Visual analytics of hotel bookings data

Julià Minguillón

April 2025

NOTE: this tutorial uses R + RStudio + some R packages to show the potential of using data visualization for inspecting and analyzing a data set. We strongly recommend you to explore the following links:

1) RStudio: https://posit.co/downloads/
2) ggplot2: https://ggplot2.tidyverse.org/
3) extensions: https://exts.ggplot2.tidyverse.org/gallery/
4) ggmosaic: this package has been removed from CRAN, it is necessary to install and older version:

Download and install RTools from "https://cran.rstudio.com/bin/windows/Rtools/rtools45/rtools.html"

Download ggmosaic running install.packages("ggmosaic", repos = c("https://haleyjeppson.r-universe.dev", "https://cloud.r-project.org"))

Load packages

```
# Install ggmosaic if not already installed
# Note: ggmosaic has been removed from CRAN, install from alternative repository
if (!require("ggmosaic", quietly = TRUE)) {
  tryCatch({
    install.packages("ggmosaic", repos = c("https://haleyjeppson.r-universe.dev", "https://cloud.r-proj
    if (!require("ggmosaic", quietly = TRUE)) {
      warning("ggmosaic installation failed. Some mosaic plots may not work.")
    }
  }, error = function(e) {
    warning("Error installing ggmosaic: ", e$message, "\nSome mosaic plots may not work.")
  })
}

# Load packages
if (require("ggmosaic", quietly = TRUE)) {
  library("ggmosaic")
} else {
  warning("ggmosaic not available. Mosaic plots will be skipped.")
}
library("ggplot2")
library("fitdistrplus")
```

```
## Loading required package: MASS
```

```
## Loading required package: survival
```

```
library("MASS")
library("survival")
library("ggstatsplot")
```

```
## You can cite this package as:
##      Patil, I. (2021). Visualizations with statistical details: The 'ggstatsplot' approach.
##      Journal of Open Source Software, 6(61), 3167, doi:10.21105/joss.03167
```

```
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.2.0     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Data loading and dimensions (N x M)

We read the dataset in CSV format, with 119,390 rows and 32 columns:

```
x=read.csv("hotel_bookings.csv", stringsAsFactors = T)
dim(x)
```

```
## [1] 119390     32
```

Data cleansing

First, we'll inspect the data using the summary() function included in R. You can find an explanation of each variable in the article that describes this dataset in detail, although the variable names are pretty much self-explanatory:

```
##           hotel        is_canceled       lead_time   arrival_date_year
##  City Hotel  :79330   Min.   :0.0000   Min.   :  0   Min.   :2015
##  Resort Hotel:40060   1st Qu.:0.0000   1st Qu.: 18   1st Qu.:2016
##                       Median :0.0000   Median : 69   Median :2016
##                       Mean   :0.3704   Mean   :104   Mean   :2016
##                       3rd Qu.:1.0000   3rd Qu.:160   3rd Qu.:2017
##                       Max.   :1.0000   Max.   :737   Max.   :2017
##
##  arrival_date_month arrival_date_week_number arrival_date_day_of_month
##  August :13877      Min.   : 1.00            Min.   : 1.0
##  July   :12661      1st Qu.:16.00            1st Qu.: 8.0
##  May    :11791      Median :28.00            Median :16.0
##  October:11160      Mean   :27.17            Mean   :15.8
##  April  :11089      3rd Qu.:38.00            3rd Qu.:23.0
##  June   :10939      Max.   :53.00            Max.   :31.0
##  (Other):47873
##  stays_in_weekend_nights stays_in_week_nights     adults
##  Min.   : 0.0000         Min.   : 0.0         Min.   : 0.000
##  1st Qu.: 0.0000         1st Qu.: 1.0         1st Qu.: 2.000
##  Median : 1.0000         Median : 2.0         Median : 2.000
##  Mean   : 0.9276         Mean   : 2.5         Mean   : 1.856
##  3rd Qu.: 2.0000         3rd Qu.: 3.0         3rd Qu.: 2.000
##  Max.   :19.0000         Max.   :50.0         Max.   :55.000
##
```

