Challenge Answers

Gustavo Mello

2022-03-28

Data loading

```
# Imports dataset made in EDA script
cwd <- getwd()
daily.revenue.listings <- read_csv(
  paste0(cwd, "/../data/output/daily_revenue_listings.csv")
)
```

```
## Rows: 289021 Columns: 26
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (6): listing, Localização, Categoria, Hotel, Status, Tipo
## dbl  (17): last_offered_price, occupancy, revenue, blocked, reservation_adva...
## date  (3): date, creation_date, Data.Inicial.do.contrato
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Question 1

What is the expected price and revenue for a listing tagged as JUR MASTER 2Q in march?

The expected value for a random variable is better estimated by its average over observed values:

```
(daily.revenue.listings %>%
  select(date, Localização, Categoria,
         Quartos, last_offered_price, commission) %>%
  mutate(month=month(date, label=TRUE)) %>%
  filter(month=="mar",
         Localização=="JUR",
         Categoria=="MASTER",
         Quartos==2))
```

```
## # A tibble: 0 x 7
## # ... with 7 variables: date <date>, Localização <chr>, Categoria <chr>,
## #   Quartos <dbl>, last_offered_price <dbl>, commission <dbl>, month <ord>
```

However the query above returns 0 results. Therefore, it is necessary to model the data to predict price and revenue (commission earned by Seazone) from the 4 features: month, location, category and number of bedrooms. Two predictive models, also considered in the EDA script, are tested and applied below:

```
commission.earned.data <- daily.revenue.listings %>%
  filter(occupancy==1, blocked==0) %>%
  select(date, Localização, Categoria, Quartos,
```

```
         last_offered_price, revenue, Comissão) %>%
  mutate(month=month(date, label=TRUE)) %>%
  # Limits earnings data to only consolidated observations
  mutate(across(c(Localização, Categoria), ~as.factor(.))) %>%
  filter(date <= "2022-03-15")


# Temporal subsetting
train <- commission.earned.data %>%
  filter(date <= "2022-01-31")

test <- commission.earned.data %>%
  filter(date > "2022-01-31")


# Linear model fitting
linear.model.price <- lm(last_offered_price ~ month + Localização
                                              + Categoria + Quartos,
                         data=train)
linear.model.revenue <- lm(revenue ~ month + Localização
                                     + Categoria + Quartos,
                           data=train)

# XGBoost model fitting
make.matrix.from <- function(data){
  as.matrix(data %>%
    select(-c(last_offered_price, revenue, Comissão, date)) %>%
    mutate(month=factor(month, levels=levels(month), ordered=FALSE)) %>%
    dummy_cols(remove_selected_columns = TRUE)
  )
}

train.matrix <- make.matrix.from(train)
test.matrix <- make.matrix.from(test%>%
                                  filter(Localização != "ILC",
                                         Localização != "JBV"))

xgboost.model.price <- xgboost(data=train.matrix,
                               label=train$last_offered_price,
                               nrounds=20,
                               max.depth=6,
                               verbose=0)
xgboost.model.revenue <- xgboost(data=train.matrix,
                                 label=train$revenue,
                                 nrounds=20,
                                 max.depth=6,
                                 verbose=0)


# Assessing model accuracy
RMSE <- function(predictions, response, filter_new_factors=FALSE){
  test.set <-
    if(filter_new_factors){
```

```r
    test %>%
      filter(Localização != "ILC",
             Localização != "JBV")
    } else{
    test
    }
  n <- dim(test.set)[1]
  sqrt(sum((predictions - test.set[,response])^2)/n)
}

## Price
mean.predictions <- rep(mean(train$last_offered_price),  #Baseline
                        length(test$last_offered_price))
linear.predictions <- predict(linear.model.price,
                              test %>%
                                filter(Localização != "ILC",
                                       Localização != "JBV"))
xgboost.predictions <- predict(xgboost.model.price, test.matrix)

