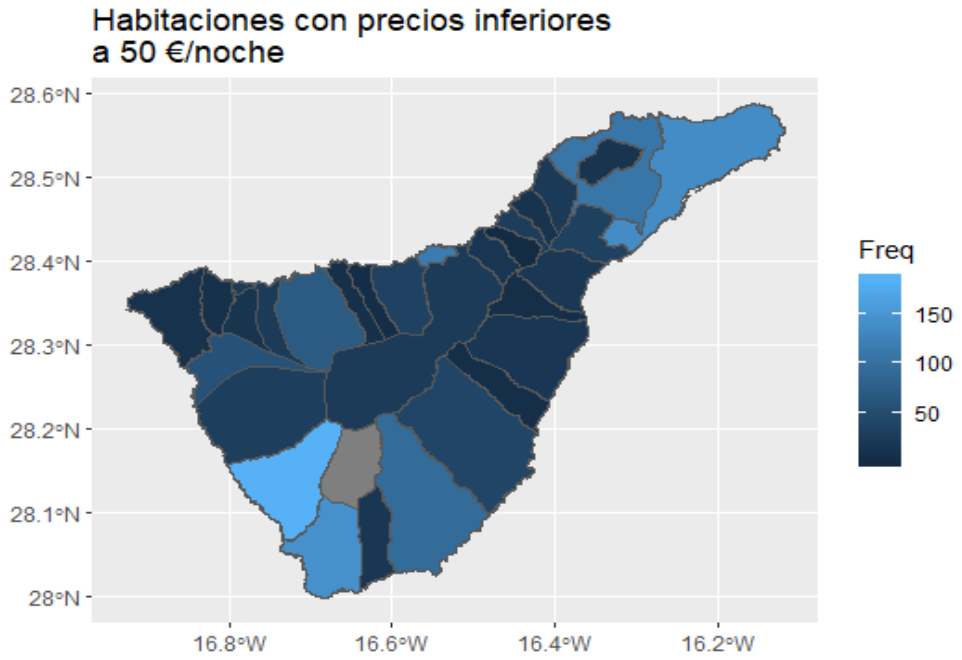


Figura 4.6. Distribución de habitaciones con precios por debajo de 50€/noche

- **<u>Habitaciones entre 50 y 100 €/noche</u>**

Para este caso hemos encontrado que los municipios que aparecen con un mayor número de habitaciones son los del Sur de Tenerife (Adeje y Arona) y la zona metropolitana.

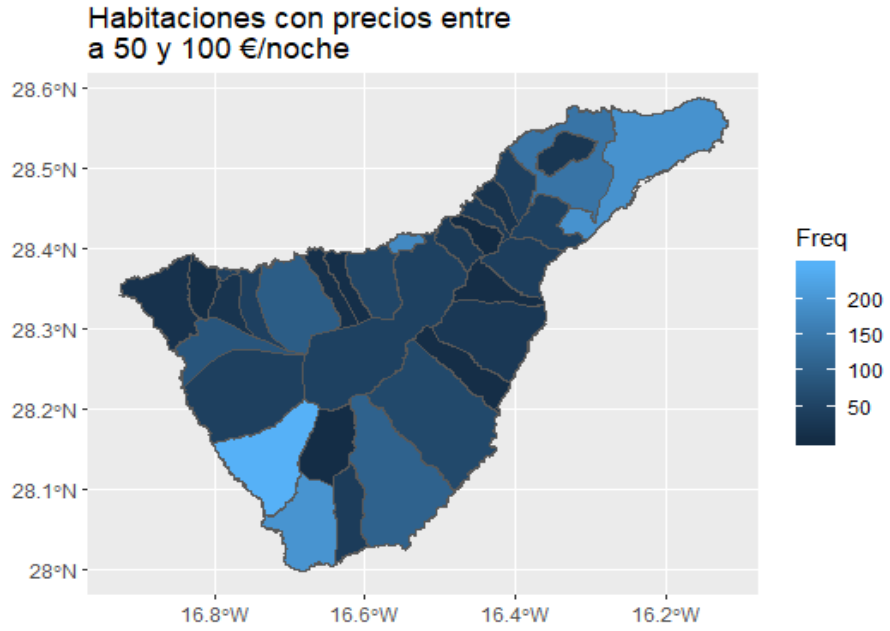


Figura 4.7. Distribución de habitaciones con precios entre 50 y 100 €/noche

- **<u>Habitaciones entre 100 y 200 €/noche</u>**

En cuanto a los datos mostrados, vemos que los alquileres vacacionales más caros (fuera de los casos extremos) se encuentran en Santa Cruz de Tenerife, Arona y Adeje.