```
##     children           babies                 meal          country
##  Min.   : 0.0000   Min.   : 0.000000   BB       :92310   PRT    :48590
##  1st Qu.: 0.0000   1st Qu.: 0.000000   FB       :  798   GBR    :12129
##  Median : 0.0000   Median : 0.000000   HB       :14463   FRA    :10415
##  Mean   : 0.1039   Mean   : 0.007949   SC       :10650   ESP    : 8568
##  3rd Qu.: 0.0000   3rd Qu.: 0.000000   Undefined: 1169   DEU    : 7287
##  Max.   :10.0000   Max.   :10.000000                     ITA    : 3766
##  NA's   :4                                               (Other):28635
##        market_segment  distribution_channel is_repeated_guest
##  Online TA    :56477   Corporate: 6677      Min.   :0.00000
##  Offline TA/TO:24219   Direct   :14645      1st Qu.:0.00000
##  Groups       :19811   GDS      :  193      Median :0.00000
##  Direct       :12606   TA/TO    :97870      Mean   :0.03191
##  Corporate    : 5295   Undefined:    5      3rd Qu.:0.00000
##  Complementary:  743                        Max.   :1.00000
##  (Other)      :  239
##  previous_cancellations previous_bookings_not_canceled reserved_room_type
##  Min.   : 0.00000       Min.   : 0.0000                A      :85994
##  1st Qu.: 0.00000       1st Qu.: 0.0000                D      :19201
##  Median : 0.00000       Median : 0.0000                E      : 6535
##  Mean   : 0.08712       Mean   : 0.1371                F      : 2897
##  3rd Qu.: 0.00000       3rd Qu.: 0.0000                G      : 2094
##  Max.   :26.00000       Max.   :72.0000                B      : 1118
##                                                        (Other): 1551
##  assigned_room_type booking_changes       deposit_type        agent
##  A      :74053      Min.   : 0.0000   No Deposit:104641   9      :31961
##  D      :25322      1st Qu.: 0.0000   Non Refund: 14587   NULL   :16340
##  E      : 7806      Median : 0.0000   Refundable:   162   240    :13922
##  F      : 3751      Mean   : 0.2211                       1      : 7191
##  G      : 2553      3rd Qu.: 0.0000                       14     : 3640
##  C      : 2375      Max.   :21.0000                       7      : 3539
##  (Other): 3530                                            (Other):42797
##     company       days_in_waiting_list         customer_type
##  NULL   :112593   Min.   :  0.000      Contract       : 4076
##  40     :   927   1st Qu.:  0.000      Group          :  577
##  223    :   784   Median :  0.000      Transient      :89613
##  67     :   267   Mean   :  2.321      Transient-Party:25124
##  45     :   250   3rd Qu.:  0.000
##  153    :   215   Max.   :391.000
##  (Other):  4354
##       adr          required_car_parking_spaces total_of_special_requests
##  Min.   :  -6.38   Min.   :0.00000             Min.   :0.0000
##  1st Qu.:  69.29   1st Qu.:0.00000             1st Qu.:0.0000
##  Median :  94.58   Median :0.00000             Median :0.0000
##  Mean   : 101.83   Mean   :0.06252             Mean   :0.5714
##  3rd Qu.: 126.00   3rd Qu.:0.00000             3rd Qu.:1.0000
##  Max.   :5400.00   Max.   :8.00000             Max.   :5.0000
##
##  reservation_status reservation_status_date
##  Canceled :43017    2015-10-21:  1461
##  Check-Out:75166    2015-07-06:   805
##  No-Show  : 1207    2016-11-25:   790
##                     2015-01-01:   763
##                     2016-01-18:   625
```

```
##                   2015-07-02:   469
##                   (Other)   :114477
```

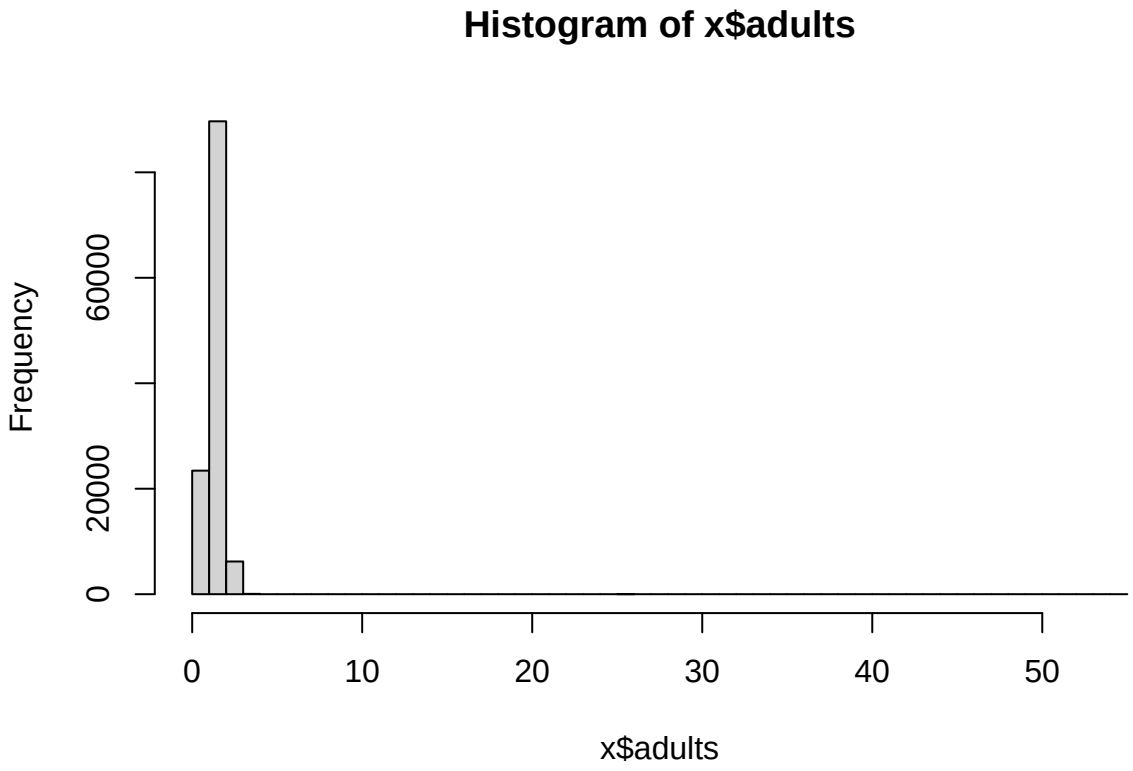
Numerical variables

Some unexpected (outliers?) values for several variables can be observed. For instance:

1) A maximum of 55 in 'adults'
2) A maximum of 10 in 'children' (including also missing values)
3) A maximum of 10 in 'babies'
4) Negative values in the average daily rate ('adr') or or very high

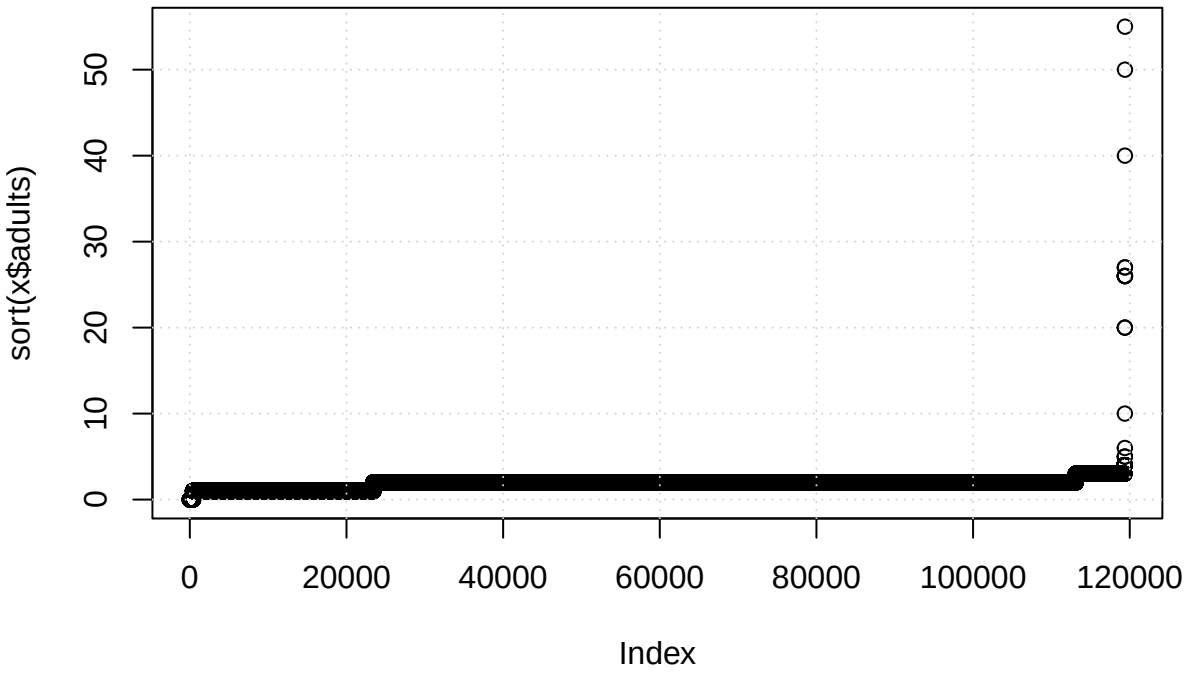
Let's visualize the histogram of the variable 'adults', with at least 55 breaks in the histogram, using the function hist() in R:

