Healthcare Challenge

Laura Pérez, Laura Humet, Adriana Nialet and Martina Massana

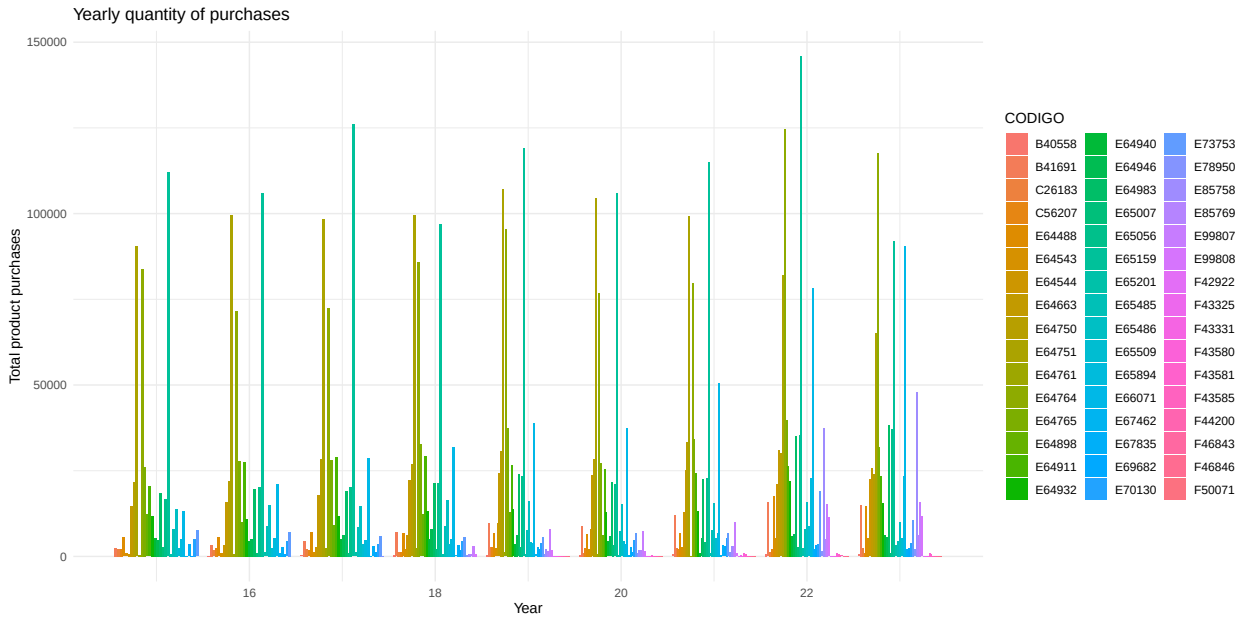
Having chosen the Healthcare Challenge, the resolution we pose consists in setting out a predictive model **based on clustering and forecasting.**
Keeping in mind the size of our *dataset*, we ought to find a clever and representative way of classifying our data. We have established several criteria with which, combined, we hope to reach a realistic prediction for 2023, the reference year in the challenge.

Firstly we are going to read our *dataset* into the Markdown document and saving it as a defined object. As well as adding a new column to the table to be able to easily identify the year each order is made.

```
data <- read_excel("C:/Users/Usuario/Desktop/DATATHON/consumo_material_clean.xlsx")
data$year <- as.numeric(substr(data$FECHAPEDIDO, 7,8)) # last two characters of the date
```

Before filtering, we are going to represent the total number of purchases of each year, using a barchart.

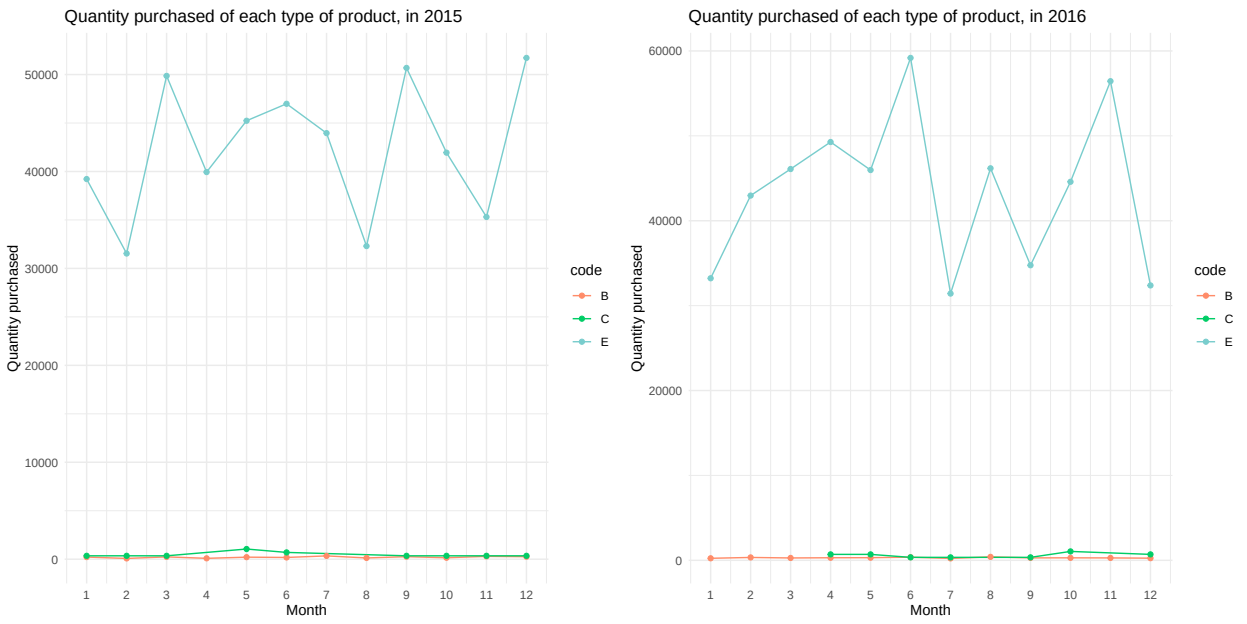
```
data_summarized <- summarise(group_by(data, year, CODIGO),
                             TOTAL_CANTIDADCOMPRA = sum(CANTIDADCOMPRA),
                             TOTAL_PRECIO = sum(PRECIO),
                             PRECIO_UNID = sum(PRECIO) / sum(CANTIDADCOMPRA))
ggplot(data_summarized, aes(x = year, y = TOTAL_CANTIDADCOMPRA, fill = CODIGO)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Yearly quantity of purchases",
       x = "Year", y = "Total product purchases") +
  theme_minimal()
```



