# Wine Prices Report

Corentin Lobet

9/10/2020

## 1. Introduction : data and objectives

In this research we use data of wine reviews that can be found here on Kaggle. It consists of about 130,000 wine reviews with information about the wine and the reviewer.

This dataset has largely been used to explain and predict the grade given to the wine (ranging from 80 to 100). Here we chose to model prices because they take values from a few dollars to several thousands dollars. We expect to get low precision using wine and reviewer specs though we wonder if the written reviews ('descrption') can help to explain unusually high prices.

```
head(wine_price)
```

```
## # A tibble: 6 x 14
##      X1 country description designation points price province region_1 region_2
##   <dbl> <chr>   <chr>       <chr>        <dbl> <dbl> <chr>    <chr>    <chr>
## 1     0 Italy   Aromas inc~ Vulkà Bian~     87    NA Sicily ~ Etna     <NA>
## 2     1 Portug~ This is ri~ Avidagos        87    15 Douro    <NA>     <NA>
## 3     2 US      Tart and s~ <NA>            87    14 Oregon   Willame~ Willame~
## 4     3 US      Pineapple ~ Reserve La~     87    13 Michigan Lake Mi~ <NA>
## 5     4 US      Much like ~ Vintner's ~     87    65 Oregon   Willame~ Willame~
## 6     5 Spain   Blackberry~ Ars In Vit~     87    15 Norther~ Navarra  <NA>
## # ... with 5 more variables: taster_name <chr>, taster_twitter_handle <chr>,
## #   title <chr>, variety <chr>, winery <chr>
```

```
names(wine_price)
```

```
##  [1] "X1"                    "country"               "description"
##  [4] "designation"           "points"                "price"
##  [7] "province"              "region_1"              "region_2"
## [10] "taster_name"           "taster_twitter_handle" "title"
## [13] "variety"               "winery"
```

```
summary(wine_price[, c(5,6)]) %>% tb
```

| points | price |
|---|---|
| Min. : 80.0 | Min. : 4 |
| 1st Qu.: 86.0 | 1st Qu.: 17 |
| Median : 88.0 | Median : 25 |

| points | price |
|---|---|
| Mean : 88.5 | Mean : 35 |
| 3rd Qu.: 91.0 | 3rd Qu.: 42 |
| Max. :100.0 | Max. :3300 |
| NA | NA's :8996 |

# 2. Methodology

As we will see in the next section, this dataset needs some cleaning before we can fit models on it.

We then pick regression ML models we'll be using to predict prices. Actually we do not provide the code we used to select models since it's simple training. The process took some days as we trained several regression models included in the caret package and we picked models based on 2 criteria : performance (RMSE) and computation time. We also verified that variables we were willing to fit were significant with a naive approach of feature selection i.e. training a linear model with different features and looking at the RMSE and the R-squared.

As a result we kept 3 main models : MARS, CIT and GBM. We also present results of 2 other models (linear regression and decision tree) as they are fast to run and it gives us the opportunity to compare them with MARS and CIT that are examples of their adaptive versions.

Firstly, we fit prices against wine and taster specs only and analyze errors of the predictions. We then add word patterns as predictors in order to see if reviews add value using our models.

## 2.1. MARS - Multivariate Adaptive Regression Splines

Introduced by Friedman in 1991, this model is a non-parametric regression technique that is an extension of linear regression. The difference with LR is that the data are fitted by multiple regression lines. The first step in building this model is to fit a regression line by OLS. Then we look for a cutoff among the predictors values such that the angle of the regression line changes at this cutoff. You can learn more bout this approach here.

## 2.2. CIT - Conditional Inference Tree

The CIT is a version of decision trees is designed to correct the variable selection bias that occurs in the successive splits. It aims to avoid partitioning that are not statistically significant. You can learn more about it here.

## 2.3. GBM - Gradient Boosting Machine

The GBM fits multiple trees and create an ensemble at each step for which the prediction is a weighted average of fitted trees. The weights are determined in function of the predictive performance of the trees. It is an alternative to random forests that often do worse and are longer to compute. You can learn more about it here.

# 3. EDA and data cleaning

## 3.1. Basic Cleaning

The table below shows the number of distinct values taken by each variable. Reviews are reported from 43 countries for up to 1,229 regions. Grades ('points') have only 21 levels (80-100) while prices have nearly 400 levels (we will assume it is a continuous variable). The table also shows that there are as few as 19 tasters (actually it is maybe more because the 19th value is NA).

Some cleaning rules arise from these observations : remove 'designation' and 'winery' as there are too many levels and we think don't bring much information.Remove 'province' but keep 'country' and 'region_1' as we want to observe if there is a country effect and we don't want repetitive informations (indeed we assume there is a multicollinearity bias for province and country).

```
# levels = apply(wine_price, 2, function(x) { factor(x) %>% levels() %>% length() })
# save(levels, file = "data/levels.RData")
load("data/levels.RData"); levels %>% tb
```

| | x |
|---|---|
| X1 | 129971 |
| country | 43 |
| description | 119955 |
| designation | 37979 |
| points | 21 |
| price | 390 |
| province | 425 |
| region_1 | 1229 |
| region_2 | 17 |
| taster_name | 19 |
| taster_twitter_handle | 15 |
| title | 118840 |
| variety | 707 |
| winery | 16757 |

Then wa take a look at NA values. We can remove rows for NA countries and varieties (less than 100 observations), for prices (it's our independent variable), for taster name (we assume it is an important variable and it's hard to deal with NAs for this one).

```
apply(wine_price, 2, function(x) { sum(is.na(x)) }) %>% tb
```

| | x |
|---|---|
| X1 | 0 |
| country | 63 |
| description | 0 |
| designation | 37465 |
| points | 0 |
| price | 8996 |
| province | 63 |
| region_1 | 21247 |
| region_2 | 79460 |

| | x |
|---|---|
| taster_name | 26244 |
| taster_twitter_handle | 31213 |
| title | 0 |
| variety | 1 |
| winery | 0 |

It appears that region_1 has many NA values (>20,000). However we can see in the title variable that it often provides the region inside parentheses. We will then try to get it back from title (a variable that we won't use then by the way).

```
wine_price[1:20, c(8,12)] %>% tb
```

| region_1 | title |
|---|---|
| Etna | Nicosia 2013 Vulkà Bianco (Etna) |
| NA | Quinta dos Avidagos 2011 Avidagos Red (Douro) |
| Willamette Valley | Rainstorm 2013 Pinot Gris (Willamette Valley) |
| Lake Michigan Shore | St. Julian 2013 Reserve Late Harvest Riesling (Lake Michigan Shore) |
| Willamette Valley | Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley) |
| Navarra | Tandem 2011 Ars In Vitro Tempranillo-Merlot (Navarra) |
| Vittoria | Terre di Giurfo 2013 Belsito Frappato (Vittoria) |
| Alsace | Trimbach 2012 Gewurztraminer (Alsace) |
| NA | Heinz Eifel 2013 Shine Gewürztraminer (Rheinhessen) |
| Alsace | Jean-Baptiste Adam 2012 Les Natures Pinot Gris (Alsace) |
| Napa Valley | Kirkland Signature 2011 Mountain Cuvée Cabernet Sauvignon (Napa Valley) |
| Alsace | Leon Beyer 2012 Gewurztraminer (Alsace) |
| Alexander Valley | Louis M. Martini 2012 Cabernet Sauvignon (Alexander Valley) |
| Etna | Masseria Setteporte 2012 Rosso (Etna) |
| Central Coast | Mirassou 2012 Chardonnay (Central Coast) |
| NA | Richard Böcking 2013 Devon Riesling (Mosel) |
| Cafayate | Felix Lavaque 2010 Felix Malbec (Cafayate) |
| Mendoza | Gaucho Andino 2011 Winemaker Selection Malbec (Mendoza) |
| Ribera del Duero | Pradorey 2010 Vendimia Seleccionada Finca Valdelayegua Single Vineyard Crianza (Ribera del Duero) |
| Virginia | Quiévremont 2012 Meritage (Virginia) |

We are now ready for this first cleaning and variable picking step. The code below provides this data manipulation

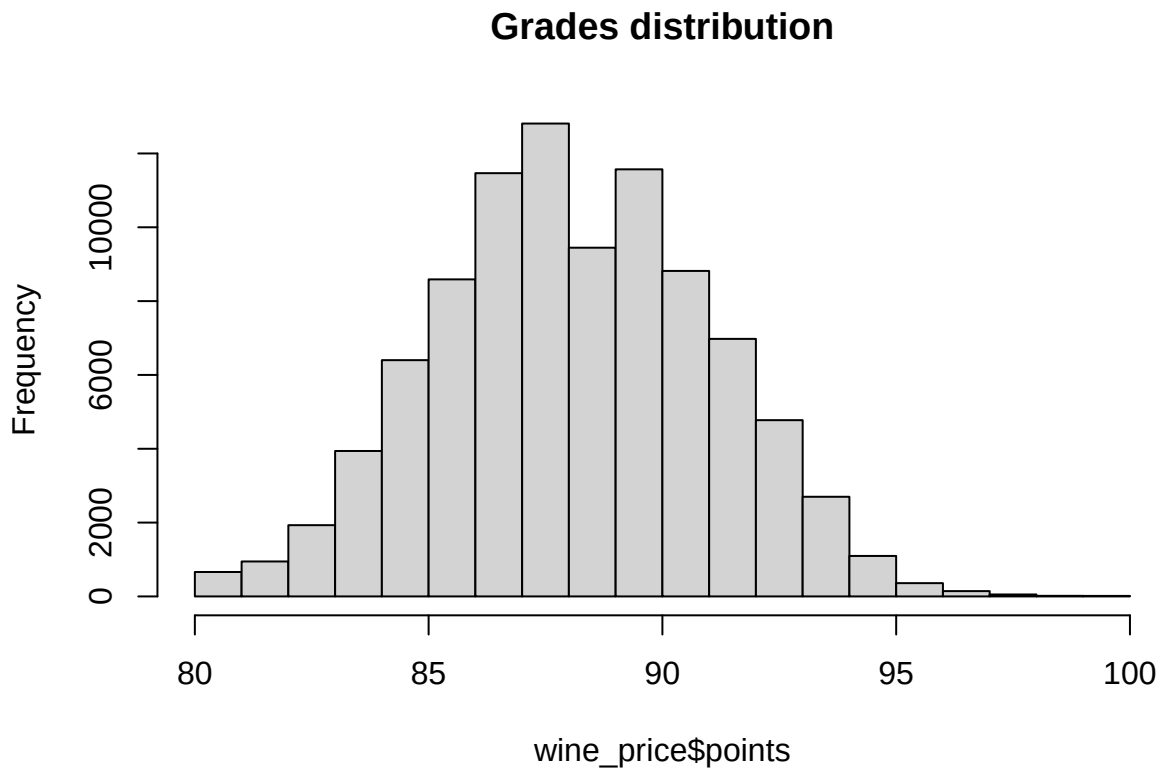
```
# Select columns and remove NAs
wine_price = wine_price %>%
   select(country, description, points, price,
          region_1, taster_name, variety, title) %>%
   filter(!is.na(price), !is.na(variety), !is.na(country), !is.na(taster_name)) %>%
   rename(region = region_1)
# Add regions from title
wine_price = wine_price %>%
```

```
   filter(!is.na(region) | str_detect(title, "\\(([:print:]+)\\)")) %>%
   mutate(region = ifelse(is.na(region), str_extract(title, "\\(([:print:]+)\\)"), region)) %>%
   mutate(region = str_remove_all(region, "\\(|\\)")) %>%
   filter(str_count(region) < 25)
wine_price = select(wine_price, -title)
```