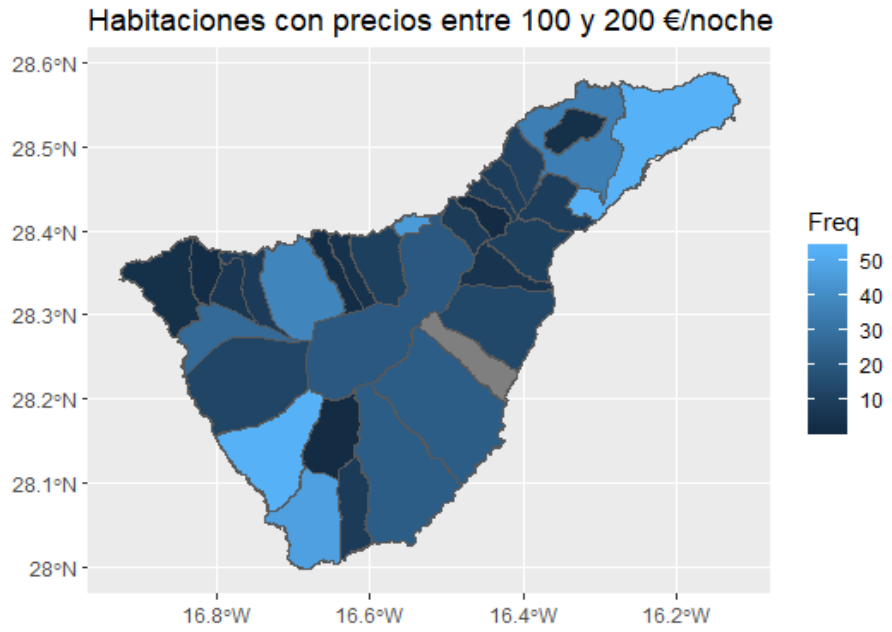


Figura 4.8. Distribución de habitaciones con precios entre 50 y 100 €/noche

Este estudio viene influenciado por la diferencia en el número de viviendas por municipio, dando como municipios con mayor número de habitaciones el sur de Tenerife (Adeje y Arona) y la zona metropolitana.

Aunque hemos descubierto un caso en el municipio de Icod de los Vinos, situado en la zona noroeste de la isla, en el que se ve aumentado el número de Habitaciones de (100,200] frente al resto de casos de estudio, sin ser un número notable, pero si se ve modificado el tono en que se pinta en el mapa. Otros aspectos son que no existen habitaciones con precios superiores a 100 €/noche para el municipio de Fasnia o con precios inferiores a 50€/noche para el municipio de Vilaflor.

4.1.3 Mejores y peores habitaciones de Tenerife según las votaciones de los usuarios de AirBnb

Para determinar este conjunto hemos determinado como nota de corte 3/5, de las cuales hemos determinado que habitaciones con nota superior a 4/5 serían las mejores e inferior a 3/5 las peores. Con ello pretendemos sacar el municipio con mejores/peores recomendaciones por los usuarios de AirBnb.

En el caso de las mejores habitaciones, dado el volumen de habitaciones, nos aparece Adeje como ganador del estudio seguido por los municipios de Arona, Santa Cruz de Tenerife y Puerto de la Cruz.

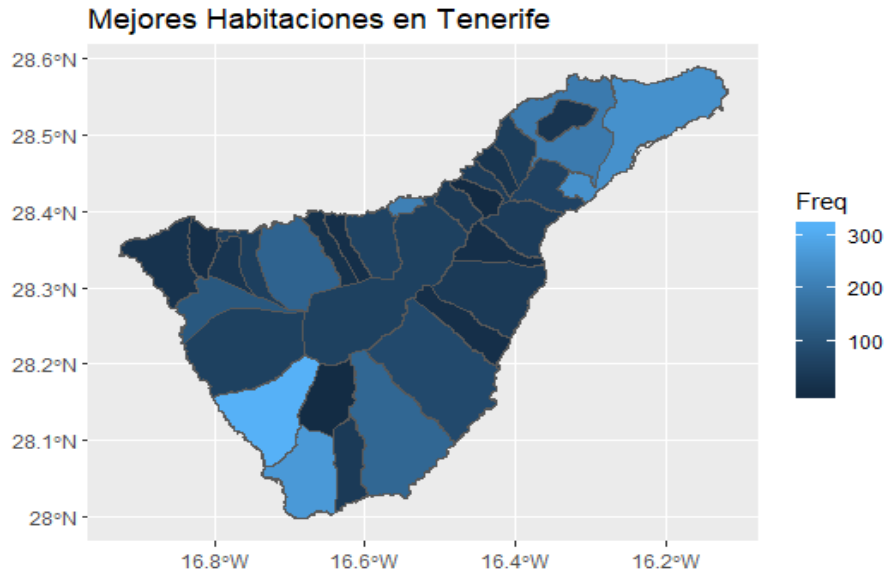


Figura 4.9. Distribución de habitaciones con mejor valoración

En cuanto a las peores habitaciones encontramos que se encuentran 4 habitaciones en municipios diferentes: Adeje, Arona, Los Realejos y Tegueste. Aunque estas no cuentan con la peor valoración de la plataforma, simplemente son las peores de la isla siguiendo lo estipulado previamente (notas inferiores a 3/5)

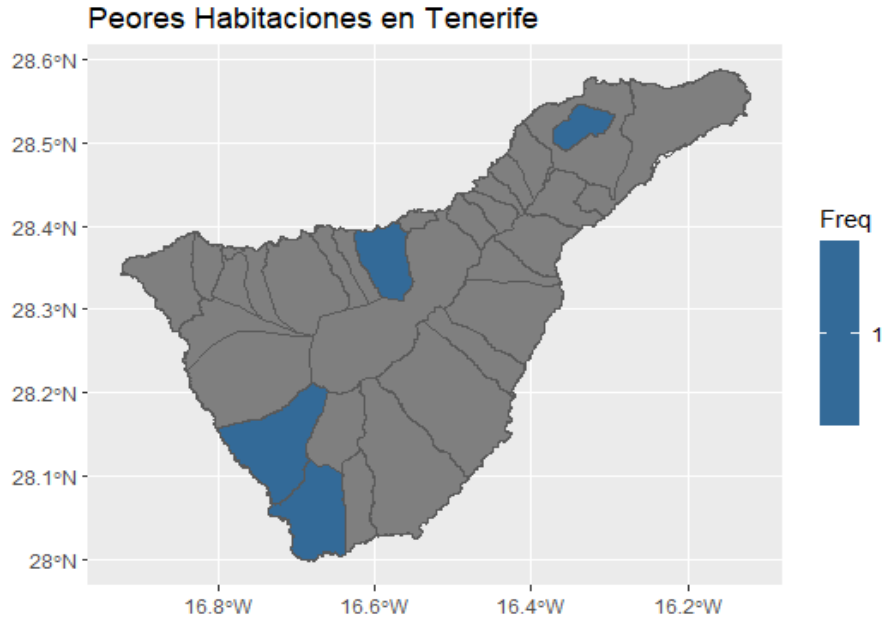


Figura 4.10. Distribución de habitaciones con peor valoración

Aunque no existe ninguna habitación en la isla con la puntuación más baja de la plataforma, por lo que podríamos decir que el servicio en general no es malo.