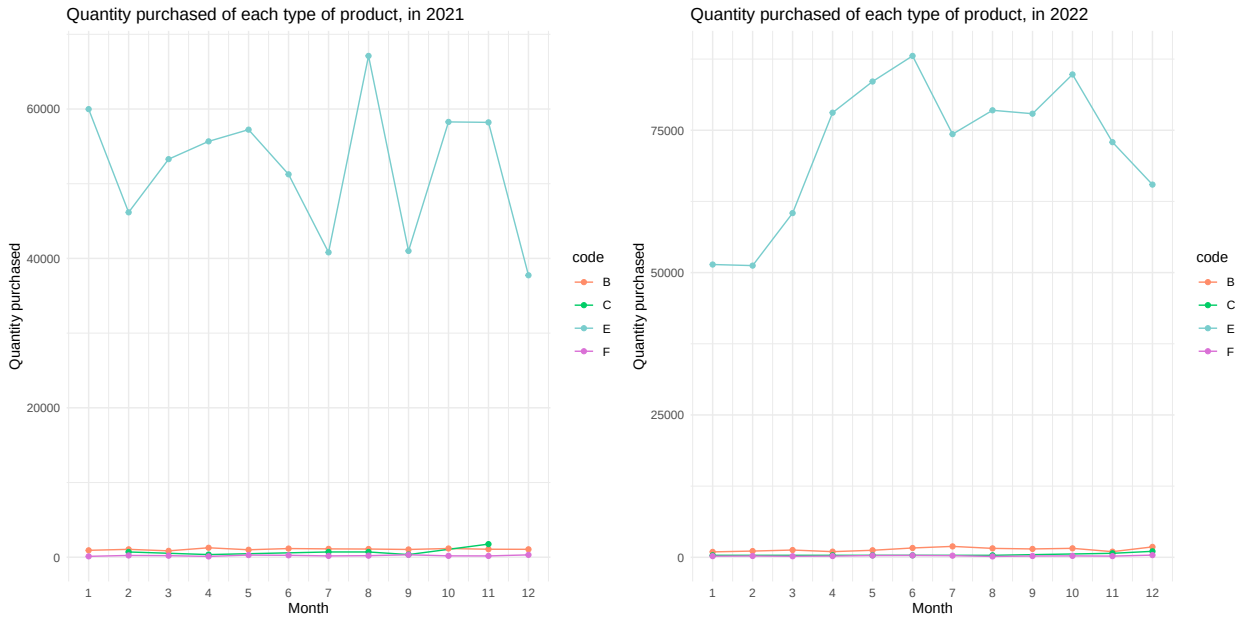
1. Filtering by year and type of product

We are going to plot the monthly total purchases for each year, considering each plot line to be a different type of product. The type of product, as specified in the problem description, is given by the letter of the product code, the first character of each entry in the column **CODIGO**.

```
data$month <- as.numeric(substr(data$FECHAPEDIDO, 4,5))
data$code <- substr(data$CODIGO, 1,1)

selected_rows15 <- data[data$year == 15, ]
data_summarized15 <- summarise(group_by(selected_rows15, month, code),
                               total_quantity = sum(CANTIDADCOMPRA))
selected_rows16 <- data[data$year == 16, ]
data_summarized16 <- summarise(group_by(selected_rows16, month, code),
                               total_quantity = sum(CANTIDADCOMPRA))

plot15 <- ggplot(data_summarized15, aes(x = month, y = total_quantity, color = code)) +
  geom_line() +
  geom_point() +
  labs(title = "Quantity purchased of each type of product, in 2015",
       x = "Month", y = "Quantity purchased") +
  theme_minimal() +
  scale_color_manual(values = c('salmon1', 'springgreen3', 'darkslategray3')) +
  scale_x_continuous(breaks = seq(1, 12, 1), limits = c(1, 12))
plot16 <- ggplot(data_summarized16, aes(x = month, y = total_quantity, color = code)) +
  geom_line() +
  geom_point() +
  labs(title = "Quantity purchased of each type of product, in 2016",
       x = "Month", y = "Quantity purchased") +
  theme_minimal() +
  scale_color_manual(values = c('salmon1', 'springgreen3', 'darkslategray3')) +
  scale_x_continuous(breaks = seq(1, 12, 1), limits = c(1, 12))

grid.arrange(plot15, plot16, ncol=2)
```



We repeat the process for the rest of years in the spectrum.



Continuing with the classification by type of product, we will narrow the search and create, for each letter, two matrices containing: the monthly evolution of the sum of prices per unit and the total amount spent in all purchases of each month; both for all years.

```
data$produnitat <- data$PRECIO/data$CANTIDADCOMPRA
dataB <- data[data$code == 'B',]

months <- c("January", "February", "March", "April", "May", "June", "July",
            "August", "September", "October", "November", "December")

produnitB <- matrix(nrow=0, ncol=12)
for (year in 15:22) {
  data_year <- dataB[dataB$year == year, ]
  v <- c()
  for (month in 1:12) {
    data_month <- data_year[data_year$month == month, ]
    v <- c(v, sum(data_month$produnitat))
  }
  produnitB <- rbind(produnitB, v)
}
rownames(produnitB) <- c(2015:2022)
colnames(produnitB) <- months
print(produnitB)
```

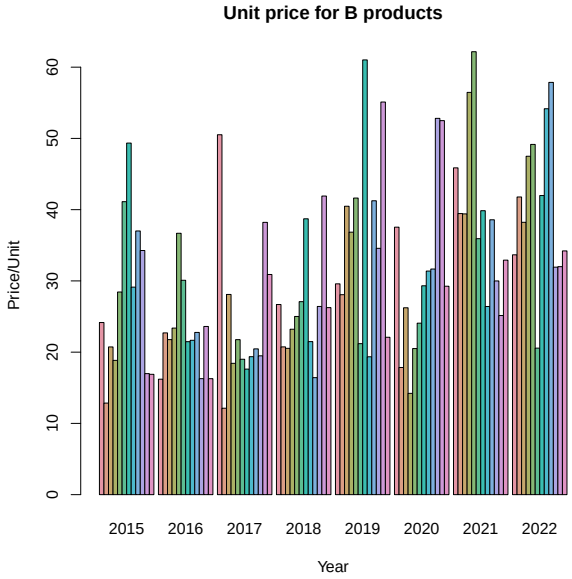
```
##       January February    March    April      May     June     July   August
## 2015 24.15888 12.85047 20.73209 18.84735 28.44237 41.12149 49.34579 29.12772
## 2016 16.20383 22.70249 21.76012 23.37842 36.68645 30.08954 21.47140 21.66500
## 2017 50.51617 12.12974 28.09962 18.43118 21.75454 19.00507 17.62214 19.36415
## 2018 26.68659 20.74006 20.53478 23.21636 25.00985 27.08682 38.71904 21.48007
## 2019 29.58655 28.06537 40.48075 36.84195 41.62956 21.19333 61.01788 19.35143
## 2020 37.54322 17.84843 26.22831 14.21818 20.50498 24.06849 29.31597 31.38267
## 2021 45.86668 39.45376 39.40694 56.46107 62.16871 35.92773 39.85441 26.41010
## 2022 33.66156 41.78017 38.22644 47.49641 49.15879 20.56507 41.98297 54.16587
```