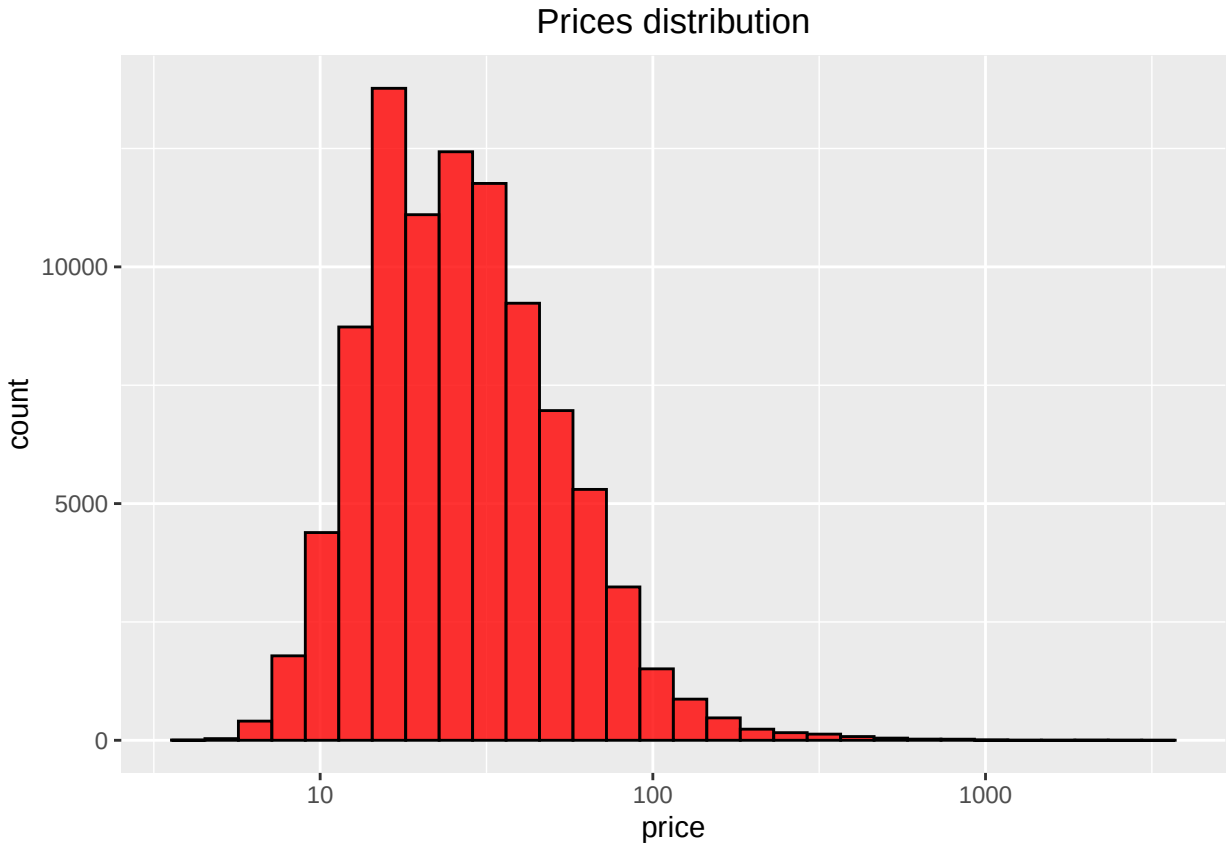
### 3.2. EDA and further data pre-processing

We can notice that prices distribution seems to be skewed (because of outliers) while ratings' scores distribution looks Gaussian. This fact makes us expect that the score won't explain well the prices. That's why we look for other variables impact.

```
# Ratings distribution
hist(wine_price$points, main = "Grades distribution")
```

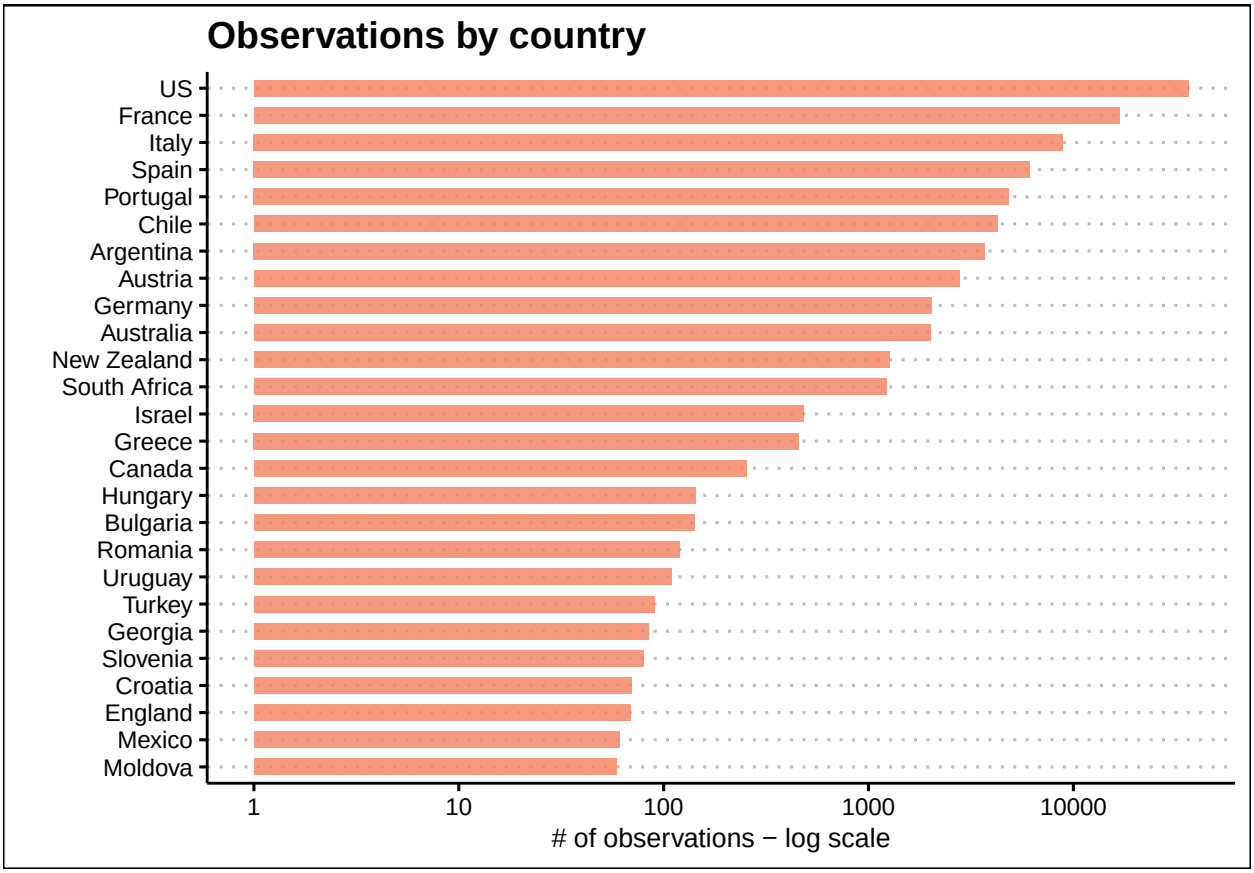



```
# Price distribution
ggplot(wine_price, aes(price)) +
   geom_histogram(color = "black", fill = "red", alpha = 0.8) +
   scale_x_log10() +
   ggtitle("Prices distribution") +
   theme(plot.title = element_text(hjust = 0.5))
```