```
hist(x$adults,breaks=55)
```



It can be observed that the histogram shows no bars around the value 55, given that this is a very large set and probably it's only one or a few cases. In these cases, to analyze the extreme values of a variable, the values of the variable in question can be represented graphically as follows, ordering and plotting the data (if they are numerical, as in this case):

```
plot(sort(x$adults))
grid()
```



The 'Index' represents the position of the element once it's sorted, but we're more interested in the Y axis, as we can see that some elements have values of 10 or higher. Since this is an integer variable with a limited set of possible values, we can use table() to visualize them:

```
table(x$adults)
```

```
##
##     0     1     2     3     4     5     6    10    20    26    27    40    50
##   403 23027 89680  6202    62     2     1     1     2     5     2     1     1
##    55
##     1
```

As you can see, there's one reservation for 10 adults, two for 20 adults, and so on, up to one for 55 adults! Without going into further detail, we'll remove all rows with reservations for 10 or more adults:

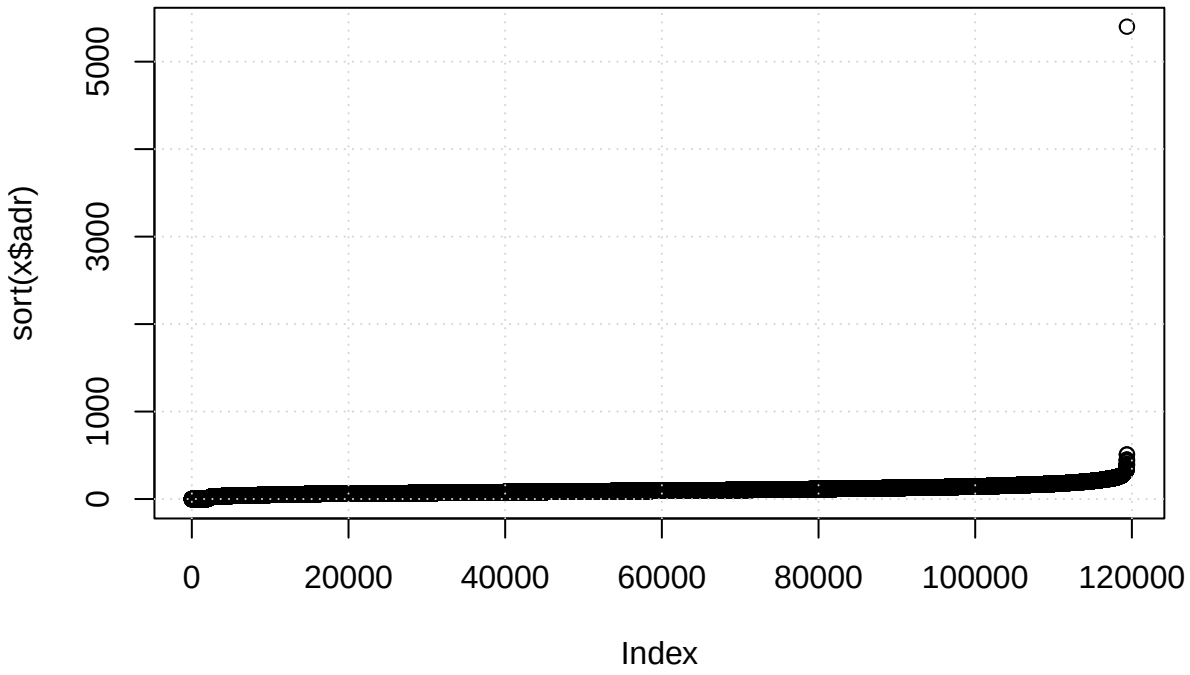
```
x=x[x$adults<10,]
```

Following the same approach, we'll clean outliers for 'children' and 'babies' variables. We use a threshold of 5 (instead of 10) for these variables since they typically have lower values than adults. This removes extreme cases that could contaminate subsequent analyses, especially distributions and group comparisons.