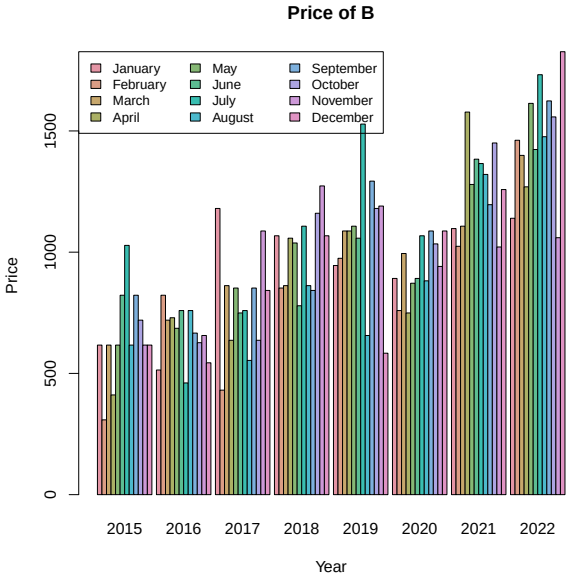
```
##      September  October November December
## 2015  37.00934 34.26791 16.98709 16.88918
## 2016  22.76914 16.27984 23.60669 16.27723
## 2017  20.46131 19.48498 38.21845 30.90098
## 2018  16.42210 26.41604 41.91185 26.25128
## 2019  41.24862 34.57225 55.10809 22.08963
## 2020  31.66915 52.81655 52.49632 29.25508
## 2021  38.57882 29.99620 25.14154 32.91552
## 2022  57.86428 31.92786 31.99963 34.20907
```

```r
priceB <- matrix(nrow=0, ncol=12)
for (year in 15:22) {
  data_year <- dataB[dataB$year == year, ]
  v <- c()
  for (month in 1:12) {
    data_month <- data_year[data_year$month == month, ]
    v <- c(v, sum(data_month$PRECIO))
  }
  priceB <- rbind(priceB, v)
}
rownames(priceB) <- c(2015:2022)
colnames(priceB) <- months
print(priceB)
```

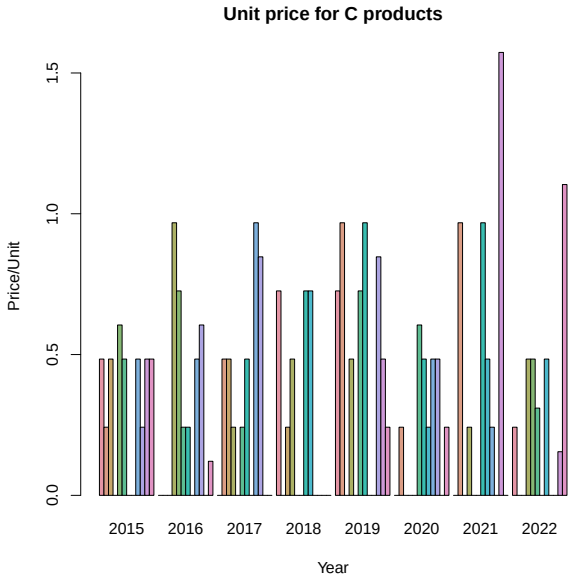
```
##        January  February     March     April       May      June      July
## 2015  616.8224  308.4112  616.8224  411.2149  616.8224  822.4298 1028.0373
## 2016  514.0186  822.4298  719.6261  729.5261  686.1224  759.2261  460.7149
## 2017 1180.3220  431.0080  862.0160  636.6120  852.1160  749.3140  759.2140
## 2018 1067.6200  852.1160  862.0160 1057.7200 1037.9200  779.0140 1107.2200
## 2019  945.0180  974.7180 1087.4200 1087.4200 1107.2200 1057.7200 1528.3280
## 2020  891.7160  759.2140  994.5180  749.3140  871.9160  891.7160 1067.6200
## 2021 1097.3200 1024.2180 1107.2200 1577.8280 1279.3220 1383.5320 1365.5400
## 2022 1139.8200 1461.5700 1399.2000 1269.5100 1613.7000 1423.4440 1731.8730
##         August September   October  November  December
## 2015  616.8224  822.4298  719.6261  616.8224  616.8224
## 2016  759.2157  666.3120  626.7120  656.4120  543.7100
## 2017  553.6100  852.1160  636.6120 1087.4200  842.2160
## 2018  862.0160  842.2160 1160.5220 1273.2240 1067.6200
## 2019  656.4120 1293.0240 1180.3220 1190.2220  583.3100
## 2020  881.8160 1087.4200 1034.1180  941.2160 1087.4200
## 2021 1320.6600 1195.9200 1450.3500 1021.3500 1258.2900
## 2022 1476.2000 1623.8200 1557.8750 1059.7180 1826.7370
```

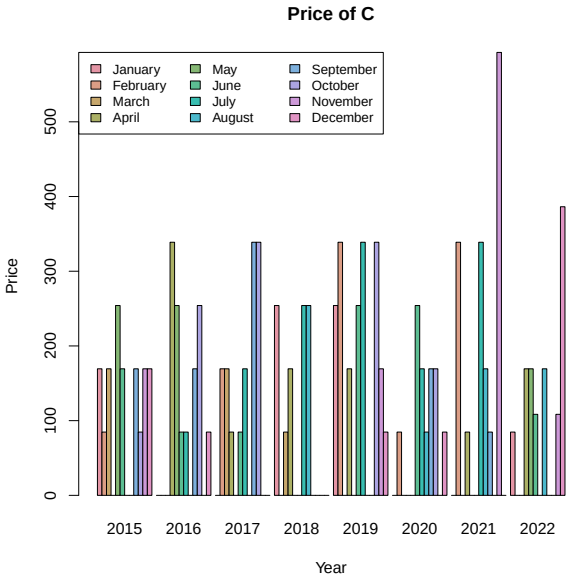
```r
# Create a barchart for each type of product
par(mfrow = c(1, 2))
barplot(t(produnitB), beside = TRUE, col = colorspace::rainbow_hcl(ncol(produnitB)),
        names.arg = 2015:2022, main = "Unit price for B products",
        xlab = "Year", ylab = "Price/Unit")

barplot(t(priceB), beside = TRUE, col = colorspace::rainbow_hcl(ncol(priceB)),
        names.arg = 2015:2022, main = "Price of B",
        xlab = "Year", ylab = "Price",
        legend.text = months, args.legend = list(x = "topleft", cex = 0.85, ncol = 3))
```