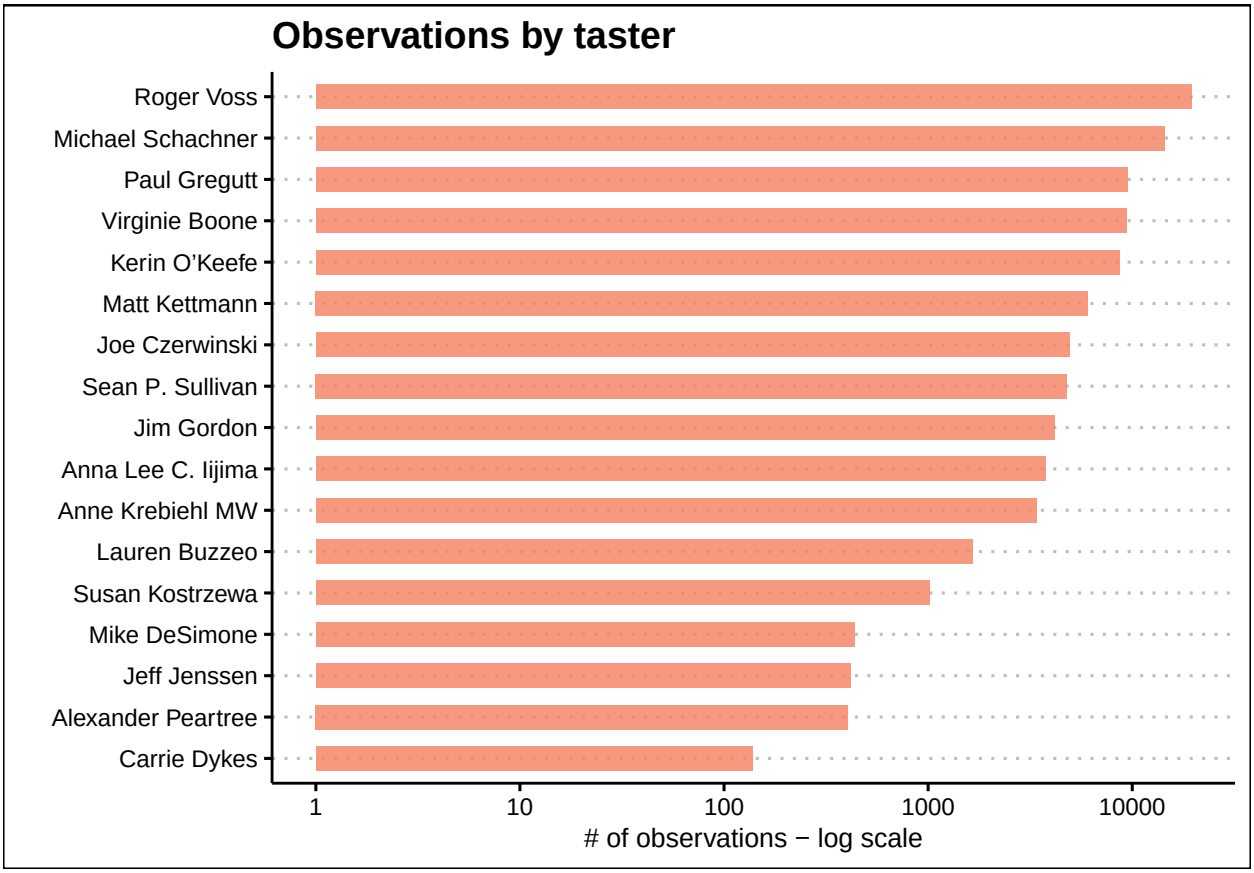

This data reveals an incredible heterogeneity of the wine production accross countries. The number of wine reviews in a country can be less than 10 or more than 10,000 (e.g. France). As we would like to compute the country effect we choose to keep countries with more than 50 observations (23 remaining countries).

```
# Keep >50 obs
temp = wine_price %>%
   group_by(country) %>%
   summarise(n = n())
countries = temp$country[which(temp$n >= 50)]
wine_price = wine_price %>%
   filter(country %in% countries)
rm(temp, countries)
# Distribution
wine_price %>%
   group_by(country) %>%
   summarise(n = n()) %>%
   mutate(country = reorder(country, n)) %>%
   ggplot(aes(y = country, x = n)) +
   geom_bar(stat = "identity", fill = "#f68060", alpha = 0.8, width = 0.6) +
   scale_x_log10() +
   xlab("# of observations - log scale") +
   ylab("") + ggtitle("Observations by country") +
   ggthemes::theme_clean()
```



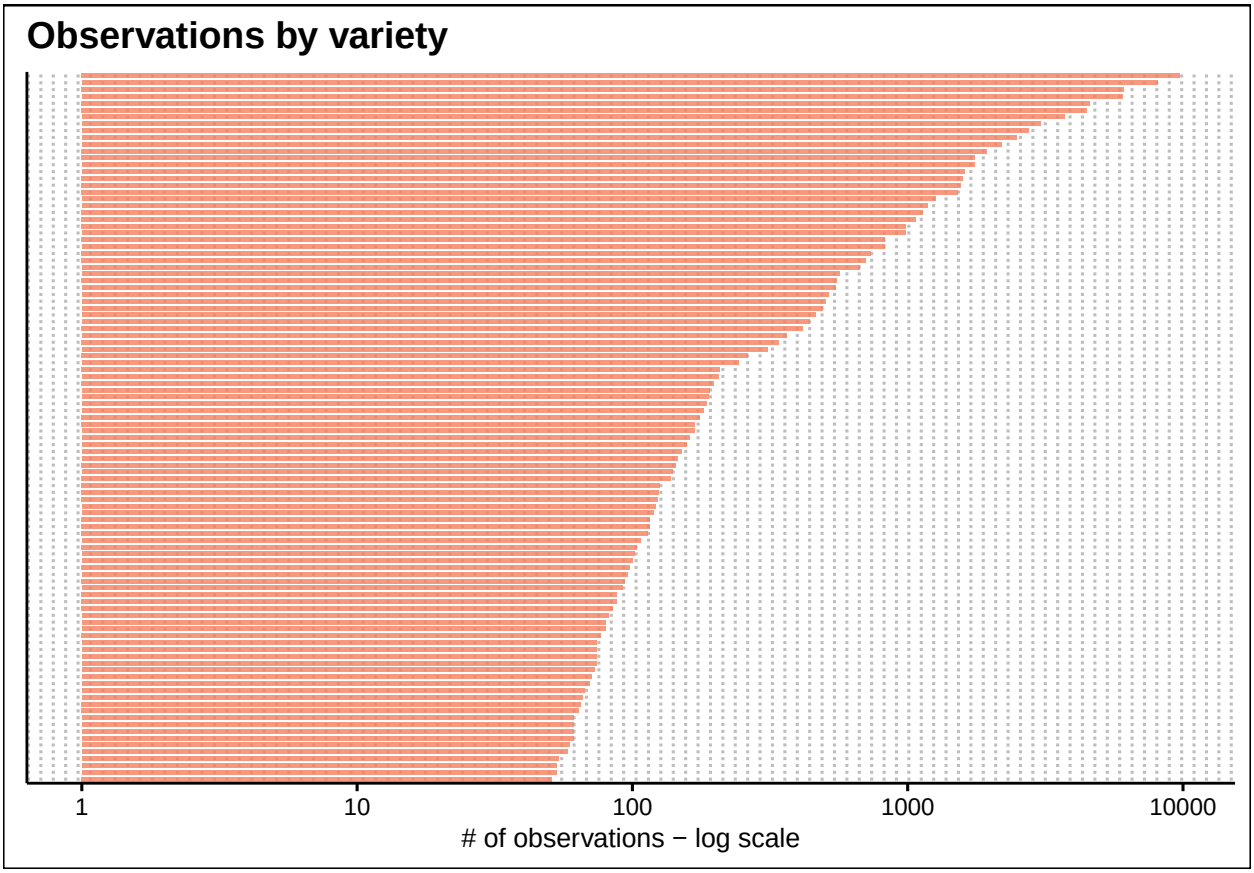

We do the same by tasters which reduces to 17 tasters each having reviewed more than 100 wines and some of them having tasted thousands.

```
temp = wine_price %>%
   group_by(taster_name) %>%
   summarise(n = n())
tasters = temp$taster_name[which(temp$n >= 50)]
wine_price = wine_price %>%
   filter(taster_name %in% tasters)
rm(temp, tasters)
wine_price %>%
   group_by(taster_name) %>%
   summarise(n = n()) %>%
   mutate(taster_name = reorder(taster_name, n)) %>%
   ggplot(aes(y = taster_name, x = n)) +
   geom_bar(stat = "identity", fill = "#f68060", alpha = 0.8, width = 0.6) +
   scale_x_log10() +
   xlab("# of observations - log scale") +
   ylab("") + ggtitle("Observations by taster") +
   ggthemes::theme_clean()
```




Same process for varieties.

```
temp = wine_price %>%
   group_by(variety) %>%
   summarise(n = n())
varieties = temp$variety[which(temp$n >= 50)]
wine_price = wine_price %>%
   filter(variety %in% varieties)
rm(temp, varieties)
wine_price %>%
   group_by(variety) %>%
   summarise(n = n()) %>%
   mutate(variety = reorder(variety, n)) %>%
   ggplot(aes(y = variety, x = n)) +
   geom_bar(stat = "identity", fill = "#f68060", alpha = 0.8, width = 0.6) +
   scale_x_log10() +
   ggthemes::theme_clean() +
   xlab("# of observations - log scale") +
   ylab("") + theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()) +
   ggtitle("Observations by variety")
```




It remains 104 wine varieties in the dataset. The table below shows the most frequent ones.

```
wine_price$variety %>% n_distinct()
```

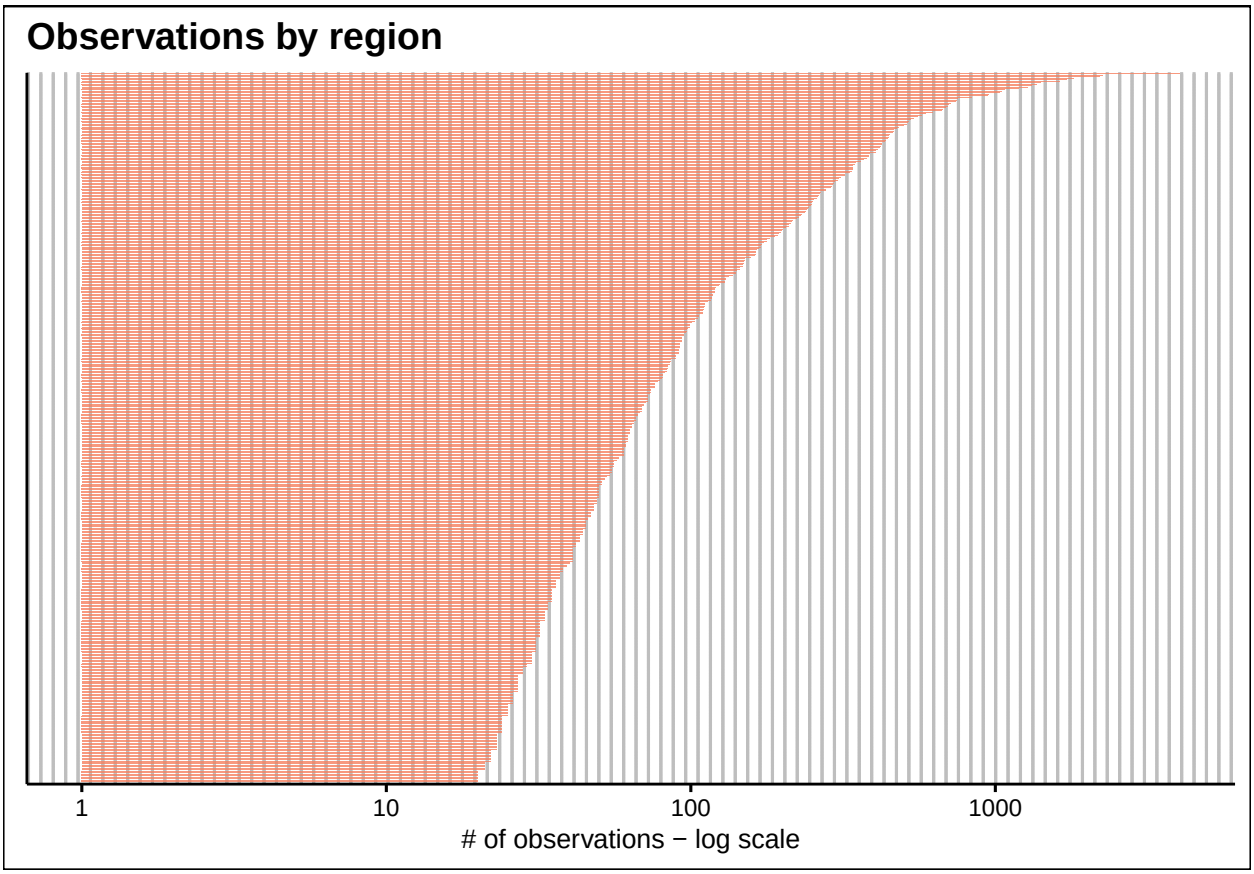
```
## [1] 104
```

```
wine_price %>%
   group_by(variety) %>%
   summarise(n = n()) %>%
   arrange(desc(n)) %>%
   head() %>% tb
```

| variety | n |
|---|---|
| Pinot Noir | 9764 |
| Chardonnay | 8099 |
| Red Blend | 6113 |
| Cabernet Sauvignon | 6020 |
| Riesling | 4597 |
| Bordeaux-style Red Blend | 4483 |

Finally we repeat the cleaning process for regions but this time with a threshold of 20. Most regions have less than 100 observations.

