Università Cattolica Sacro Cuore

Statistical and Actuarial Sciences

mj: Data Business Analytics

End-to-End Real Estate Rental app, a Bayesian Spatial approach wtih INLA

Supervisor:
Dr. Marco DellaVedova

Author:
Niccolò Salvini

Assistant Supervisor:
Dr. Vincenzo Nardelli

AY 2019 / 2020

End-to-End Real Estate Rental app, a Bayesian spatial modelling approach wtih INLA

Niccolò Salvini[1]

date: Last compiled on 13 ottobre, 2020

[1]https://niccolosalvini.netlify.app/

Contents

[1] mailto:structure%7B@webstructure

List of Tables

List of Figures

Chapter 1

Introduction

Main themes:

- General introduction to the problem to solve
- Open Data discussion
- Research Question
- Milan Real Estate Controversies
-
- Why a Bayesian approach

Chapter 2

Scraping

2.1 What is Web Scraping

Web Scraping is a techniques that involves informatics and statistics aimed at extracting data from static or dynamic internet web pages. It can be done automatically and simultaneously as well as by a scheduler that palns its execution at a given time. Due to the absence of API's to call regarding Milan Real Estate rental markets, this practice has to be forcly applied. The reason behind that and personal hope is to open source the scraping API so that in the future further analysis will be available without putting effort in gathering data fresh data. *Content distribution is managed by HTTP protocols, which are a common standard to connect web clients. Web browsers (i.e. one among the others web clients) download the content and parses them making possible to read them_The athomical unit of measurement in scraping is the url, which are the locations where this data exchanges abstractly happens.* [questo in API infra] Data/content in webpages is the most of the times well organized and accessible by urls. This is made possible by the effort put into building both the *website structure* and the *content architecture*. For website structure it is meant the way urls, pointing to webpages, are arranged throughout the website. Website structure constitutes a *first dimension* of hierarchy. Some popular structures examples might regard social-networks where posts can be scrolled down within a single page named the wall. Scrolling down might end due to fewer posts, but the perception is a never-ending webpage associated to a single url. Instead personal profiles are dedicated to a specified unique url and even in profiles posts are allocated into a sub domain and might be scrolled down arranged by time since the day of social subscription. Online newspapers display their articles in the front wall and by accessing to one of them all the related following articles sometimes can be reached by an arrow, pointing right or left. Articles can also be suggested and the more the website is explored the more articles are likely to be seen twice within the same session. It will soon end up into a suggestion loop that it will recursively shows the same contents; recursive structures are popular in newspaper-type websites. Online Retailers as Amazon, based on filters, groups inside a single webpage (i.e. page n° 1) a fixed set of items, having their dedicated personal url attached to them. Furthermore Amazon offers the opportunity to skip to the following page (i.e. page n° 2), searching for another different and fixed set of items and so on until the last page/url. Generally website structures try to reflect both the user expectations with respect to the product on the website and the design expression of the web developer. For these reasons for each websites usage category exists a multitude of content architectures. For each content architectures there are multiple front end languages which are destinated to multiple end users. In the future expectations are tailor made webpages on users with respect to its personal preferences. Moreover web design in scraping plays an important role since the more sophisticated graphical technologies are implied, the harder will be to scrape information.

A *second dimension* of hierarchy is brought by content architecture in the name of the language used for content creation and organization i.e. HTML. HTML stands for Hyper Text Markup Language and ... HTML drives the hierarchy structure that is then generalized to the website structure. According to this point of view the hierarchical website structure is a consequence of the content architecture by means of HTML language (*arborescence*: direction from root to leaves). CSS language stands for Cascading Style Sheets and takes care of

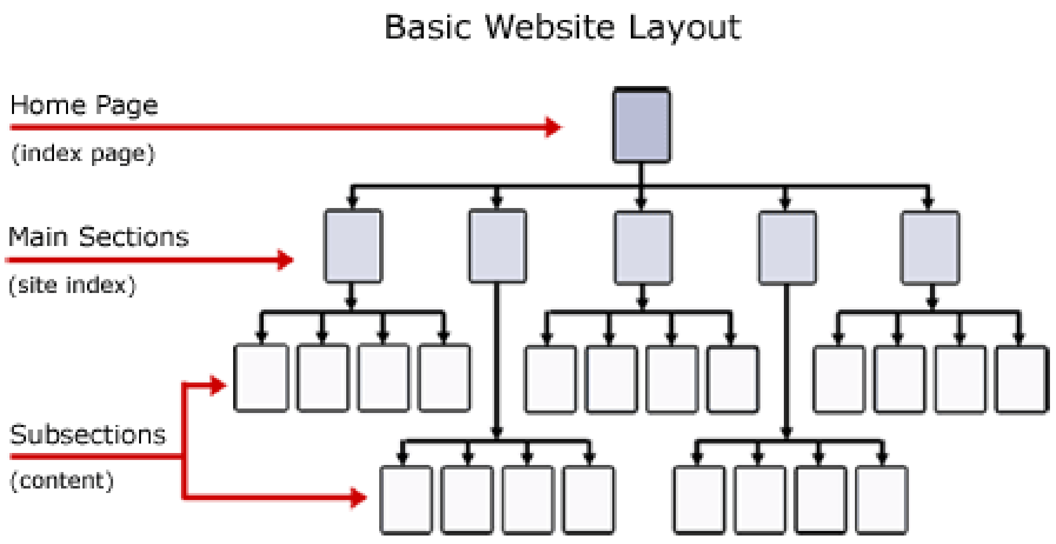


Figure 2.1: general website structure

the style of the webpage. The combination of HTML and CSS offers a wide flexibility in building web sites, once again expressed by the vast amount of different designs on the web. Some websites' components also might be tuned by Javascript language, which in the context of scraping adds a further layer of difficulty. As a matter of fact since Javascript components are dynamic within the webpage, scraping requires specialized libraries to enable different parser to get the content. CSS allows the scraper to target a class of objects in the web page that shares same style (e.g. same css query) so that each element that belongs to the class (i.e. share same style) can be gathered. This practice provides enormous advantages since by CSS a set of objects can be obtained within a single function call. First and Second dimension of the scraping problem imply hierarchy, a simple way to approach the problem is to represent it through already known data structures. One way to imagine hierarchy in both of the two dimensions are graph based data structures named as **Rooted Trees**. By analyzing the first dimension through the lenses of Rooted trees it is possible to compress the whole problem into the general graph based jargon (Diestel, 2006). Rooted trees must start with a root node which is the domain of the web page. Each *Node* is a url destination and each *Edge* is the connection between nodes. Connections have been made possible in the website by nesting urls inside webpages so that within a single webpage the user can access to a number of other related links. Furthermore, as an extension to the rooted tree framework a general graph theory component is introduced, i.e. the *Weight*. Each edge is associated to a weight whose interpretation is the run time cost to walk from a node to its conncted other nodes (e.g. from a url to the other). In addition the content inside each node takes the name of payload, which ultimately is the scope of the scraping processes. The walk from node 17 to node 8 in figure below (even though that is the case of a binary rooted tree) is called path and it represented as an ordered list of nodes connected by edges. In this context each node can have both a fixed and variable outgoing sub-nodes that are called *Children* . When root trees have a fixed set of children are called *k-ary* rooted trees. A node is said to be *Parent* to other nodes when it is connected to them by outgoing edge, in right figure below "head" is the parent of nodes "title" and "meta". Nodes in the tree that shares the same parent node are said *Siblings*, "head" and "body" are siblings in figure @ref(tree_html). Moreover *Subtrees* are a set of nodes and edges comprised of a parent and its descendants e.g. node "body" with all of its descendants might constitute a subtree. The concept of subtree in both of the dimensions plays crucial role in cutting run time scraping processes and fake headers provision (see section 2.3.1). If the website strucuture is locally reproducible and the content architecture within webpages tends to be equal, then functions for a single subtree might be extended to the rest of others subtrees that are siblings to the same parent node. Local reproducibility is a property according to which starting from a single url all the related urls can be inferred from a pattern. Equal content architecture throughout different single links means to have sort of standard shared-within advs criteria that each single rental advertisement has to refer. In addition two more metrics have might better describe the tree: *level* and *height*. The level of a node **L** counts the number of edges on the path from the root node to **L**. The height is the maximum level for any node in the tree, from now on **H**. What is worth to be anticipating is that functions are not going to be applied directly to siblings in the more general rooted tree. Instead it would be better segmenting the highest level rooted tree into a sequence of single subtrees whose roots are the siblings for reasons explained in section 2.3.1.

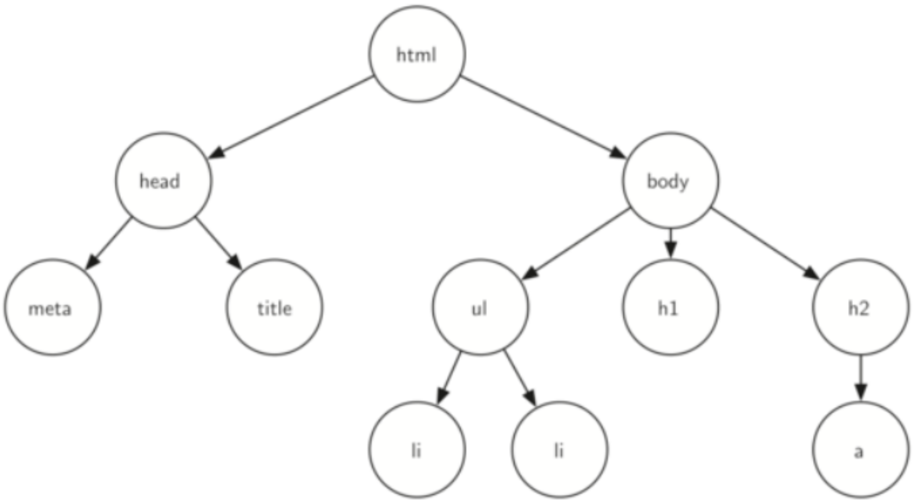


Figure 2.2: html_tree

2.1.1 Immobiliare.it Webscraping website structure\ {@webstructure[1]}

The structure of the website resembles the one encountered for the popular online retailer Amazon. According to the query filters (e.g. number of rooms 5, price less than 600 p.m. etc), the url is shaped so that each further filter added is appended at the end to the domain url `https://www.immobiliare.it/` root node (see figure below). Once filters are all applied to the root domain this constitutes the new url root domain node that might have this appereance: `https://www.immobiliare.it/affitto-case/milano/?criterio=rilevanza&superficieMi` Since this is true only for page n°1 and contains the first 25 advs all the remaining siblings nodes corresponding to next pages have to be generated. Here comes handy the Local reproducibility property introduced in the previous section. The rest of the siblings, i.e. the ones belonging

[1]mailto:structure%7B@webstructure

to page 2 (with the attached 25 advs) and to page 3 can be generated by appending `&pag=n` at the end of the url, where n is the page number reference (from now on referred as *pagination*). For page number 4 the exact url that points to the web page, being equal the filters from above, is `https://www.immobiliare.it/affitto-case/milano/?criterio=rilevanza&superficieM`
Author customary choice is to stop pagination up to 300 pages since spatial data can not be to large due to computational burdens. Moreover thanks to the reverse engineer applied to url building tree height **H** is pruned, so the parsing part is cut short. At this point pagination has generated a list of siblings urls whose children number is fixed (i.e. 25 per page). That makes those trees k-ary, where k is 25 indicating the number of children. K-ary trees with equal content structure shared across siblings allow to design a single function to call that could be mapped to all the other siblings. In addition in order to further disassemble the website and making available the whole set of single rental advertisement links for each page (ranging from 1 to 300) a specific function has been made. As a consequence a single function call `scrape_href()` can grab all the links inside page 1. The function is then iterated along all the previously generated siblings (i.e. pages) obtaining a collection of all the single ads links belonging to the set of pages generated.

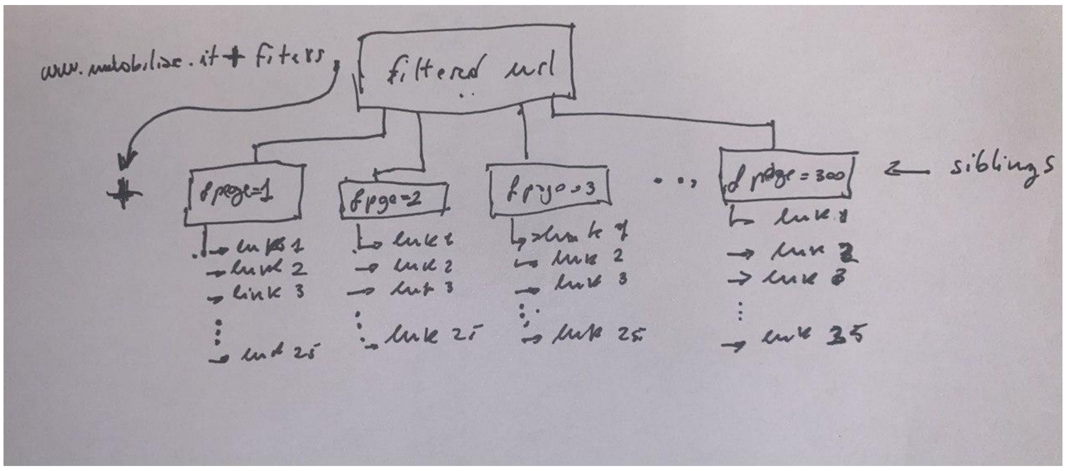


Figure 2.3: website_tree

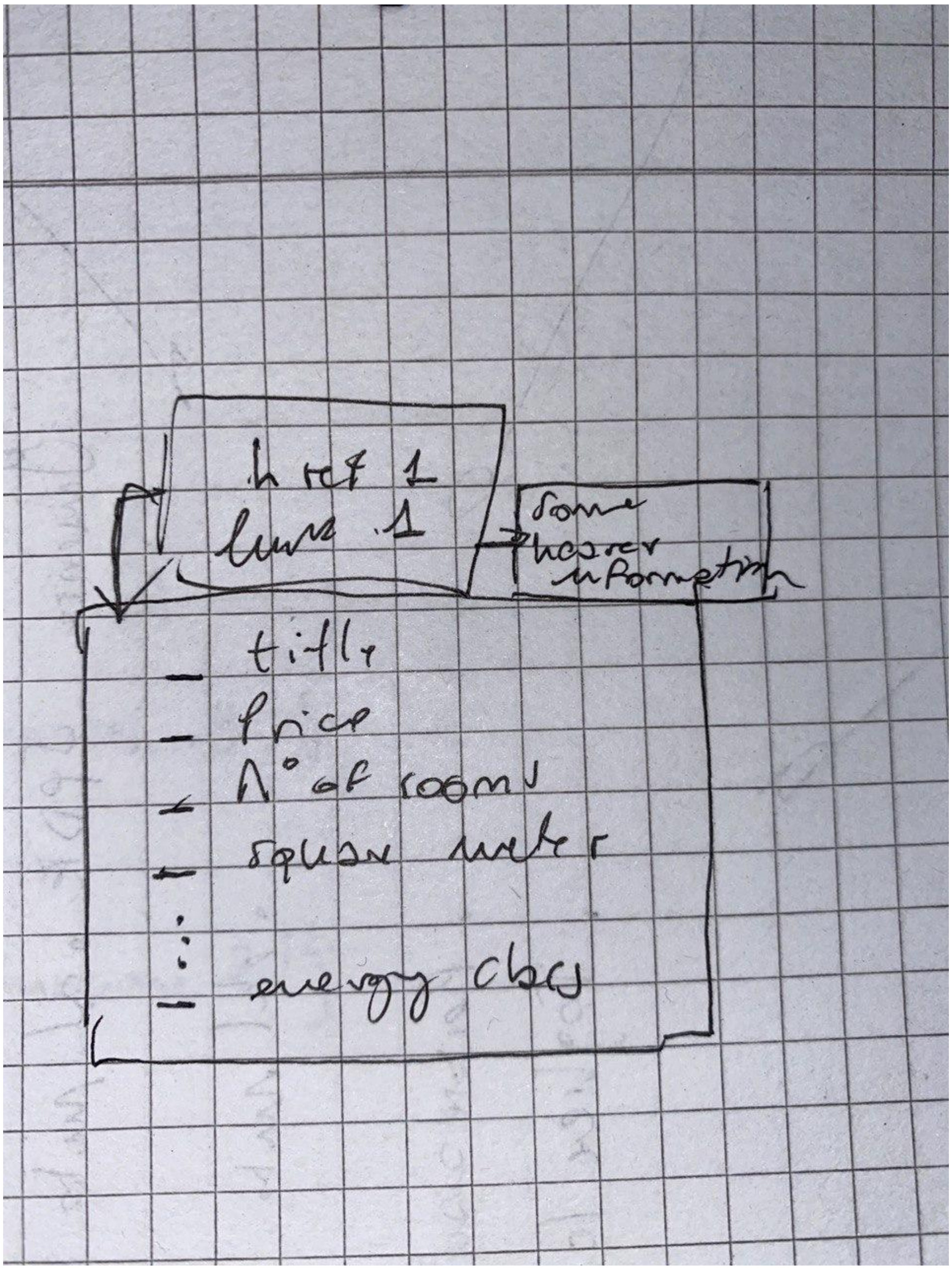


Figure 2.4: children_str

2.1.2 Immobiliare.it Webscraping content architecture with `rvest`

To start a general scraping function it is required a target url (i.e. the filtered root node url). Then a `html_session` nested list object is opened by specifying the url and the request data that the user need to send to the web server (see left part to dashed line image ??). Information to be attached to the web server request will be further explored later, tough they are mainly three: User Agents, emails references and proxy servers. `html_session` objects contains a number of useful information such as: the url, the response, coockies, session times etc. Once the connection is established (request response 200) all the following operations rely on the opened session, in other words for the time being in the session the user will be authorized with the before-provided characteristics through the request. The list object contains mostly the html content of the webpage and that is where data needs to be parsed and collected. The list as said can disclose other interesting metadata that might be interesting but is beyond the scope of the analysis. The workflow below schematize what scraping is doing:

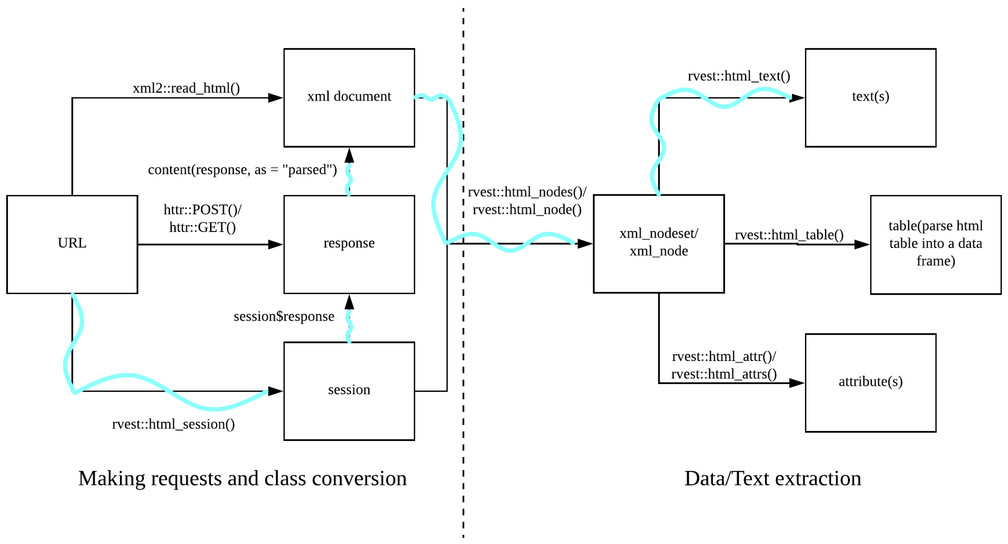

xml2::read_html()
xml document
content(response, as = "parsed")
URL
httr::POST()/ httr::GET()
response
session$response
session
rvest::html_session()
rvest::html_nodes()/ rvest::html_node()
rvest::html_text()
text(s)
xml_nodeset/ xml_node
rvest::html_table()
table(parse html table into a data frame)
rvest::html_attr()/ rvest::html_attrs()
attribute(s)
Making requests and class conversion
Data/Text extraction

Figure 2.5: workflow

In the right part of the dashed vertical line is represented a sequence of `rvest` functions that follow a general step by step text comprehension rules. `rvest` first handles parsing the html content of the web page within the session object `read_html()`, secondly looking for a single node `html_nodes()` through CSS. CSS is the way to tell `rvest` to point to a precise node in the webpage and this is brought by query. Thirdly it converts the content into readable text with `html_text()`. The entire process can be also thought as an autoencoder, where adding decoding layer after layer on top of the starting url, it can be reached the final requested payload data wanted. The reason why the process appear straightforward and simple relies in the blending with `rvest` by the pipe %>% operator. Below it is shown a function that exemplifies scrapping the price.

```
scrapeprice.imm = function(session) {

opensess = read_html(session) price = opensess %>% html_nodes(css
    =".im-mainFeatures__title") %>% html_text() %>% str_trim()

if(is.null(price) || identical(price, character(0))) { price2 =
    opensess %>% html_nodes(css ='.im-features__value ,
    .im-features__title') %>% html_text() %>% str_trim()

if ("prezzo" %in% price2) { pos = match("prezzo",price2)
    return(price2[pos+1]) %>% str_replace_all(c("€"="","\\."="")) %>%
    str_extract( "\\-*\\d+\\.*\\d*") %>% str_replace_na() %>%
    str_replace("NA", "Prezzo Su Richiesta") } else {
    return(NA_character_) } } else { return(price) %>%
    str_replace_all(c("€"="","\\."="")) %>% str_extract(
    "\\-*\\d+\\.*\\d*") %>% str_replace_na() %>% str_replace("NA",
    "Prezzo Su Richiesta")

}
}
```

The function takes as a single argument a session object which is initialized in one other function. Then It reads the inner html content in the session storing the information into an obj called the `opensess`. Another obj is created, namely price, right after the pipe operator a css query into the html is called. The css query `.im-mainFeatures__title` points to a precise data stored in immobiliare web page header, right below the main title. Expectation are that price is a one-element chr vector, containing the price and some other unnecessary characters. Then the algorithm enters into the first `if` statement. The handler checks if the object `price` is empty. If it doesn't the algorithm jumps to the end of the algorithm and returns the cleaned quantity. But If it does it takes again the `opensess` and redirect to a second css query `.im-features__value , .im-features__title` where price, once again in the webpage, could be found. Please note that This is all done within the same session, so no more request additional information has to be sent. Since the latter css query points to data stored inside a list, for the time being the newly created obj price2 is a list containing various information. Then the algorithm flow enters into the second `if` statement that checks whether the `"prezzo"` is matched the list or not, if it does it returns the +1 position index element with respect to the "prezzo" positioning. This happens because data in price2 list are stored by couples sequentially, e.g. [title, "Appartamento Sempione", energy class, "G", "prezzo", 1200/al mese]. When it returns the element corresponding to +1 position index it applies also some data wrangling with `stringr` package to keep out overabundant characters. The function then escapes in the else statement by setting `price2 = NA_Character_` once no css query could be finding the price information. the *character-string* type has to be imposed due to fact that later they can not be bind. In other words if the function is evaluated for a url and returns the price quantity, but then is evaluated for url2 and outputs NA (no character) then results can not be combined into dataframe due to different types.

Once all the functions have been created they need to be called together and then data coming after them need to be combined. This is done by `get,data.catsing()` which at first checks the validity of the url, then takes the same url as input and filters it as a session object. Then simultaneously all the functions are called and then combined. All this happens inside a `foreach` parallel loop called by `scrape.all.info()`

```
scrape.all.info = function(url =
    "https://www.immobiliare.it/affitto-case...",
vedi = FALSE,
scrivi = FALSE,
silent = FALSE){
if (silent) {
start = as_hms(Sys.time()); cat('Starting the process...\n\n')
message('\nThe process has started in',format(start,usetz = TRUE))
}
# open parallel multisession
cl = makeCluster(detectCores()-1) #using max cores - 1 for parallel
    processing
registerDoParallel(cl)
start = as_hms(Sys.time())

if (silent) {
message('\n\nStart all the requests at time:', format(start,usetz =
    T))
}
ALL = foreach(i = seq_along(links),
.packages = lista.pacchetti,
.combine = "bind_rows",
.multicombine = FALSE,
.export ="links" ,
.verbose = TRUE,
.errorhandling='pass') %dopar% {
source("utils.R")
sourceEntireFolder("functions_singolourl")
get.data.catsing = function(singolourl){

# dormi()
#
if(!is_url(singolourl)){
stop(paste0("The following url does not seem either to exist or it is
    invalid", singolourl))
}

session = html_session(singolourl, user_agent(agents[sample(1)]))
if (class(session) == "session") {
session = session$response
}

id = tryCatch({scrapehouse.ID(session)},
error = function(e){ message("some problem occured in
    scrapehouse.ID") })
lat = tryCatch({scrapelat.imm(session)},
error = function(e){ message("some problem occured in scrapelat.imm")
    })
long = tryCatch({scrapelong.imm(session)},
error = function(e){ message("some problem occured in
    scrapelong.imm") })
location = tryCatch({take.address(session)},
error = function(e){ message("some problem occured in take.address")
    })
condom = tryCatch({scrapecondom.imm(session)},
error = function(e){ message("some problem occured in
    scrapecondom.imm") })
buildage = tryCatch({scrapeagebuild.imm(session)},
error = function(e){ message("some problem occured in
    scrapeagebuild.imm") })

...

combine = tibble(ID = id,
LAT = lat,
LONG = long,
LOCATION = location,
CONDOM = condom,
BUILDAGE = buildage,

...


return(combine)
}
stopCluster(cl)
return(ALL)
}
```

The skeleton constitutes a standard format adopted for many other scraping function used in the analysis. Being equal the css query what it changes is the matching term, i.e. “numero camere” instead of “prezzo” to look for how many rooms there are in the house. This is true for all the information contained in the list accessed by the fixed css query. Those that are not they are a few and they do not need to be scraped. In addition some other functions outputs need to undergo to further heavy cleaning steps in order to be usable As a consequence oh that functions need also to be broken down by pieces into many single .R files whose names correspond to each important information. Below it is printed the tree structure folder that composes the main elements of the scraping procedure. It can be notices that the two folders, namely functions_singolourl and functions_url enclose all the single functions that allows to grab single information from session. Folders with a customized function are then sourced within the two main functions, scrape.all and scrape.all.info so data can be extracted.

```
 levelName
1  immobiliare.it-WebScraping
2   ¦--functions_singolourl
3   ¦   ¦--0scrapesqfeetINS.R
4   ¦   ¦--0scrapenroomINS.R
5   ¦   ¦--0scrapepriceINS.R
6   ¦   ¦--0scrapetitleINS.R
7   ¦   ¦--ScrapeAdDate.R
8   ¦   ¦--ScrapeAge.R
9   ¦   ¦--ScrapeAgeBuilding.R
10  ¦   ¦--ScrapeAirConditioning.R
11  ¦   ¦--ScrapeAptChar.R
12  ¦   ¦--ScrapeCatastInfo.R
13  ¦   ¦--ScrapeCompart.R
14  ¦   ¦--ScrapeCondom.R
15  ¦   ¦--ScrapeContr.R
16  ¦   ¦--ScrapeDisp.R
17  ¦   ¦--ScrapeEnClass.R
18  ¦   ¦--ScrapeFloor.R
19  ¦   ¦--ScrapeHasMulti.R
20  ¦   ¦--ScrapeHeating.R
21  ¦   ¦--ScrapeHouseID.R
22  ¦   ¦--ScrapeHouseTxtDes.R
23  ¦   ¦--ScrapeLAT.R
24  ¦   ¦--ScrapeLONG.R
25  ¦   ¦--ScrapeLoweredPrice.R
26  ¦   ¦--ScrapeMetrature.R
27  ¦   ¦--ScrapePhotosNum.R
28  ¦   ¦--ScrapePostAuto.R
29  ¦   ¦--ScrapePropType.R
30  ¦   ¦--ScrapeReaReview.R
```

```
¦   ¦--ScrapeStatus.R
¦   ¦--ScrapeTotPiani.R
¦   ¦--ScrapeType.R
¦   °--take_location.R
¦--scrapeALL.R
¦--scrapeALLINFO.R
¦--functions_url
¦   ¦--ScrapeHREF.R
¦   ¦--ScrapePrice.R
¦   ¦--ScrapePrimaryKey.R
¦   ¦--ScrapeRooms.R
¦   ¦--ScrapeSpace.R
¦   °--ScrapeTitle.R
¦--libs.R
¦--utils.R
¦--README.Rmd
¦--README.md
°--simulations
    ¦--rt_match_vs_forloop.R
    °--runtime_simul.R
```

2.2 Scraping Best Practices and Robottxt

Robots.txt files are (rivedi citation) a way to kindly ask webbots, spiders, crawlers, wanderers and the like to access or not access certain parts of a webpage. The de facto 'standard' never made it beyond a informal "Network Working Group INTERNET DRAFT". Nonetheless, the use of robots.txt files is widespread (e.g. https://en.wikipedia.org/robots.txt, https://www.google.com/robots.txt) and bots from Google, Yahoo and the like will adhere to the rules defined in robots.txt files, although their *interpretation* of those rules might differ.

Robots.txt files are plain text and always found at the root of a website's domain. The syntax of the files in essence follows a fieldname: value scheme with optional preceding user-agent: … lines to indicate the scope of the following rule block. Blocks are separated by blank lines and the omission of a user-agent field (which directly corresponds to the HTTP user-agent field) is seen as referring to all bots. # serves to comment lines and parts of lines. Everything after # until the end of line is regarded a comment. Possible field names are: user-agent, disallow, allow, crawl-delay, sitemap, and host. For further notions (Meissner and Ren, 2020, goo (2020))

Some interpretation problems:

- finding no robots.txt file at the server (e.g. HTTP status code 404) implies that everything is permitted
- subdomains should have there own robots.txt file if not it is assumed that everything is allowed
- redirects involving protocol changes - e.g. upgrading from http to https - are followed and considered no domain or subdomain change - so whatever is found at the end of the redirect is considered to be the - robots.txt file for the original domain
- redirects from subdomain www to the doamin is considered no domain change - so whatever is found at the end of the redirect is considered to be the robots.txt file for the subdomain originally requested

For the thesis purposes it has been designed a dedicated function to inspect whether the domain requires specific actions or prevents some activity on thw target website. The following `checkpermission()` function has been integrated inside the scraping architecture and it is called once at the very beginning.

```r
library(robotstxt)
dominio = "immobiliare.it"
```

```
checkpermission = function(dom) {


    robot = robotstxt(domain = dom)
    vd = robot$check()[1]
    if (vd) {
        cat("\nrobot.txt for", dom, "is okay with scraping!")
    } else {
        cat("\nrobot.txt does not like what you're doing")
        ## stop()
    }
}
checkpermission(dominio)
```

```
##
## robot.txt for immobiliare.it is okay with scraping!
```

Further improvements in this direction came from the `polite` package (Perepolkin, 2019) which combines the power of the `robotstxt`, the `ratelimitr` to rate-limiting requests and the `memoise` for response caching. This package is wrapped up around 3 simple but effective ideas:

> The three pillars of a polite session are seeking permission, taking slowly and never asking twice.

The three pillars constitute the Ethical web scraping manifesto (Densmore, 2019) which are common shared practises that are aimed to self regularize scrapers. This has not nothing to do with law but since many scrapers themselves, as website administrators or analyst, have fought with bots. Bots might fake out real client logs and might stain analytics, so here it is born the choice to fine common ground and politely ask for permission.

2.3 User agents, Proxies, Handlers

HTTP requests to the website server by web clients come with some mandatory information packed in it. The process according to which HTTP protocols allow to exchange information can be easily thought with an everyday real world analogy. As a generic person A rings the door's bell of person B's house. A comes to B door with its personal information, its name, surname, where he lives etc. At this point B may either answer to A requests by opening the door and let him enter given the set of information he has, or it may not since B is not sure of the real intentions of A. This typical everyday situation in nothing more what happens billions of times on the internet everyday, the user (in the example above A) is interacting with a server website (part B) sending packets of information. If a server does not trust the information provided by the user, if the requests are too many, if the requests seems to be scheduled due to fixed sleeping time, a server can block requests. In certain cases it can even forbid the user log to the website. The language the two parties exchanges are coded in numbers that ranges from 100 to 511, each of which has its own specific significance. A popular case of this type of interaction occurs when users are not connected to internet so the server responds 404, page not found. Servers are built with a immune-system like software that raises barriers and block users to prevent dossing or other illegal practices.

This procedure is a day to day issue to people that are trying to collect information from websites. Google performs it everyday with its spider crawlers, which are very sophisticated bots that scrapes over a enormous range of websites. This challenge can be addressed in multiple ways, there are some specific Python packages that overcome this issue. The are also certain types of scraping as the Selenium web driver automation that simulates browser automation. Selenium allows the user not to be easily detected by the server immune system and peaceful. Precautions have not been taken lightly, and a simple but effective approach is proposed.