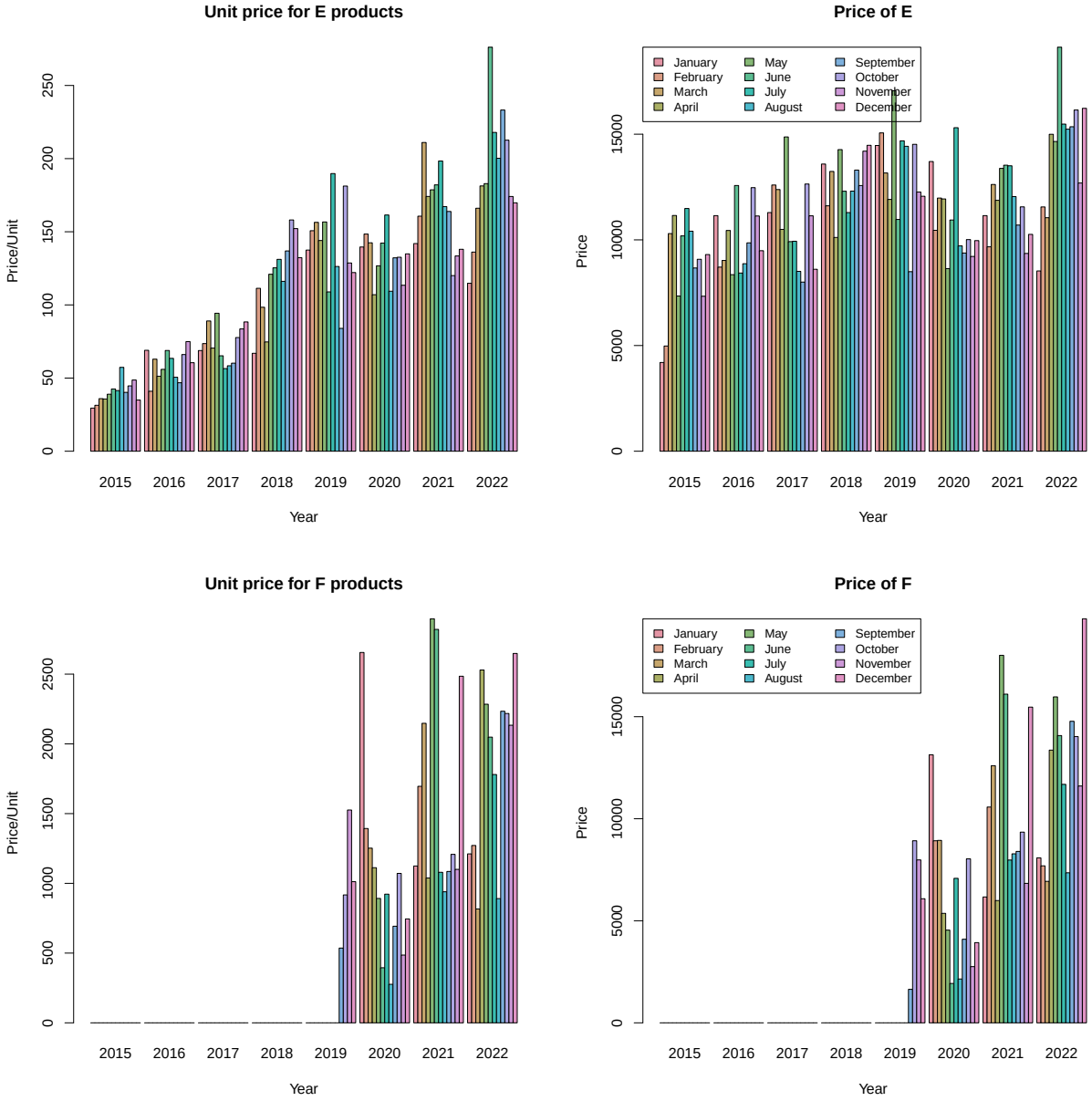




We repeat the process for the rest of product types.





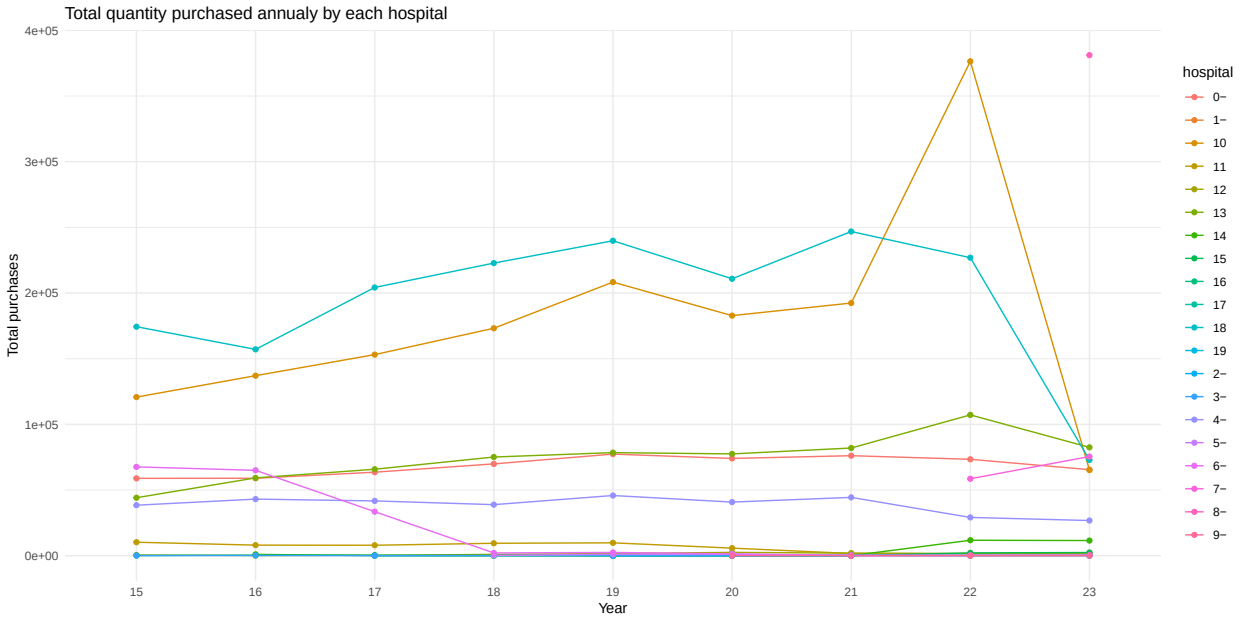


The goal with these charts is to see whether a price fluctuation affects the quantity of purchases or not. Comparing both charts to the previous section's plot, we are able to see the tendency is the same, so the effect is neither positive or negative. We can conclude **purchases ar based on necessity** and not conditioned by price.

Furthermore, it is necessary to note the behavior of different hospitals is similar, so we will consider them as a whole, not finding necessary having to analyse each one's progression separately.

```
data$hospital <- substr(data$ORIGEN, 3, 4)
data_hospital <- summarise(group_by(data, year, hospital),
                           total_quantity = sum(CANTIDADCOMPRA))

ggplot(data_hospital, aes(x = factor(year), y = total_quantity, color = hospital)) +
  geom_line(aes(group = hospital)) +
```

```
  geom_point() +
  labs(title = "Total quantity purchased annualy by each hospital",
       x = "Year", y = "Total purchases") +
  theme_minimal() +
  scale_x_discrete(breaks = unique(data_hospital$year))
```



Something we need to keep in mind when analyzing this plot is the fact the data we have from 2023, which we added only for credibility reasons, to make it as real as possible; **only goes from January to mid October**. It makes sense the quantities are quite lower than the previous years, for which we have data from the twelve months; especially when we have previously seen November and December are months with large amounts of registered purchases.

