The “obvious” type of analysis  
to do is NLP, but there is also a lot of numerical data inside historical newspapers.  
For instance, you can find these tables that show the market prices of the day in the *L’Indépendance Luxembourgeoise*:

I wanted to see how easy it was to extract these tables from the newspapers and then make it available.  
It was a bit more complicated than anticipated.

**Download data**

The first step is to download the data. For this, I have used the code [@yvesmaurer](https://twitter.com/yvesmaurer) which you  
can find [here](https://github.com/ymaurer/eluxemburgensia-opendata-ark). This code makes it easy to download individual  
pages of certain newspapers,  
for instance [this one](https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F1drtkj%2Fpages%2F1/full/full/0/default.jpg). The  
pages I am interested in are pages 3, which contain the tables I need, for example  
[here](https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F1drtkj%2Fpages%2F3/full/full/0/default.jpg).  
[@yvesmaurer](https://twitter.com/yvesmaurer)’s code makes it easy to find the download links, which look like  
this: https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F1drtkj%2Fpages%2F3/full/full/0/default.jpg. It is also possible to  
crop the image by changing some parameters [like so](https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fwsvhwh%2Fpages%2F3/pct:74,0,100,100/full/0/default.jpg).  
This is helpful, because it makes the image smaller. The tables I’m interested in are always in the last column, so I can can use  
this feature to get smaller images. However, not every issue contains these tables, and I only want to download the ones  
that have these tables. So I wrote the following code to download the images I’m interested in:

library(tidyverse)

library(magick)

library(tesseract)

library(furrr)

download\_image <- function(link){

print(link)

isok <- image\_read(link) %>%

ocr(engine = "fra") %>%

str\_to\_lower() %>%

str\_detect("marché de luxembourg")

if(isok){

date\_link <- link %>%

str\_replace("pages%2f3", "pages%2f1") %>%

str\_replace("pct:74,0,100,100", "pct:76,1,17,5")

paper\_date <- image\_read(date\_link) %>%

ocr(engine = "fra") %>%

str\_squish() %>%

str\_remove("%") %>%

str\_remove("&") %>%

str\_remove("/")

ark <- link %>%

str\_sub(53, 60)

download.file(link, paste0("indep\_pages/", ark, "-", paper\_date, ".jpg"))

} else {

NULL

}

}

This code only downloads  
an image if the ocr() from the {tesseract} (which does, you guessed it, OCR) detects the string “marché de luxembourg” which  
is the title of the tables. This is a bit extreme, because if a single letter cannot be correctly detected by the OCR, the page will not  
be downloaded. But I figured that if this string could not be easily recognized, this would be a canary telling me that the text  
inside the table would also not be easily recognized. So it might be extreme, but my hope was that it would make detecting  
the table itself easier. Turned out it wasn’t so easy, but more on this later.

**Preparing images**

Now that I have the images, I will prepare them to make character recognition easier. To do this, I’m using the {magick}  
package:

library(tidyverse)

library(magick)

library(tesseract)

library(furrr)

prepare\_image <- function(image\_path){

image <- image\_read(image\_path)

image <- image %>%

image\_modulate(brightness = 150) %>%

image\_convolve('DoG:0,0,2', scaling = '1000, 100%') %>%

image\_despeckle(times = 10)

image\_write(image, paste0(getwd(), "/edited/", str\_remove(image\_path, ".jpg"), "edited.jpg"))

}

image\_paths <- dir(path = "indep\_pages", pattern = "\*.jpg", full.names = TRUE)

plan(multiprocess, workers = 8)

image\_paths %>%

future\_map(prepare\_image)

The picture below shows the result:

Now comes the complicated part, which is going from the image above, to the dataset below:

good\_fr,good\_en,unit,market\_date,price,source\_url

Froment,Wheat,hectolitre,1875-08-28,23,https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F02grxj%2Fpages%2F1/full/full/0/default.jpg

Métail,Meslin,hectolitre,1875-08-28,21,https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F02grxj%2Fpages%2F1/full/full/0/default.jpg

Seigle,Rye,hectolitre,1875-08-28,15,https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F02grxj%2Fpages%2F1/full/full/0/default.jpg

Orge,Barley,hectolitre,1875-08-28,16,https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F02grxj%2Fpages%2F1/full/full/0/default.jpg

Orge mondé,Pot Barley,kilogram,1875-08-28,0.85,https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F02grxj%2Fpages%2F1/full/full/0/default.jpg

Orge perlé,Pearl barley,kilogram,1875-08-28,0.8,https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F02grxj%2Fpages%2F1/full/full/0/default.jpg