RMSE(mean.predictions, "last_offered_price")    # 352.3382
RMSE(linear.predictions, "last_offered_price",  # 260.2715
     filter_new_factors = TRUE)
RMSE(xgboost.predictions, "last_offered_price", # 239.7306
     filter_new_factors = TRUE)

## Revenue
mean.predictions <- rep(mean(train$revenue),  #Baseline
                        length(test$revenue))
linear.predictions <- predict(linear.model.revenue,
                              test %>%
                                filter(Localização != "ILC",
                                       Localização != "JBV"))
xgboost.predictions <- predict(xgboost.model.revenue, test.matrix)

RMSE(mean.predictions, "revenue")    # 351.9307
RMSE(linear.predictions, "revenue",  # 260.7388
     filter_new_factors = TRUE)
RMSE(xgboost.predictions, "revenue", # 239.9097
     filter_new_factors = TRUE)
```

Above, new locations are filtered in the test set since these models aim to estimate price and revenue under known conditions.

Finally, predicting for the required case:

```r
case <- train %>%
  add_row(date=ymd("2022-03-01"), Localização="JUR", Categoria="MASTER",
          Quartos=2, last_offered_price=NA, revenue=NA, Comissão=0.2,
          month="mar") %>%
  mutate(month=ordered(month, levels=levels(train$month)))

case.matrix <- make.matrix.from(case)
case.index <- dim(case.matrix)[1]
case.row <- case %>% dplyr::slice(n())
```

```r
predict(linear.model.price, case.row)                       # 586.2073
predict(xgboost.model.price, case.matrix)[case.index]       # 312.2011
predict(linear.model.revenue, case.row)                     # 586.2073
predict(xgboost.model.revenue, case.matrix)[case.index] # 312.2011
```

Comparing predictions to actual data, we might see that a good estimate should be in between R$ 383,00 and R$ 531,64.

```r
(comparison <- commission.earned.data %>%
  filter(Localização=="JUR", month=="mar", Categoria=="MASTER") %>%
  group_by(month, Localização, Categoria, Quartos) %>%
  summarise(avg_revenue=mean(revenue)))
```

```
## # A tibble: 3 x 5
## # Groups:   month, Localização, Categoria [1]
##   month Localização Categoria Quartos avg_revenue
##   <ord> <fct>       <fct>       <dbl>       <dbl>
## 1 mar   JUR         MASTER          1        383.
## 2 mar   JUR         MASTER          3        532.
## 3 mar   JUR         MASTER          4       2244.
```

To get inside this interval, an average between XGBoost and Linear model is taken. Only one value is predicted since the models set the same for both revenue and price.

```r
average.prediction <- mean(c(312.2011, 586.2073))
average.prediction
```

```
## [1] 449.2042
```

This process suggests an ensemble model:

```r
RMSE(0.1*linear.predictions + 0.9*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 238.2749
```

```r
RMSE(0.2*linear.predictions + 0.8*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 237.484
```

```r
RMSE(0.3*linear.predictions + 0.7*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 237.5454
```

```r
RMSE(0.4*linear.predictions + 0.6*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 238.4585
```

```r
RMSE(0.5*linear.predictions + 0.5*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 240.2135
```

```
RMSE(0.6*linear.predictions + 0.4*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 242.7922
```

```
RMSE(0.7*linear.predictions + 0.3*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 246.1687
```

```
RMSE(0.8*linear.predictions + 0.2*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 250.3108
```

```
RMSE(0.9*linear.predictions + 0.1*xgboost.predictions,
     "revenue",
     filter_new_factors = TRUE)
```

```
## [1] 255.1811
```

Answer

Using the ensemble with the least RMSE as final predictor:

```
final.prediction.price <- 0.2*586.2073 + 0.8*312.2011
# R$ 367,00

final.prediction.commission <- 0.2*final.prediction.price
# R$ 73,40
```

Even though it is not contained in the proposed interval, it is informed by all the data points rather than those few in the comparison table and should be a more robust prediction therefore.

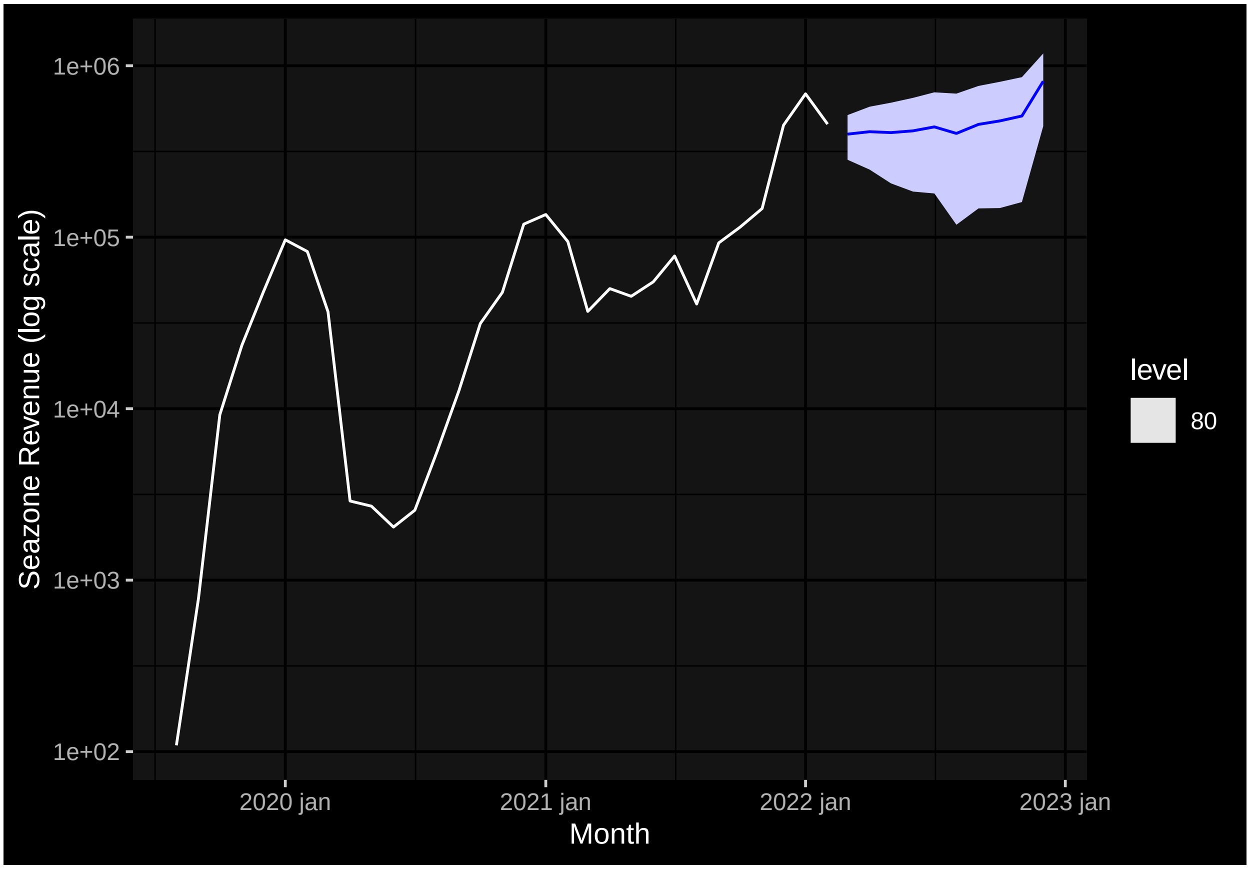
Question 2

What is Seazone expected revenue for 2022? Why?

As developed in the Revenue Modeling script, a SARIMA(1, 0, 0)(0, 1, 0)[12] model, the best performer in cross-validation tests, forecasts the below results in terms of revenue for Seazone (commissions earned total):