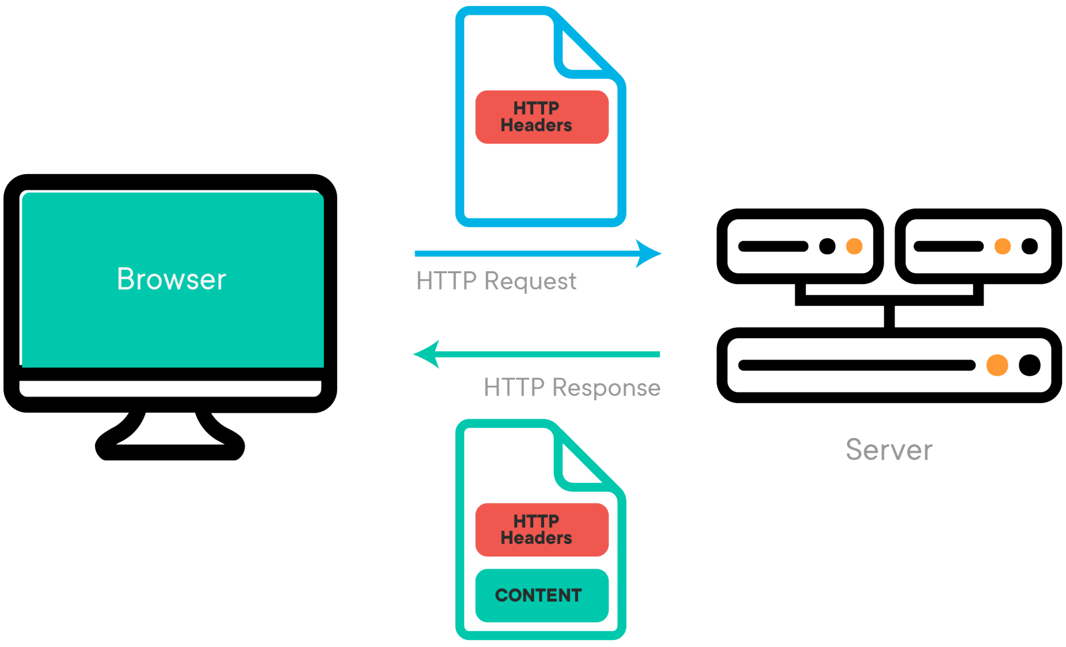


Figure 2.6: How Web Works

2.3.1 User agents Spoofing

A user agent (who, 2020) is a string of characters in each web browser that serves as an identification agent. The user agent permits the web server to be able to identify the user operating system and the browser. Then, the web server uses the exchanged information to determine what content should be presented to particular operating systems and web browsers on a series of devices. The user agent string contains the user application or software, the operating system (and their versions), the web client, the web client's version, and the web engine responsible for the content display (such as AppleWebKit). The user agent string is sent in form of a HTTP request header. Since the User Agents acts as middle man between the client request and the server response what it would be better doing is to actively faking it so that each time a web browser presents himself to a web server it has a different specifications, different web client, different operating system and so on.

The simple approach followed was building a vector of samples of different existing and updated User Agents (UA). Then whenever a request from a web browser is served to a web server 1 random sample string is drawn from the user agents deck. So each time the user is sending the requests it appears to have a different "identity". Below the user agents rotating pool:

```
set.seed(27)
agents = c("Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36",
    "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36",
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/602.2.14 (KHTML, like Gecko) Version/10.0.1 Safari/602.2.14",
    "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.98 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.98 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36",
    "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0")
agents[sample(1)]
```

```
## [1] "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36"
```

An improvement to this could be further using rotating proxies. A proxy server acts as a gateway between the web user and the web server. It's an intermediary server himself, separating end clients from the websites they are browsing. Proxy servers provide varying layers of functionality, security, and privacy are some of the examples. While the user is exploiting a proxy server, internet traffic flows through the proxy server on its way to the server you requested. The request then comes back through that same proxy server and then the proxy server forwards the data received from the website to you. Many proxy servers are offered in a paid version so in this case since security barriers of the target website are not high they will not be implemented. It has to be mentioned that many online services are providing free proxies but the turnaround of this solutions are many, two of them are: - Proxies to be free are widely shared among people, so as long as someone has used them for illegal purposes the user is inheriting their mistakes when caught. - Some of those proxies, pretty all the ones coming from low ranked websites, are tracked so there might be a user privacy violation issue.

2.3.2 Handlers

During scraping many difficulties might be met. Starting from the things that have been previously explained at the chapter start. Some of them are: a

varying url structure so that the html tree hierarchies are changed, payloads are not correctly parsed with respect to previous scraping sessions. ending with the ones that have just been said a few lines ago. Handlers and trycatch error workarounds are explicitly built in this sense. The continuous testing of the scraping functioning while developing has required the maintainer to track where the error occurs. A few numbers: the "agglomerative" function `get.data.catsing()` triggers more than 36 scrapping functions that are going to catch 36 different data pieces. If one of them went missing then the other one would be missing too. Then when row-data is binded together one entry column might not exists making the process fail.
Then the solution to that is to call inside the aggolmerative function as much as trycatch as many scrapping functions are involved. The trycatch can leverage the gap by introducing a specified quantity and alerting that something went wrong. On top of that many other handlers are called throughout the procedure:

- `get_ua()` verifies that the user agent coming from the session request is not the default one

```
get_ua = function(sess) {
    stopifnot(is.session(sess))
    stopifnot(is_url(sess$url))
    ua = sess$response$request$options$useragent
    return(ua)
}
```

- `is_url()` verifies that the url input needed has the canonic form. This is done by a REGEX query.

```
is_url = function(url) { re =
    "^(?:(?:http(?:s)?|ftp)://)(?:\\S+(?::(?:\\S)*)?@)?(?:(?:[a-z0-9¡-<ef><U+00BF><U+00BF>](?:-)*)*(?:[a-z0-9¡-<ef><U+00BF><U+00BF>])+)(?:\\.(?:[a-z0-9¡
    grepl(re, url) }
```

- `.get_delay()` checks through the robotxt file if a delay between each request is kindly welcomed.

```
.get_delay = function(domain) {

message(sprintf("Refreshing robots.txt data for %s...", domain))

cd_tmp = robotstxt::robotstxt(domain)$crawl_delay

if (length(cd_tmp) > 0) {
star = dplyr::filter(cd_tmp, useragent=="*")
if (nrow(star) == 0) star = cd_tmp[1,]
as.numeric(star$value[1])
} else {
10L
}

}

get_delay = memoise::memoise(.get_delay)
```

2.3.3 Parallel Computing

Since are opened as many sessions as single links and since for each link are going to be called many functions computations can take a while. Reporting it into numbers in order to get a usable dataset 300 pages are considered, which at their own time contains 25 links per page, for which almost 34 different functions are called. For the sake of the analysis and the app this should not bother the end user because scraping tasks are performed daily by a scheduler and a single day is sufficient amount of runtime. In any case functions are optimized following optimization criteria. Run time is crucial when dealing with active web pages and time to market in real estate is very important, here comes the need to have always fresh data. A way to secure fresh new data is to have lightweight computation on a single machine o heavy computation spread among a pool of different machines, in this case multi-threaded sessions. All the following runtime examinations are performed on the `scrape` functions a lightweight version of the final scraping API function. A first attempt was using `furrr` package (Vaughan and Dancho, 2018) which enables mapping through a list with the `purrr`, along with a `future` parallel backend. The approach has shows decent results, but its run time drastically increases when more requests are sent. This leads to a preventive conclusion about the computational complexity: it has to be at least linear. Empirical demonstrations have been made:

```
library(furrr)

vecelaps = c()
start = c()
end = c()
for (i in 1:len(list.of.pages.imm[1:20])) { start[i] = Sys.time()
    cat("\n\n run iteration", i, "over 20 total\n")
    list.of.pages.imm[1:i] %>% furrr::future_map(get.data.caturl,
    .progress = T) %>% bind_rows()

end[i] = Sys.time() vecelaps[i] = end[i] - start[i] }
```

```
furrrmethod = tibble(start,
end,
vettoelaps)

# ggplot (themed) run time meausurament method 1
p = ggplot(furrrmethod,aes(x=1:20, y=vettoelaps)) +
geom_line( color="steelblue") +
geom_point() +
xlab("Num URLS evaluated") +
ylab("run time (in seconds)") +
ggtitle("Run-Time for First method (furrr multisession)") +
stat_smooth(method=lm) +
theme_nicco()
p
```

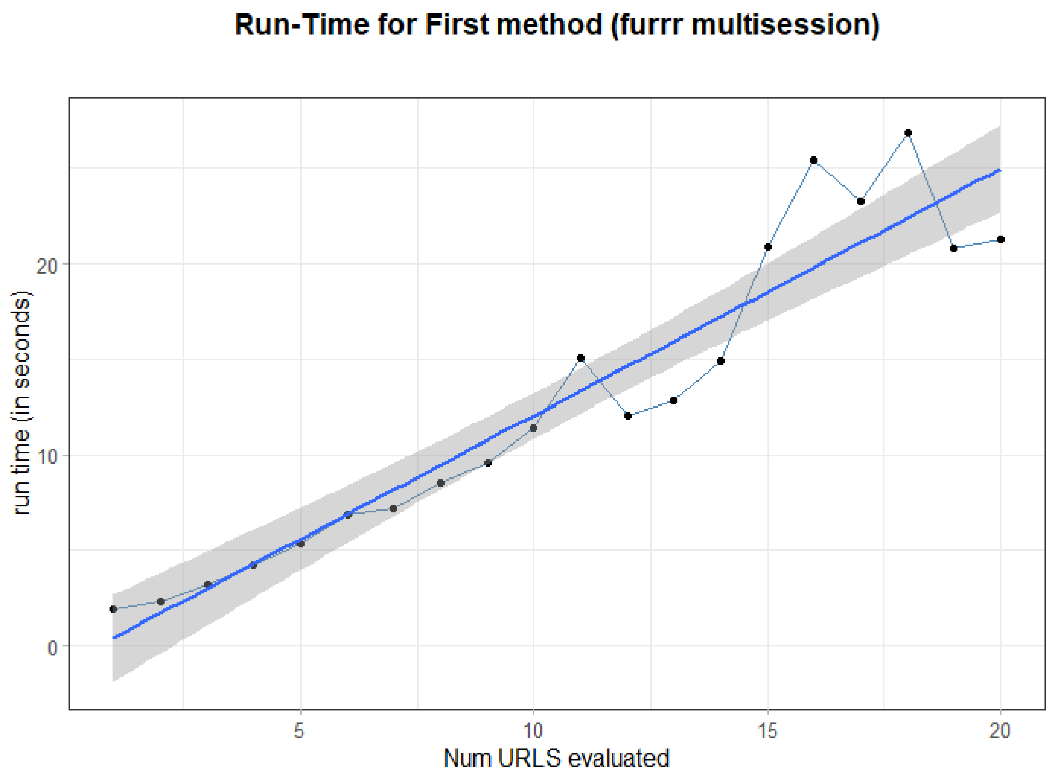


Figure 2.7: computational complexity analysis with Furrr

On the x-axis the number of urls which are evaluated together, on y axis the run time taken measured in seconds. Iteration after iteration the urls considered are increased by a unity. Looking at the smooth curve in between confidence lines a first guess might be linear time $\mathbf{O}(n)$, where n are the links considered.

A second attempt tried to explore the `foreach` package (Microsoft and Weston, 2020). This interesting package enables a new looping construct for executing R code in an iterative way. The main reason for using the `foreach` package is that it supports *parallel execution*, that is, it can execute those repeated operations on multiple processors/cores on the computer, or on multiple nodes of a cluster. The construction is straightforward:

- start clusters on processors cores
- define the iterator, in this case i = to the elements that are going to be looped through
- `.packages`: Inherits the packages that are used in the tasks define below
- `.combine`: Define the combining function that bind results at the end (say cbind, rbind or tidyverse::bind_rows). It has to be a string.
- `.errorhandling`: specifies how a task evaluation error should be handle.
- `%dopar%`: the dopar keyword suggests foreach with parallelization method
- then the function within the elements are iterated
- close clusters

One major important thing concerns the fact that th function within iterators repeats itself should be standalone. For standalone it is meant that the body function should be defined inside, as it would be a an empty environment. As a matter of fact packages has to be taken inside each time, and if the function is not defined inside body (or is not source from some other locations) the clusters can not operate and an error is printed.

```
cl = makeCluster(detectCores()-1)
registerDoParallel(cl)


vettoelaps1 = c()
start1 = c()
end1 = c()
for (j in 1:len(list.of.pages.imm[1:20])) {
start1[j] = Sys.time()
cat("\n\n run iteration",j,"over 20 total\n")
foreach(i = seq_along(list.of.pages.imm[1:j]),
.packages = lista.pacchetti,
.combine = "bind_rows",
.errorhandling='pass') %dopar% {
```

```
source("main.R")
x = get.data.caturl(list.of.pages.imm[i])
}
end1[j] = Sys.time()
vettoelaps1[j] = end1[j] -start1[j]
}
stopCluster(cl)
```

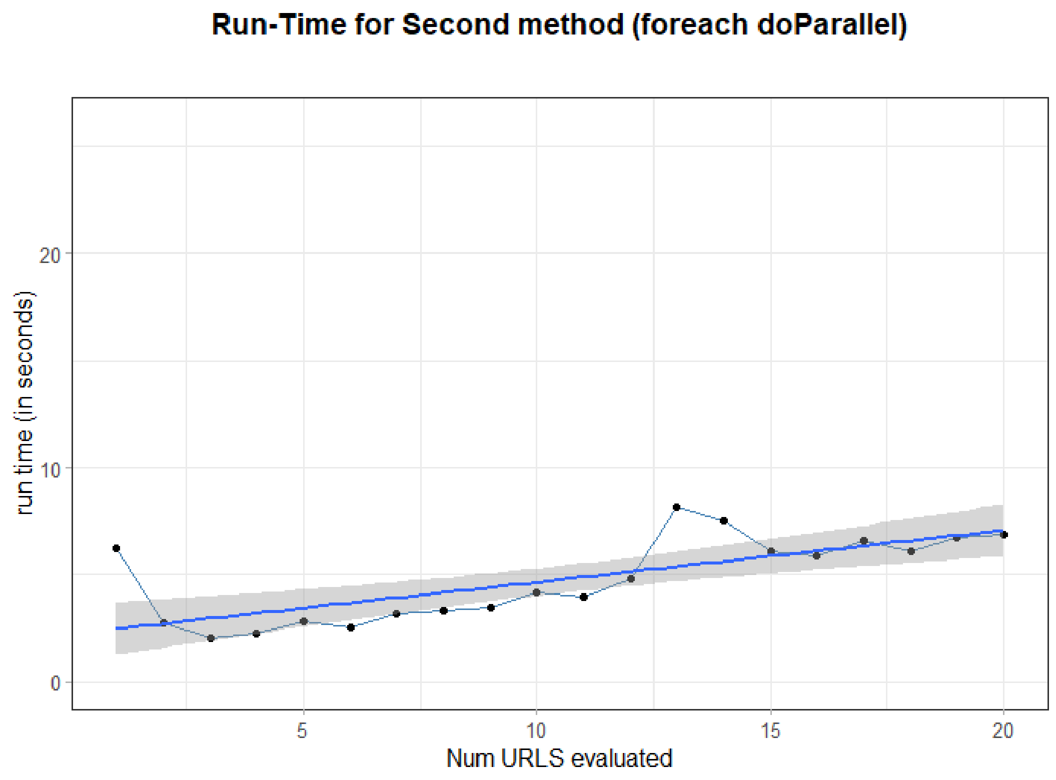


Figure 2.8: computational complexity analysis with Furrr

It can be seen quite easily that the curve now is flattened and first guess is logarithmic time $\mathbf{O}(log(n))$.

A further improvement could be obtained using a new package called `doAzureParallel` which is built on top of foreach. doAzureParallel enables different Virtual Machines operates parallel computing throughout Microsoft Azure cloud, but this comes at a substantial monetary cost. This would be a perfect match given that parallel methods seen before accelerates the number of requests sent among different processors or cluster, even though actually what it is really needed it is something that separates session. Unleashing Virtual Machines permits from one hand to further increase computational power and the number of potential requests, from the other it can splits requests among different user agents (a pool for each VM) masquerading even better the scraping automation.

2.4 Further Improvements

The main challenge remains unsolved since each single elements has been finely optimized but API continues to be unstable. What it can not be optimized are the system ad choices to change the layout of the page or to change the url structure (allocation of datain the web page). The way the API is designed really facilitates responsive fast debugging but this can not be by any means automatized. The API necessitates frequently to resort to continuous integration (i.e. CI) review to verify the working status. Moreover Error messages can not really be undestood sometimes, this is due to functions that are called within a parallel beckend that does not allow [refererenza su stack overflow di errore di print] to print error on console. So each time an error occur the "main" functions needs to be taken out of from the foreach cluster and evaluated in isolation where the loop stops. This is trivial and time consuming but for the time being no solutions have been provided.

2.5 Legal Challenges (ancora non validato)

"Data that are online and public are always available for all" is never a good answer to the question "Can I use those web data to my scope". Immobiliare.it[2] is not providing any open source data from its own database neither it is planning to do so in the future. It has not even provided a paid API

[2]https://www.immobiliare.it/

through which might be possible to perform analysis. A careful reading of thier terms has been done to get this service running without any legal consequence, as a reference further information can be collected in their specialized section (**?**)(https://www.immobiliare.it/terms/) Nevertheless the golden standard for scraping was respected since the robottxt file talks clearly permitting any action as demonstrated above. So if it might be the case of misuse of their intellectual their property, it will be also the case of misunderstanding between servers response and immobiliare.it intent to preserve their own intellectual property. What it has been really surprising are the low barriers to request information to their severs with respect to other online players. Best practices are applied and delayed requests have been sent to normalized traffic congestion in compliance to anti-dossing rules. But scraping criteria followed are fully based on common shared best practises (see section 2.2), and not any sort of agreements between parties (i.e. general terms and agreements). As a result a possible approach could be applying scraping procedures without any prevention. It would not surely cause any sort of disservice for the website but in the long run when scrapers are few, but in the long run it will cause delays as soon as subjects will increase. Totally different was the approach proposed by Idealista.com, which is a comparable to immobiliare.it. Idealista does block requests if they are not in compliance with their servers predefined ones. User agents in this case must be rotated quite frequently and as soon as a request does not fall within the pool of user agents (i.e. is labled as web bot) it is immediately blocked and 404 response is sent back.

- Idealista content is composed by Javascript so and html parser can no get that.
- Idealista blocks also certain web browser that have a demonstrated "career" in scraping procedures.