```
# Remove outliers for children (threshold: < 5)
x=x[x$children<5,]
# Remove outliers for babies (threshold: < 5)
x=x[x$babies<5,]
```

The histogram of the 'adr' variable (average daily rate) presents the same problem as the 'adults' variable, so we will simply create a graph with the ordered values again:

```
plot(sort(x$adr))
grid()
```



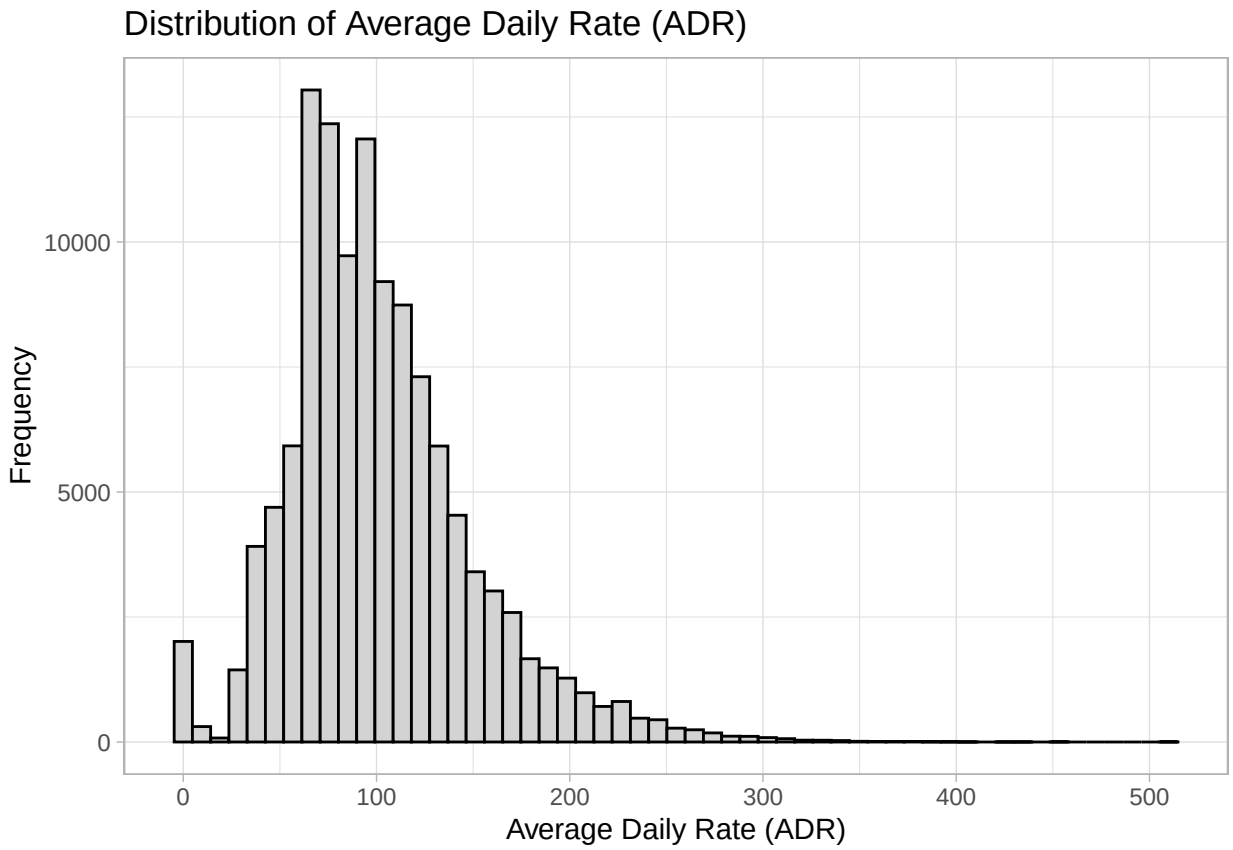
In this case, we observe that only one value is significantly higher than the rest. We consider it an outlier and eliminate it, as well as the negative values which have no a clear explanation, although we keep the 0 values:

```
x=x[x$adr>=0 & x$adr<1000,]
```

The histogram now provides us with some relevant information. We draw it using the ggplot2 package, which offers many more options than hist():

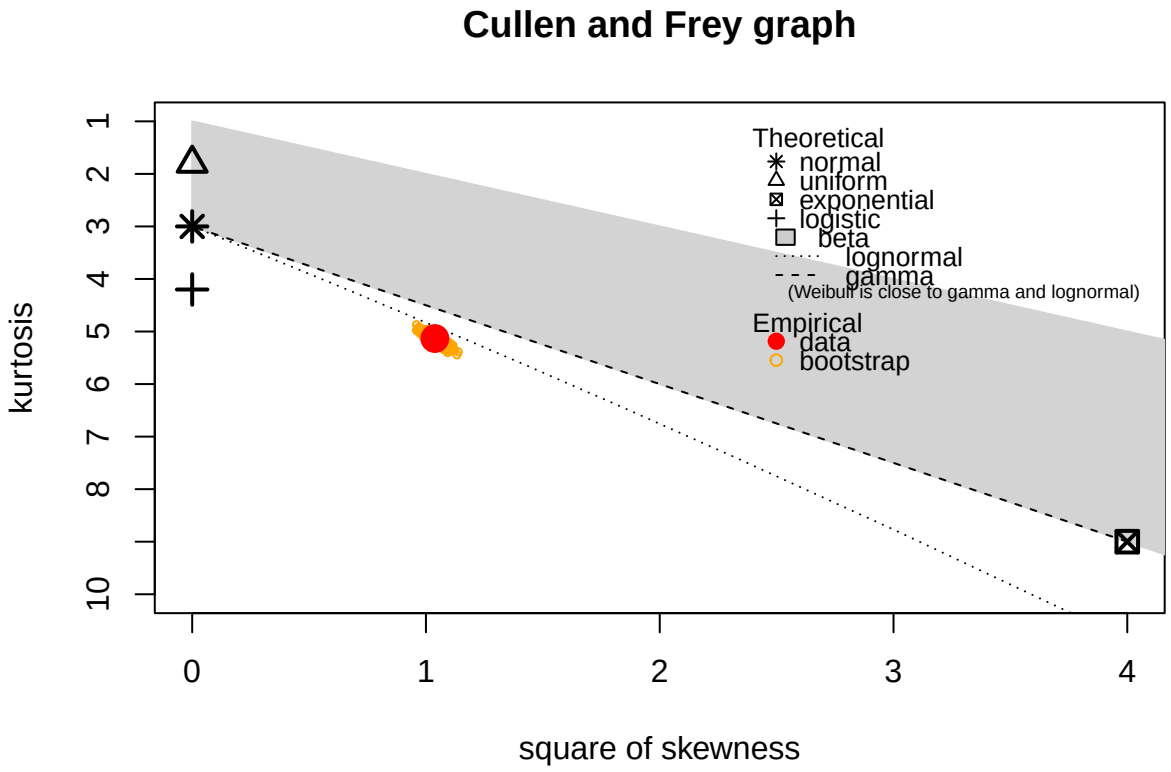
```
ggplot(data=x, aes(x=adr)) +
  geom_histogram(bins=55, colour="black", fill = "lightgray") +
  theme_light() +
  labs(x="Average Daily Rate (ADR)", y="Frequency",
       title="Distribution of Average Daily Rate (ADR)")
```

```
## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



We can see that there is a set of approximately 2,000 zero values, which could be analyzed separately, for example. There are R packages that help us estimate this distribution and the parameters that determine it visually, such as the fitdistrplus package, which provides the descdist() function (caution, slow!):

```r
require(fitdistrplus)
# Filter out non-finite values and ensure we have valid data
adr_clean = x$adr[is.finite(x$adr) & !is.na(x$adr)]
if (length(adr_clean) > 0 && var(adr_clean, na.rm = TRUE) > 0) {
  descdist(adr_clean, boot = 1000)
} else {
  warning("Cannot compute descdist: insufficient valid data or no variance in ADR values.")
}
```




```
## summary statistics
## ------
## min:  0   max:  510
## median:  94.62
## mean:  101.8011
## estimated sd:  48.14287
## estimated skewness:  1.018971
## estimated kurtosis:  5.13332
```

As you can see, the real data (observations, a colored dot) and the simulated data (in other color) approximate what a lognormal distribution might look like. However, to experiment with the cleanest possible data set, we will:

1) remove 0-day stays
2) remove 0-cost stays
3) remove stays with no guests
4) replace the NAs in the children variable with 0

```
# Replace NA values in children with 0
x[is.na(x$children),'children']=0
# Remove invalid stays: must have positive ADR, at least one night, and at least one guest
x=x[x$adr>0 &
    (x$stays_in_week_nights+x$stays_in_weekend_nights)>0 &
    (x$adults+x$children+x$babies)>0,]
```