```
monthly.commissions <- daily.revenue.listings %>%
  select(date, commission) %>%
  mutate(date=yearmonth(date)) %>%
  group_by(date) %>%
  summarise(commission=sum(commission)) %>%
  as_tsibble(index=date) %>%
  filter_index(~"2022-02")

monthly.commissions.forecast <- monthly.commissions %>%
  model(ARIMA(commission)) %>%
  forecast(h=10)

revenue.forecast.plot <- monthly.commissions.forecast %>%
```

```r
  autoplot(monthly.commissions, level=80) +
  scale_y_log10() +
  labs("Revenue forecast",
       x="Month",
       y="Seazone Revenue (log scale)")

revenue.forecast.plot
```

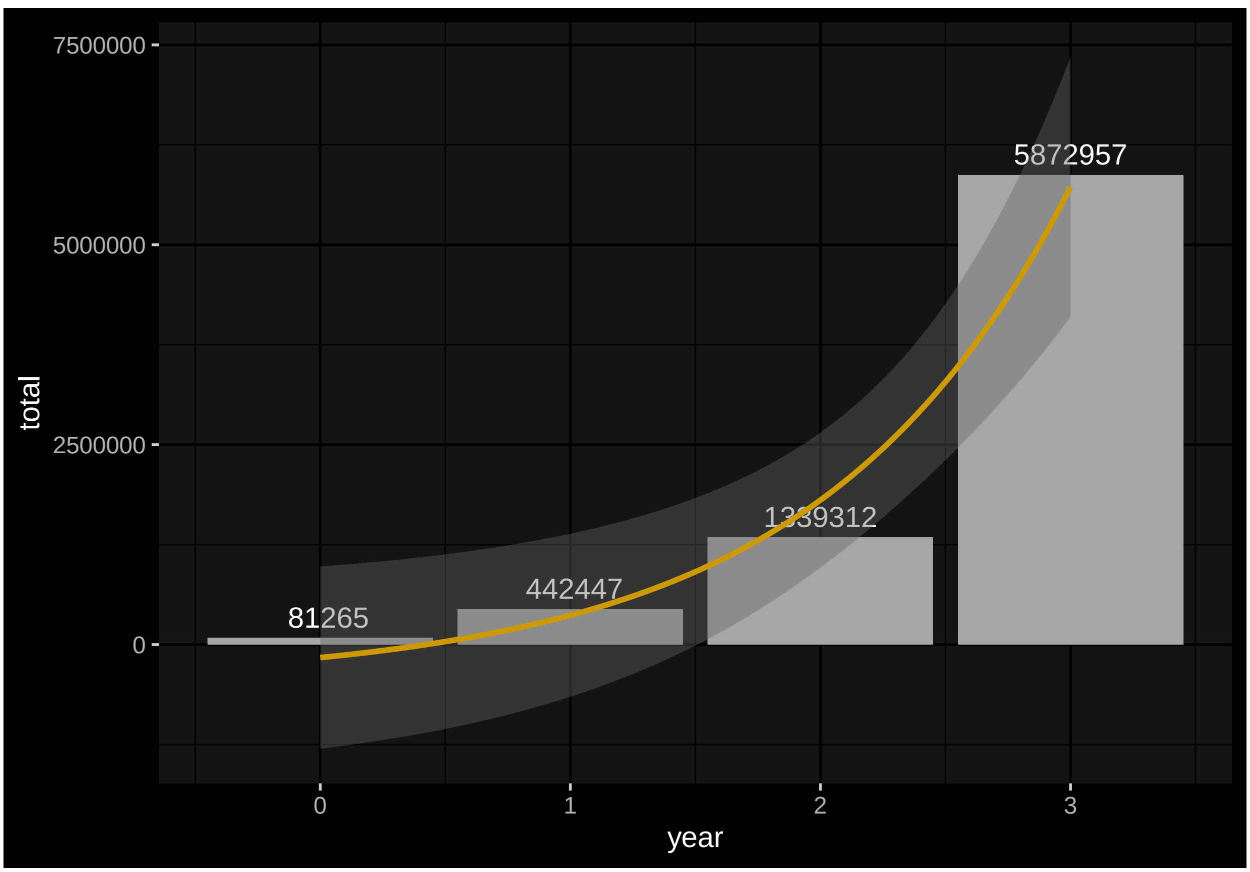



```r
monthly.commissions.forecasted <- monthly.commissions.forecast %>%
  hilo() %>%
  unpack_hilo(c("80%", "95%")) %>%
  select(-c(.model, commission)) %>%
  rename(forecast=.mean)

monthly.commissions.actual.and.forecasted <- monthly.commissions %>%
  full_join(monthly.commissions.forecasted, by=c("date"="date"))

revenue.by.year <- monthly.commissions.actual.and.forecasted %>%
  as_tibble() %>%
  select(date, commission, forecast, "80%_lower", "80%_upper") %>%
  rename(upper80="80%_upper", lower80="80%_lower") %>%
  mutate(year=year(date)) %>%
  group_by(year) %>%
  summarise(realized=sum(commission, na.rm=TRUE),
            forecasted_mean=sum(forecast, na.rm=TRUE),
            upper80=sum(upper80, na.rm=TRUE),
```

```
            lower80=sum(lower80, na.rm=TRUE)) %>%
  mutate(total_lower=realized+lower80,
         total=realized+forecasted_mean,
         total_upper=realized+upper80)

revenue.by.year
```

```
## # A tibble: 4 x 8
##    year realized forecasted_mean  upper80 lower80 total_lower  total total_upper
##   <dbl>    <dbl>           <dbl>    <dbl>   <dbl>       <dbl>  <dbl>       <dbl>
## 1  2019   81265.               0        0   0          81265. 8.13e4      81265.
## 2  2020  442447.               0        0   0         442447. 4.42e5     442447.
## 3  2021 1339312.               0        0   0        1339312. 1.34e6    1339312.
## 4  2022 1142237.        4730720. 7342358.   2.12e6    3261319. 5.87e6    8484595.
```