Avoine,Oats,hectolitre,1875-08-28,8.5,https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F02grxj%2Fpages%2F1/full/full/0/default.jpg

Pois,Peas,hectolitre,1875-08-28,NA,https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F02grxj%2Fpages%2F1/full/full/0/default.jpg

**OCR with {tesseract}**

The first step was to get the date. For this, I have used the following function, which will then  
be used inside another function, which will extract the data and prices.

library(tidyverse)

library(lubridate)

library(magick)

library(tesseract)

library(furrr)

library(janitor)

is\_empty\_line <- function(line){

ifelse(line == "", TRUE, FALSE)

}

Sys.setlocale('LC\_TIME', "fr\_FR")

get\_date <- function(string, annee){

liste\_mois <- c("janvier", "février", "mars", "avril", "mai", "juin", "juillet",

"août", "septembre", "octobre", "novembre", "décembre")

raw\_date <- string %>%

str\_to\_lower() %>%

str\_remove\_all("\\.") %>%

str\_extract("\\d{1,2} .{3,9}(\\s+)?\\d{0,4}") %>%

str\_split("\\s+", simplify = TRUE)

if(ncol(raw\_date) == 2){

raw\_date <- cbind(raw\_date, "annee")

}

raw\_date[1, 3] <- annee

raw\_date <- str\_to\_lower(raw\_date[1:1, 1:3])

long\_month <- case\_when(

raw\_date[2] == "janv" ~ "janvier",

raw\_date[2] == "févr" ~ "février",

raw\_date[2] == "sept" ~ "septembre",

raw\_date[2] == "oct" ~ "octobre",

raw\_date[2] == "nov" ~ "novembre",

raw\_date[2] == "dec" ~ "décembre",

TRUE ~ as.character(raw\_date[2]))

raw\_date[2] <- long\_month

is\_it\_date <- as.Date(paste0(raw\_date, collapse = "-"), format = "%d-%b-%Y") %>%

is.na() %>% `!`()

if(is\_it\_date){

return(as.Date(paste0(raw\_date, collapse = "-"), format = "%d-%b-%Y"))

} else {

if(!(raw\_date[2] %in% liste\_mois)){

raw\_date[2] <- liste\_mois[stringdist::amatch(raw\_date[2], liste\_mois, maxDist = 2)]

return(as.Date(paste0(raw\_date, collapse = "-"), format = "%d-%b-%Y"))

}

}

}

This function is more complicated than I had hoped. This is because dates come in different formats.  
For example, there are dates written like this “21 Janvier 1872”, or “12 Septembre” or “12 sept.”.  
The biggest problem here is that sometimes the year is missing. I deal with this in the next  
function, which is again, more complicated than what I had hoped. I won’t go into details and  
explain every step of the function above, but the idea is to extract the data from the raw text,  
replace abbreviated months with the full month name if needed, and then check if I get a valid date.  
If not, I try my luck with stringdist::amatch(), to try to match, say “jonvier” with “janvier”.  
This is in case the OCR made a mistake. I am not very happy with this solution, because it is very  
approximative, but oh well.

The second step is to get the data. I noticed that the rows stay consistent, but do change  
after June 1st 1876. So I simply hardcoded the goods names, and was only concerned with extracting  
the prices. I also apply some manual corrections inside the function; mainly dates that were  
wrongly recognized by the OCR engine, and which were causing problems. Again, not an optimal solution,  
the other alternative was to simply drop this data, which I did not want to do. Here is the function:

extract\_table <- function(image\_path){

image <- image\_read(image\_path)

annee <- image\_path %>%

str\_extract("187\\d")

ark <- image\_path %>%

str\_sub(22, 27)

source\_url <- str\_glue("https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F{ark}%2Fpages%2F1/full/full/0/default.jpg",

ark = ark)

text <- ocr(image, engine = "fra")

text <- text %>%

str\_split("\n") %>%

unlist %>%

str\_squish() %>%

str\_remove\_all("^.{1,10}$") %>%

discard(is\_empty\_line) %>%

str\_replace("Mercuriale du \\+ Nov. 1831.", "Mercuriale du 4 Nov. 1831.") %>%

str\_replace("….u .T juillet.", "du 7 juillet") %>%

str\_replace("octobré", "octobre") %>%

str\_replace("AT octobre", "17 octobre") %>% # correction for "f8g6kq8-18 LUNDI 19 OCTOBRÉ 1874. BUREAUX de fa RÉDACTIGedited.jpg"