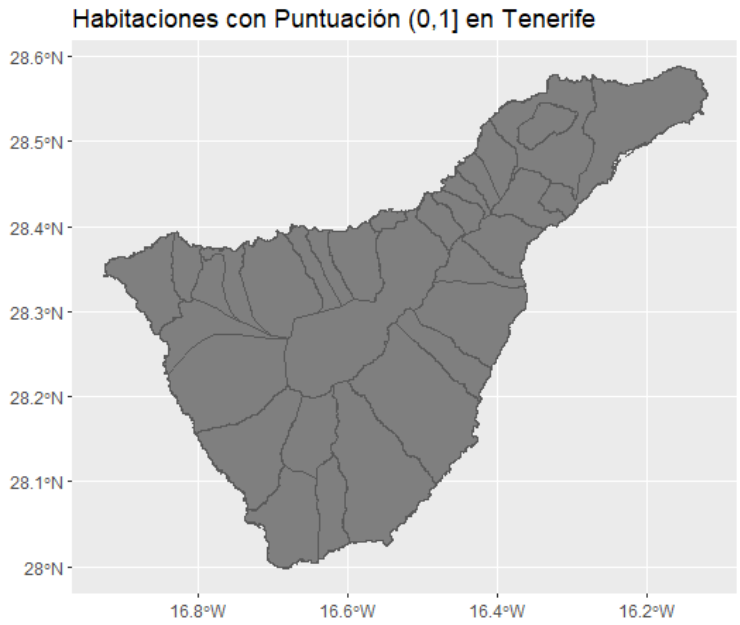


Figura 4.11. Distribución de habitaciones con valoración entre 0 y 1

4.1.4 Habitaciones basándonos en el número de Huéspedes máximos

Otra visión del estudio es la de buscar aquellas habitaciones que se ajusten a un tipo de viaje, según el número de integrantes. Por ello hemos estudiado aquellos viajes en tres grupos, siempre filtrando aquellas habitaciones por encima de la media 3/5:

- **Viajes en solitario**

Para aquellos viajeros a la isla que viajen solos, según el estudio realizado, el mayor número de habitaciones con ocupación máxima de una persona, sería en el municipio de Arona, además de poder descartar otros municipios donde no existe disponibilidad de habitaciones con esa característica.

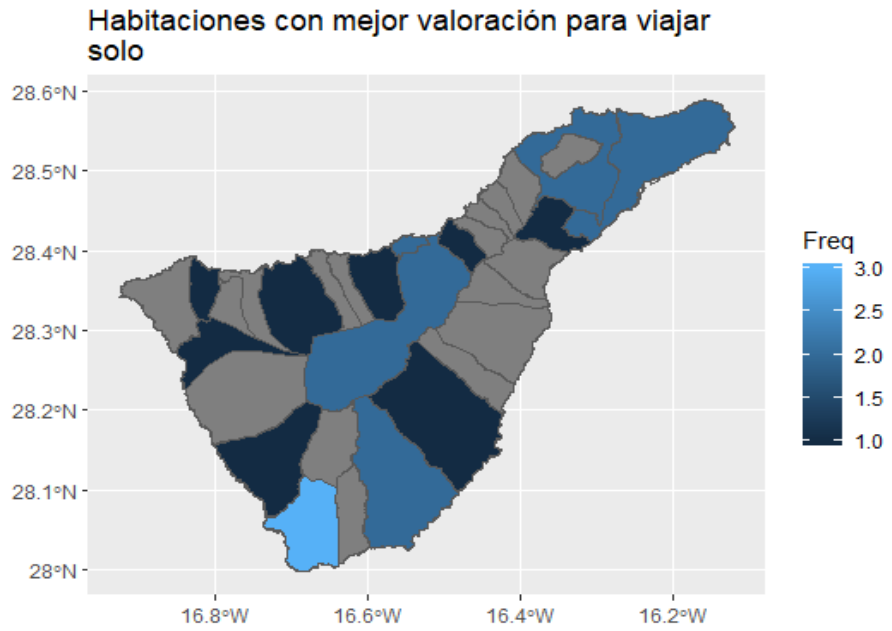


Figura 4.12. Distribución de las mejores habitaciones para viajar solo

- **<u>En Pareja</u>**

Para los viajes en pareja existen nuevas opciones, entre las que destacan: Puerto de la Cruz, San Cristóbal de la Laguna y Adeje. En segundo plano existen otros municipios, como son: Icod de los Vinos y Santa Cruz de Tenerife.