```
revenue.by.year %>%
  mutate(year=year-2019) %>%
  ggplot(aes(x=year, y=total, label=format(total, digits=2))) +
  geom_col() +
  geom_text(vjust=-0.5) +
  geom_smooth(method = "lm", formula = y ~ exp(x))
```



Answer

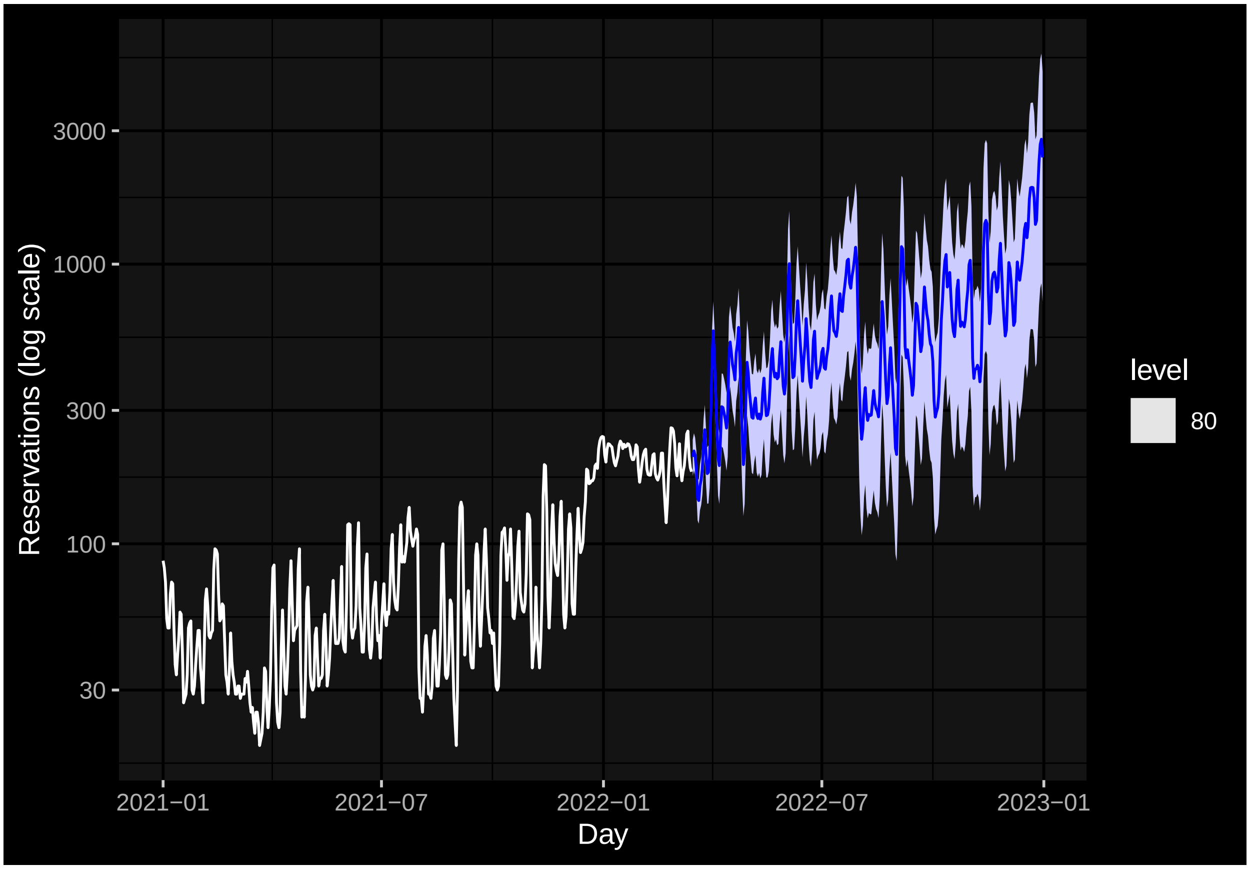
The predicted total revenue is in an 80% confidence interval ranging from R$ 3.261.318,82 to R$ 8.484.595,04 with expected value R$ 5.872.956,93

Question 3

How many reservations should we expect to sell per day? Why?

As thoroughly discussed over the Demand Modeling script, the expected number of reservations per day in 2022 can be estimated by a dynamic harmonic regression with 75 pairs of sine and cosine fourier harmonics, a dummy variable to distinguish between workdays and non-workdays and an ARMA error component.

```
dynamic_harmonic_regression <- function(K, log_transform=FALSE){
  K_weekly <- if (K > 3) {3} else {K}

  if(log_transform){
    ARIMA(log(count) ~ PDQ(0, 0, 0)
                     + fourier(K=K_weekly, period=7)
                     + fourier(K=K, period=365.25)
                     + is.workday)
  } else{
    ARIMA(count ~ PDQ(0, 0, 0)
                + fourier(K=K_weekly, period=7)
                + fourier(K=K, period=365.25)
                + is.workday)
  }
}

reservations.daily <- daily.revenue.listings %>%
  filter(date <= "2022-03-15", occupancy==1, blocked==0) %>%
  select(date) %>%
  group_by(date) %>%
  summarise(count=n()) %>%
  tsibble(index=date) %>%
  fill_gaps(count=0, .full=TRUE) %>%
  mutate(is.workday = !(timeDate::isWeekend(date)
                      | (date %in% holidaysANBIMA)))

reservations.daily.COVID.excluded <- reservations.daily %>%
  filter_index("2021-01-01" ~ "2022-03-15")

reservations.daily.COVID.excluded.new.data <- reservations.daily.COVID.excluded %>%
  new_data(291L) %>% # predictions until 31-12-2022
  mutate(is.workday = !(timeDate::isWeekend(date)
                      | (date %in% holidaysANBIMA)))