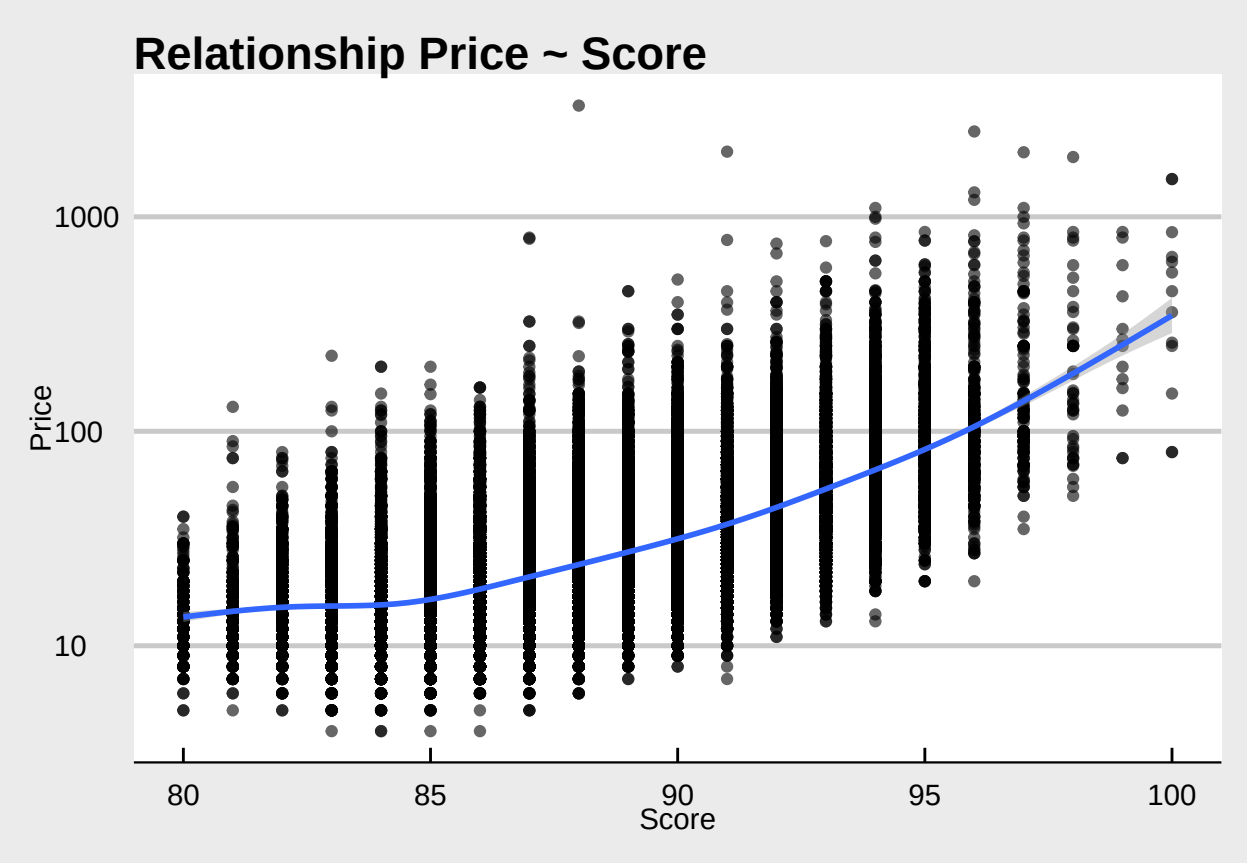
```r
temp = wine_price %>%
   group_by(region) %>%
   summarise(n = n())
regions = temp$region[which(temp$n >= 20)]
wine_price = wine_price %>%
   filter(region %in% regions)
rm(temp)
wine_price %>%
   group_by(region) %>%
   summarise(n = n()) %>%
   mutate(region = reorder(region, n)) %>%
   ggplot(aes(y = region, x = n)) +
   geom_bar(stat = "identity", fill = "#f68060", alpha = 0.8, width = 0.6) +
   scale_x_log10() +
   ggthemes::theme_clean() +
   xlab("# of observations - log scale") +
   ylab("") + theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()) +
   ggtitle("Observations by region")
```




### 3.3. Look at relationships

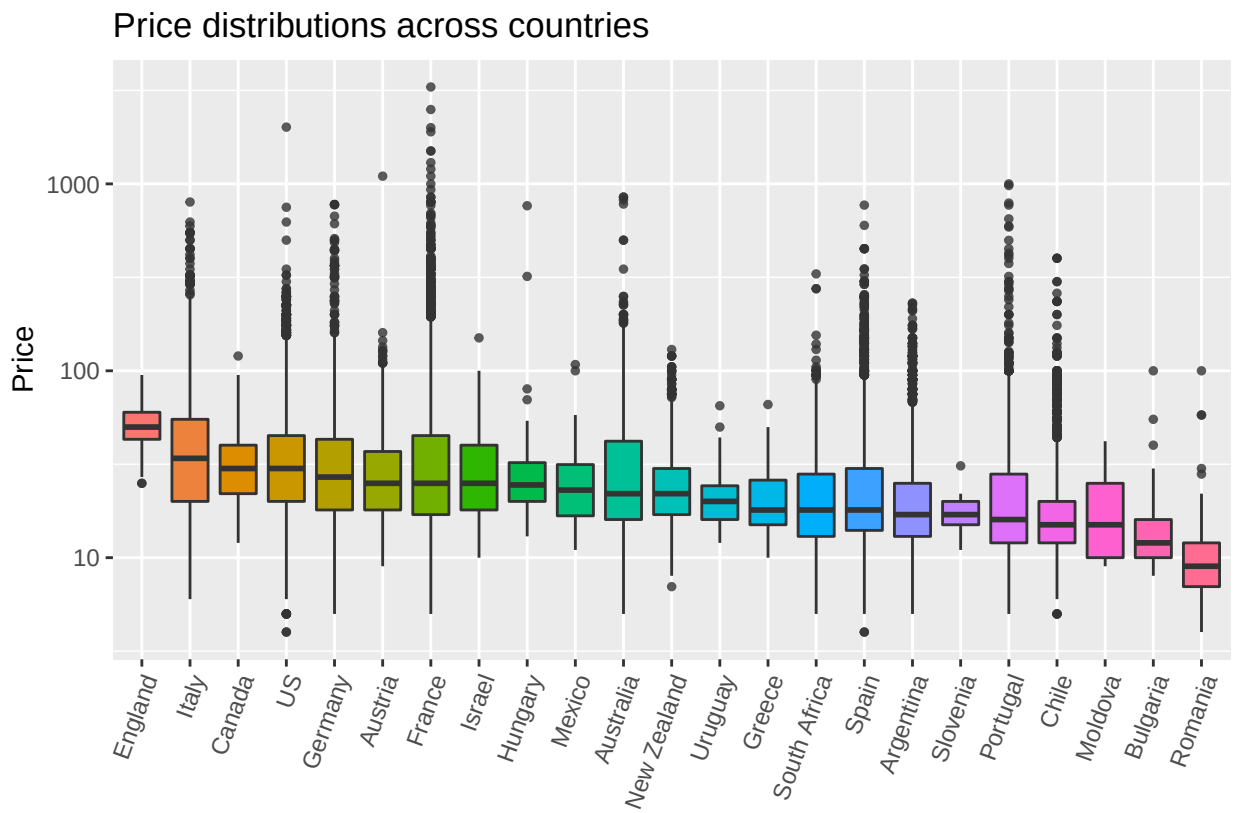
Score effect : it seems that there is a log-lin relationship between price and score though the score doesn't explain outsiders i.e. luxury wines.

```
# wine_price %>%
#     ggplot(aes(points, price)) +
#     geom_point(alpha = 0.6) +
#     geom_smooth() +
#     ggthemes::theme_economist_white() +
#     xlab("Score") + ylab("Price") +
#     scale_y_log10()
load("data/plot1.RData"); plot1
```




Country effect : the heterogeneity is striking. Some countries have tiny price range while others have many outliers. Actually these countries are well known for their wines (France, Italy, ...).

```
wine_price %>%
   group_by(country) %>%
   mutate(median = median(price)) %>%
   ggplot(aes(reorder(country, desc(median)), price, fill = reorder(country, desc(median)))) +
   geom_boxplot(show.legend = F, outlier.size = 1, outlier.alpha = 0.8) +
   scale_y_log10() +
   theme(axis.text.x = element_text(angle = 70, hjust = 1)) +
   xlab("") + ylab("Price") +
   ggtitle("Price distributions across countries")
```




Region effect : we will study the region effect in countries with the most regions recorded for wines : France and USA.

```
wine_price %>%
   group_by(country) %>%
   summarise(regions = n_distinct(region)) %>%
   arrange(desc(regions))
```

```
## # A tibble: 23 x 2
##    country      regions
##    <chr>          <int>
##  1 France           128
##  2 US               125
##  3 Italy             60
##  4 Spain             33
##  5 Australia         21
##  6 Chile             20
##  7 Argentina         19
##  8 Austria           16
##  9 Portugal          14
## 10 South Africa      10
## # ... with 13 more rows
```