Now, we are going to analyse how the orders are changing annually. To reach this goal, our first option is to group by similarity (clusters) and then, adjust an adequate prediction model.

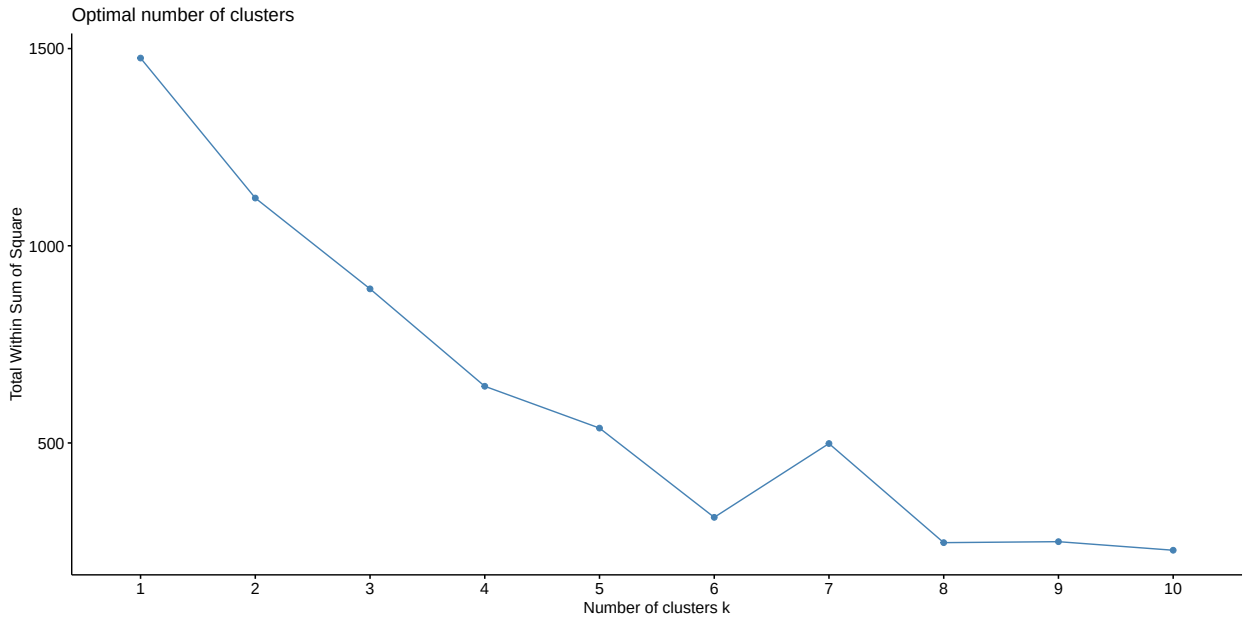
2. Cluster construction

In order to be able to define coherent clustering, we need to consider we are taking an annual approach so we need to select a variable subset of: **total quantity purchased per year, total price, and price per unit.**

2.1. Elbow Method

Our first option is to apply the **Elbow Method**, with which we search the value where an *elbow* (subtle peak, similar to the concept of a local maximum) is created. The point where we evaluate this difference marks the optimal amount of clusters with which we should assess our model.
In our case:

```
numeric_data <- data_summarized[, sapply(data_summarized, is.numeric)]
data_scaled <- scale(numeric_data)

fviz_nbclust(data_scaled, kmeans, method = "wss")
```

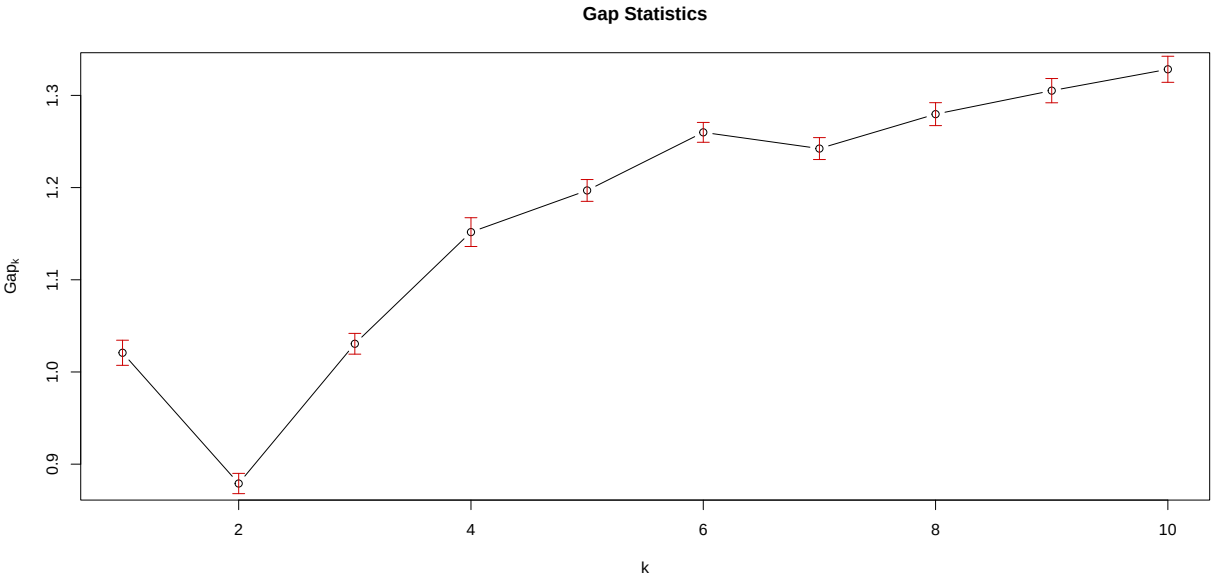


With said graphic representation we are able to conclude the Elbow Method doesn't work for this model specifically; it doesn't show for which k amount of clusters we would reach an optimal adjust. If we wanted to be safe and decided to apply a large k we would encounter serious problems of overfitting which would cause the adjust to be incorrect either way.

2.2. Gap Method

In using the **Gap Method** we establish the optimal k amount of clusters as the point in which the plot's tendency changes and begins to "stabilize". We want to maximize the distance between each plot gap, assessing the maximum one as the selected optimum.
Once again, in our case:

```
gap_stat <- clusGap(data_scaled, FUN = kmeans, K.max = 10, B = 50)
plot(gap_stat, main = "Gap Statistics")
```



As we see, 4 seems a promising candidate for optimal k as it shows a clear change in tendency. But we also have to consider said stabilization reached after 4 only remains until 6, so it doesn't seem to be correct enough. This type of clustering, as well as the other one, doesn't seem to be our best option.