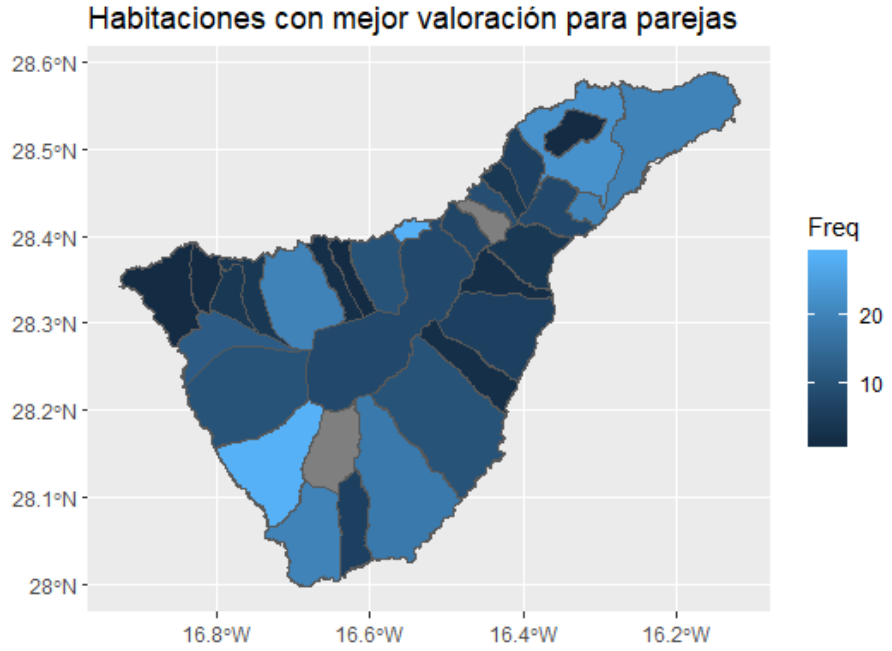


Figura 4.13. Distribución de las mejores habitaciones para viajar en pareja

- **Con Amigos**

Para viajar con amigos, grupos de 3 o más personas, podemos comprobar cómo la zona sur empieza a tomar fuerza: Adeje y Arona. Santa Cruz de Tenerife también tiene presencia junto a San Cristóbal de la Laguna.

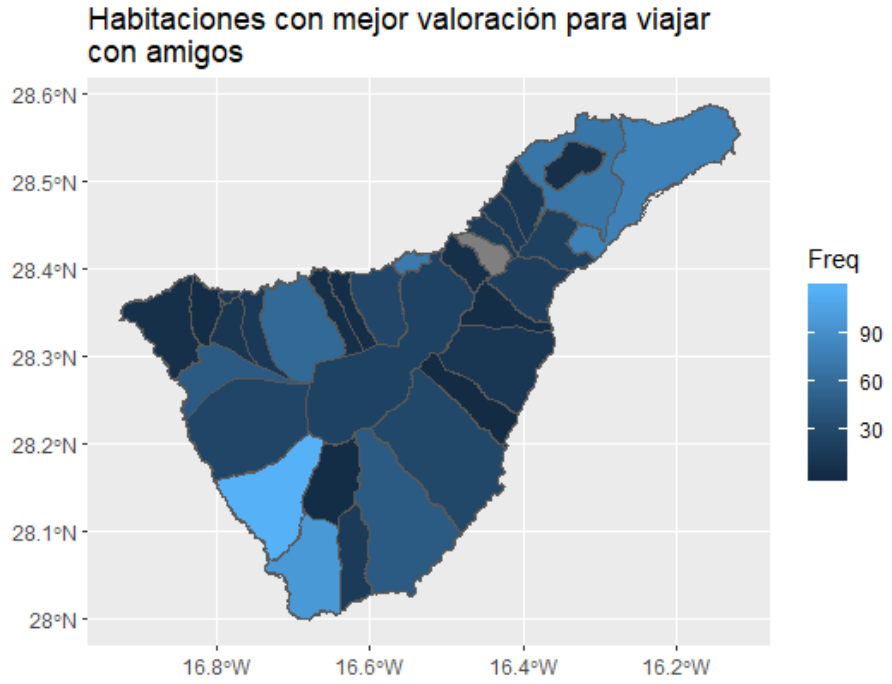


Figura 4.14. Distribución de las mejores habitaciones para viajar en grupo con amigos

4.2 Gráficos de relación entre variables.

Otro estudio interesante basado en nuestros datos podría ser la relación que existe entre las diferentes variables presentes en el *dataset,* respecto al precio de las habitaciones.

4.2.1 Relación entre habitaciones y baños

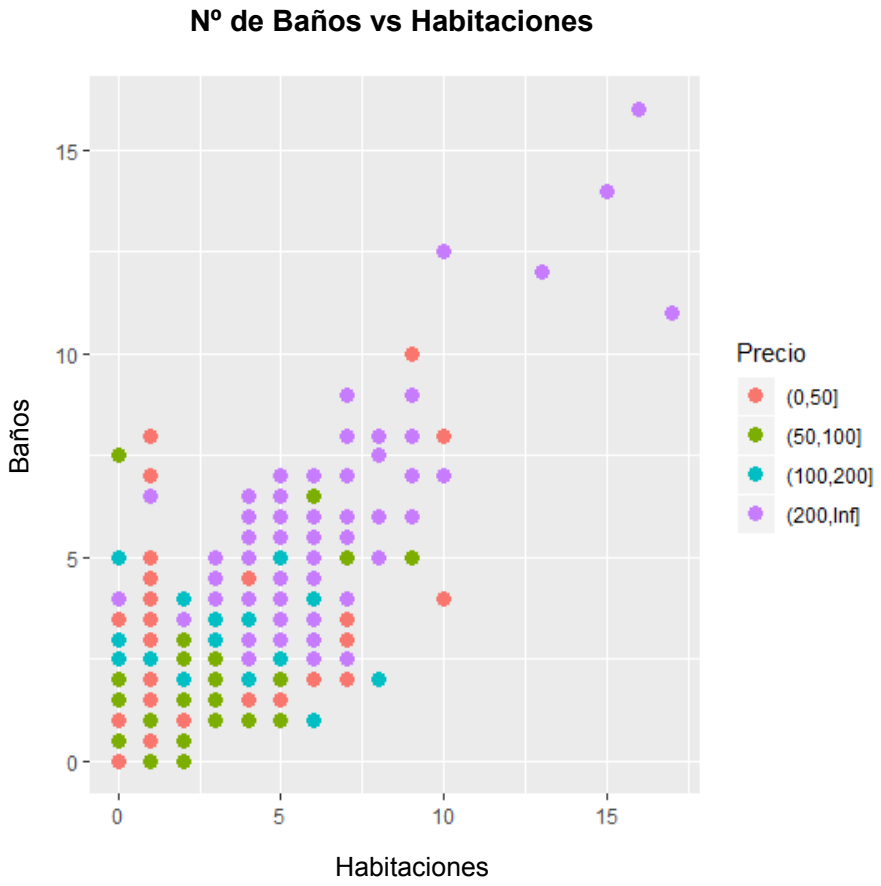


Figura 4.15. Relación entre el número de baños y habitaciones

El primero de los estudios realizados será comprobar cuál es la relación entre el número de habitaciones y baños con respecto al precio que tiene una habitación.