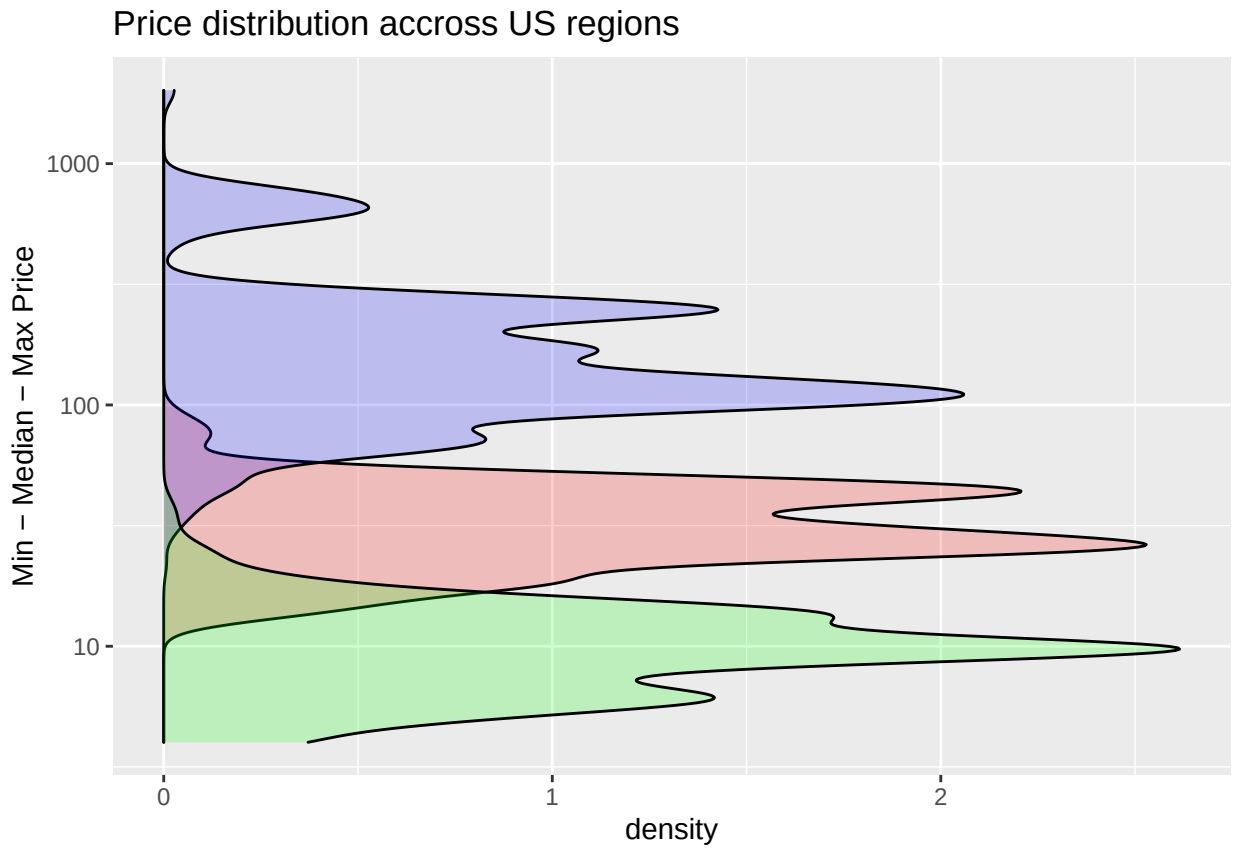
**USA**

```
temp = wine_price %>%
   filter(country == "US") %>%
   group_by(region) %>%
   mutate(median = median(price),
          max = max(price),
          min = min(price))
summary(temp[, 8:10])
```

```
##      median          max            min
##  Min.   :14.0   Min.   :  22   Min.   : 4.0
##  1st Qu.:25.0   1st Qu.: 100   1st Qu.: 7.0
##  Median :30.0   Median : 125   Median :10.0
##  Mean   :32.6   Mean   : 200   Mean   :10.2
##  3rd Qu.:42.0   3rd Qu.: 240   3rd Qu.:13.0
##  Max.   :90.0   Max.   :2013   Max.   :40.0
```

These 3 density plots below show the distributions of the min, median and max prices across USA regions. Median price ranges between 14 and 90 which is very large. Some regions produce expansive wines. Min price ranges from 4 to 40. Max price rockets to 2,013 while some regions have max price lower than 100 (minmax = 22).

```
temp %>% ggplot(aes(median)) +
   geom_density(fill = "red", alpha = 0.2, bw = 0.05) +
   geom_density(aes(max), fill = "blue", alpha = 0.2, bw = 0.05) +
   scale_x_log10() +
   geom_density(aes(min), fill = "green", alpha = 0.2, bw = 0.05) +
   coord_flip() +
   xlab("Min - Median - Max Price") +
   ggtitle("Price distribution accross US regions")
```

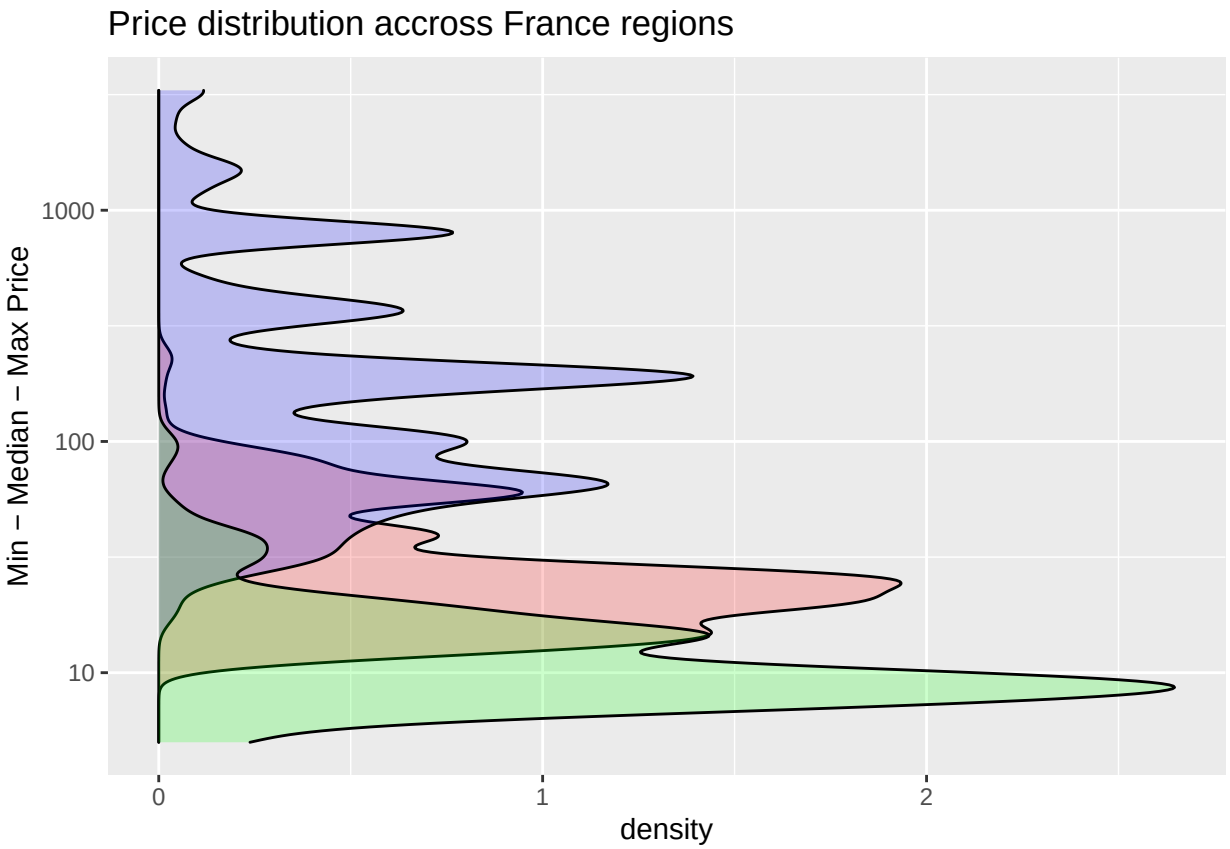



**France**

```
temp = wine_price %>%
   filter(country == "France") %>%
   group_by(region) %>%
   mutate(median = median(price),
          max = max(price),
          min = min(price))
summary(temp[, 8:10])
```

```
##      median           max            min
##  Min.   : 12.0   Min.   :  17   Min.   :  5.0
##  1st Qu.: 19.0   1st Qu.:  65   1st Qu.:  8.0
##  Median : 25.0   Median : 141   Median : 10.0
##  Mean   : 32.7   Mean   : 318   Mean   : 13.4
##  3rd Qu.: 40.0   3rd Qu.: 350   3rd Qu.: 15.0
##  Max.   :231.0   Max.   :3300   Max.   :100.0
```

```
temp %>% ggplot(aes(median)) +
   geom_density(fill = "red", alpha = 0.2, bw = 0.05) +
   geom_density(aes(max), fill = "blue", alpha = 0.2, bw = 0.05) +
   scale_x_log10() +
   geom_density(aes(min), fill = "green", alpha = 0.2, bw = 0.05) +
   coord_flip() +
   xlab("Min - Median - Max Price") +
   ggtitle("Price distribution accross France regions")
```




In France the median goes up to 231 and there are 3 regions showing a median price above 150.

```
temp %>% filter(median > 150) %>% .$region %>% unique()
```

```
## [1] "Clos de Vougeot"    "Corton-Charlemagne" "Echézeaux"
```

Five regions have a minimum price above 50. And 4 regions have a maximum price below 25. Therefore, as it can be deduced from the previous plot, the price heterogeneity across regions can be significantly different between countries. That's a good argument for fitting ML models with both variables.

```
temp %>% filter(min > 50) %>% .$region %>% unique()
```

```
## [1] "Clos de Vougeot"    "Condrieu"           "Corton-Charlemagne"
## [4] "Corton"             "Echézeaux"
```

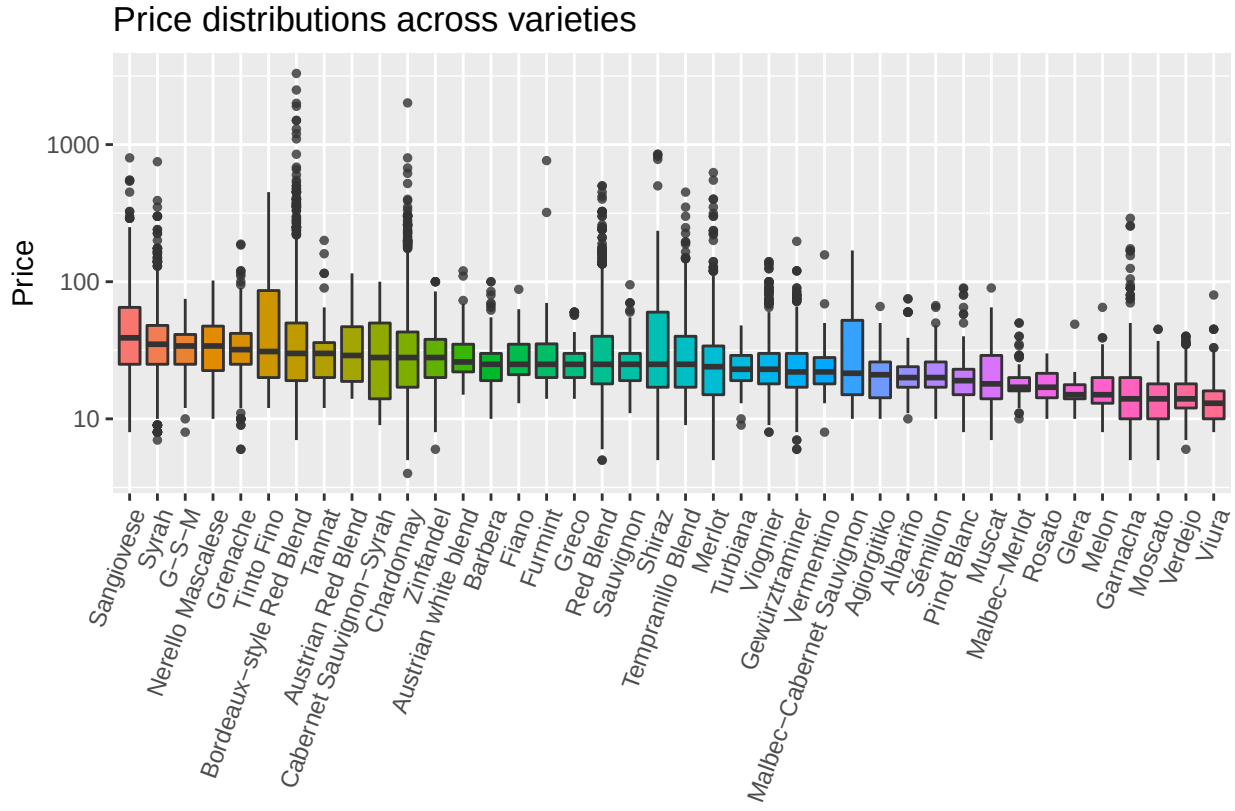
```
temp %>% filter(max < 25) %>% .$region %>% unique()
```

```
## [1] "Entre-Deux-Mers" "Mediterranée"    "Val de Loire"    "Rosé d'Anjou"
```

```
rm(temp)
```