All of this leads to accept that entry barriers to scrape are for sure higher than the one faced for Immobiliare. The reticence to share data could be a reflex on how big idealista is; as a matter of fact it has a heavy market presence in some of the Europe real estates country as Spain and France. So what they thought was to raise awareness on scraping procedure that in a certain way can hurt their business. This has been validated by the fact that prior filtering houses on their website a checkbox has to be signed. The checkbox make the user sign an agreement on their platform according to which data can not be misused and it belongs their intellectual property.

Chapter 3

Infrastructure

In order to provide a fast and easy to use service to the end user many technologies have been involved. Challenges in scraping as pointed out in section 2.4 are many and still some remains unsolved. Challenges regards not only scraping but also the way the service has to respond to the users. Service has to be fast otherwise data become obsolete and so happen to analysis that relied on those data. Service has to be deployed so that in does not only run locally. Service needs to be scaled when needed since when the number of users increases the run time performance should not decrease. Service has to be maintained up to date so that each function can be reshaped with respect to immobiliare.it layout changes. On the other hand code behind the service has to be kept freezed to certain version, so that when packages are updated service still runs. Furthemore service has to be also secured granting access only to the ones authorized. In the end Service has to be run at a certain given times and storing data on cloud, so that it can be tracked back the evolution of the phenomenon. Open source solutions are available for back-end and front-end to meet the requirements before. Documentations related to technologies served are up to date and offer flexible solutions to embed the R environment. As a general discussion technologies used can be thought as the distance between something running locally on the laptop and something that is actually put into production, seen by company stakeholders, solving business problem. When such technologies are applied data scientist and counterparts gradually close the gap. Insights are better communicated (they can be interactive or automated) and services can be shared over a wider range of subjects. Nonetheless when the infrastructure is made with vision then integrating or substituting existing technologies is not trivial. Anyway technologies can not be always embedded because they might be exclusively designed to work only on certain back ends, therefore some choices are not into discussion. With foresight RStudio by setting future-oriented guidelines has spent a lot of effort giving its users an easy, integrated and interconnected environment. By that it is meant that the RStudio community has tried to either integrate or open the possibility to a number of technologies that fill the blanks in their weaker parts. On top of many, an entire package has been dedicated to democratize REST APIs (Plumber (Trestle Technology, LLC, 2018)). As a further example developers in RStudio have created an entire new paradigm i.e. Shiny (Chang et al., 2020), a popular web app development package, that enforces the developer to have front-end and back-end technologies tied up in the same IDE. They also added performance monitoring and optimization packages that are fitted into shiny such as shinytest [metti tag] and shinyloadtest [metti tag] to simulate sessions and verify network traffic congestion. The actual idea is to provide an API with 4 endpoints which calls parallelized scraping functions. On the other hand (chapter 2) a daily scheduler, exposing one API endpoint, produces and later stores a .csv file in a NOSQL mongoDB Atlas could database. It is all meant to be containerized in a Linux env (Ubuntu distr) docker container hosted by a AWS EC2 server. API endpoints are going to be secured with https protocols and protected with authentication by nginx reverse proxy. A Shiny app calls an endpoint with specified parameters which returns up to date data from the former infrastructure. It then models data with bayesian spatial methods 6.

Technologies involved are:

- GitHub version control

- Scheduler cron job, section 3.1
- Docker containers, section 3.2
- Plumber REST API, section 3.3.1
- NGINX reverse proxy, section 3.4
- AWS (Amazon Web Services) EC2 3.5
- MongoDB Atlas
- Shiny, see chapter 7

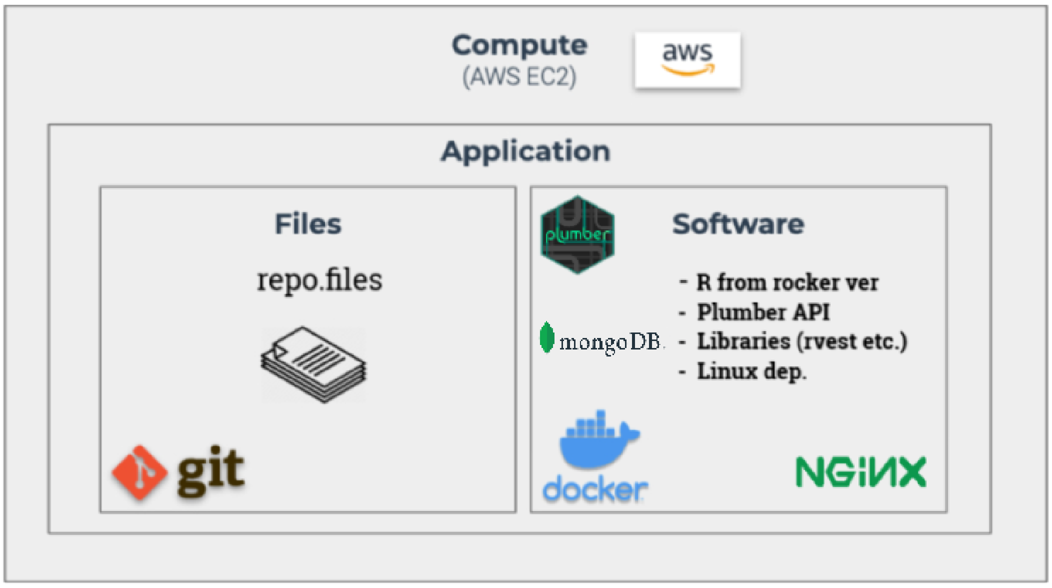


Figure 3.1: complete infrastructure (Matt Dancho source)

As a side note even each single part of this thesis has been made stand alone and can be easily accessed and modified through a gitbook deployed at @link[1]. RMarkdown knits the .rmd files extension and coverts them into .html files (the book's chapters). All the documents are then pushed to a Github repository with git. By a simple trick, since all the files are static html, they can be displayed through GH pages as it is a website whose link is a github subdomain. The pdf output for the thesis can be obtained by clicking the download button, then choosing output pdf in the upper banner. A Latex engine (Xelatex) wrapped into the website compiles the sequence of RMarkdown documents according to a predefined .tex template. Formatting can be tuned by modifying

[1]https://niccolosalvini.github.io/Thesis/

the .yml file, that sets general instructions for the pdf document. All of this has been possible thanks to Bookdown (Xie, 2020) once again a R well documented package (Xie, 2016) to build interactive books along with RMarkdown (Allaire et al., 2020).

Some of the main technologies implied will be viewed singularly, nonetheless for brevity some of them will be skipped.

3.1 Scheduler

A Scheduler in a process is a component on a OS that allows the computer to decide which activity is going to be executed. In the context of multi-programming it is thought as a tool to keep CPU occupied as much as possible. As an example it can trigger a process while some other is still waiting to finish. There are many type of scheduler and they are based on the frequency of times they are executed considering a certain closed time neighbor.

- Short term scheduler: it can trigger and queue the "ready to go" tasks
  - with pre-emption
  - without pre-emption

The ST scheduler selects the process and It gains control of the CPU by the dispatcher. In this context we can define latency as the time needed to stop a process and to start a new one.

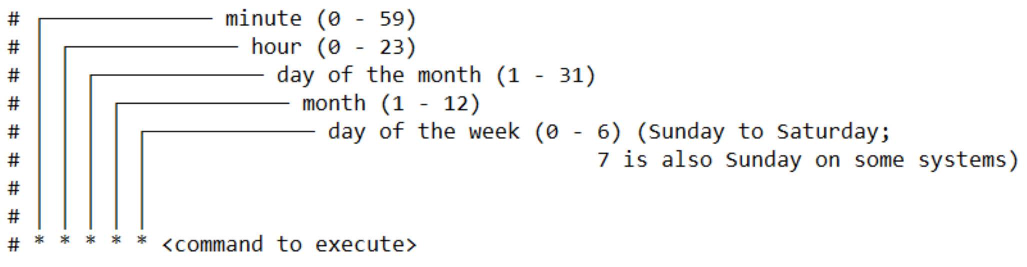
- Medium term scheduler
- Long term scheduler

for some other useful but beyond the scope refereces, such as the scheduling algorithm the reader can refer to (Wikiversità, 2020).

3.1.1 Cron Jobs

Cron job is a software utility which acts as a time-based job scheduler in Unix-like OS. Linux users that set up and maintain software environments exploit cron to schedule their day-to-day routines to run periodically at fixed times, dates, or intervals. It typically automates system maintenance but its usage is very flexible to whichever needed. It is lightweight and it is widely used since it is a common option for Linux users. The tasks by cron are driven by a crontab file, which is a configuration file that specifies a set of commands to run periodically on a given schedule. The crontab files are stored where the lists of jobs and other instructions to the cron daemon are kept.

Each line of a crontab file represents a job, and has this structure



```
# ┌───────────── minute (0 - 59)
# │ ┌───────────── hour (0 - 23)
# │ │ ┌───────────── day of the month (1 - 31)
# │ │ │ ┌───────────── month (1 - 12)
# │ │ │ │ ┌───────────── day of the week (0 - 6) (Sunday to Saturday;
# │ │ │ │ │                                   7 is also Sunday on some systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * <command to execute>
```

Figure 3.2: crontab

Each line of a crontab file represents a job. This example runs a shell named scheduler.sh at 23:45 (11:45 PM) every Saturday. .sh commands can update mails and other minor routines.

45 23 * * 6 /home/oracle/scripts/scheduler.sh

Some rather unusual scheduling definitions for crontabs can be found in this reference (Wikipedia contributors, 2020). Crontab's syntax completion can be made easier through this[2] GUI.

The cron job needs to be ran on scraping fucntions at 11:30 PM every single day. The get_data.R script first sources an endpoint function, then it applies the function with fixed parameters. Parameters describe the url specification,

[2]https://crontab.guru/

so that each time the scheduler runs the get_data.R collects data from the same source. Day after day .json files are generated and then stored into a NOSQL *mongoDB* database whose credentials are public. Data are collected on a daily basis with the explicit aim to track day-by-day changes both in the new entries an goners in rental market, and to investigate the evolution of price differentials over time. Spatio-Temporal modeling is still quite unexplored, data is saved for future used. Crontab configuration for daily 11:30 PM schedules has this appearance:

```
30 11 * * * /home/oracle/scripts/get_data.R
```

Since now the computational power comes from the machine on which the system is installed. A smarter solution takes care of it by considering run time limits and the substantial inability to share data. To a certain extent what it has been already done since now might fit for personal use: a scheduler can daily execute the scraping scripts and generate a .csv file. Furthermore an application can rely on those data, but evident reasons suggest that it does not suite any need. What it will do the trick would be an open source dedicated software environment or *container* that will contains scraping functions and a scheduler on cloud solving a pair of the problems arisen. This problem can be addressed with a technology that has seen a huge growth in its usage in the last few years.

3.2 Docker

3.2.1 What is Docker

Docker is a software tool to create and deploy applications using containers. *Docker containers* are a standard unit of software (i.e. software boxes) where everything needed for applications, such as libraries or dependencies can be run reliably and quickly. Furthermore they are also portable, in the sense that they can be taken from one computing environment to the following. Docker containers by default run on kernel Linux OS. Containers can be thought as an abstraction at the app layers that groups code and dependencies together. One major advantage of containers is that multiple containers can run on the same machine with the same OS. Each container can run its own isolated process in the user space, so that each task is complementary to the other. Containers are lightweight and take up less space than Virtual Machines (container images are files which can take up typically tens of MBs in size), can handle more applications and require fewer Virtual Machines and OSs.

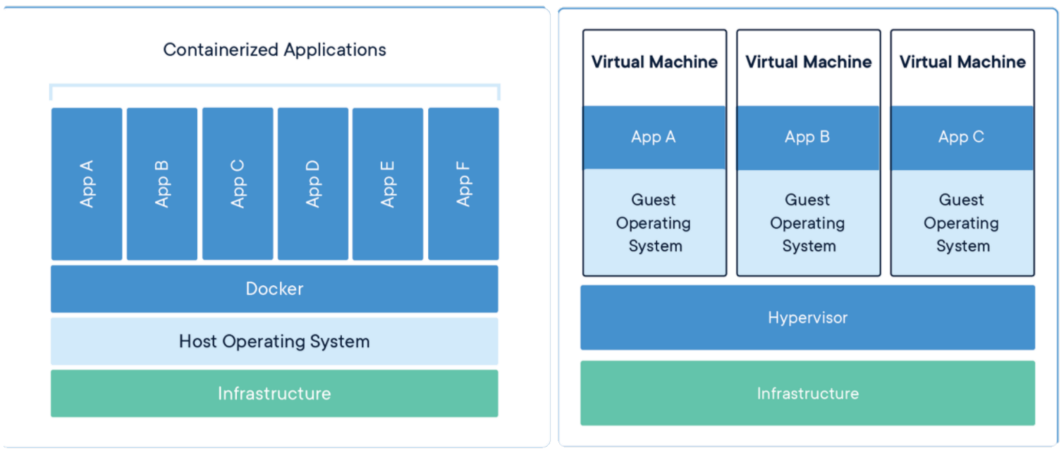


Figure 3.3: docker container vs VM

When containers are built *Docker container Images* are created and can be open sourced with Docker Hub *Docker Hub* is a web service provided by Docker for searching and sharing container images with other teams or developers in the community. Docker Hub behind authentication allows to integrate GitHub in the Docker project repository. Once the connection is authorized on local machine changes are made and then pushed by version control to the remote GH repository. The push command triggers the automatic building (pre-set by the user, branch should be given) of the image in the docker hub repository. The just created docker image can be tagged so that firstly it is recognizable and secondly can be reused in the future. Once the building stage is completed the DH repository can be pulled and then run locally on machine or cloud, see section 3.5. Docker building and testing images can be very time consumin. R packages can take a long time to install because code has to be compiled, especially if using R on a Linux server or in a Docker container. Rstudio package manager[3] includes beta support for pre-compiled R packages that can be installed faster. This dramatically reduces packages time installation (Nolis, 2020). In addition an open source project named rocker[4] have already narrowed the path building custom R images for R and docker users. What can be read from their own website about them: “The rocker project provides a collection of containers suited for different needs. find a base image to extend or images with popular software and optimized libraries pre-installed. Get the latest version or a reproducible fixed environment.” Enabling caching when container images are built heavily shorten the total duration for containerization.

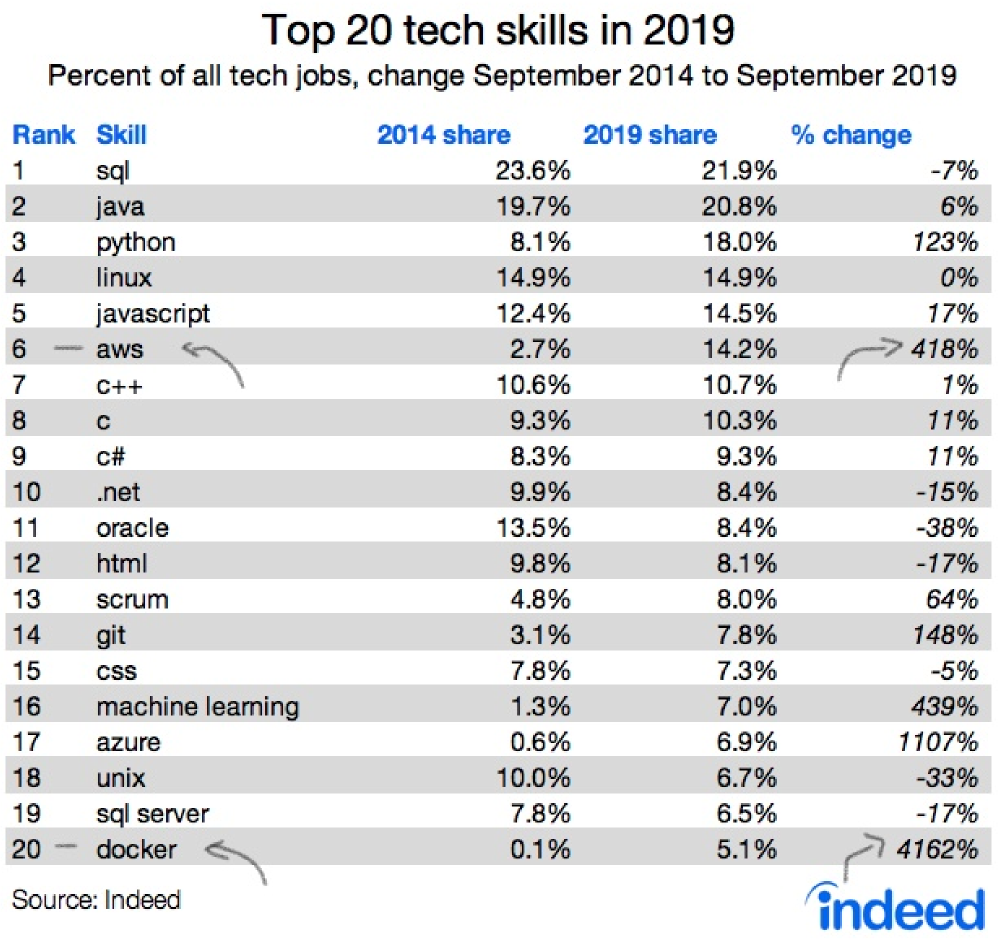
3.2.2 Why Docker

Indeed, an employment-related search engine, released an article on 2019 displaying changing trends from 2015 to 2019 in Technology Job market. Many changes are relevant in key technologies. Two among the others technologies (i.e. docker and Azure) have experienced a huge growth and both refer to the same demand input: *containers* . The landscape of Data Science is changing (was previously an Economist at the Indeed Hiring Lab, 2020) from reporting to application building: In 2015 - Businesses reports drive better decisions In 2020 - Businesses need apps to empower better decision making at all levels

For all the things said what docker is bringing to business (red, 2020):

- *Speed application deployment* : containers include the minimal run time requirements of the application, reducing their size and allowing them to be deployed quickly.
- *Portability across machines* : an application and all its dependencies can be bundled into a single container that is independent from the host version of Linux kernel, platform distribution, or deployment model.