Como se muestra en el gráfico anterior, aun existiendo algunos casos extremos, podemos comprobar como la gran mayoría de los alojamientos se concentran en menos de 10 habitaciones y menos de 10 baños, y dada su forma tenemos que el número de baños aumenta según aumenta el número de habitaciones hasta llegar a las 5-10 baños por cada 5-10 habitaciones, que se estabiliza el crecimiento. También aparecen fuera de este rango algunos alojamientos de lujo, previsiblemente villas o caseríos privados de la isla cuyos dueños alquilan.

También nos sorprende encontrar habitaciones de precios inferiores a 50 €/noche, con más de 10 habitaciones. Esto puede llevar a confusión, ya que este tipo de casos suele tratarse de establecimientos del tipo *"Bed & Breakfast"* o pensiones, muy populares en este tipo de plataformas, aunque no sea su nicho de mercado.

4.2.2 Relación entre habitaciones y huéspedes máximos.

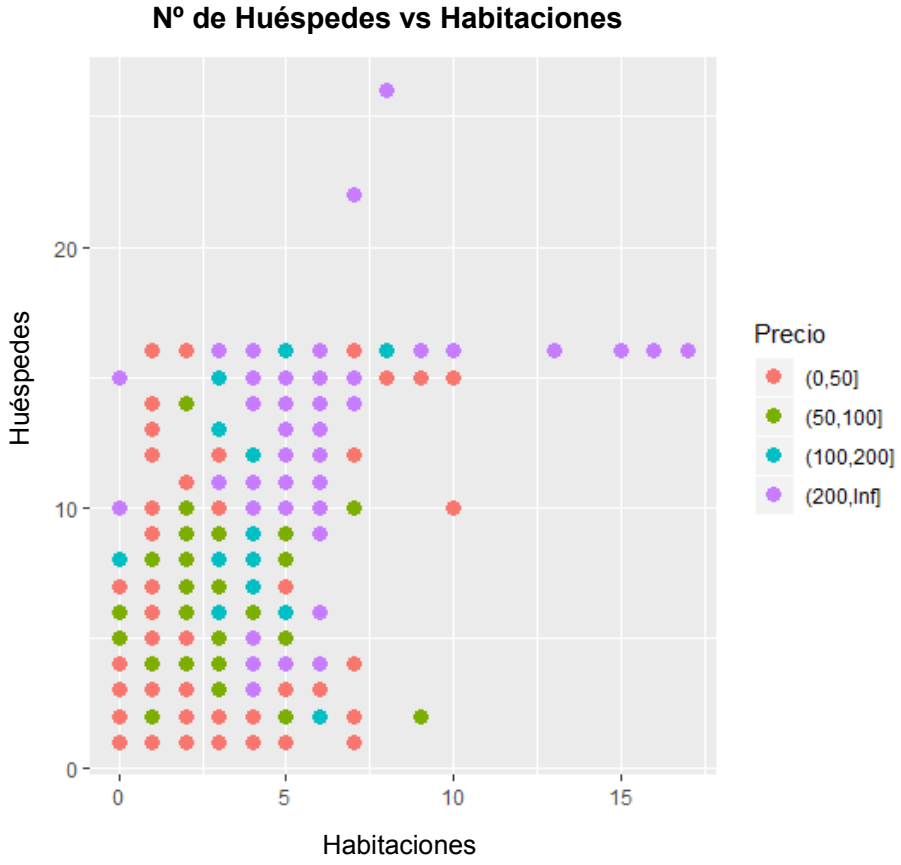


Figura 4.16. Relación entre el número de huéspedes y habitaciones

Aquí podemos comprobar como los rangos de precio más bajos están destinados a habitaciones con menos huéspedes y habitaciones. Existen como comentamos anteriormente, pensiones en la plataforma, dado que existen habitaciones de precios muy bajos con más de 10 huéspedes y menos de 5 habitaciones. A su vez podemos comprobar como también podríamos ver algunos casos de residencias o villas de lujo, con más de 15 habitaciones para cerca de 20 huéspedes.