Given this two past failed verdicts we need to make a decision based on approximate conclusions extracted from them. That's why, based on plot tendencies we will take $\boxed{k = 5}$ as our **optimal amount of clusters**, which curiously enough also coincides with the quantity of different types of products (as we mentioned earlier, letters in the alphanumeric **CODIGO** column).

2.3 Clustering with *k-means*

Having chosen 5 as our official optimal clustering amount, we can now define this decision as an object, using the R function ***kmeans***. But we have to keep in mind our decision was approximate and *kmeans* is unsupervised learning, so we cannot guarantee a 100% success rate for our prediction using this classification method.

```
km <- kmeans(data_scaled, centers = 5)
```

Because the optimization goal is mainly **economic**, the 5 clusters can be defined with two parameters each, as follows:
The specification **TOTAL** refers to the total aggregated amount of units purchased of products in said cluster (in units). **UNIT PRICE** is the result of the quotient between price per "package" and units per package (in euros). The two sections of every clustered is represented with an **approximate range**.

- $\boxed{\text{A}}$ **TOTAL:** $20,000 - 50,000$. **UNIT PRICE:** < 1.
- $\boxed{\text{B}}$ **TOTAL:** $80,000 - 130,000$. **UNIT PRICE:** $2 - 7$.
- $\boxed{\text{C}}$ **TOTAL:** $< 40,000$. **UNIT PRICE:** $150 - 500$.
- $\boxed{\text{D}}$ **TOTAL:** $200,000 - 300,000$. **UNIT PRICE:** $30 - 50$.
- $\boxed{\text{E}}$ **TOTAL:** $60,000 - 300,000$. **UNIT PRICE:** $1 - 3$.

We are now going to add a variable with the name 'cluster' to our table, from which we are going to be able to analyze each group by different additional criteria.

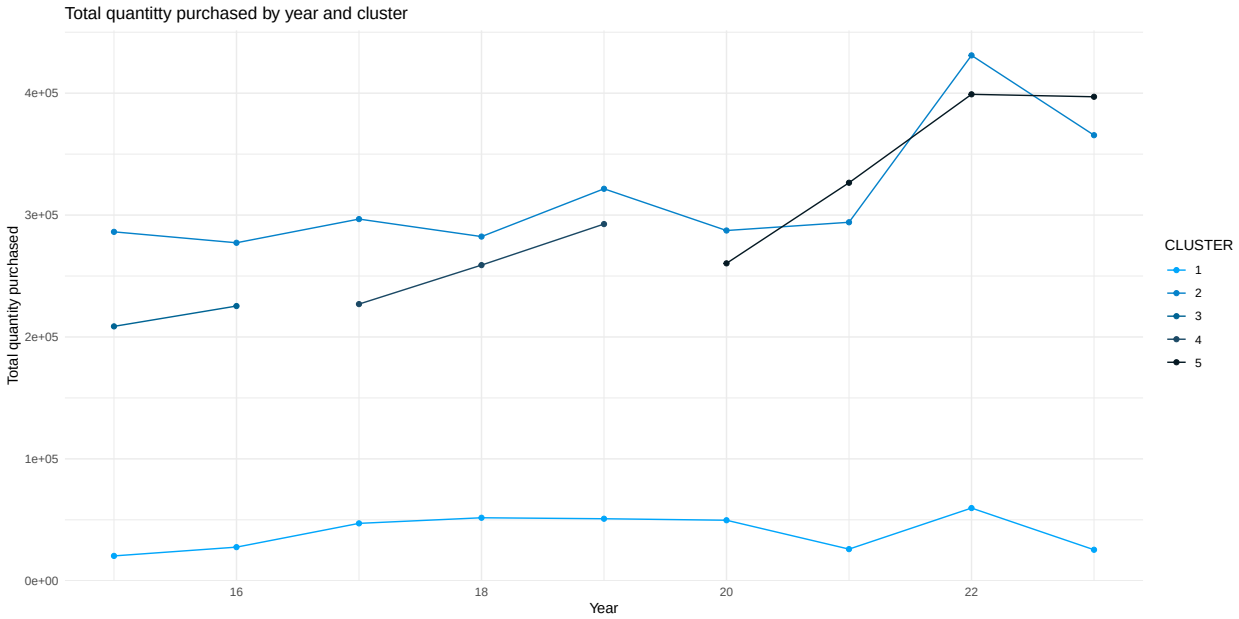
```
data_summarized$cluster <- km$cluster
```

Setting out three new classifications, our goal is to see how clusters behave by themselves or if there is any direct relation between certain ones.

- **By year and cluster, displaying the total quantity purchased**

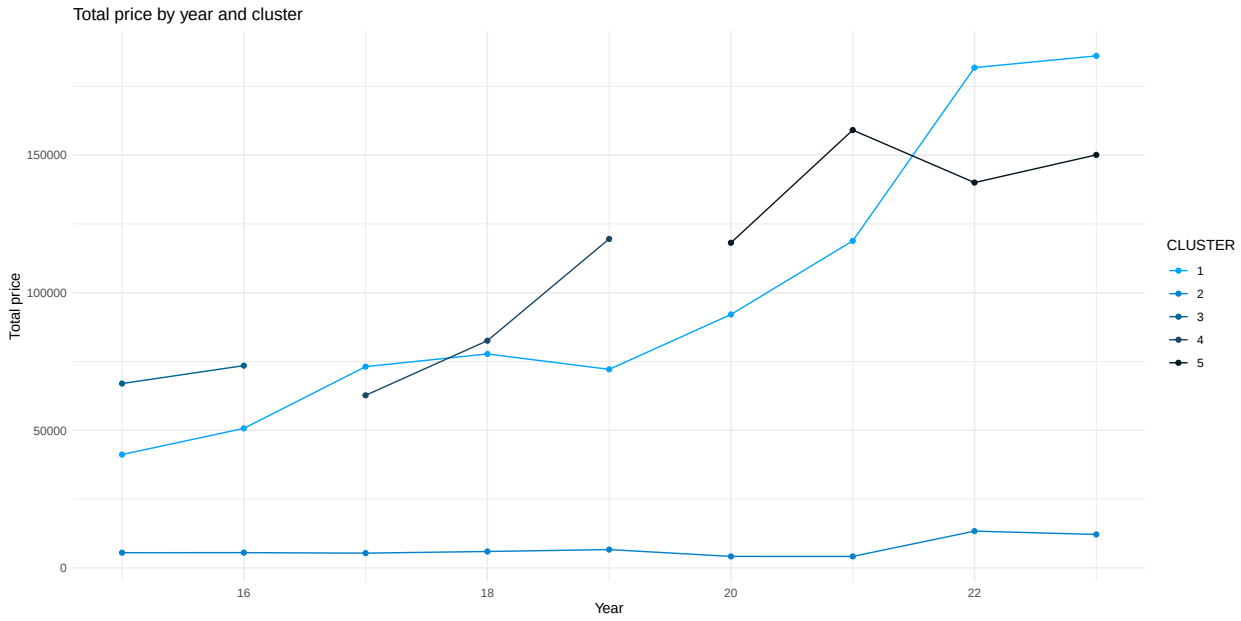
```
colors <- c("#00a6fb", "#0582ca", "#006494", "#1b4965", "#051923")

data_clusters <- summarise(group_by(data_summarized, cluster, year),
                           TOTAL = sum(TOTAL_CANTIDADCOMPRA),
                           TOTAL_PRECIO = sum(TOTAL_PRECIO),
                           PRECIO_UNID = sum(TOTAL_PRECIO) / sum(TOTAL_CANTIDADCOMPRA))

ggplot(data_clusters, aes(x = year, y = TOTAL, color = as.factor(cluster))) +
```

```
  geom_point() +
  geom_line(aes(group = cluster), linetype = "solid") +
  labs(title = "Total quantitty purchased by year and cluster",
       x = "Year", y = "Total quantity purchased", color = "CLUSTER") +
  theme_minimal() +
  scale_color_manual(values = colors)
```