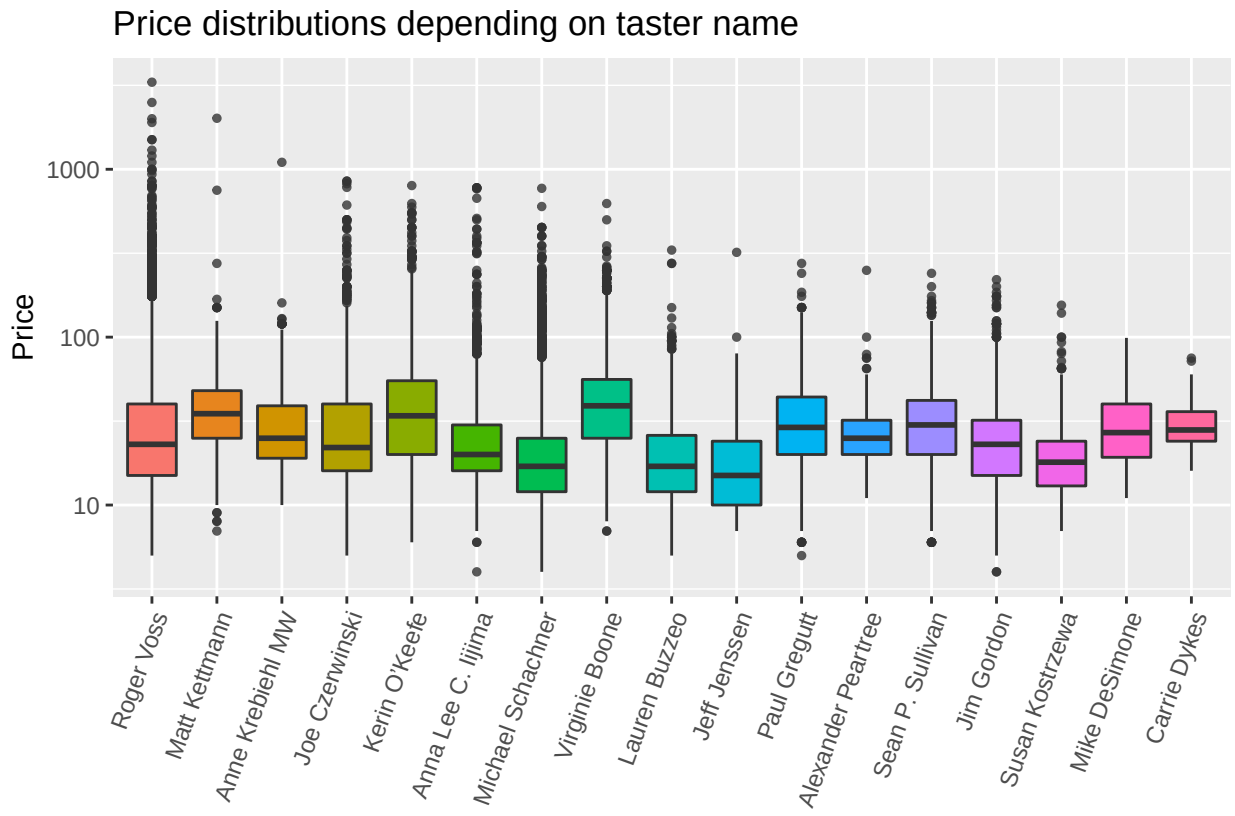
Variety effect : We see less variability across varieties and we expect the effect to be minor. For visualization comfort purposes we randomly select 40 of the 104 varieties.

```
seed(10)
selection = sample(unique(wine_price$variety), 40)
wine_price %>%
   filter(variety %in% selection) %>%
   group_by(variety) %>%
   mutate(median = median(price)) %>%
   ggplot(aes(reorder(variety, desc(median)), price, fill = reorder(variety, desc(median)))) +
   geom_boxplot(show.legend = F, outlier.size = 1, outlier.alpha = 0.8) +
   scale_y_log10() +
   theme(axis.text.x = element_text(angle = 70, hjust = 1)) +
   xlab("") + ylab("Price") +
   ggtitle("Price distributions across varieties")
```




Taster effect : We do not observe strong variability in medians and interquartiles but it seems that some reviewers allow for higher prices, maybe due to their reputation. For example, Roger Voss, who made 20% of the reviews for the remaining data, is related to wines that have a low median price though many outliers.

```
wine_price %>%
   group_by(taster_name) %>%
   mutate(max = max(price)) %>%
   ggplot(aes(reorder(taster_name, desc(max)), price, fill = reorder(taster_name, desc(max)))) +
   geom_boxplot(show.legend = F, outlier.size = 1, outlier.alpha = 0.8) +
   scale_y_log10() +
   theme(axis.text.x = element_text(angle = 70, hjust = 1)) +
   xlab("") + ylab("Price") +
   ggtitle("Price distributions depending on taster name")
```




## 4. Fitting models on basic predictors

At this point we have 5 predictors to fit prices : country, region, variety, score and taster. In this section we train the 5 selected models and show the results of predictions (RMSEs). For the same reason mentioned in the Methodology section, we do not provide the code for parameter tuning.

Code for partitionning data :

```
# Load and coerce to factors
load(file = "data/wine_price_2.RData")
wine_price = wine_price %>%
   mutate(country = factor(country),
          region = factor(region),
          variety = factor(variety),
          taster_name = factor(taster_name))
# RMSE function
RMSE = function(actual, predicted) sqrt(mean((actual - predicted)^2))
# Split into training and validation sets
seed(66)
ind = createDataPartition(wine_price$price, p = 0.15, list = F)
wp_train = wine_price[-ind,]; validation = wine_price[ind,]
validation = validation %>% filter(region %in% wp_train$region)
```

### 4.1. Models results :

(We do not run the code again as it may take a lot of time)

```
# seed(66)
# wp_train = wine_price[-ind,]; validation = wine_price[ind,]
# validation = validation %>% filter(region %in% wp_train$region)
#
# lm_fit = lm(log(price) ~ points + country + region + variety + taster_name,
#             data = wp_train)
# metrics = data.frame(Model = "lm",
#                      RMSE = RMSE(predict(lm_fit, validation) %>% exp(), validation$price))
# tree_fit = rpart(log(price) ~ points + country + region + variety + taster_name,
#                  data = wp_train, cp = 0.001)
# metrics = rbind(metrics, c("tree",
#                            RMSE(predict(tree_fit, validation) %>% exp(), validation$price)))
# ctree_fit = party::ctree(log(price) ~ points + country + region + variety + taster_name,
#                          data = wp_train, controls = party::ctree_control(mincriterion = 0.01))
# metrics = rbind(metrics, c("ctree",
#                            RMSE(predict(ctree_fit, validation) %>% exp(), validation$price)))
# earth_fit = earth::earth(log(price) ~ points + country + region + variety + taster_name,
#                          data = wp_train)
# metrics = rbind(metrics, c("earth",
#                            RMSE(predict(earth_fit, validation) %>% exp(), validation$price)))
# gbm_fit = gbm::gbm(log(price) ~ points + country + region + variety + taster_name,
#                    data = wp_train, n.trees = 150,
#                    shrinkage = 0.25, interaction.depth = 8, n.minobsinnode = 5)
# metrics = rbind(metrics, c("gbm",
#                            RMSE(predict(gbm_fit, validation) %>% exp(), validation$price)))
# metrics[,2] = metrics[,2] %>% as.numeric()
load("data/metrics.RData")
metrics %>% knitr::kable(align = "c")
```

| Model | RMSE |
|:---:|:---:|
| lm | 27.81 |
| tree | 27.60 |
| ctree | 27.53 |
| earth | 28.04 |
| gbm | 24.60 |

With 'lm' the linear regression, 'tree' the standard regression tree, 'ctree' the conditional inference tree, 'earth' the MARS model and 'gbm' the GBM. More details about these models are given in the Methodology section.

We see that linear regression and the two tree methods have similar performance. What's striking though is that the MARS model, which is more adaptive than the linear regression, performs worse than it. The GBM is the best performing model here (>10% better based on RMSE).

Now let's build ensemble models : the first predicts prices by computing the average predicted prices of the 5 models. The second ensemble performs a weigted average of the 5 models predictions in order to give more weight to better performing models.

What we can observe in the table below is that the ensembles greatly increase performance compared to 4

models but they don't do better than the GBM. However giving even more weight to the GBM will make converge the RMSE. We conclude that the GBM is the best model we have fitted.