str\_replace("T norembre", "7 novembre") %>% # correction for fcrhrn5-LE 8 LUNDI 9 NOVEMBRE 1874 BUREAUX de la RÉDedited.jpg

str\_replace("À oc demain 5", "27 mai") %>% # correction for fd61vzp-MARDI 50. MAI 1876 BUREAUX de la. RED, n VE DE L’ADMINISTRAedited.jpg

str\_replace("G", "6") %>%

str\_replace("Hercariale du 80 nov. 1872,", "du 30 novembre 1872") %>%

str\_replace("….u .T juillet.", "du 7 juillet") %>%

str\_replace("Rs ne its du 28-octobré.: :!: :", "28 octobre") %>%

str\_replace("De routes due 98-juilléle. à eat", "28 juillet") %>%

str\_replace("\\| Mereariale dn 14 dre. 1872,", "14 décembre 1872")

start <- text %>%

str\_which("MARCH(É|E).\*D(E|É).\*LUXEMBOUR(G|6)") + 2

start <- ifelse(is\_empty(start), str\_which(text, ".\*D.\*UXEM.\*") + 2, start)

end <- start + 40

pricing\_date <- text[start - 1] %>%

str\_remove("%") %>%

str\_remove("er") %>%

str\_remove("\\.+") %>%

str\_remove("\\\*") %>%

str\_remove("®") %>%

str\_remove(":") %>%

str\_remove("\\?") %>%

str\_replace("\\$", "9") %>%

str\_remove("°") %>%

str\_replace("‘du 14août.. - ; En", "14 août") %>%

str\_replace("OP PE CN AP PP", "du 28 juin") %>%

str\_replace("‘ du 81 janvi Le", "31 janvier") %>%

str\_replace("\\| \\| du AT août", "17 août") %>%

str\_replace("Su” du 81 juillet. L", "31 juillet") %>%

str\_replace("0 du 29 avril \" \\|", "29 avril") %>%

str\_replace("LU 0 du 28 ail", "28 avril") %>%

str\_replace("Rs ne its du 28-octobre :!: :", "23 octobre") %>%

str\_replace("7 F \\| du 13 octobre LA LOTS", "13 octobre") %>%

str\_replace("À. du 18 juin UT ET", "13 juin")

market\_date <- get\_date(pricing\_date, annee)

items <- c("Froment", "Métail", "Seigle", "Orge", "Orge mondé", "Orge perlé", "Avoine", "Pois", "Haricots",

"Lentilles", "Pommes de terre", "Bois de hêtre", "Bois de chêne", "Beurre", "Oeufs", "Foin",

"Paille", "Viande de boeuf", "Viande de vache", "Viande de veau", "Viande de mouton",

"Viande fraîche de cochon", "Viande fumée de cochon", "Haricots", "Pois", "Lentilles",

"Farines de froment", "Farines de méteil", "Farines de seigle")

items\_en <- c("Wheat", "Meslin", "Rye", "Barley", "Pot Barley", "Pearl barley", "Oats", "Peas", "Beans",

"Lentils", "Potatoes", "Beech wood", "Oak wood", "Butter", "Eggs", "Hay", "Straw", "Beef meat",

"Cow meat", "Veal meat", "Sheep meat", "Fresh pig meat", "Smoked pig meat", "Beans", "Peas",

"Lentils", "Wheat flours", "Meslin flours", "Rye flours")

unit <- c("hectolitre", "hectolitre", "hectolitre", "hectolitre", "kilogram", "kilogram", "hectolitre",

"hectolitre", "hectolitre", "hectolitre", "hectolitre", "stere", "stere", "kilogram", "dozen",

"500 kilogram", "500 kilogram", "kilogram", "kilogram", "kilogram", "kilogram", "kilogram",

"kilogram", "litre", "litre", "litre", "kilogram", "kilogram", "kilogram")

# starting with june 1876, the order of the items changes

items\_06\_1876 <- c("Froment", "Métail", "Seigle", "Orge", "Avoine", "Pois", "Haricots", "Lentilles",

"Pommes de terre", "Farines de froment", "Farines de méteil", "Farines de seigle", "Orge mondé",

"Beurre", "Oeufs", "Foins", "Paille", "Bois de hêtre", "Bois de chêne", "Viande de boeuf", "Viande de vache",

"Viande de veau", "Viande de mouton", "Viande fraîche de cochon", "Viande fumée de cochon")

items\_06\_1876\_en <- c("Wheat", "Meslin", "Rye", "Barley", "Oats", "Peas", "Beans", "Lentils",