[3]https://packagemanager.rstudio.com/client/#/
[4]https://www.rocker-project.org/images/



Top 20 tech skills in 2019

Percent of all tech jobs, change September 2014 to September 2019

Rank	Skill	2014 share	2019 share	% change
1	sql	23.6%	21.9%	-7%
2	java	19.7%	20.8%	6%
3	python	8.1%	18.0%	123%
4	linux	14.9%	14.9%	0%
5	javascript	12.4%	14.5%	17%
6	aws	2.7%	14.2%	418%
7	c++	10.6%	10.7%	1%
8	c	9.3%	10.3%	11%
9	c#	8.3%	9.3%	11%
10	.net	9.9%	8.4%	-15%
11	oracle	13.5%	8.4%	-38%
12	html	9.8%	8.1%	-17%
13	scrum	4.8%	8.0%	64%
14	git	3.1%	7.8%	148%
15	css	7.8%	7.3%	-5%
16	machine learning	1.3%	7.0%	439%
17	azure	0.6%	6.9%	1107%
18	unix	10.0%	6.7%	-33%
19	sql server	7.8%	6.5%	-17%
20	docker	0.1%	5.1%	4162%

Source: Indeed

indeed

Figure 3.4: docker-stats

This container can be transfered to another machine that runs Docker, and executed there without compatibility issues.

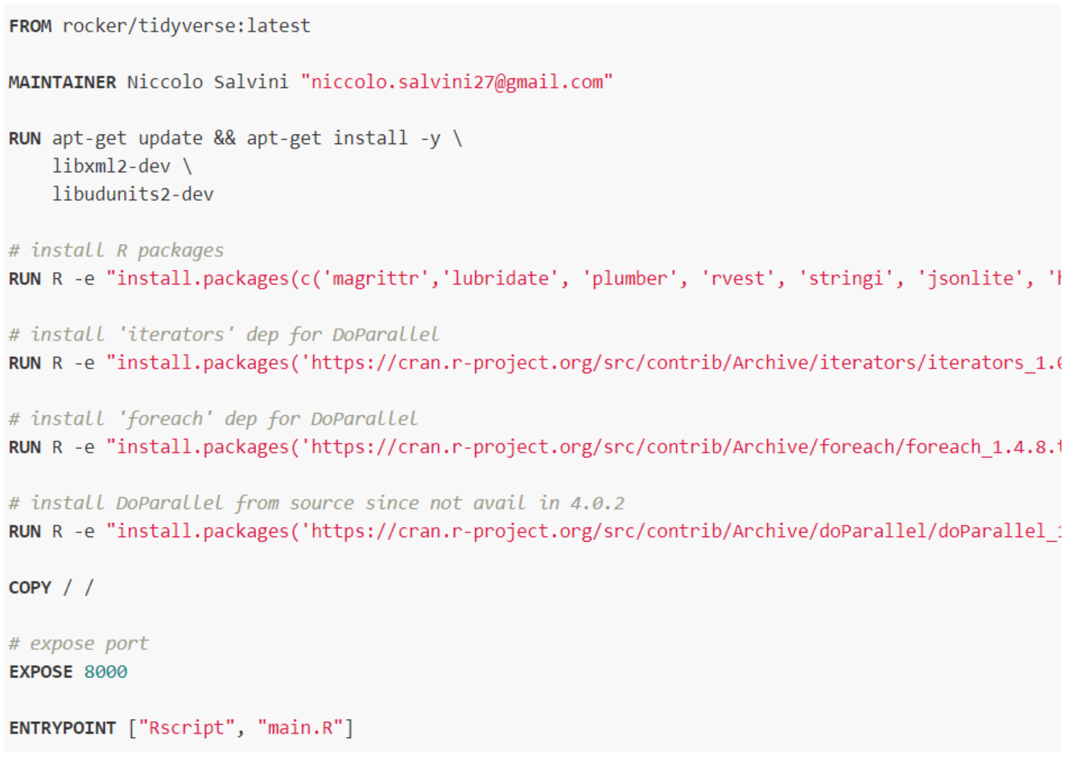
- *Version control and component reuse* : you can track successive versions of a container, inspect differences, or roll-back to previous versions. Containers reuse components from the preceding layers, which makes them noticeably lightweight. In addition due to Docker Hub it is possible to establish a connection between Git and DockerHub. Vesion
- *Sharing* : you can use a remote repository to share your container with others. It is also possible to configure a private repository hosted on Docker Hub.
- *Lightweight footprint and minimal overhead* : Docker images are typically very small, which facilitates rapid delivery and reduces the time to deploy new application containers.
- *Fault isolation* :Docker reduces effort and risk of problems with application dependencies. Docker also freezes the environment to the preferred packages version so that it guarantees continuity in deployment and isolate the container from system fails coming from package version updates.

The way to tell docker which system requirements are needed in the newly born software is a *Dockerfile.*

3.2.3 Dockerfile

Docker can build images automatically by reading instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands/rules a generic user could call on the CLI to assemble an image. Executing the command `docker build` from shell the user can trigger the image building. That executes sequentially several command-line instructions. For thesis purposes a dockerfile is written with the specific instructions and then the file is pushed to GitHub repository. Once pushed DockerHub automatically parses the repository looking for a plain text file whose name is “Dockerfile”. When It is matched then it trriggers the building of the image.

The Dockerfile used to trigger the building of the service docker container has the following set of instructions:



```
FROM rocker/tidyverse:latest

MAINTAINER Niccolo Salvini "niccolo.salvini27@gmail.com"

RUN apt-get update && apt-get install -y \
    libxml2-dev \
    libudunits2-dev

# install R packages
RUN R -e "install.packages(c('magrittr','lubridate', 'plumber', 'rvest', 'stringi', 'jsonlite', '

# install 'iterators' dep for DoParallel
RUN R -e "install.packages('https://cran.r-project.org/src/contrib/Archive/iterators/iterators_1.

# install 'foreach' dep for DoParallel
RUN R -e "install.packages('https://cran.r-project.org/src/contrib/Archive/foreach/foreach_1.4.8.

# install DoParallel from source since not avail in 4.0.2
RUN R -e "install.packages('https://cran.r-project.org/src/contrib/Archive/doParallel/doParallel_

COPY / /

# expose port
EXPOSE 8000

ENTRYPOINT ["Rscript", "main.R"]
```

Figure 3.5: dockerfile

- `FROM rocker/tidyverse:latest` : the command imports a pre-built image by the rocker team that contains the latest (tag latest) version of base-R along with the tidyverse packages.

- `MAINTAINER Niccolo Salvini "niccolo.salvini27@gmail.com"` : The command tags the maintainer and its e-mail contact information.

- `RUN apt-get update && apt-get install -y \ libxml2-dev \ libudunits2-dev` :The command update and install Linux dependencies needed for running R packages. `rvest` requires libxml2-dev and `magrittr` needs libudunits2-dev. If they are not installed then associated libraries can not be loaded. Linux dependencies needed have been found by trial and error while building containers. Building logs messages print errors and suggest which dependency is mandatory.

- `RUN R -e "install.packages(c('plumber','tibble','...',dependencies=TRUE)` : the command install all the packages required to execute the files (R files) containerized for the scraping. Since all the packages have their direct R dependencies the option `dependencies=TRUE` is needed.

- `RUN R -e "install.packages('https://cran.r-project.org/.../iterators, type='source')` `RUN R -e "install.packages('https://cran.r-project.org/.../` `type='source')` `RUN R -e "install.packages('https://cran.r-project.org/.../` `type='source')` DoParallel was not available in package manager for R version later than 4.0.0. For this reason the choice was to install a previous source version by the online repository, as well as its dependencies.

- `COPY \\` The command tells Docker copies all the files in the container.

- `EXPOSE 8000` : the commands instructs Docker that the container listens on the specified network ports 8000 at runtime. It is possible to specify whether the port exposed listens on UDP or TCP, the default is TCP (this part needs a previous set up previous installing, for further online documentation It is recommended (doc, 2020))

- `ENTRYPOINT ["Rscript", "main.R"]` : the command tells docker to execute the file main.R within the container that triggers the API start. In main.R it are pecified both the port and the host where API expects to be exposed (in this case port 8000).

In order to make the system stand-alone and make the service available to a wider range of subjects a choice has to be made. The service has to have both the characteristics to be run on demand and to specify query parameters.

3.3 API

API stands for application programming interface and it is a set of definitions and protocols for building and integrating application software. APIs let a product or a service communicate with other products and services without having to know how they're implemented. This can simplify app development, saving time and impacting positively on the budget. APIs are sometimes thought of as contracts, with documentation that represents an agreement between parties. There are many types of API that exploit different medium to communicate with apps or services. HTTP API are a special type of API that accepts http requests as input and elaborate them through end points. An end point identifies the operation through traditional http methods (e.g. /GET /POST) that the API caller wants to perform. HTTP APIs have now become the predominant medium by which software exchanges information, further documentation and differences between REST and RESTful API can be found here[5].

API examples: - Google Maps API: allows developers to embed geo-location data using JavaScript. The Google Maps API is designed to work on mobile and desktop. - YouTube API: allows developers integrate YouTube videos and functionalities into websites or applications. - Google Analytics API: allows to track website performance in terms of audience, monetization and other important metrics through the Google Analytics interface. The website the thesis come from has this implementation working.

This is obtained by embedding the existing R source code into the Plumber API framework.

[5]https://docs.aws.amazon.com/it_it/apigateway/latest/developerguide/http-api-vs-rest.html

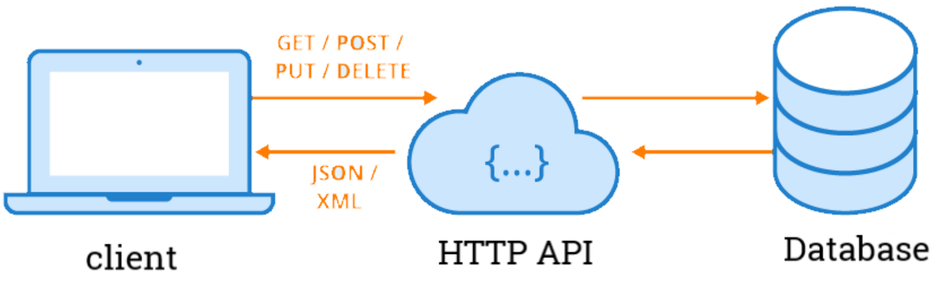


Figure 3.6: API functioning

3.3.1 Plumber API

Plumber (api, 2020) allows the user to create a web API by simply adding decoration comments to the existing R source code. Decorations are a special type of comments that suggests to plumber where and when the API parts are. Below the original API source code.

```r
# plumber.R

# * Echo back the input * @param msg The message to echo * @get /echo
function(msg = "") {
    list(msg = paste0("The message is: '", msg, "'"))
}

# * Plot a histogram * @serializer png * @get /plot
function() {
    rand <- rnorm(100)
    hist(rand)
}

# * Return the sum of two numbers * @param a The first number to add * @param
# The second number to add * @post /sum
function(a, b) {
    as.numeric(a) + as.numeric(b)
}
```

three endpoints associated to 2 /GET and 1 /POST requests are made available. Functions are made clear without names so that whenever the endpoint is called functions are directly executed. Decorations are marked as this `#*` and they are followed by specific keywords denoted with `@`. - the `@params` keyword refers to parameter that specifies the corpus of the HTTP request, i.e. the inputs with respect to the expected output. If default parameters are inputted then the API response is the elaboration of the functions with default parameters. As opposite endpoint function elaborates the provided parameters and returns a response. - `#* @serializer` specifies the extension of the output file when needed. - `#* @get` specifies the method of HTTP request sent. - `/echo` is the end point name. - `@filter` decorations activates a filter layer which are used to track logs and to parse request before passing the argbody to the end points. - many more…

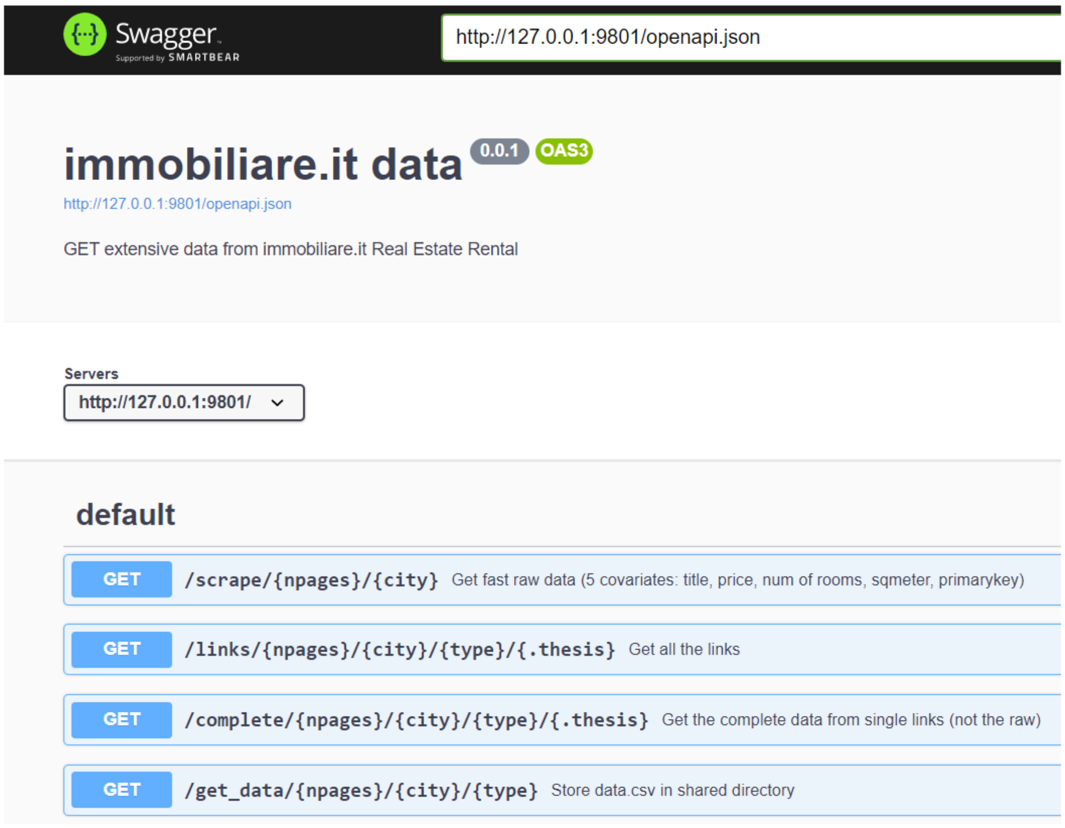
3.3.2 Immobiliare.it HTTP API

The API service is composed by three endpoints */scrape* , */links* and */complete*:

- */scrape performs a fast scraping that extracts 5 covariates directly from filtered url. url from which data extraction takes place might be composed through parameters. By default the end point scrape data from

Milan real estate rental market. Fast scraping is reached thanks to avoiding to access to single links. It is a superficial scraping and does not contain geospatial, however it might fit for regression settings.