4.2.3 Relación entre Tipo de Habitación y Precio

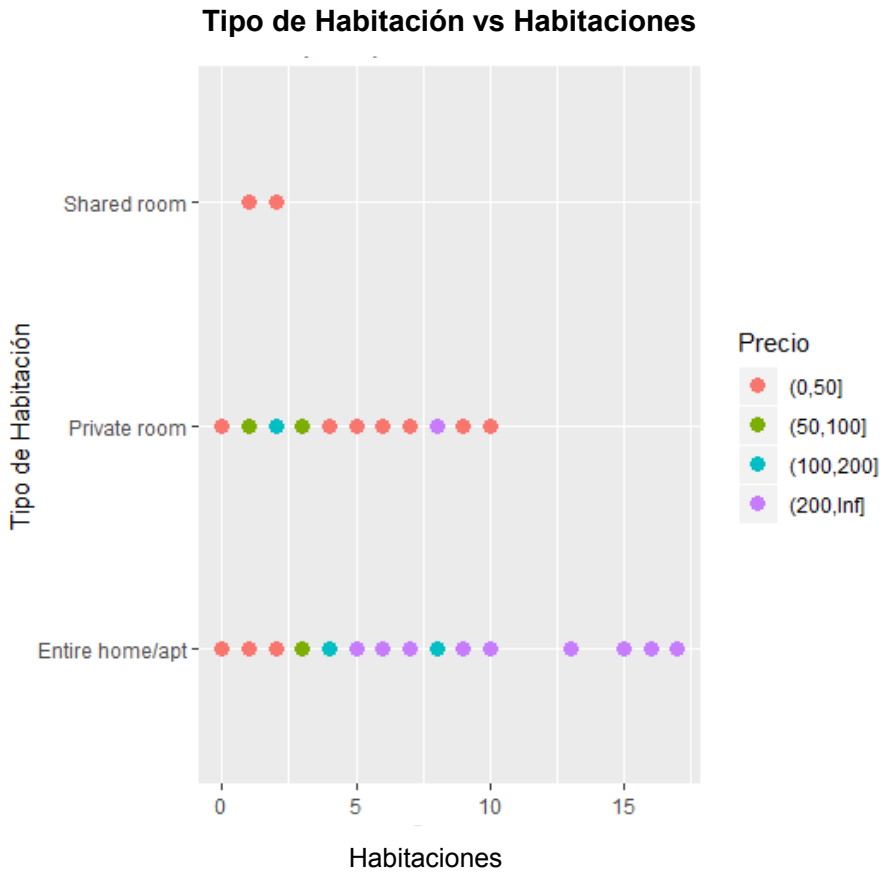


Figura 4.17. Relación entre el tipo de Habitación y habitaciones

Aquí podemos ver una distribución por tipo de habitación y precios, según el número de habitaciones disponibles. De esta representación podríamos obtener que: la mayoría de los alojamientos con precios superior a 200€/noche se tratan de apartamentos o residencias completas para el huésped. Y por contra, aquellos con precios inferiores a 50 €/noche, se encuentran en el grupo de habitaciones privadas o compartidas, aunque existan algunos casos de apartamentos o residencias completas.

4.3 Modelos Predictivos

En cuanto a los modelos predictivos, hemos generado dos modelos para medir cuán acertados son según su gráfica. Ambos son modelos de regresión lineal, de tal forma que se pretende calcular el precio de una habitación en función del resto de las variables del dataset:

```
Y = price
X_1,..,X_n=room_type, reviews, overall_satisfaction, accommodates, bedrooms,
bathrooms
```

O lo que es lo mismo:

```
Y_est = b0 + b1*room_type + b2*reviews + ..... + b_n*bathrooms
Y_real = Y_est + residual
```

De esta manera generamos dos modelos: el primero de ellos se trata de un modelo básico en el que se calcula el precio directamente en función del resto de variables.

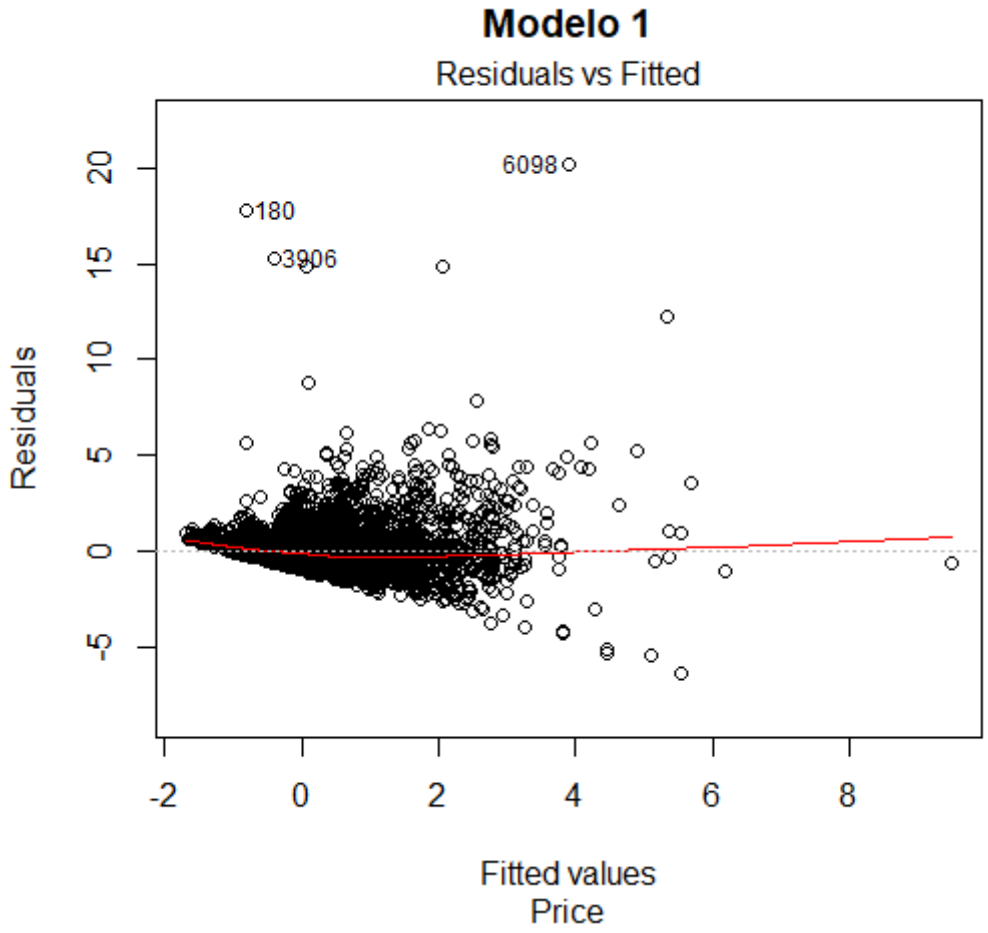


Figura 4.18. Modelo Predictivo 1

De este modelo querríamos conocer el error de predicción (**Y_est**), frente al valor real del caso (**Y_real**). Para ello haremos uso de dos indicadores como son:

Error Cuadrático Medio (MSE)

```
mse.lm <- function(lm_model) summary(lm_model)$sigma ^ 2
```

Error Porcentual Absoluto Medio (MAPE)

```
mape.lm <- function(lm_model) 100*sum(abs(residuals(lm_model) /
lm_model$model[,1]))/length(lm_model$model[,1])
```

Para este modelo estos indicadores arrojan los siguientes datos:

Modelo 1	
MSE	0.5752547
MAPE	213.0076 %

Tabla 4.1. Error Modelo 1

El segundo de los modelos es también lineal. Mediante el uso de una función logarítmica, intenta ajustar la predicción de dicha variable en función del resto de ellas. Esta es una de las técnicas más básicas de ajuste.