"Potatoes", "Wheat flours", "Meslin flours", "Rye flours", "Pot barley",

"Butter", "Eggs", "Hay", "Straw", "Beechwood", "Oakwood", "Beef meat", "Cow meat",

"Veal meat", "Sheep meat", "Fresh pig meat", "Smoked pig meat")

units\_06\_1876 <- c(rep("hectolitre", 9), rep("kilogram", 5), "douzaine", rep("500 kilogram", 2),

"stere", "stere", rep("kilogram", 6))

raw\_data <- text[start:end]

prices <- raw\_data %>%

str\_replace\_all("©", "0") %>%

str\_extract("\\d{1,2}\\s\\d{2}") %>%

str\_replace("\\s", "\\.") %>%

as.numeric

if(is.na(prices[1])){

prices <- tail(prices, -1)

} else {

prices <- prices

}

if(market\_date < as.Date("01-06-1876", format = "%d-%m-%Y")){

prices <- prices[1:length(items)]

tibble("good\_fr" = items, "good\_en" = items\_en, "unit" = unit, "market\_date" = market\_date,

"price" = prices, "source\_url" = source\_url)

} else {

prices <- prices[1:length(items\_06\_1876\_en)]

tibble("good\_fr" = items\_06\_1876, "good\_en" = items\_06\_1876\_en, "unit" = units\_06\_1876,

"market\_date" = market\_date, "price" = prices, "source\_url" = source\_url)

}

}

As I wrote previously, I had to deal with the missing year in the date inside this function. To do  
that, I extracted the year from the name of the file, and pasted it then into the date. The file  
name contains the data because the function in the function that downloads the files I also performed  
OCR on the first page, to get the date of the newspaper issue. The sole purpose of this was to  
get the year. Again, the function is more complex than what I hoped, but it did work well overall.  
There are still mistakes in the data, for example sometimes the prices are in the wrong order;  
meaning that they’re “shifted”, for example instead of the prices for eggs, I have the prices of the  
good that comes next. So obviously be careful if you decide to analyze the data, and double-check  
if something seems weird. I have made the data available on Luxembourg Open Data Portal,  
[here](https://data.public.lu/fr/datasets/digitised-luxembourg-historical-newspapers-journaux-historiques-luxembourgeois-numerises/#resource-community-27293c42-22e5-4811-aee8-89d6f7fa9533).

**Analyzing the data**

And now, to the fun part. I want to know what was the price of smoked pig meat, and how it varied  
through time:

library(tidyverse)

library(ggplot2)

library(brotools)

market\_price <- read\_csv("https://download.data.public.lu/resources/digitised-luxembourg-historical-newspapers-journaux-historiques-luxembourgeois-numerises/20190407-183605/market-price.csv")

## Parsed with column specification:

## cols(

## good\_fr = col\_character(),

## good\_en = col\_character(),

## unit = col\_character(),

## market\_date = col\_date(format = ""),

## price = col\_double(),

## source\_url = col\_character()

## )

market\_price %>%

filter(good\_en == "Smoked pig meat") %>%

ggplot(aes(x = market\_date, y = price)) +

geom\_line(aes(group = 1), colour = "#82518c") +

theme\_blog() +

labs(title = "Prices of smoked pig meat at the Luxembourg-City market in the 19th century")

## Warning: Removed 2 rows containing missing values (geom\_path).

As you can see, there is a huge spike somewhere in 1874. Maybe there was a very severe smoked pig  
meat shortage that caused the prices to increase dramatically, but the more likely explanation is  
that there was some sort of mistake, either in the OCR step, or when I extracted the prices, and somehow  
that particular price of smoked pig meat is actually the price of another, more expensive good.

So let’s only consider prices that are below, say, 20 franks, which is already very high:

market\_price %>%

filter(good\_en == "Smoked pig meat") %>%

filter(price < 20) %>%

ggplot(aes(x = market\_date, y = price)) +

geom\_line(aes(group = 1), colour = "#82518c") +

theme\_blog() +

labs(title = "Prices of smoked pig meat at the Luxembourg-City market in the 1870s")

Now, some prices are very high. Let’s check if it’s a mistake:

market\_price %>%

filter(good\_en == "Smoked pig meat") %>%

filter(between(price, 5, 20)) %>%

pull(source\_url)

## [1] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fbs2fs6%2Fpages%2F1/full/full/0/default.jpg"

## [2] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fd61vzp%2Fpages%2F1/full/full/0/default.jpg"

## [3] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fjdwb6m%2Fpages%2F1/full/full/0/default.jpg"

## [4] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fng14m3%2Fpages%2F1/full/full/0/default.jpg"

## [5] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fw9jdrb%2Fpages%2F1/full/full/0/default.jpg"

If you go to the first url, you will land on the first page of the newspaper. To check the table,  
you need to check the third page, by changing this part of the url “pages%2F1” to this “pages%2F3”.