- */links: extracts the list of single links belonging to each of the page, looking at section ?? each 25 single links for each sibling. It displays sufficient performances in terms of run time. It is propaedeutic to apply the following endpoint.
- */complete: both the function all.links and complete are sourced. The former with the aim to grab each single links and store it into an object. The latter to actually iterate scraping on each of the links.



http://127.0.0.1:9801/openapi.json

immobiliare.it data 0.0.1 OAS3

http://127.0.0.1:9801/openapi.json

GET extensive data from immobiliare.it Real Estate Rental

Servers
http://127.0.0.1:9801/

default

GET `/scrape/{npages}/{city}` Get fast raw data (5 covariates: title, price, num of rooms, sqmeter, primarykey)

GET `/links/{npages}/{city}/{type}/{.thesis}` Get all the links

GET `/complete/{npages}/{city}/{type}/{.thesis}` Get the complete data from single links (not the raw)

GET `/get_data/{npages}/{city}/{type}` Store data.csv in shared directory

Figure 3.7: swagger

3.3.3 API source code

- Get FAST data, it covers 5 covariates: title, price, num of rooms, sqmeter, primarykey

```
# * Get all the links * @param city [chr string] the city you are interested
# extract data (lowercase without accent) * @param npages [positive integer]
# number of pages to scrape default = 10, min = 2, max = 300 * @param type [c
# string] affitto = rents, vendita = sell (vendita no available for now) * @g
# /links/<npages:int>/<city:chr>/<type:chr>/<.thesis:bool>
function(npages = 10, city = "milano", type = "affitto", .thesis = F, req) {
    cat("\n\n port:", req$SERVER_PORT, "\n server_name:", req$SERVER_NAME, "\n
    if (npages > 300 & npages > 0) {
        stop("npages must be between 1 and 1,000")
    }
    if (.thesis) {
        list(all.links(npages, city, type, .thesis = TRUE))
    } else {
        list(all.links(npages, city, type))
    }
}



# * Get the complete data from single links (not the raw) * @param city [chr
# string] the city you are interested to extract data (lowercase without acce
# * @param npages [positive integer] number of pages to scrape default = 10,
# = 2, max = 300 * @param type [chr string] affitto = rents, vendita = sell
# (vendita no available for now) * @get
# /complete/<npages:int>/<city:chr>/<type:chr>/<.thesis:bool>
function(npages = 10, city = "milano", type = "affitto", .thesis = F, req) {
    if (npages > 300 & npages > 0) {
        stop("npages must be between 1 and 1,000")
    }
    if (.thesis) {
        links = all.links(npages, city, type, .thesis = TRUE)
        list(complete(links))
    } else {
        links = all.links(npages, city, type)
        list(complete(links))
    }
}
```

3.3.4 API documentation

- Get FAST data, it covers 5 covariates: title, price, num of rooms, sqmeter, primarykey

```
      GET */scrape
      @param city [chr string] the city you are interested in (e.g. "roma", "m
      @param npages [positive integer] number of pages to scrape, default = 10
      @param type [chr string] "affitto" = rents, "vendita"  = sell
      content-type: application/json
```

- Get all the links

```
      GET */link
      @param city [chr string] the city you are interested to extract data (lo
      @param npages [positive integer] number of pages to scrape default = 10,
      @param type [chr string] "affitto" = rents, "vendita"  = sell
      @param .thesis [logical] data used for master thesis
      content-type: application/json
```

- Get the complete set of covariates (52) from each single links, takes a while

```
GET */complete
@param city [chr string] the city you are interested to extract data (lo
@param npages [positive integer] number of pages to scrape default = 10,
@param type [chr string] "affitto" = rents, "vendita"  = sell
@param .thesis [logical] data used for master thesis
content-type: application/json
```

3.4 NGINX reverse proxy server

For analysis purposes NGINX is open source software for reverse proxying and load balancing. Proxying is typically used to distribute the load among several servers, seamlessly show content from different websites, or pass requests for processing to application servers over protocols other than HTTP. [...]

When NGINX proxies a request, it sends the request to a specified proxied server, fetches the response, and sends it back to the client. It is possible to proxy requests to an HTTP server (another NGINX server or any other server) or a non-HTTP server (which can run an application developed with a specific framework, such as PHP or Python) using a specified protocol. Supported protocols include FastCGI, uwsgi, SCGI, and memcached. [...]

.conf file and installation on Linux server. Security and Authentication.

3.5 AWS EC2 server

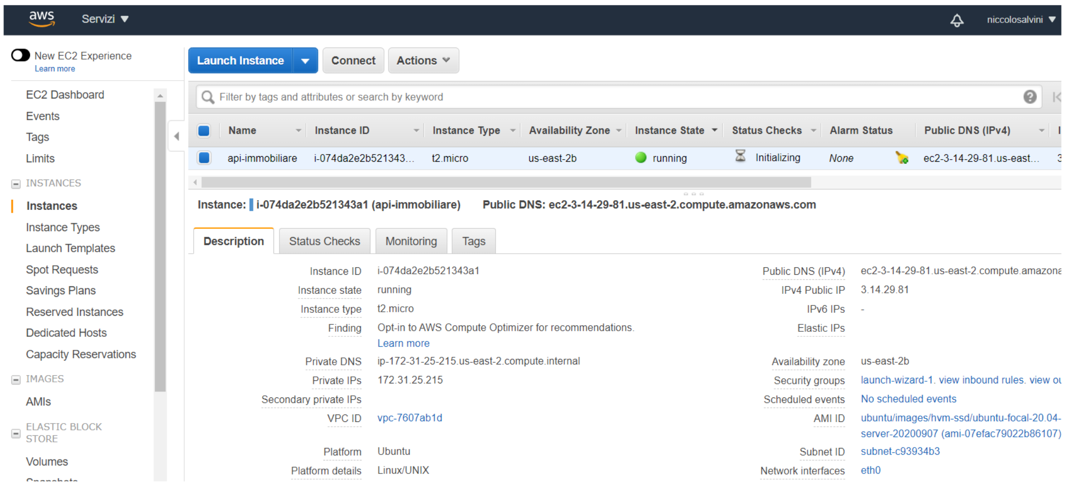
Executing API on a public server allows to share data with a multitude of subjects. Since it can not be specified a-priori how many users are going to enjoy the service a scalable solution fills the needs. Scalable infrastructure through a flexible cloud provider combined with nginx load balancing can offer a stable and reliable infrastructure for relatively a cheap price. AWS offers a wide range of services each of which for a wide range of budgets and integration. Free tier servers can be rent up to a certain amount of storage and computation that nearly 0s the total bill. The cloud provider also has a dedicated webpage to configure the service needed with respect to the usage named amazon cost manager[6]. Amazon Elastic Compute Cloud (EC2) is a web service that contributes to a secure, flexible computing capacity in the AWS cloud. EC2 allows to rent as many virtual servers as needed with customized capacity, security and storage. [few words still on EC2]

3.5.1 Launch an EC2 instance

The preliminary step is to pick up an AMI (Amazon Machine Image). AWS AMI are already-set-up machines with stadardized specification designed to speed up the process of choosing the a customed machine. Since the project is planned to be nearly 0-cost a "Free Tier Eligible" server is chosen. By checking the Free Tier box all the available free tiers are displayed. The machine selected has this specification: t2.micro with 1 CPU and 1GB RAM and runs on a Ubuntu distribution OS. First set up settings needs to be left as-is, networking and VPC can always be updated when needed. In the "add storage" step 30 GB storage are selected, moreover 30 represent the upper limit since the server can be considered free tier. Tags windows are beyond the scope. Secondly configuration needs to account security and a new entry below SSH connection (port 22) has to be set in. New security configuration has to have TCP specification and should be associated to port 8000. Port 8000, as in dockerfile section 3.2.3, has been exposed and needs to be linked to the security port opened.

At this point instance is prepared to run and in a few minutes is deployed. Key pairs, if never done before, are generated and .pem file is saved and securely stored. Key pairs are mandatory to access to the Ubuntu server via SSH. SSH

[6]https://aws.amazon.com/en/aws-cost-management/



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
api-immobiliare	i-074da2e2b521343...	t2.micro	us-east-2b	running	Initializing	None	ec2-3-14-29-81.us-east...

Instance: i-074da2e2b521343a1 (api-immobiliare) **Public DNS:** ec2-3-14-29-81.us-east-2.compute.amazonaws.com

Instance ID	i-074da2e2b521343a1	Public DNS (IPv4)	ec2-3-14-29-81.us-east-2.compute.amazona
Instance state	running	IPv4 Public IP	3.14.29.81
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-172-31-25-215.us-east-2.compute.internal	Availability zone	us-east-2b
Private IPs	172.31.25.215	Security groups	launch-wizard-1. view inbound rules. view ou
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-7607ab1d	AMI ID	ubuntu/images/hvm-ssd/ubuntu-focal-20.04-server-20200907 (ami-07efac79022b86107)
Platform	Ubuntu	Subnet ID	subnet-c93934b3
Platform details	Linux/UNIX	Network interfaces	eth0

Figure 3.8: aws_dashboard

connection in Windows OS can be handled with PuTTY[7], which is a SSH and telnet client designed for Windows. At first PuTTYgen has to convert the key pair .pem file into a .ppk extension (otherwise Putty can not read it). Once converted .ppk is sourced in the authorization panel. If everything works and authentication is verified then the Ubuntu server CLI appears and an interaction with the server is made available.

posso far vedere: - come si fa deplyment su AWS quindi - inizializzazione istanza - settare paramtri - mettere spazio macchina - spiegare perchè conviene tenere un server AWS

[7]https://www.putty.org/

Chapter 4

Exploratory Analysis

4.1 What data is

Data come packed from the API end point /complete in JSON format. A preliminary data assessment evidences 34 covariates and a 250 rows, even though the number could be varyign accoriding to the API query sent. Json file is then parsed by the convenient read_csv from the readr package mutating columns by their respective type. below the explanation of the columns:

ID: *numeric* a primary key number from immobiliare to identify univocally the advertisement LAT: *numeric* Exact latitude where the apartement is located (that compose the spatial covariate) LONG: *numeric* Exact longitude where the apartement is located (that compose the spatial covariate) CONDOM: *numeric* The condominium price which is an important component in the final cost structure BUILDAGE: *numeric* the year in which the building has been built FLOOR: *numeric* the floor at which is located the apartement/ house INDISVAP: *categorical* If it the rental ad regards an indipendent structure or an apartement in a building …

At a first glance it could be possible to see some missing values in some consistent measure in LOWPRCE and its derivations, Energy calss Disp Pauto.

4.2 Data Glimpse

```r
library(knitr)
glimpse(data)
```

```
## Rows: 250
## Columns: 34
## $ X                          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
## $ ID                         <int> 82585523, 82710269, 80532997, 80880239, 82
## $ LAT                        <dbl> 45.4770, 45.4709, 45.4605, 45.4709, 45.470
## $ LONG                       <dbl> 9.15101, 9.20868, 9.18927, 9.20868, 9.2086
## $ LOCATION                   <chr> "piazzale arduino C.A.", "via antonio kram
## $ CONDOM                     <int> 310, 117, 417, 117, 133, 170, 333, 317, 35
## $ BUILDAGE                   <int> 1940, 1920, 1967, 1920, 1920, 1990, 1900,
## $ FLOOR                      <chr> "6°, con ascensore", "1° piano", "4°, con
## $ INDIVSAPT                  <chr> "Appartamento", "Appartamento", "Appartame
## $ LOCALI                     <chr> "3 (2 camere da letto, 1 altro), 2 bagni,
## $ TPPROP                     <chr> "Residenziale", "Residenziale", "Residenzi
## $ STATUS                     <chr> "Nuovo / In costruzione", "Ottimo / Ristru
## $ HEATING                    <chr> "Centralizzato, a radiatori, alimentato a
## $ AC                         <chr> "Autonomo, freddo", NA, "Autonomo, freddo"
## $ PUB_DATE                   <chr> "2020-09-22", "2020-09-22", "2020-09-24",
## $ CATASTINFO                 <chr> "Classe A/3, rendita \200 697", "Classe A/
## $ APTCHAR                    <chr> "- - fibra ottica- - - videocitofono- - -
## $ PHOTOSNUM                  <int> 20, 20, 20, 20, 20, 20, 20, 20, 16, 20, 18
## $ AGE                        <chr> "Skyline RE Milano", "Skyline RE Milano",
## $ LOWRDPRICE.originalPrice <chr> NA, NA, NA, "\200 1.400", NA, NA, NA, NA,
## $ LOWRDPRICE.currentPrice  <chr> NA, NA, NA, "\200 1.300", NA, NA, NA, NA,
## $ LOWRDPRICE.passedDays    <int> NA, NA, NA, 8, NA, NA, NA, NA, NA, NA, NA,
## $ LOWRDPRICE.date          <int> NA, NA, NA, 8, NA, NA, NA, NA, NA, NA, NA,
## $ ENCLASS                    <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
## $ CONTR                       <chr> "Affitto", "Affitto", "Affitto", "Affitto"
## $ DISP                        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
## $ TOTPIANI                    <chr> "7 piani", "5 piani", "4 piani", "5 piani"
## $ PAUTO                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, "1 in gara
## $ REVIEW                      <chr> "Rif: CITY LIFE CUBE - OK CONTRATTO SOCIET
## $ HASMULTI                    <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
## $ PRICE                       <int> 2450, 1200, 3000, 1300, 1700, 3850, 5000,
## $ SQFEET                      <int> 110, 82, 180, 82, 82, 157, 208, 160, 90, 4
## $ NROOM                       <int> 3, 2, 5, 2, 3, 5, 5, 4, 2, 2, 1, 3, 2, 3,
## $ TITLE                       <chr> "Trilocale piazzale ARDUINO, Milano", "Bil
```

4.3 Explorative Analysis

4.3.1 Semivariogram Covariogram

[en passant anche isotropy and anisotropy. Con stazionarietà forte e debole]

4.4 Gaussian random fields

Chapter 5

INLA computation

va parafrasatoo

Several types of models are used with spatial and spatio-temporal data, depend- ing on the aim of the study. If we are interested in summarizing spatial and spatio-temporal variation between areas using risks or probabilities then we could use statistical methods like disease mapping to compare maps and identify clusters. Moran Index is extensively used to check for spatial autocorrelation (Moran, 1950), while the scan statistics, implemented in SaTScan (Killdorf, 1997), has been used for cluster detection and to perform geographical surveillance in a non-Bayesian approach. The same types of models can also be used in studies where there is an aetiological aim to assess the potential effect of risk factors on outcomes. A different type of study considers the quantification of the risk of experienc- ing an outcome as the distance from a certain source increases. This is typically framed in an environmental context, so that the source could be a point (e.g., waste site, radio transmitter) or a line (e.g., power line, road). In this case, the meth- ods typically used vary from nonparametric tests proposed by Stone (1988) to the parametric approach introduced by Diggle et al. (1998). In a different context, when the interest lies in mapping continuous spatial (or spatio-temporal) variables, which are measured only at a finite set of specific points in a given region, and in predicting their values at unobserved locations, geostatistical methods – such as kriging – are employed (Cressie, 1991; Stein, 1991). This may play a significant role in environmental risk assessment in order to identify areas where the risk of exceeding potentially harmful thresholds is higher. Bayesian methods to deal with spatial and spatio-temporal data started to appear around year 2000, with the development of Markov chain Monte Carlo (MCMC) simulative methods (Casella and George, 1992; Gilks et al., 1996). Before that the Bayesian approach was almost only used for theoretical models and found little applications in real case studies due to the lack of numerical/analytical or simula- tive tools to compute posterior distributions. The advent of MCMC has triggered the possibility for researchers to develop complex models on large datasets without the need of imposing simplified structures. Probably the main contribution to spatial and spatio-temporal statistics is the one of Besag et al. (1991), who developed the Besag–York–Mollié (BYM) method (see Chapter 6) which is commonly used for disease mapping, while Banerjee et al. (2004), Diggle and Ribeiro (2007) and Cressie and Wikle (2011) have concentrated on Bayesian geostatistical models. The main advantage of the Bayesian approach resides in its taking into account uncertainty in the estimates/predictions, and its flexibility and capability of dealing with issues like missing data. In the book, we follow this paradigm and introduce the Bayesian philosophy and inference in Chapter 3, while in Chapter 4 we review Bayesian computation tools, but the reader could also find interesting the follow- ing: Knorr-Held (2000) and Best et al. (2005) for disease mapping and Diggle et al. (1998) for a modeling approach for continuous spatial data and for prediction

va parafrasatoo

5.1 INLA

For many years, Bayesian inference has relied upon Markov chain Monte Carlo methods (Gilks et al. 1996; Brooks et al. 2011) to compute the joint posterior distribution of the model parameters. This is usually computationally very expensive as this distribution is often in a space of high dimension.

Havard Rue, Martino, and Chopin (2009) propose a novel approach that makes Bayesian inference faster. First of all, rather than aiming at estimating the joint posterior distribution of the model parameters, they suggest focusing on individual posterior marginals of the model parameters. In many cases, marginal inference is enough to make inference of the model parameters and latent effects, and there is no need to deal with multivariate posterior distributions that are difficult to obtain. Secondly, they focus on models that can be expressed as latent Gaussian Markov random fields (GMRF). This provides the computational advantages (see Rue and Held 2005) that reduce computation time of model fitting. Furthermore, Havard Rue, Martino, and Chopin (2009) develop a new approximation to the posterior marginal distributions of the model parameters based on the Laplace approximation (see, for example, MacKay 2003). A recent review on INLA can be found in Rue et al. (2017).