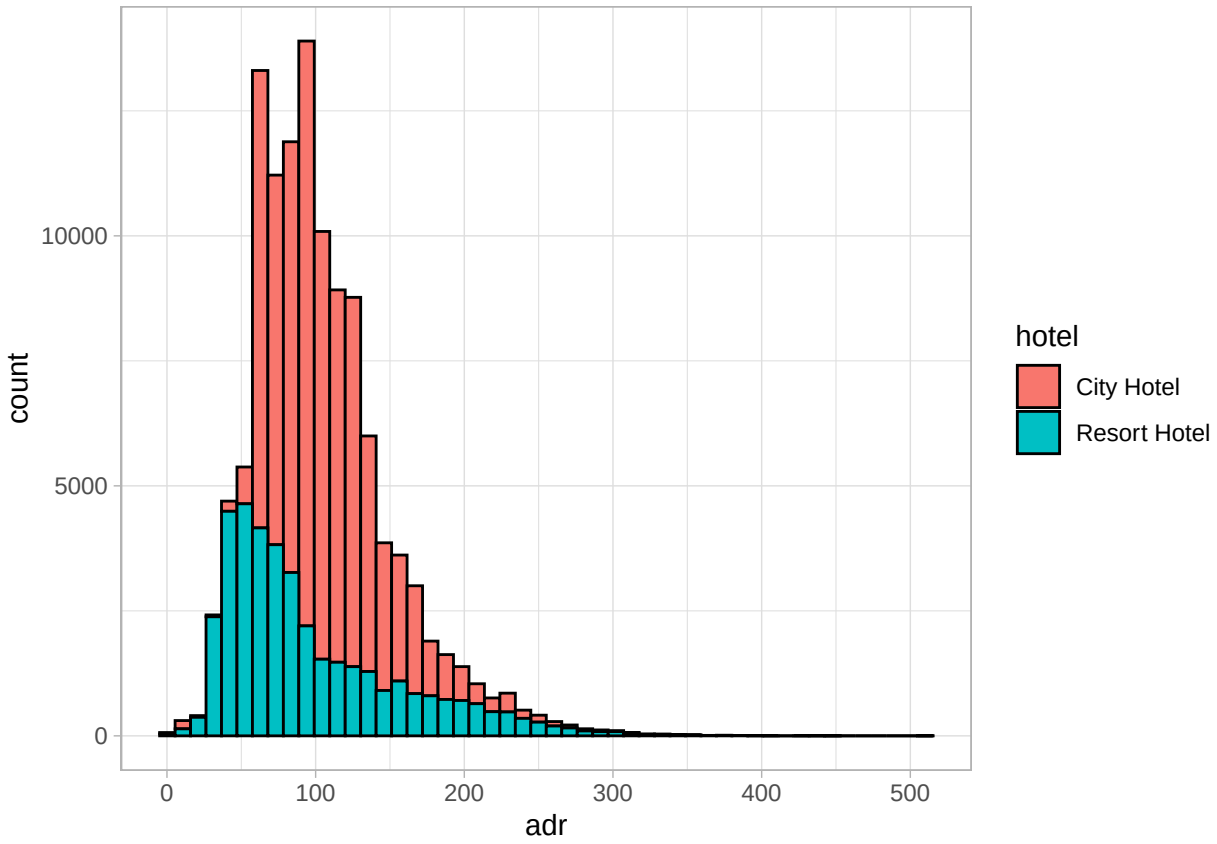
Categorical variables

For categorical variables, the summary() function gives us a first idea of the possible values each can take. For example, in the original set (before removing outliers), there are 79,330 reservations at a city hotel (Lisbon) and 40,060 at a resort (Algarve). We can ask ourselves whether the cost distribution is the same for both groups, either by using the appropriate statistical test or simply by comparing histograms, in this case using

the ggplot2 package, which is much more powerful for creating all kinds of graphs:

```
# require(ggplot2)
ggplot(data=x, aes(x=adr, fill=hotel)) +
  geom_histogram(bins=50, colour="black") +
  theme_light()
```

```
## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

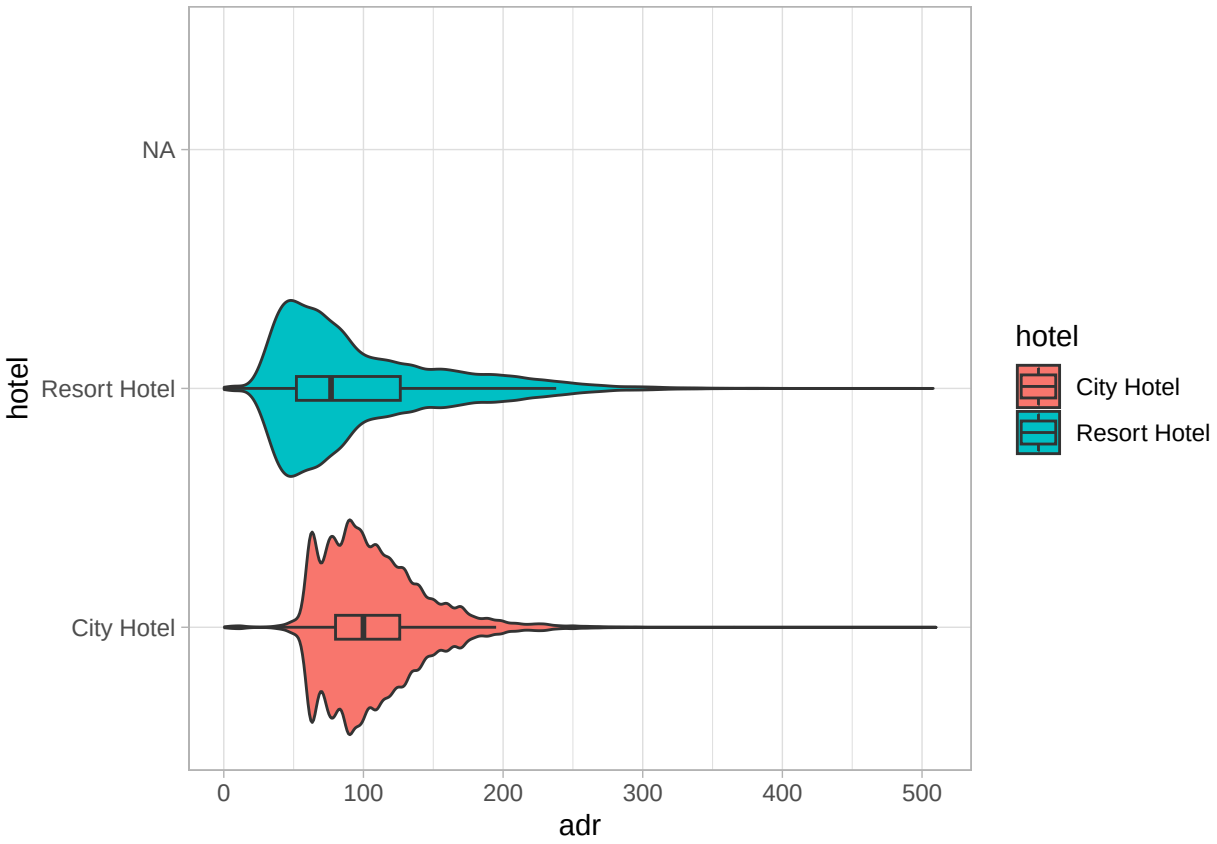


It can be seen that the most common prices in Lisbon (city hotels) are slightly to the right of the most common prices in the Algarve (resort hotels), although the highest prices in Lisbon decrease more rapidly than in the Algarve. By using a violin plot, we can see more detail, especially if we also show the typical quartiles of a box plot:

```
ggplot(data=x, aes(x=hotel, y=adr, fill=hotel)) +
  geom_violin() + geom_boxplot(width=.1, outlier.shape = NA) +
  coord_flip() +
  theme_light()
```

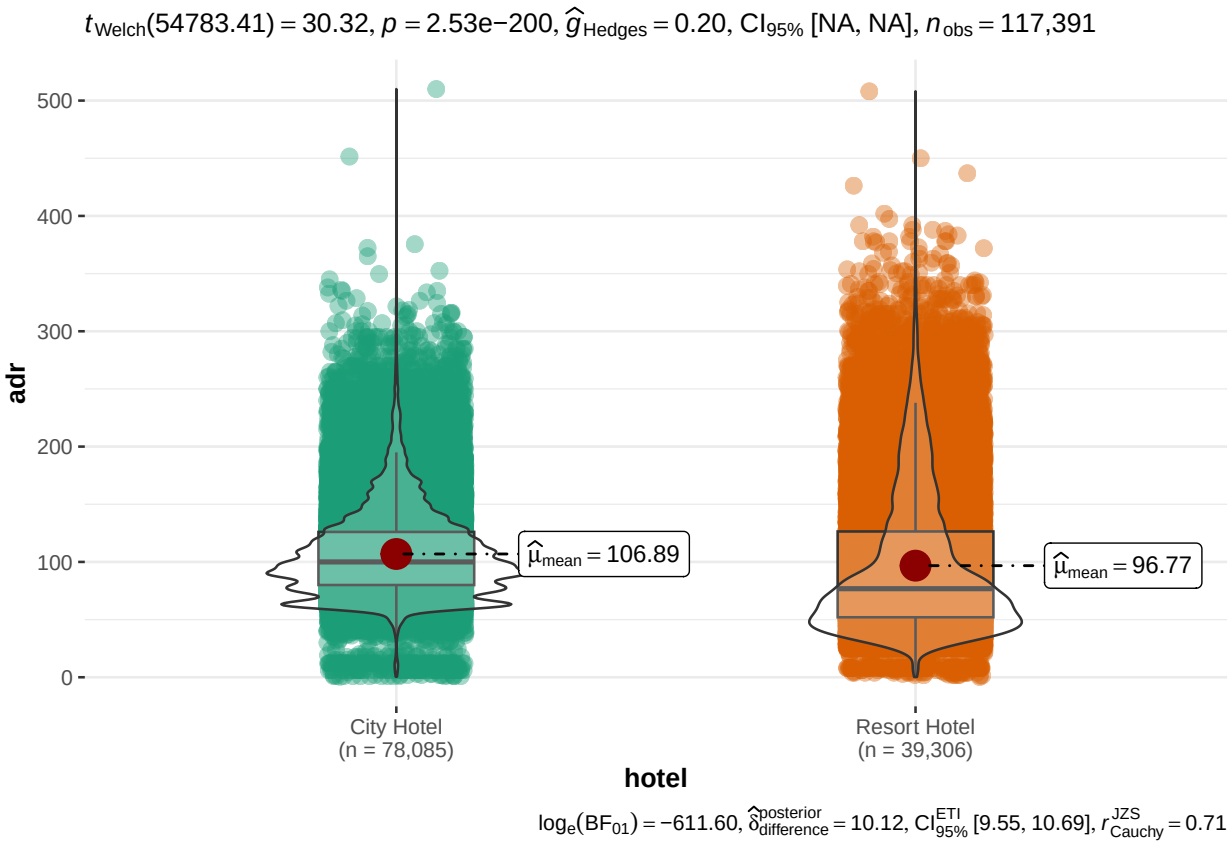
```
## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_ydensity()`).
```

```
## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



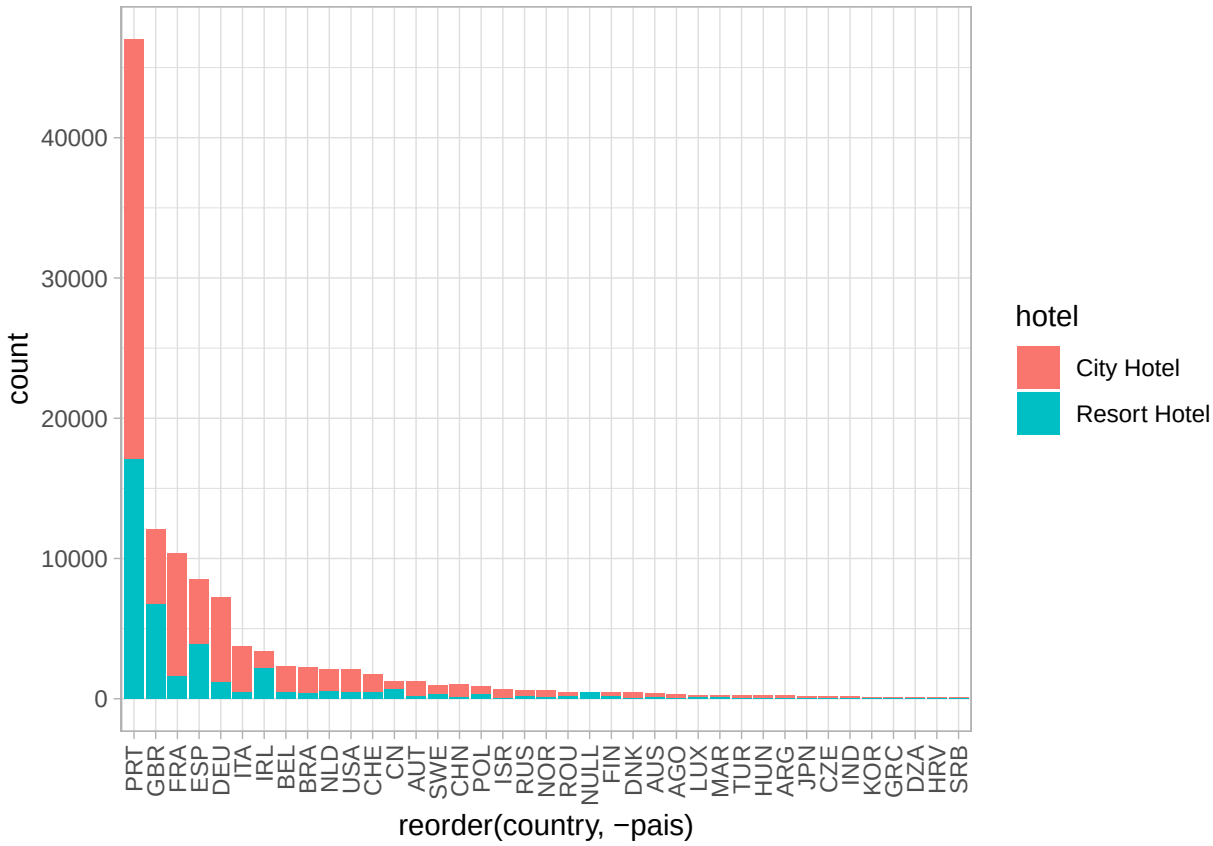
There is an R package called ggstatsplot that has specific functions for each type of graph, including appropriate statistical tests to determine if there are differences between groups:

```
# require(ggstatsplot)
ggbetweenstats(data=x, x=hotel, y=adr)
```