You will then find the following:

As you can see, the price was 2.5, but the OCR returned 7.5. This is a problem that is unavoidable  
with OCR; there is no way of knowing a priori if characters were not well recognized. It is actually  
quite interesting how the price for smoked pig meat stayed constant through all these years.  
A density plot shows that most prices were around 2.5:

market\_price %>%

filter(good\_en == "Smoked pig meat") %>%

filter(price < 20) %>%

ggplot() +

geom\_density(aes(price), colour = "#82518c") +

theme\_blog()

What about another good, say, barley?

market\_price %>%

filter(good\_en == "Barley") %>%

ggplot(aes(x = market\_date, y = price)) +

geom\_line(aes(group = 1), colour = "#82518c") +

theme\_blog() +

labs(title = "Prices of barley at the Luxembourg-City market in the 1870s")

Here again, we see some very high spikes, most likely due to errors. Let’s try to limit the prices  
to likely values:

market\_price %>%

filter(good\_en == "Barley") %>%

filter(between(price, 10, 40)) %>%

ggplot(aes(x = market\_date, y = price)) +

geom\_line(aes(group = 1), colour = "#82518c") +

theme\_blog() +

labs(title = "Prices of barley at the Luxembourg-City market in the 1870s")

market\_price %>%

filter(good\_en == "Barley") %>%

ggplot() +

geom\_density(aes(price), colour = "#82518c") +

theme\_blog()

## Warning: Removed 39 rows containing non-finite values (stat\_density).

Let’s finish this with one of my favourite legume, lentils:

market\_price %>%

filter(good\_en == "Lentils") %>%

ggplot(aes(x = market\_date, y = price)) +

geom\_line(aes(group = 1), colour = "#82518c") +

theme\_blog() +

labs(title = "Prices of lentils at the Luxembourg-City market in the 1870s")

market\_price %>%

filter(good\_en == "Lentils") %>%

ggplot() +

geom\_density(aes(price), colour = "#82518c") +

theme\_blog()

## Warning: Removed 79 rows containing non-finite values (stat\_density).

All these 0’s might be surprising, but in most cases, they are actually true zeros! For example,  
you can check this  
[issue](https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fwsvhwh%2Fpages%2F3/pct:74,0,100,100/full/0/default.jpg).  
This very likely means that no lentils were available that day at the market.  
Let’s get rid of the 0s and other extreme values:

market\_price %>%

filter(good\_en == "Lentils") %>%

filter(between(price, 1, 40)) %>%

ggplot(aes(x = market\_date, y = price)) +

geom\_line(aes(group = 1), colour = "#82518c") +

theme\_blog() +

labs(title = "Prices of lentils at the Luxembourg-City market in the 1870s")

I would like to see if the spikes above 30 are errors or not:

market\_price %>%

filter(good\_en == "Lentils") %>%

filter(between(price, 30, 40)) %>%

pull(source\_url)

## [1] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F04mb5t%2Fpages%2F1/full/full/0/default.jpg"

## [2] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fb8zp31%2Fpages%2F1/full/full/0/default.jpg"

## [3] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fkzrj53%2Fpages%2F1/full/full/0/default.jpg"

## [4] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fs8sw2v%2Fpages%2F1/full/full/0/default.jpg"

## [5] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fsjptsk%2Fpages%2F1/full/full/0/default.jpg"

## [6] "https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2Fwk65b6%2Fpages%2F1/full/full/0/default.jpg"

The price was recognized as being 35, and turns out it was correct as you can see  
[here](https://iiif.eluxemburgensia.lu/iiif/2/ark:%2F70795%2F04mb5t%2Fpages%2F3/full/full/0/default.jpg).  
This is quite interesting, because the average price was way lower than that:

market\_price %>%

filter(good\_en == "Lentils") %>%

filter(between(price, 1, 40)) %>%

summarise(mean\_price = mean(price),

sd\_price = sd(price))

## # A tibble: 1 x 2

## mean\_price sd\_price

##

## 1 20.8 5.82

I’m going to finish here; it was an interesting project, and I can’t wait for more newspapers to be  
digitized and OCR to work even better. There is a lot more historical data trapped in these newspapers  
that could provide a lot insights on Luxembourg’s society in the 19th century.