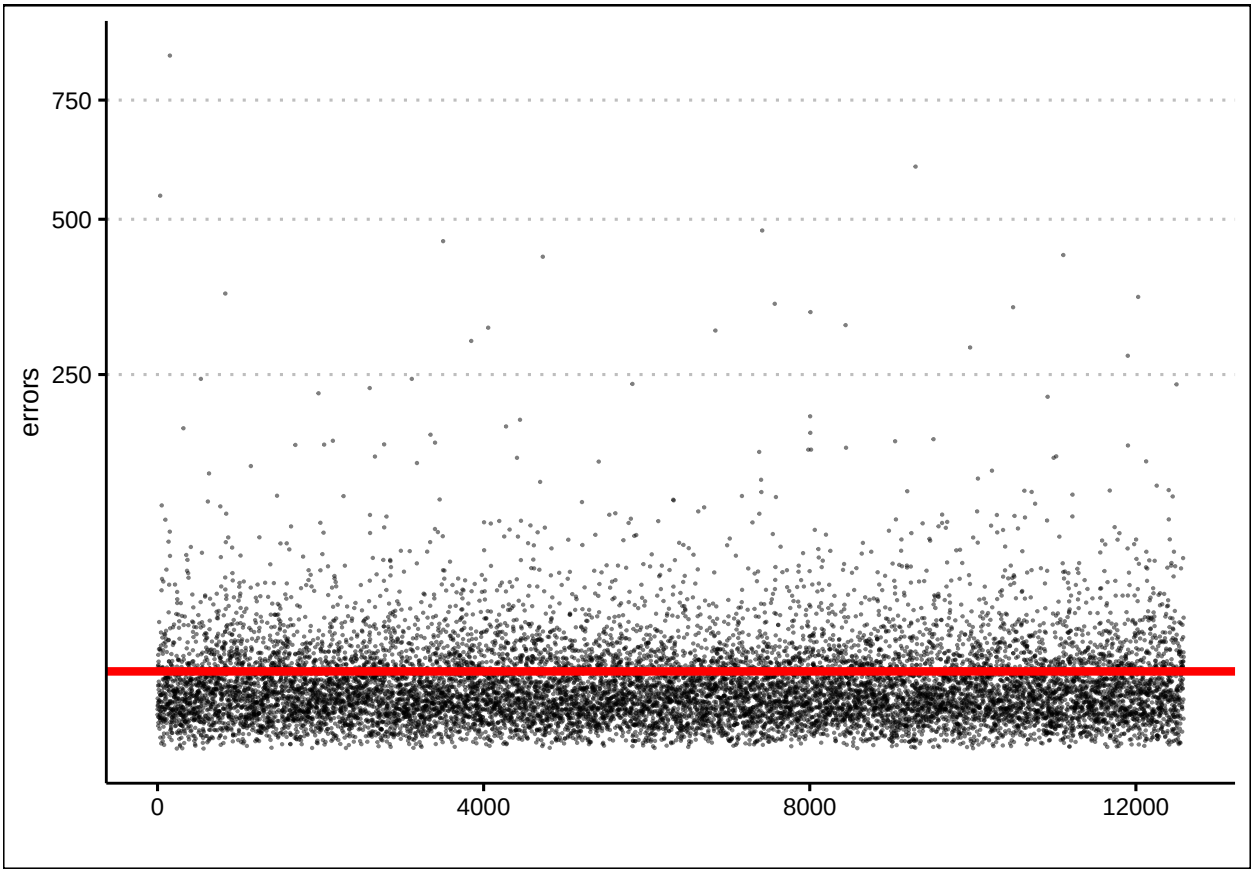
```
# predictions set
load("data/lm.RData"); load("data/tree.RData"); load("data/ctree.RData")
load("data/earth.RData"); load("data/gbm.RData")
predictions = data.frame(lm, tree, ctree, earth, gbm)
# ensemble 1
ens_arit = predictions %>% as.matrix() %>%
   rowMeans()
# ensemble 2
weights = 1/metrics$RMSE^10
ens_weighted = apply(predictions %>% as.matrix, 1, function(x){
   (x %*% weights %>% t()) / sum(weights)
})
# display results
metrics = rbind(metrics,
                c("Naive Ens", RMSE(ens_arit, validation$price)),
                c("Weighted Ens", RMSE(ens_weighted, validation$price)))
metrics[,2] = metrics[,2] %>% as.numeric()
metrics %>% knitr::kable(align = "c")
```

| Model | RMSE |
|---|---|
| lm | 27.81 |
| tree | 27.60 |
| ctree | 27.53 |
| earth | 28.04 |
| gbm | 24.60 |
| Naive Ens | 25.71 |
| Weighted Ens | 25.05 |

## 4.2. Errors analysis

The chart below shows prediction errors we obtained with the weighted ensemble model. The red line is the average error. The errors scale is log-2 transformed. What we see is that the average error is pulled up by very large errors i.e. unusually high wine prices. Our model is not capable of predicting large prices.

```
errors = sqrt((ens_weighted - validation$price)^2)
data.frame(errors) %>%
   ggplot(aes(1:length(errors), errors)) +
   geom_point(alpha = 0.5, size = 0.1) +
   xlab("") + scale_y_continuous(trans = "sqrt") +
   geom_hline(yintercept = mean(errors), color = "red", lwd = 1.5) +
   ggthemes::theme_clean()
```

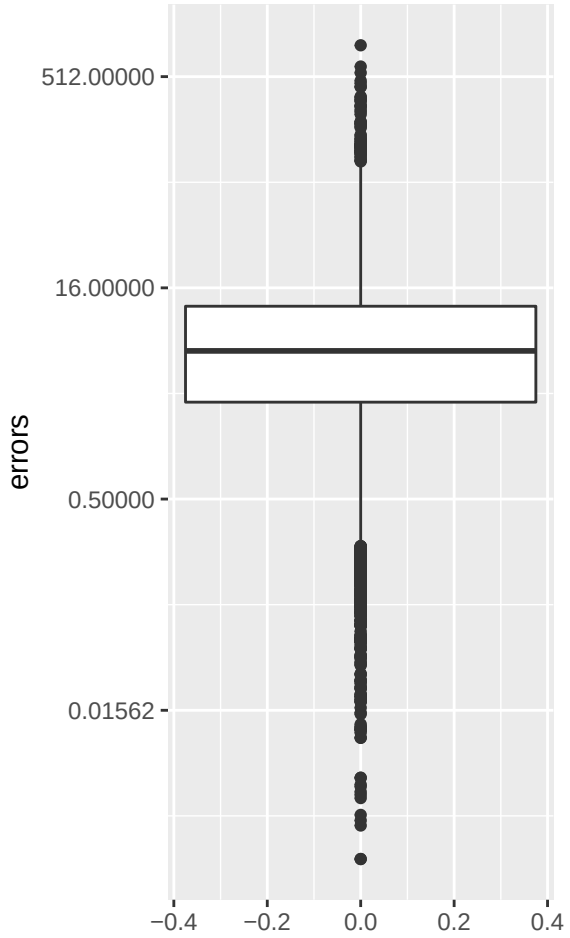



Indeed errors range from nearly 0 to more than 850.

```
errors %>% range()
```

```
## [1]   0.001361 856.347581
```

```
errors %>% data.frame() %>%
   ggplot(aes(y = errors)) +
   geom_boxplot() +
   scale_y_continuous(trans = "log2")
```




As we said, the mean is propulsed by outliers : only 25.87% of errors are above their mean (11.5), 2.7% are above 50 and less than 1% are above 100.

```
mean(errors > 11.5)
```

```
## [1] 0.2587
```

```
mean(errors > 50)
```

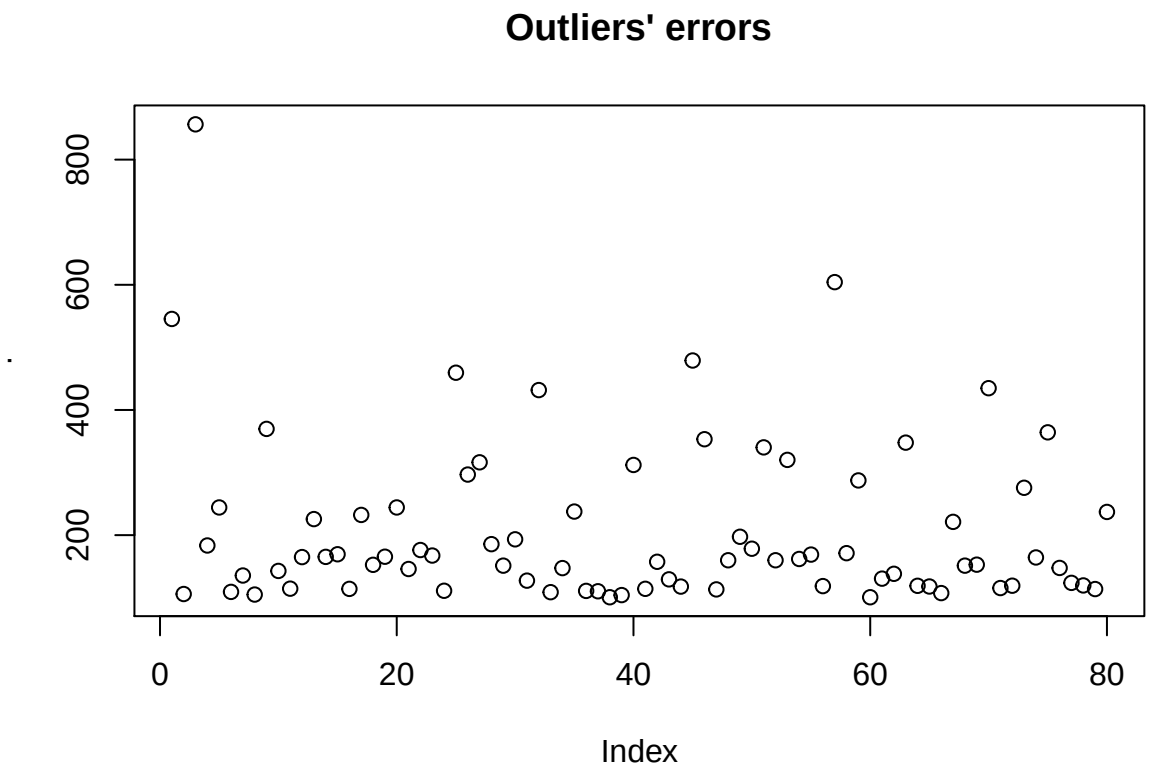
```
## [1] 0.02701
```

```
mean(errors > 100)
```

```
## [1] 0.006356
```

The plot below shows errors taht are above 100$.Some go above 400.

```
errors[errors > 100] %>% plot(main = "Outliers' errors")
```




# 5. Implementing descriptions as a predictor

## 5.1. Exctracting words from reviews

Now we want to add written reviews to the set of predictors so that we can see if (at least using our models) it can improve the predicting performance. Reviews are stored in the variable 'description' :

```
load(file = "data/wine_price_2.RData")
descriptions = wine_price$description

head(descriptions)
```

```
## [1] "This is ripe and fruity, a wine that is smooth while still structured. Firm tannins are filled
## [2] "Tart and snappy, the flavors of lime flesh and rind dominate. Some green pineapple pokes through
## [3] "Pineapple rind, lemon pith and orange blossom start off the aromas. The palate is a bit more opu
## [4] "Much like the regular bottling from 2012, this comes across as rather rough and tannic, with rus
## [5] "This dry and restrained wine offers spice in profusion. Balanced with acidity and a firm texture
## [6] "Savory dried thyme notes accent sunnier flavors of preserved peach in this brisk, off-dry wine.
```