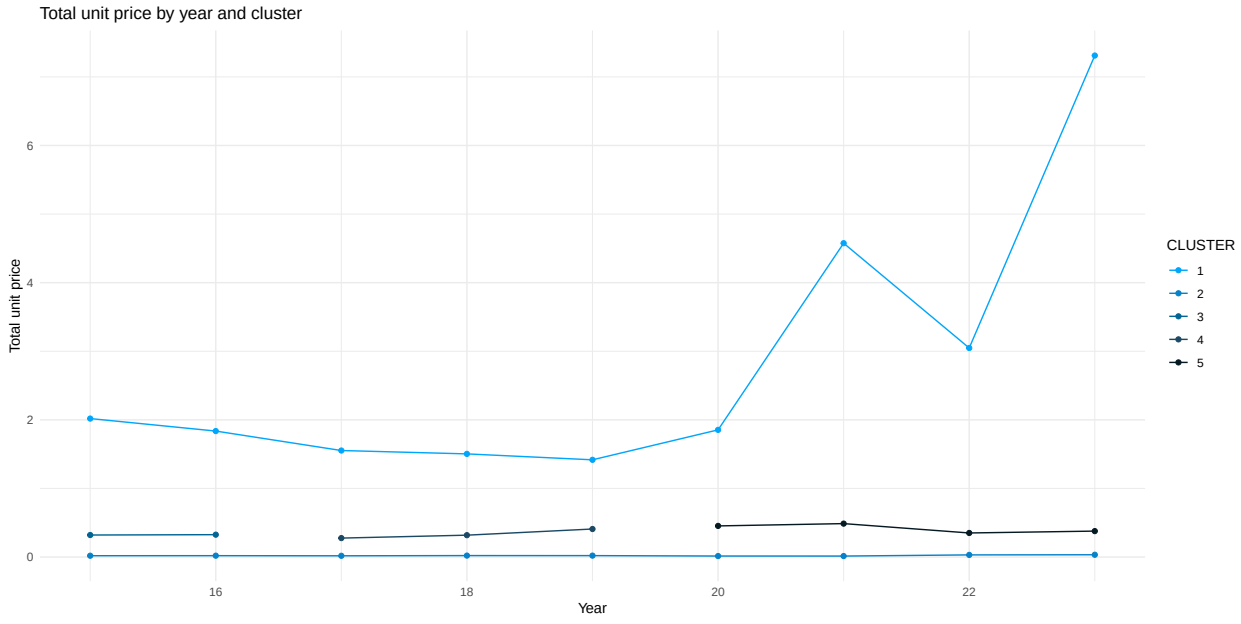
- **By year and cluster, displaying the total money (regarding price) spent**

```
ggplot(data_clusters, aes(x = year, y = TOTAL_PRECIO, color = as.factor(cluster))) +
  geom_point() +
  geom_line(aes(group = cluster), linetype = "solid")  +
  labs(title = "Total price by year and cluster",
       x = "Year", y = "Total price", color = "CLUSTER") +
  theme_minimal()+
  scale_color_manual(values = colors)
```



- **By year and cluster, displaying their unit price**

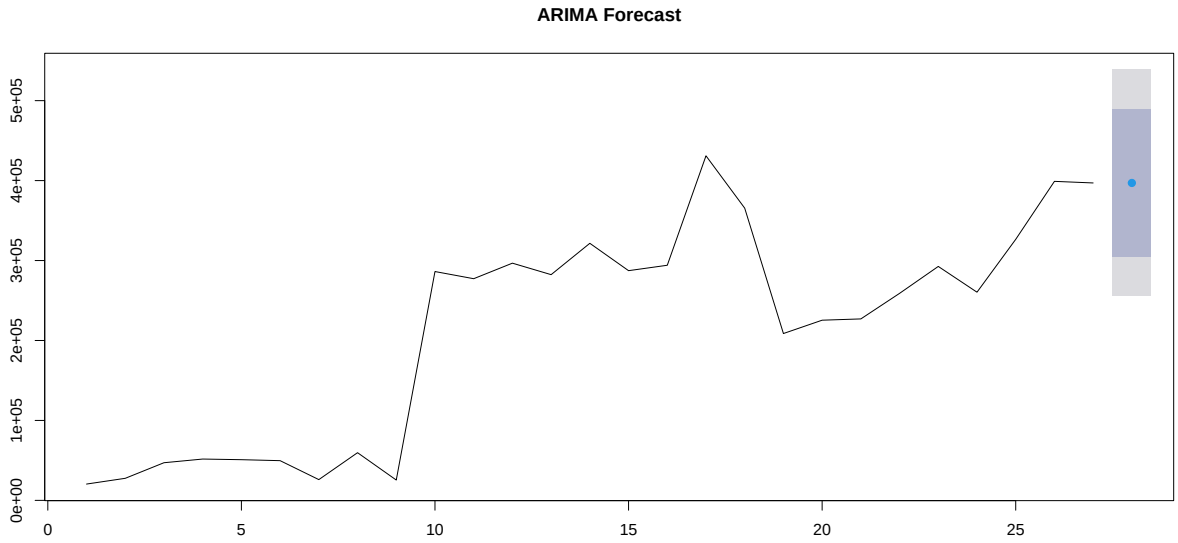
```
ggplot(data_clusters, aes(x = year, y = PRECIO_UNID, color = as.factor(cluster))) +
  geom_point() +
  geom_line(aes(group = cluster), linetype = "solid")  +
  labs(title = "Total unit price by year and cluster",
       x = "Year", y = "Total unit price", color = "CLUSTER") +
  theme_minimal()+
  scale_color_manual(values = colors)
```