5.2 Laplace Approximation

An alternative approach to the simulation-based MC integration is analytic approx- imation with the Laplace method. Suppose we are interested in computing the following integral:

$$\int f(x)\mathrm{d}x = \int \exp(\log f(x))\mathrm{d}x$$

where $f(x)$ is the density function of a random variable X. We represent $logf(x)$ by means of a Taylor series expansion evaluated in x = x0:

$$\log f(x) \approx \log f\left(x_0\right) + \left(x - x_0\right) \left.\frac{\partial \log f(x)}{\partial x}\right|_{x=x_0} + \frac{\left(x - x_0\right)^2}{2} \left.\frac{\partial^2 \log f(x)}{\partial x^2}\right|_{x=x_0}$$

If x0 is set equal to the mode $x* = argmax$, log f(x) then log f(x) $\frac{\partial \log f(x)}{\partial x}\Big|_{x=x^*} = 0$ and the approximation becomes

$$\log f(x) \approx \log f(x^*) + \frac{(x-x^*)^2}{2} \frac{\partial^2 \log f(x)}{\partial x^2}\Bigg|_{x=x^*}$$

The integral of interest is then approximated as follows:

$$\int f(x)\mathrm{d}x \approx \int \exp\left(\log f(x^*) + \frac{(x-x^*)^2}{2} \frac{\partial^2 \log f(x)}{\partial x^2}\Bigg|_{x=x^*}\right)\mathrm{d}x$$

$$= \exp(\log f(x^*)) \int \exp\left(\frac{(x-x^*)^2}{2} \frac{\partial^2 \log f(x)}{\partial x^2}\Bigg|_{x=x^*}\right)\mathrm{d}x$$

where the integrand can be associated with the density of a Normal distribution. In fact, by setting

$$\sigma^{2*} = -1/ \frac{\partial^2 \log f(x)}{\partial x^2}\Bigg|_{x=x^*}$$

we obtain:

$$\int f(x)\mathrm{d}x \approx \exp(\log f(x^*)) \int \exp\left(-\frac{(x-x^*)^2}{2\sigma^{2*}}\right)\mathrm{d}x$$

where the integrand is the kernel of a Normal distribution with mean equal to x∗ and variance σ^{2*}. More precisely, the integral evaluated in the interval (α, β) is approximated by:

$$\int_\alpha^\beta f(x)\mathrm{d}x \approx f(x^*)\sqrt{2\pi\sigma^{2*}}(\Phi(\beta) - \Phi(\alpha))$$

where $\Phi(\cdot)$ denotes the cumulative density function of th $Normal(x_i, \sigma^{2*})$ distri- bution.

qui volendo un esempio fatto da me

5.3 The Class of Latent Gaussian Models

The first step in defining a latent Gaussian model within the Bayesian framework is to identify a distribution for the observed data y = (y1, … , yn). A very general approach consists in specifying a distribution for yi characterized by a parameter i (usually the mean E(yi)) defined as a function of a structured additive predictor i through a link function g( ), such that g(i) = i. The additive linear predictor i is defined as follows:

$$\eta_i = \beta_0 + \sum_{m=1}^{M} \beta_m x_{mi} + \sum_{l=1}^{L} f_l\left(z_{li}\right)$$

Here 0 is a scalar representing the intercept; the coefficients $\beta = \{\beta_1, \dots, \beta_M\}$ quantify the (linear) effect of some covariates $x = (x_1, \dots, x_M)$ on the response; and $f = \{f_1(\cdot), \dots, f_L(\cdot)\}$ is a collection of functions defined in terms of a set of covariates $z = (z_1, ..., z_L)$. The terms $f_l(\cdot)$ can assume different forms such as smooth and

nonlinear effects of covariates, time trends and seasonal effects, random intercept and slopes as well as temporal or spatial random effects. For this reason, the class of latent Gaussian models is very flexible and can accomodate a wide range of models ranging from generalized and dynamic linear models to spatial and spatio-temporal models (see Martins et al., 2013 for a review). We collect all the latent (nonobservable) components of interest for the inference in a set of parameters named defined as $\theta = \{\beta_0, \beta, f\}$. Moreover, we denote with $\psi = \{\psi_1, \dots, \psi_K\}$ the vector of the K hyperparameters. By assuming conditional independence, the distribution of the n observations (all coming from the same distribution family) is given by the likelihood

$$p(y \mid \theta, \psi) = \prod_{i=1}^{n} p\left(y_i \mid \theta_i, \psi\right)$$

where each data point yi is connected to only one element i in the latent field θ. Martins et al. (2013) discuss the possibility of relaxing this assumption assuming that each observation may be connected with a linear combination of elements in θ; moreover, they take into account the case when the data belong to several distri- butions, i.e., the multiple likelihoods case.

We assume a multivariate Normal prior on with mean and precision matrix $Q(\psi)$, i.e., $\theta \sim \text{Normal}\left(\mathbf{0}, Q^{-1}(\psi)\right)$ with density function given by

$$p(\theta \mid \psi) = (2\pi)^{-n/2}|Q(\psi)|^{1/2} \exp\left(-\frac{1}{2}\theta' Q(\psi)\theta\right)$$

where | | denotes the matrix determinant and is used for the transpose operation. The components of the latent Gaussian field are supposed to be conditionally independent with the consequence that $Q(\psi)$ is a sparse precision matrix.8 This specification is known as Gaussian Markov random field (GMRF, Rue and Held, 2005). Note that the sparsity of the precision matrix gives rise to computational ben- efits when making inference with GMRFs. In fact, linear algebra operations can be performed using numerical methods for sparse matrices, resulting in a considerable computational gain (see Rue and Held, 2005 for algorithms). The joint posterior distribution of and ψ is given by the product of the likelihood (4.13), of the GMRF density (4.14) and of the hyperparameter prior distribution $p(\psi)$:

$$\begin{aligned}
p(\theta, \psi \mid y) &\propto p(\psi) \times p(\theta \mid \psi) \times p(y \mid \theta, \psi) \\
&\propto p(\psi) \times p(\theta \mid \psi) \times \prod_{i=1}^{n} p\left(y_i \mid \theta_i, \psi\right) \\
&\propto p(\psi) \times |Q(\psi)|^{1/2} \exp\left(-\frac{1}{2}\theta' Q(\psi)\theta\right) \times \prod_{i=1}^{n} \exp\left(\log\left(p\left(y_i \mid \theta_i, \psi\right)\right)\right) \\
&\propto p(\psi) \times |Q(\psi)|^{1/2} \exp\left(-\frac{1}{2}\theta' Q(\psi)\theta + \sum_{i=1}^{n} \log\left(p\left(y_i \mid \theta_i, \psi\right)\right)\right)
\end{aligned}$$

5.4 Approximate Bayesian inference with INLA

The objectives of Bayesian inference are the marginal posterior distributions for each element of the parameter vector

$$p\left(\theta_i \mid y\right)=\int p\left(\theta_i, \psi \mid y\right) \mathrm{d} \psi=\int p\left(\theta_i \mid \psi, y\right) p(\psi \mid y) \mathrm{d} \psi$$

and for each element of the hyperparameter vector

$$p\left(\psi_k \mid y\right)=\int p(\psi \mid y) \mathrm{d} \psi_{-k}$$

Thus, we need to perform the following tasks: (i) compute $p(\psi \mid y)$, from which also all the relevant marginals p(k|y) can be obtained; (ii) compute $p\left(\theta_i \mid \psi, y\right)$ which is needed to compute the parameter marginal posteriors $p\left(\theta_i \mid y\right)$

The INLA approach exploits the assumptions of the model to produce a numerical approximation to the posteriors of interest based on the Laplace approximation method introduced in Section 4.7 (Tierney and Kadane, 1986). The first task (i) consists of the computation of an approximation to the joint posterior of the hyperparameters as:

$$\begin{aligned}
p(\psi \mid y) &= \frac{p(\theta, \psi \mid y)}{p(\theta \mid \psi, y)} \\
&= \frac{p(y \mid \theta, \psi) p(\theta, \psi)}{p(y)} \frac{1}{p(\theta \mid \psi, y)} \\
&= \frac{p(y \mid \theta, \psi) p(\theta \mid \psi) p(\psi)}{p(y)} \frac{1}{p(\theta \mid \psi, y)} \\
&\propto \frac{p(y \mid \theta, \psi) p(\theta \mid \psi) p(\psi)}{p(\theta \mid \psi, y)} \\
&\approx \left.\frac{p(y \mid \theta, \psi) p(\theta \mid \psi) p(\psi)}{\tilde{p}(\theta \mid \psi, y)}\right|_{\theta=\theta^* \psi} =: \tilde{p}(\psi \mid y)
\end{aligned}$$

where $\tilde{p}(\theta \mid \psi, y)$ is the Gaussian approximation – given by the Laplace method – of $p(\theta \mid \psi, y)$ and $\theta^*(\psi)$ is the mode for a given ψ the Gaussian approximation turns out to be accurate since $p(\theta \mid \psi, y)$appears to be almost Gaussian as it is a priori dis- tributed like a GMRF, y is generally not informative and the observation distribution is usually well-behaved. The second task (ii) is slightly more complex, because in general there will be more elements in than in , and thus this computation is more expensive. A first easy possibility is to approximate the posterior conditional distributions $p(\theta \mid \psi, y)$ directly as the marginals from $\tilde{p}(\theta \mid \psi, y)$ i.e. using a Normal distribution, where the Cholesky decomposition is used for the precision matrix (Rue and Martino, 2007). While this is very fast, the approximation is generally not very good. The second possibility is to rewrite the vector of parameters as $\theta = (\theta_i, \theta_{-i})$ and use again Laplace approximation to obtain

$$\begin{aligned} p\left(\theta_i \mid \psi, y\right) &= \frac{p\left(\left(\theta_i, \theta_{-i}\right) \mid \psi, y\right)}{p\left(\theta_{-i} \mid \theta_i, \psi, y\right)} \\ &= \frac{p(\theta, \psi \mid y)}{p(\psi \mid y)} \frac{1}{p\left(\theta_{-i} \mid \theta_i, \psi, y\right)} \\ &\propto \frac{p(\theta, \psi \mid y)}{p\left(\theta_{-i} \mid \theta_i, \psi, y\right)} \\ &\approx \left. \frac{p(\theta, \psi \mid y)}{\tilde{p}\left(\theta_{-i} \mid \theta_i, \psi, y\right)} \right|_{\theta_{-i}=\theta^*_{-i}(\theta_i, \psi)} =: \tilde{p}\left(\theta_i \mid \psi, y\right) \end{aligned}$$

where $\tilde{p}\left(\theta_{-i} \mid \theta_i, \psi, y\right)$ is the Laplace Gaussian approximation to $p\left(\theta_{-i} \mid \theta_i, \psi, y\right)$ and $\theta^*_{-i}\left(\theta_i, \psi\right)$ is its mode. Because the random variables $\theta_{-i} \mid \theta_i, \psi, y$ re in general reasonably Normal, the approximation provided by (4.20) typically works very well. This strategy, however, can be very expensive in computational terms as $\tilde{p}\left(\theta_{-i} \mid \theta_i, \psi, y\right)$ must be recomputed for each value of and (some modifications to the Laplace approximation in order to reduce the computational costs are described in Rue et al., 2009).

Operationally, INLA proceeds as follows: (i) first it explores the hyperparameter joint posterior distribution $\tilde{p}(\psi \mid y)$ of Eq. (4.18) in a nonparametric way, in order to detect good points $\left\{\psi^{(j)}\right\}$ for the numerical integration required in Eq. (4.22). Rue et al. (2009) propose two different exploration schemes, both requiring a reparameterization of the ψ-space – in order to deal with more regular densities – through the following steps:

a) Locate the mode ψ^* of $\tilde{p}(\psi \mid y)$ by optimizing log $\tilde{p}(\psi \mid y)$ with respect to (e.g., through the Newton–Raphson method).
b) Compute the negative Hessian H at the modal configuration.
c) Compute the eigen-decomposition $= V\Lambda^{1/2}V'$, with $\Sigma = H^{-1}$.
d) Define the new variable z, with standardized and mutually orthogonal components, such that:

$$\psi(z) = \psi^* + V\Lambda^{1/2}z$$

The first exploration scheme (named grid strategy) builds, using the z-parameterization, a grid of points associated with the bulk of the mass of $\tilde{p}(\psi \mid y)$. This approach has a computational cost which grows exponentially with the number of hyperparameters; therefore the advice is to adopt it when K, the dimension of ψ, is lower than 4. Otherwise, the second explo- ration scheme, named central composite design (CCD) strategy, should be used as it reduces the computational costs. With the CCD approach, the integration problem is seen as a design problem; using the mode ψ^* and the Hessian H, some relevant points in the -space are selected for performing a second-order approximation to a response variable (see Section 6.5 of Rue et al., 2009 for details). In general, the CCD strategy uses much less points, but still is able to capture the variability of the hyperparameter distribution. For this reason it is the default option in R-INLA.

5.5 Stochastic partial differential equation

approach

As described in the previous chapters, in R-INLA the linear predictor η_i is defined using the formula which includes `f()` terms for nonlinear effects of covariates or random effects. In the same way, the Matérn GF will be included in the formula using a proper specification for `f()`. For making the connection between the linear predictorη i and the formula more explicit, it is convenient to follow Lindgren and Rue (2015) and to rewrite the linear predictor as:

$\eta_i = \sum_k h_k \left(z_i^k\right)$

where k denotes the k th term of the `formula`, z_i^k represents the covariate value for a fixed/nonlinear effect or, in the case of a random effect, the index of a second-order unit (e.g., area or point ID). The mapping function $h_k(\cdot)$ links z_i^k to the actual value of the latent field for the k th formula component. As an example consider the case where the linear predictor η_i includes a fixed effect of a covariate (named `z1` in R), a nonlinear effect (e.g., RW2) of the variable `time` (given by a sequence of time points), and a random effect with iid components indexed by the variable `index.random`. Thus the corresponding `R-INLA` `formula` is

```
formula <- -1 + z1 + f(time, model="rw2") + f(index.random, model="iid")
```

The default intercept is not included as we specify `-1` in the formula. With the new linear pwredictor, we have that $h_1\left(z_i^1\right)$ is equal to $z_i^1\beta$, $h_2\left(z_i^2\right)$ is the smooth effect evaluated at z_i^2 (the ith element of vector time), and $h_3\left(z_i^3\right)$ is the random effect component with index equal to z_i^3 (the ith element of `index.random`). As usual, the latent field θ is the joint vector of all the latent Gaussian variables included in the linear predictor.

The formulation only allows each observation to directly depend on a single element z_i^k from each effect $h_k(\cdot)$ and this does not cover the case when a random effect is defined as a linear combination of temporal or areal values, such as the SPDE representation. In such cases each observation y_i depends on a linear combination of the elements of and the observation distribution will be defined as:

$y_i \mid \theta, \psi \sim p\left(y_i \mid \sum_j A_{ij}\theta_j, \psi\right)$

instead of $y_i \mid \theta_i, \psi \sim p\left(y_i \mid \theta_i, \psi\right)$ as in Eq. (4.13). The term A_{ij} is the generic element of the matrix **A** referred to as the observation or projector matrix. In Sections 6.7.2 and 6.7.3, we will show how to create the **A** matrix using the helper func tion `inla.spde.matrix.A` and how to include it in the inla call through the `control.predictor` option. The **A** matrix defines a mapping between the spatial latent field (defined on the mesh) and the observations (defined in a set of locations) and this allows the SPDE models to be treated as standard indexed random effects (the mapping is done by placing appropriate $\varphi_g(s)$ values in the **A** matrix, see Eq. (6.18)). Internally, `R-INLA` creates a new linear predictor $\eta^\star$ defined as a linear combination of the original one η:

$\eta^\star = A\eta$

and in this case the likelihood is linked to the latent field through$\eta_i^\star$ instead of η_i:

$p(y \mid \theta, \psi) = \prod_{i=1}^{n} p\left(y_i \mid \eta_i^\star, \psi\right)$

5.6 The Projector Matrix

We need to construct a projection matrix **A** to project the GRF from the observations to thetriangulation vertices. The matrix **A** as the number of rows equal to the number of observations, and the number of columns equal to the number of vertices of the triangulation. Row i of **A** corresponding to an observation at location s_i ossibly has three non-zero values at the columns that correspond to the vertices of the triangle that contains the location. If s_i within the triangle, these values are equal to the barycentric coordinates. That is, they are proportional to the areas of each of the three subtriangles defined by the location s_i and the triangle's vertices, and sum to 1. If s_i s equal to a vertex of the triangle, row

i has just one non-zero value equal to 1 at the column that corresponds to the vertex. Intuitively, the value $Z(s)$ at a location that lies within one triangle is the projection of the plane formed by the triangle vertices weights at location

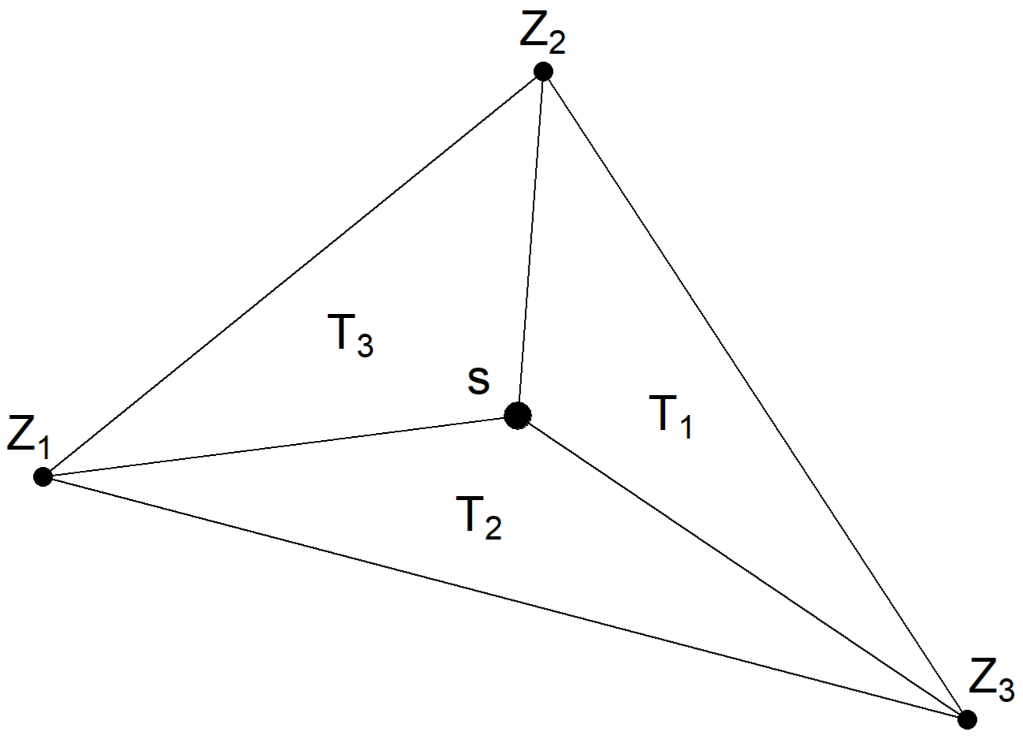
s .