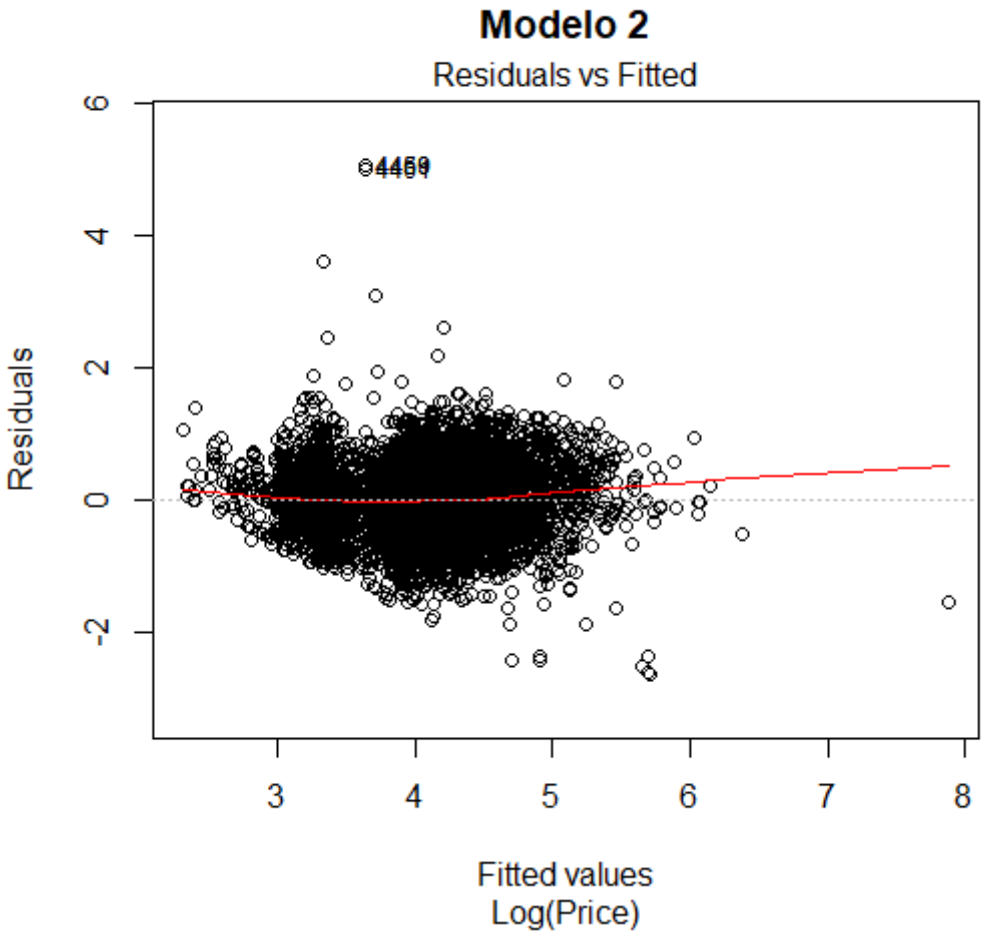


Figura 4.19. Modelo Predictivo 2

Como podemos apreciar en este segundo modelo, la tendencia es mucho más uniforme que en el primero Para comprobar que esto es cierto, calcularemos los valores de los indicadores anteriores y compararemos ambos resultados.

	Modelo 1	**Modelo 2**
MSE	0.5752547	0.1870816
MAPE	213 %	8 %

Tabla 4.2. Error Modelo 2

Fijándonos en el valor del MSE, vemos como este valor es inferior en el segundo modelo, por lo que el error ha disminuido. Por ello, determinamos que el segundo de los modelos es mejor que el primero.

4.4 Modelo de Clasificación

Cómo último estudio realizaremos un árbol de clasificación para intentar predecir de manera gráfica el grupo de precios de una habitación mediante el resto de variables.

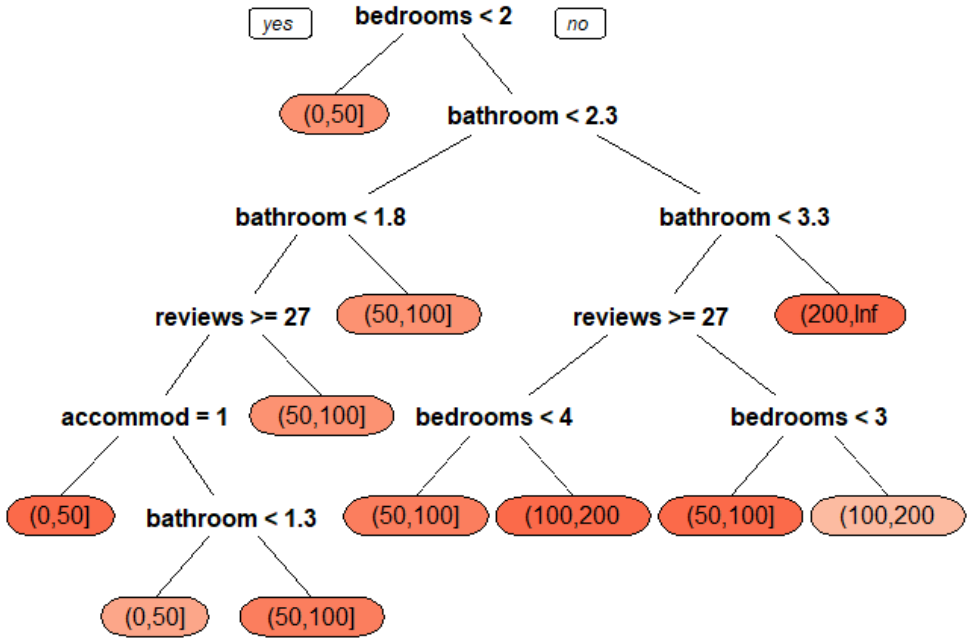


Figura 4.20. Árbol de Clasificación

Del gráfico anterior podríamos ver como el modelo predice el precio de una habitación en función de las otras variables. Cabe destacar que aparezcan variables con valores decimales, cuando no es factible, esto lo realiza internamente el algoritmo, por lo que deberemos utilizar la lógica y realizar un redondeo a la baja.