4. Our solution: Applying ARIMA Forecasting

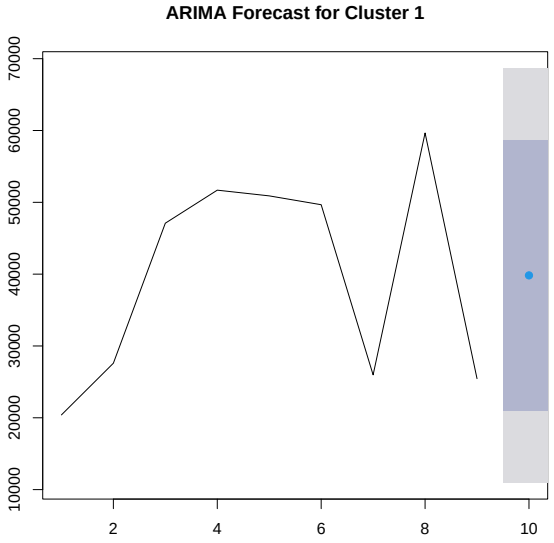
Finally, considering all the information we have seen until this point, we can conclude the desired model is **not a linear model**, so we will use **ARIMA Forecasting** to predict one year ahead.
For that, we are going to turn our data into an adequate object the ARIMA model can correctly forecast. Following this step, we specify we want to forecast 1 year (12 months) ahead, and with that we are able to plot our result. It is important to keep in mind this result corresponds to **modeling without considering clusters** yet.

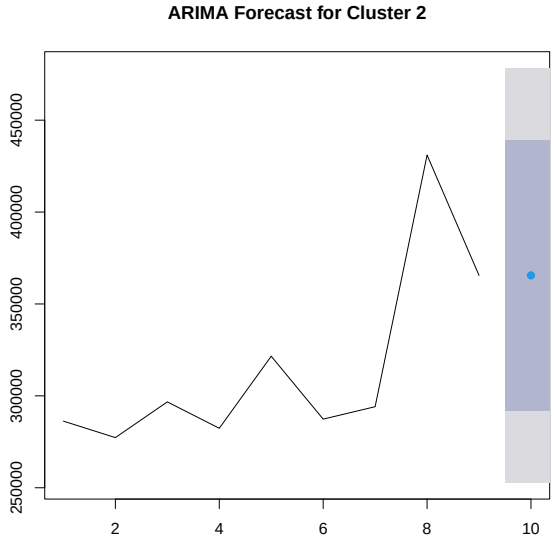
```
ts_data <- ts(data_clusters$TOTAL, frequency = 1)

arima_model <- auto.arima(ts_data)
forecast_result <- forecast(arima_model, h = 1)  # Forecasting 12 periods ahead

plot(forecast_result, main = "ARIMA Forecast")
```

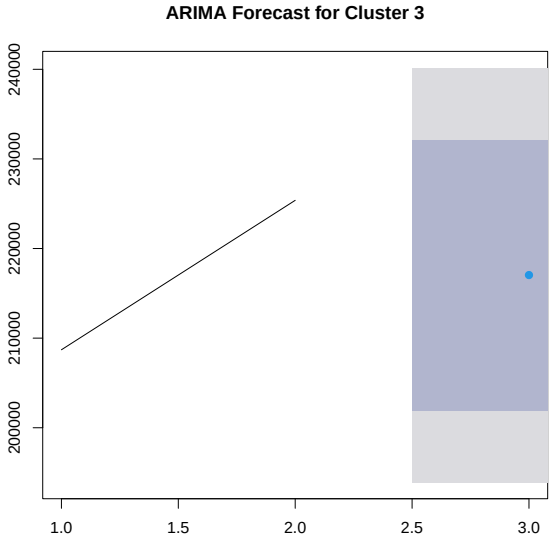


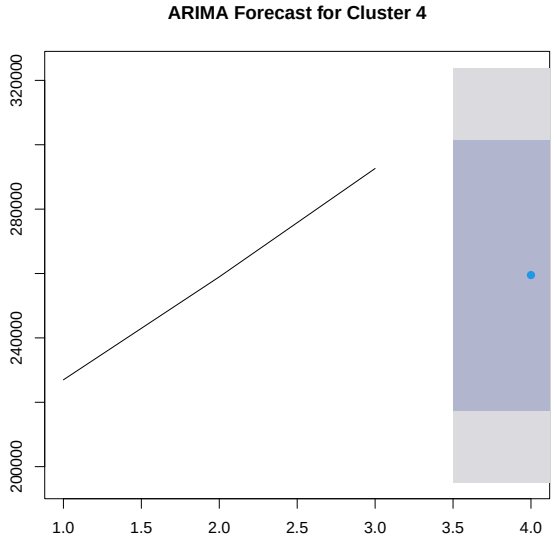
We can now apply ARIMA Forecasting also to the cluster distribution.

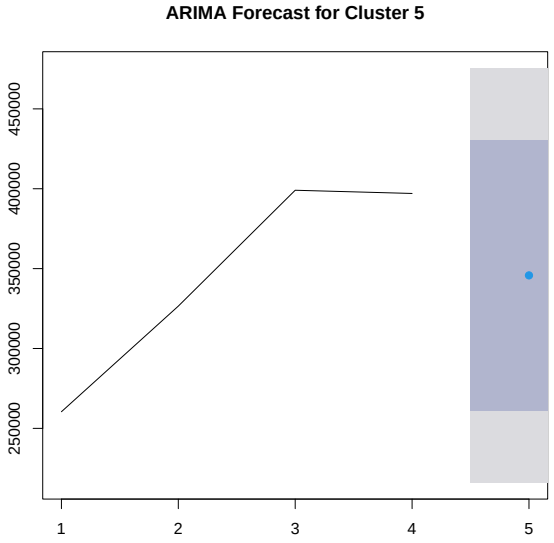
```
par(mfrow = c(1, 2))
for (cluster_value in unique(data_clusters$cluster)) {
  data_cluster <- data_clusters[data_clusters$cluster == cluster_value, ]
  ts_data <- ts(data_cluster$TOTAL, frequency = 1)
  arima_model <- auto.arima(ts_data)
  forecast_result <- forecast(arima_model, h = 1)
  plot(forecast_result, main = paste("ARIMA Forecast for Cluster", cluster_value))
}
```


ARIMA Forecast for Cluster 1
70000
60000
50000
40000
30000
20000
10000
2
4
6
8
10


ARIMA Forecast for Cluster 2
450000
400000
350000
300000
250000
2
4
6
8
10


ARIMA Forecast for Cluster 3
240000
230000
220000
210000
200000
1.0
1.5
2.0
2.5
3.0


ARIMA Forecast for Cluster 4
320000
280000
240000
200000
1.0
1.5
2.0
2.5
3.0
3.5
4.0



In conclusion, **we have achieved the goal of predicting the demand for 2023.**
However, it is also important to note we have found out this model is also especially effective for **monthly predictions** (and frequencies). Unfortunately, because of time constraints we haven't been able to decentralize data into months, but we are aware that said option could be a better prediction. With it we could have taken into account the **difference in purchase density and distribution throughout the year's months** and narrow the predicted interval for each type of product. This approach would help anticipate orders to be placed monthly, reducing smaller and last-minute orders by half due to a lack of foresight.