# Model training may take a little while
final.model <- reservations.daily.COVID.excluded %>%
  model(dynamic_harmonic_regression(K=75, log_transform=TRUE))

demand.forecast.plot <- final.model %>%
  forecast(reservations.daily.COVID.excluded.new.data) %>%
  autoplot(reservations.daily.COVID.excluded, level=80) +
  labs("Daily Number of Reservations Forecast",
       x="Day",
       y="Reservations (log scale)") +
  scale_y_log10()

demand.forecast.plot
```



Question 4

At what time of the year should we expect to have sold 10% of our new year's night? And 50%? And 80%? How can this information be useful for pricing our listings?

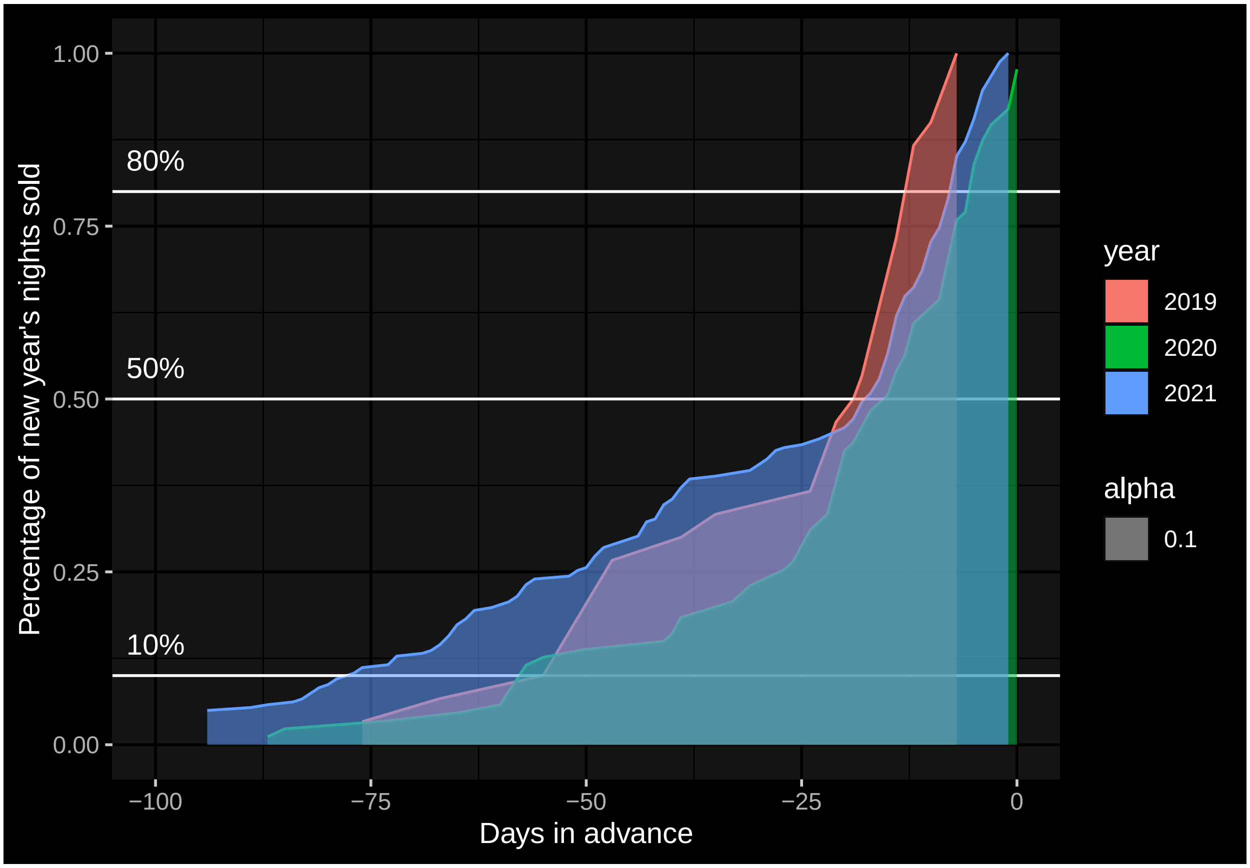
```
ny.nights <- daily.revenue.listings %>%
  filter(occupancy==1, blocked==0) %>%
  select(date, creation_date) %>%
  filter(day(date)==31, month(date)==12) %>%
  group_by(date, creation_date) %>%
  summarise(reservations=n()) %>%
  group_by(date) %>%
  mutate(percentage_until=cumsum(reservations)/sum(reservations),
         advance=as.numeric(creation_date-date),
         year=as.character(year(date)))
```