Because we want our predictors to be word patterns, the first thing we need to do is to split descriptions in words. The following code is designed to achieve that. There is a total of 3,449,502 words (i.e. approximately 41 per description) and 61,159 unique words.

```r
# Remove some symbols
descriptions = descriptions %>%
   str_remove_all("[\\.|\\,|\\;|\\:|\\(|\\)|\\?|\\!|\\-|\\'|\"]")

words = descriptions %>% str_split(" ") %>% unlist()

length(words); unique(words) %>% length()
```

```
## [1] 3449502
```

```
## [1] 41603
```

We want to remove stop words from it since we assume they have no impact on prices. Only 600 unique words have been removed but it represents a total of 1,341,657 words.

```r
data("stop_words", package = "tidytext")
words = words[!(words %in% stop_words$word)]

length(words); unique(words) %>% length()
```

```
## [1] 2073704
```

```
## [1] 41024
```

Now we analyze the frequency of each remaining word in the data. Actually we get 48 problems of words not detected in the description variable. Because this number of issues is very low we decide to remove them from words.

```r
# wine_price = wine_price %>%
#    mutate(description = description %>%
#              str_remove_all("[\\.|\\,|\\;|\\:|\\(|\\)|\\?|\\!|\\-|\\'|\"]"))

# freq = sapply(words, function(word){
#    str_detect(wine_price$description, word) %>%
#       sum()
# })

load("data/freq.RData")
sum(freq == 0)
```

```
## [1] 48
```

```r
words = words[freq != 0]
```

## 5.2. Analyze words impact

Before we can analyze words impact on prices we need to remove words that are not in the training set since we won't train our model with the validation set. That's what the following code is designed for :

```
# index = logical(length(words))
# for(i in 1:length(words)){
#    print(i)
#    sum = wp_train %>%
#       mutate(word = ifelse(str_detect(description, words[i]), 1, 0)) %>%
#       .$word %>% sum()
#    index[i] = (sum != 0)
# }
# words = words[index]

load("data/words.RData")
```

In order to quantify the impact of words we compute a linear regression of prices against every word. We therefore store the beta coefficient and its p-value in a matrix.

```
# mat = matrix(0, length(words), 2)
# for(i in 1:length(words)) {
#    lm = (wp_train %>%
#             mutate(word = ifelse(str_detect(description, words[i]), 1, 0)) %>%
#             lm(price ~ word, data = .) %>%
#             summary())[[4]][2,]
#    mat[i,] = c(lm[1], lm[4])
# }

# colnames(mat) = c("beta", "pvalue")
# mat[,1] = round(mat[,1], 3); mat[,2] = round(mat[,2], 3)

# head(mat)
# save(mat, file = "data/words_reg.RData")

load("data/words_reg.RData")
head(mat)
```

```
##         beta pvalue
## [1,]   0.277  0.415
## [2,]   3.975  0.000
## [3,] -10.516  0.000
## [4,]   5.474  0.000
## [5,]   0.774  0.270
## [6,]  15.929  0.000
```

We can see the top impactful words i.e. those with the highest absolute beta. We filter with a p-value below 5%.

```
mat %>% as.data.frame() %>%
   mutate(word = words) %>%
   filter(pvalue < 0.05) %>%
   mutate(type = ifelse(beta >= 0, "+", "-")) %>%
   mutate(beta = abs(beta)) %>%
   arrange(desc(beta)) %>%
   .$word %>% .[1:10]
```

```
##  [1] "rainwet"         ""rescued""       "recesses"        "90th"
```

```
##  [5] "Laville"          "emotions"         "embryonic"        "vintages-expect"
##  [9] "Yin"              "Granges"
```

## 5.3. Fitting models with word patterns

Before training our models we need to create word predictors. Firstly we load the data we need :

```
# Load datasets
load("data/words_reg.RData")
load("data/words.RData")
seed(66)
ind = createDataPartition(wine_price$price, p = 0.15, list = F)
wp_train = wine_price[-ind,]; validation = wine_price[ind,]
validation = validation %>% filter(region %in% wp_train$region)

# Remove symbols in descriptions
wp_train = wp_train %>%
   mutate(description = description %>%
             str_remove_all("[\\.|\\,|\\;|\\:|\\(|\\)|\\?|\\!|\\-|\\'|\"]"))
validation = validation %>%
   mutate(description = description %>%
             str_remove_all("[\\.|\\,|\\;|\\:|\\(|\\)|\\?|\\!|\\-|\\'|\"]"))
```

Then we pick 3000 words among all the words we have computed the effect. We choose to keep p-values above 5% and we want the words to appear at least 10 times in all the reviews for the training set.

```
# keep pvalue < 5%
words_reg = mat %>% as.data.frame() %>%
   mutate(word = words) %>%
   filter(pvalue < 0.05) %>%
   mutate(type = ifelse(beta >= 0, "+", "-")) %>%
   mutate(beta = abs(beta))

# Compute their frequency
# occurences = numeric(nrow(words_reg))
# for(i in 1:nrow(words_reg)){
#    occurences[i] = wp_train %>%
#       mutate(word = ifelse(str_detect(description, words_reg$word[i]), 1, 0)) %>%
#       .$word %>% sum()
# }
load("data/occurences1.RData")

# Keep the first 3000 after sorting by impact
words_reg = words_reg %>%
   mutate(occurences) %>%
   filter(occurences > 10,
          !(word %in% colnames(wine_price))) %>%
   arrange(desc(beta)) %>%
   .$word %>% .[1:3000]
```

We do not want to have 3000 predictors though. That's why we create 100 clusters of 30 words. We keep the beta ordering for clusters making so that all clusters have different range of betas. Thus the first cluster

of words contains the 30 first words in terms of beta. We then create 100 logical predictors for each word groups (TRUE if at least one of the 30 words of the group is in the description).

```
### Create words clusters
# words_groups = split(words_reg, sapply(1:100, function(x){rep(x, 30)}))
# patterns = c()
# for(i in 1:length(words_groups)) {
#    patterns[i] = paste0(words_groups[[i]], collapse = "|")
# }

### Create 100 logical predictors for word detection
# for(i in 1:length(patterns)){
#    wp_train = wp_train %>%
#       mutate(ifelse(str_detect(description, patterns[i]), 1, 0) %>%
#                 factor(levels = c(0,1)))
#    validation = validation %>%
#       mutate(ifelse(str_detect(description, patterns[i]), 1, 0) %>%
#                 factor(levels = c(0,1)))
#    colnames(wp_train)[i+7] = paste0("pattern", as.character(i))
#    colnames(validation)[i+7] = paste0("pattern", as.character(i))
# }
# wp_train = wp_train[, -2]; validation = validation[, -2]

load("data/validation.RData"); load("data/wp_train.RData")
```

Now we are ready to fit prices with all the variables. The results are shown in the table below :

```
# lm_fit = lm(log(price) ~ ., data = wp_train)
# metrics2 = data.frame(Model = "lm",
#                       RMSE = RMSE(predict(lm_fit, validation) %>% exp(), validation$price))
# tree_fit = rpart(log(price) ~ ., data = wp_train, cp = 0.001)
# metrics2 = rbind(metrics2, c("tree",
#                              RMSE(predict(tree_fit, validation) %>% exp(), validation$price)))
# ctree_fit = party::ctree(log(price) ~ ., data = wp_train,
#                          controls = party::ctree_control(mincriterion = 0.01))
# metrics2 = rbind(metrics2, c("ctree",
#                              RMSE(predict(ctree_fit, validation) %>% exp(), validation$price)))
# earth_fit = earth::earth(log(price) ~ ., data = wp_train)
# metrics2 = rbind(metrics2, c("earth",
#                              RMSE(predict(earth_fit, validation) %>% exp(), validation$price)))
# gbm_fit = gbm::gbm(log(price) ~ ., data = wp_train, n.trees = 150,
#                    shrinkage = 0.25, interaction.depth = 8, n.minobsinnode = 5)
# metrics2 = rbind(metrics2, c("gbm",
#                              RMSE(predict(gbm_fit, validation) %>% exp(), validation$price)))
# metrics2[,2] = metrics2[,2] %>% as.numeric()
load("data/metrics3.RData")
metrics2 %>% knitr::kable(align = "c")
```

| Model | RMSE |
|:---:|:---:|
| lm | 27.20 |
| tree | 26.50 |
| ctree | 26.87 |

| Model | RMSE |
|---|---|
| earth | 27.78 |
| gbm | 24.86 |

Finally we build again ensemble models and we are ready to compare results with and without word predictors. Generally we observe a little improvement of our predictions as revealed in the last table. However the GBM was better without word predictors and as a result the weighted ensemble did not improve at all.

```
# load("data/lm2.RData"); load("data/tree2.RData"); load("data/ctree2.RData")
# load("data/earth2.RData"); load("data/gbm2.RData")
# predictions = data.frame(exp(lm), exp(tree), exp(ctree), exp(earth), exp(gbm))
#
# # ensemble 1
# ens_arit2 = predictions %>% as.matrix() %>%
#    rowMeans()
#
# # ensemble 2
# weights = 1/metrics2$RMSE^10
# ens_weighted2 = apply(predictions %>% as.matrix, 1, function(x){
#    (x %*% weights %>% t()) / sum(weights)
# })
#
# metrics2 = rbind(metrics2,
#                  c("Naive Ens", RMSE(ens_arit2, validation$price)),
#                  c("Weighted Ens", RMSE(ens_weighted2, validation$price)))
# metrics2[,2] = metrics2[,2] %>% as.numeric() %>% round(4)

load("data/metrics4.RData")
data.frame(Model = metrics2$Model, No.words = metrics$RMSE,
           With.words = metrics2$RMSE) %>% knitr::kable(align = "c")
```

| Model | No.words | With.words |
|---|---|---|
| lm | 27.81 | 27.20 |
| tree | 27.60 | 26.50 |
| ctree | 27.53 | 26.87 |
| earth | 28.04 | 27.78 |
| gbm | 24.60 | 24.86 |
| Naive Ens | 25.71 | 25.38 |
| Weighted Ens | 25.05 | 25.05 |

# 6. Conclusion

We can conclude that using our 5 models we are not able to make accurate forecasts of wine prices, especially for expensive wines. Furthermore analyzing written reviews doesn't add much value to the prediction process as it appears that words are not repeated in wine reviews of the same price. We think that it may exists better models to predict prices with these information but that the improvement wouldn't be much better. As a result we believe that there's a strong lack of information. Indeed, luxury wines that we call Grand Cru get their power to raise price for many reasons we don't have access to in this dataset (year, harvesting method, vine height, protected designation of origin, rarity, market demand, etc.).