An example of a projection matrix is given below. This projection matrix projects n observations to G triangulation vertices. The first row of the matrix corresponds to an observation with location that coincides with vertex number 3. The second and last rows correspond to observations with locations lying within triangles.

$$4 = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1G} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2G} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \dots & A_{nG} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & \dots & 0 \\ A_{21} & A_{22} & 0 & \dots & A_{2G} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \dots & 0 \end{bmatrix}$$

igure 8.6 shows a location s that lies within one of the triangles of a triangulated mesh. The value of the process $Z(\cdot)$ at s is expressed as a weighted average of the values of the process at the vertices of the triangle (Z_1, Z_2 and Z_3)and with weights equal to T_1/T, T_2/T and T_3/T where Tdenotes the area of the big triangle that contains s and T_1, T_2 and T_3 are the areas of the subtriangles

$Z(s) \approx \frac{T_1}{T} Z_1 + \frac{T_2}{T} Z_2 + \frac{T_3}{T} Z_3$



`R-INLA` provides the `inla.spde.make.A()` function to easily construct a projection matrix **A** We create the projection matrix of our example by using `inla.spde.make.A()` passing the triangulated mesh `mesh` and the coordinates `coo`.

```
A <- inla.spde.make.A(mesh = mesh, loc = coo)
```

Chapter 6

Point Referenced Data Modeling

Geostatistical data are a collection of samples of geo type data indexed by coordinates (e.g. latlong, eastings and northings) that originate from a spatially continuous phenomenon (Moraga, 2019). What It can be observed is just a sample, some sites, from a continuous unobserved process which we aim to estimate. Data as such can monitor a vast range of natural or economic meausuraments, as example cancer detection (Bell et al., 2006) at several sites, COVID19 spread in China (Li et al., 2020) etc. For simplicity lets consider y of interest observations $y\left(s_1\right), y\left(s_2\right), \ldots, y\left(s_n\right)$ from a random spatial process Y $Y\left(s_1\right), Y\left(s_2\right), \ldots, Y\left(s_n\right)$ observed at location $s_1, \ldots, s_n$. In the context of geostatistical data each observation, once again, has to be considered as a partial realization of a random spatial process. $\left\{Y(s): s \in D \subset \mathbb{R}^2\right\}$, where D is a subset of $\mathbb{R}^r$ a r-dimensional Euclidean space. When $r = 1$ it is a stochastic process widely encountered in literature e.g. time series process, nevertheless in this context there will be always $r = 2$ or eventually $r = 3$. The stochastic process Y is observed in a fixed set of "monitoring stations" and inference can be done regarding moments of the realized process. This information are essential to build a spatially continuous surface over the y-studied variable in order to predict the phenomenon at locations not yet observed.

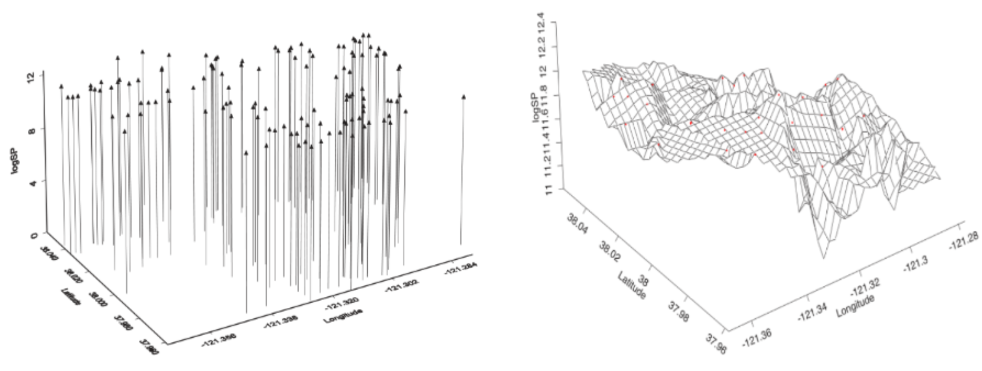


Figure 6.1: 3D scatterplot and surface, Stockton data.

6.1 Gaussian Random Fields (GRF)

A spatial process is said to be a **GF** if for any set of spatial index n and for each set of corresponding locations $\left(y\left(s_1\right), \ldots, y\left(s_n\right)\right)$ follows a multivariate *Gaussian* distribution with mean $\mu = \left(\mu\left(s_1\right), \ldots, \mu\left(s_n\right)\right)$ and covariance matrix . The covariance matrix relates each observation to each others and it is expressed through a spatial covariance function defined as $\mathcal{C}(\cdot)$, such that $\Sigma_{ij} = \operatorname{Cov}\left(y\left(s_i\right), y\left(s_j\right)\right) = \mathcal{C}\left(y\left(s_i\right), y\left(s_j\right)\right)$. There are many covariance functions and their application with respect to the others depend on data and the scope on the analysis. Examples of spatial covariance functions are: Linear, Spherical, Matérn etc. The latter displays higher degree of flexibility and it the most used in geostatistics (Krainski et al., 2018), therefore it will be implied in later analysis. The spatial indexes i and j move around an $r = 2$ euclidean space and can be thought as coordinates on a x, y euclidean axis. Moreover a spatial process can be either *first* order stationary or *second* order stationary. The former asserts that for any given spatial index n and its correspondent $s_{1,\ldots,n}$ locations the distribution of the process $\left\{Y\left(s_1\right), \ldots, Y\left(s_n\right)\right\}$ is the same of the one in $\left\{Y\left(s_1 + h\right), \ldots, Y\left(s_n + h\right)\right\}$, where h is a number belonging to R^2.

The second order stationarity states that the if the mean moment of the spatial process in constant over the study domain $\mu(\mathbf{s}) \equiv \mu$ (e.g. $E[L(s)] = \mu, \forall s \in D$) then the covariance functions does depend only on the distance between each couple points (euclidean $\|s_i - s_j\|$) separation. The natural consequence is that It does not matter whether the observations are placed either in a specific region, nor the direction towards they are oriented, all it counts it is their raw euclidean distance. In addition the process is said to be **Isotropic** if the covariance function depends only on the between-points distance. If the last assumption does not hold and direction towards point are distant from each other in the spatial domain D then the process is said to be **Anisotropic**.

Moreover since it is assumed to be true $E[Y(\mathbf{s}+\mathbf{h}) - Y(\mathbf{s})] = 0$, the second moment of the latter expression can be written as $E[Y(\mathbf{s}+\mathbf{h}) - Y(\mathbf{s})]^2$ that can be rewritten also as $\operatorname{Var}(Y(\mathbf{s}+\mathbf{h}) - Y(\mathbf{s}))$. Last expression is called *variogram* and can be expressed though $2\gamma(\mathbf{h})$, even tough its half is more often used and $\gamma(\mathbf{h})$ is named *semivariogram* [metti citation Cressie 2006]. The intuition behind this expression is that the difference in value between two near points $Y(\mathbf{s}+\mathbf{h})$ and $Y(\mathbf{h})$ is expected to be small with respect to the ones farther, in compliance with the first law of geography by Tobler:

> "everything is related to everything else, but near things are more related than distant things"

Semivariograms are an effient tool to asses spatial continuity but they are theretical. However semivargiogframs can be fitted to existing data giving birth to empirical semivariograms which are then plotted against the separation distance. The plot can be used to verify the null hypothesis of spatial independece and variability of the process. Below the functional form:

$\hat{\gamma}(t) = \frac{1}{2}|N(t)| \sum_{(\mathbf{s}_i, \mathbf{s}_j) \in N(t)} \left(Y\left(\mathbf{s}_i\right) - Y\left(\mathbf{s}_j\right)\right)^2$

where $N(t)$ is the set of location pairs such that $\left\|\mathbf{s}_i - \mathbf{s}_j\right\| = t$ and so $|N(t)|$ is the number of pairs in the set. As already guessed before empirical semivariogram values are expected to be small at short pairs distance and tends

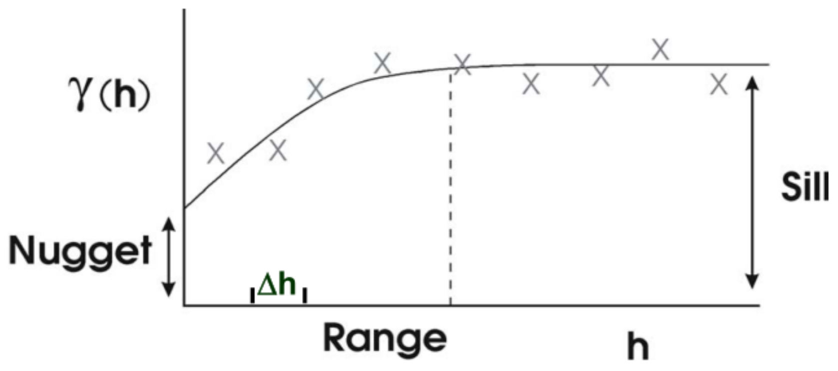


Figure 6.2: variogram example

to increase when distance increases. The rational behind is that similar observations are expected to lay close together (small h) leading to lower semivariogram values($\gamma(\mathbf{h})$), as opposite farther pairs obervations (big h) tend to be different and associated to greater semivariogram values. Flat semivariogram might indicate small spatial variance, since whether distance h increases or not, semivariogram values remains the same $\frac{\Delta\gamma(\mathbf{h})}{\Delta\mathbf{h}} \approx 0$. Semivariograms are also implied later in the modelling process to evaluate if any spatial pattern in still present in the residuals. The three main arguments in semivariograms are: *range* σ^2, *nugget* τ^2 and *sill* $\tau^2 + \sigma^2$. [parameters explanation]

6.2 The Stochastic Partial Differential Equation approach (SPDE)

SPDE [Lindegren] allows to extend the GF continuous process observations starting from the GMRF. Equation to start is a fractional partial differential:

$(\kappa^2 - \Delta)^{\alpha/2} (\tau\xi(s)) = \mathcal{W}(s)$

where $s \in \mathbb{R}^r$, Δ is the Laplacian function, α tunes the smoothness, $\kappa > 0$ is the scale parameter, τ controls the variance, and $\mathcal{W}(s)$ is a Gaussian spatila white noise process. The solution blended with conveninet Matèrn covariance function is:

$\operatorname{Cov}\left(\xi\left(s_i\right), \xi\left(s_j\right)\right)=\operatorname{Cov}\left(\xi_i, \xi_j\right)=\frac{\sigma^2}{\Gamma(\lambda) 2^{\lambda-1}}\left(\kappa\left\|s_i-s_j\right\|\right)^\lambda K_\lambda\left(\kappa\left\|s_i-s_j\right\|\right)$

6.3 Hedonic class models in points ref bayesian data analysis

The theoretical foundation of the hedonic price [Xiao et al., 2017] is based on the hedonic utility in theory of consumer's demand. The price is given by the unit cost of each characteristic. Hedonic Price function establishes a functional relationship between the observed household expenditures on housing, $P(H)$, and the level of characteristics contained in vector H. $P(H)=f\left(h_1, h_2, h_3, \ldots, h_n\right)$. The characteristics belong to different categories:

- residential structure, such as its style, lot size, and the number of rooms
- externalities associated with the geographic location (locational effects) both inbetween houses and within other existing ifrastructures

For the scope of the anlysis both of them are considered, even thougbh the effect of external-to-the-house existing infrastructures are not taken into consideration. Since the relationships between the house characteritics and the price is not explicited in economic literature it is possibile to assume whichever needed [linear non linear functional etc aggiungi qualcosa], which comes very handy in the INLA setting. Furthemore if it assumed a linear relationship a classical non-spatial linear regression can relate the y response variable with its covariates and error through the expression

$$y_i=\mu_i+\varepsilon_i$$

where y_i is normally distributed as $y_i \sim \operatorname{Normal}\left(\mu_i, \sigma^2\right)$ and μ_i is the mean structure that might linearly depend on some $\mathbf{x}$ covariates, β coefficients and ε white noise error. Keeping in mind the INLA notation, the linear predictor can be written as $\eta_i=\beta_0+\sum_{j=1}^J \beta_j x_{im}$, where i pedix is the observation number and j pedix is the mth covariate. Moreover it is possible to specify η as a function of μ. The link function between them in this case is identity, so that $\eta_i = g(\mu_i) = \mu_i$. In other modeling cases where response has to stay between $[0, 1]$ (e.g. probabilities) the link function has to Logit, giving birth to logistic regression. More generally expressed in vector notation, incorporating the spatial index:

$y(\mathbf{s}_i) = \mathbf{x}(\mathbf{s}_i)' \beta_j + \varepsilon(\mathbf{s}_i)$

where its OLS estimator is $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{y}$

However in many cases non-linear functions, like basis function might be a smoother solution to model coordinates when used as covariates (Krainski et al., 2018).

In the context of bayesian analysis a prior distribution has to be imposed on the regression coefficients $\beta = \{\beta_0, \ldots, \beta_J\}$ as well ad on the variance σ^2 of y_i. When no expert information is provided vague priors are introduced, meaning that the regression should not be driven by the priors choice. Vague priors might be:

- $\beta_m \sim \text{Normal}(0, 10^6)$ for the beta coefficients
- $\log(\tau) = \log(1/\sigma^2) \sim \log\text{Gamma}(1, 10^{-5})$ for precision

[qui piccolo regresso su bayesian modelling and hierarchical regression]

When leveraging information from location an added has to be introduced in the previous equation in order to fully incorporate spatial properties,

$y(\mathbf{s}) = \mathbf{x}(\mathbf{s})'\beta + w(\mathbf{s}) + \varepsilon(\mathbf{s})$

where the $w(\mathbf{s})$ is the entire surface D that encloses observations. The $w(\mathbf{s})$ in the context of the analysis is approached as a GF, seen in the previous section, whose distribution is multivariate with mean $\mu(\mathbf{s})$ in function of the spatial index $\mathbf{s}$ and covariance matrix . $\varepsilon(\mathbf{s})$ is iid and mean centered in 0 with variance τ^2 and is called the non-spatial error since it contributes to the nugget. The error term is nit pure since it interfers with the covariance function so the model can be fully spatial

Chapter 7

Shiny Web App

Some *significant* applications are demonstrated in this chapter.

7.1 Example one

7.2 Example two

Chapter 8

Final Words

We have finished a nice book.

Bibliography

(2020). 7.2. advantages of using docker red hat enterprise linux 7.

(2020). An api generator for r.

(2020). Get docker.

(2020). Presentazione dei file robots.txt - guida di search console.

(2020). User agent: Learn your web browsers user agent now.

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2020). *rmarkdown: Dynamic Documents for R.* R package version 2.3.

Bell, B. S., Hoskins, R. E., Pickle, L., and Wartenberg, D. (2006). *International Journal of Health Geographics*, 5(1):49.

Chang, W., Cheng, J., Allaire, J., Xie, Y., and McPherson, J. (2020). *shiny: Web Application Framework for R.* R package version 1.4.0.2.

Densmore, J. (2019). Ethics in web scraping.

Diestel, R. (2006). *Graph theory.* Graduate Texts in Mathematics. Springer, 3rd edition.

Krainski, E., Gómez-Rubio, V., Bakka, H., Lenzi, A., Castro-Camilo, D., Simpson, D., Lindgren, F., and Rue, H. (2018). *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA*. Chapman and Hall/CRC.

Li, H., Li, H., Ding, Z., Hu, Z., Chen, F., Wang, K., Peng, Z., and Shen, H. (2020). Spatial statistical analysis of coronavirus disease 2019 (covid-19) in china. *Geospatial Health*, 15(1).

Meissner, P. and Ren, K. (2020). *robotstxt: A 'robots.txt' Parser and 'Webbot'/'Spider'/'Crawler' Permissions Checker*. R package version 0.7.7.

Microsoft and Weston, S. (2020). *foreach: Provides Foreach Looping Construct*. R package version 1.5.0.

Moraga, P. (2019). *Geospatial Health Data*. Chapman and Hall/CRC.

Nolis, J. (2020). R docker faster.

Perepolkin, D. (2019). *polite: Be Nice on the Web*. R package version 0.1.1.

Trestle Technology, LLC (2018). *plumber: An API Generator for R*. R package version 0.4.6.

Vaughan, D. and Dancho, M. (2018). *furrr: Apply Mapping Functions in Parallel using Futures*. R package version 0.1.0.

was previously an Economist at the Indeed Hiring Lab, A. F. F. (2020). Indeed tech skills explorer: Today's top tech skills.

Wikipedia contributors (2020). Cron — Wikipedia, the free encyclopedia. [Online; accessed 15-September-2020].

Wikiversità (2020). Scheduling — wikiversità,. [Online; accesso il 15-settembre-2020].

Xie, Y. (2016). *bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 978-1138700109.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.20.