```
## `summarise()` has grouped output by 'date'. You can override using the
## `.groups` argument.
```

```
percentage.by.advance.plot <- ny.nights %>%
  ggplot(aes(x=advance, y=percentage_until)) +
  geom_line(aes(color=year)) +
  scale_x_continuous("Days in advance",
                     limits = c(-100, 0)) +
  ylab("Percentage of new year's nights sold") +
  geom_hline(yintercept = 0.1) +
```

```
  geom_hline(yintercept = 0.5) +
  geom_hline(yintercept = 0.8) +
  geom_text(aes(-100, 0.1, label="10%", vjust = -1), data=data.frame()) +
  geom_text(aes(-100, 0.5, label="50%", vjust = -1), data=data.frame()) +
  geom_text(aes(-100, 0.8, label="80%", vjust = -1), data=data.frame())

# For some reason, the area plot does not look right
# in Plotly's Chart Studio, so I isolated the area layer here
(percentage.by.advance.plot
+ geom_area(aes(fill=year, alpha=0.1), position="identity"))
```

```
## Warning: Removed 11 row(s) containing missing values (geom_path).
```



Answers

```
advances.by.year <- ny.nights %>%
  group_by(year) %>%
  summarise(ten=advance[which(percentage_until >= 0.1)][1],
            fifty=advance[which(percentage_until >= 0.5)][1],
            eighty=advance[which(percentage_until >= 0.8)][1])

advances.by.year
```

```
## # A tibble: 3 x 4
##   year    ten fifty eighty
##   <chr> <dbl> <dbl>  <dbl>
## 1 2019    -55   -19    -13
## 2 2020    -57   -15     -5
```

```
## 3 2021    -77   -17     -7
```

```r
predicted.advances <- advances.by.year %>% select(ten:eighty) %>% colMeans
predicted.dates <- as_date(sapply(predicted.advances, function(advance){
  ymd("2022-12-31") + period(ceiling(advance), units="day")
}))

predicted.dates
```

```
##          ten        fifty       eighty
## "2022-10-29" "2022-12-14" "2022-12-23"
```

As uncovered in the EDA script, reservation advance, apart from the cleaning fees when applied, is the most relevant predictor for a listing's last offered price. Pricing is considerably dependent on reservation advance since it carries a trade-off: on one hand, a costumer is willing to pay more for booking to a close date; on the other, there is a strong incentive to sell as many bookings as possible up to the target date (Guizzardi et al., 2017).

To set an optimal price is to reach an agreement between what is mutually interesting for both the business and its clients.

Furthermore, a model that predicts the odds of selling a booking given the percentage of available listings for a particular date at a particular price may be used to maximize a revenue function through the price variable.