El siguiente paso será, calcular usando una matriz de confusión, la validez y *accuracy* (exactitud) de nuestro modelo.

```
Confusion Matrix and Statistics

           Reference
Prediction  (0,50] (50,100] (100,200] (200,Inf]
  (0,50]       550      329        54         3
  (50,100]     215      376       175        26
  (100,200]     13       40        61        29
  (200,Inf]      4        8        22        38

Overall Statistics

               Accuracy : 0.5275
```

Vistos los resultados nos aparece que nuestro modelo tiene una cuota de acierto del 52%, por lo cual es un modelo predictivo que aún tiene margen de mejora.

Capítulo 5.
Conclusiones y líneas futuras

En este proyecto se ha realizado un estudio básico del estado del alquiler vacacional en la isla de Tenerife. Un tema de actualidad en algunas ciudades españolas como Madrid o Barcelona, donde los precios de los alquileres generales se han disparado, debido a la rentabilidad del uso vacacional.

Hemos encontrado algunos datos anecdóticos como que no existen hospedajes con la mínima nota de la plataforma, y que hay un número muy bajo de éstos con notas bajas.

Por otro lado, los modelos estadísticos obtenidos a partir de los datos se pueden mejorar con el fin de aumentar su potencia predictiva y de clasificación. Necesitaríamos categorizar muchas más variables que actualmente son de tipo discreto, e intentar mejorar los modelos usando técnicas de análisis de datos. Sin embargo, se puede mencionar que se han alcanzado los objetivos principales del trabajo, y que consistían en aplicar técnicas de *scraping* para la captura de la información de la plataforma y su posterior análisis.

Entre las líneas futuras de trabajo cabría mencionar las siguientes:

- La mejora de los modelos predictivos y de clasificación.
- La ampliación del estudio a otras regiones geográficas con oferta de alojamientos en la plataforma.
- La creación de informes con herramientas de inteligencia de negocio interactivas.

Capítulo 6.
Summary and Conclusions

In this project, a basic study of the state of the holiday rental on the island of Tenerife has been carried out. This is actually a current issue in some Spanish cities such as Madrid or Barcelona, where the prices of general rentals have skyrocketed, due to the profitability of holiday rentals.

We have found some data that show that there are no accommodations with the minimum rating on the platform, and that there are hardly any of these with low grades.

On the other hand, the statistical models obtained from the data can be improved in order to increase their predictive power and classification. We would need to categorize many more variables that currently have a discrete typology, and try to improve the models using data analyses techniques. However, it can be mentioned that the main objectives of the work have been achieved, concretely that which consists in applying scraping techniques for the capture of the information of the platform and its subsequent analysis.

Among the future lines of work, the following should be mentioned:

- *The improvement of predictive and classification models.*
- *The extension of the study to other geographical regions with accommodations on the platform.*
- *The creation of reports with interactive business intelligence tools.*

Capítulo 7. Presupuesto

En cuanto al presupuesto utilizado para este proyecto, hemos conseguido reducir los costes del mismo con el uso únicamente de licencias libres y aplicaciones sin pago por uso. En cuanto al equipo utilizado para el desarrollo se trata de un terminal personal, por lo que tampoco implica coste alguno.

Bibliografía

[1] “Web Scraping” *Wikipedia*, Wikimedia Foundation, 3 Dec. 2018, http://es.wikipedia.org/wiki/Web_scraping.

[2] “Screen Scraping” *Wikipedia*, Wikimedia Foundation, 17 July 2018, http://es.wikipedia.org/wiki/Screen_scraping.

[3] “Alternativas Para Realizar Web Scraping” *Feliciano Borrego*, 9 Aug. 2018, http://felicianoborrego.com/alternativas-para-realizar-web-scraping/.

[4] Baquía, Redacción de. “Cómo Evitar Que Alguien Se Apropie De Los Contenidos De Nuestra Página Web.” *BAQUIA*, 15 Mar. 2016, www.baquia.com/emprendedores/2014-01-26-imperva-seguridad-innovae-como-evitar-que-alguien-se-apropie-de-los-contenidos-de-nuestra-pagina-web.

[5] “Web Scraping: ¿Legal o Ilegal?” *ECIJA*, 13 Sept. 2017, http://ecija.com/web-scraping-legal-ilegal/.

[6] @chris_70736. “HiQ v. LinkedIn and the Legality of Web Scraping - Knowmad Law.” *Medium*, Medium, 24 Apr. 2018, https://medium.com/@chris_70736/hiq-v-linkedin-and-the-legality-of-web-scraping-e80b9ab06f1d.

[7] “What Is R?” *R*, http://www.r-project.org/about.html.

[8] Martín-Santana S., Pérez-González C.J., Colebrook M., Roda-García J.L., González-Yanes P. (2019) “Deploying a Scalable Data Science Environment Using Docker”. In: García Márquez F., Lev B. (eds) *Data Science and Digital Business*. Springer, Cham.

[9] AirBnb, plataforma de listado y alquiler de alojamientos, https://www.airbnb.es