Another interesting variable is the hotel guests' origin ('country'). The problem is that this variable has many different values (178), so we should focus on the countries with the most tourists, also showing whether they choose a city hotel or a resort:

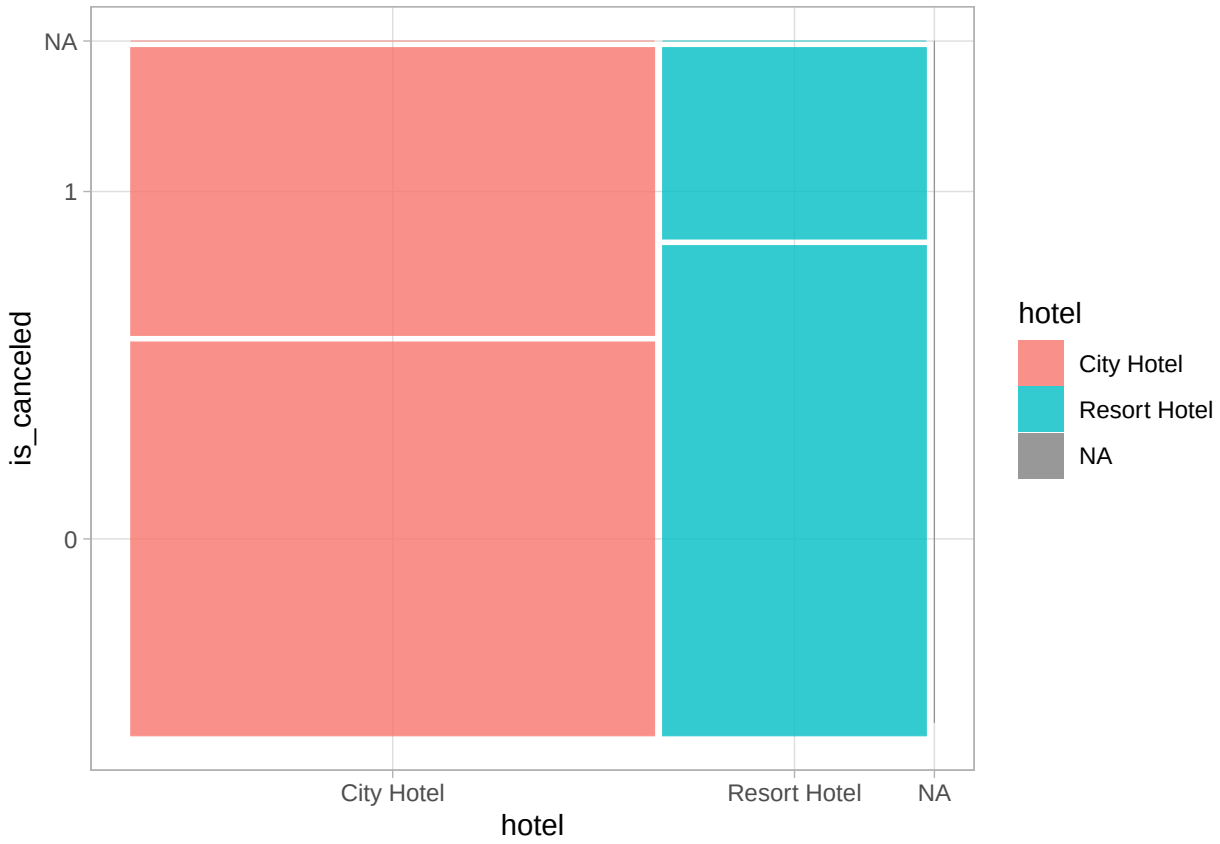
```
# require(tidyverse)
# countries with at least 100 bookings
xx = x %>% group_by(country) %>% mutate(pais=n()) %>% filter(pais>=100)
xx$country=factor(xx$country)
ggplot(data=xx, aes(x=reorder(country, -pais))) +
  geom_bar(stat="count", aes(fill=hotel)) +
  theme_light() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Obviously, Portugal (PRT) ranks first, followed by neighboring countries such as Great Britain, France, and Spain. Visitors from Great Britain and Ireland are most likely to choose a resort, while those from France, Germany, and Italy primarily visit Lisbon.

Another interesting variable is 'is_canceled', which indicates whether a reservation was canceled or not (37.0% of the time). We can observe the relationship between two categorical variables using a mosaic chart:

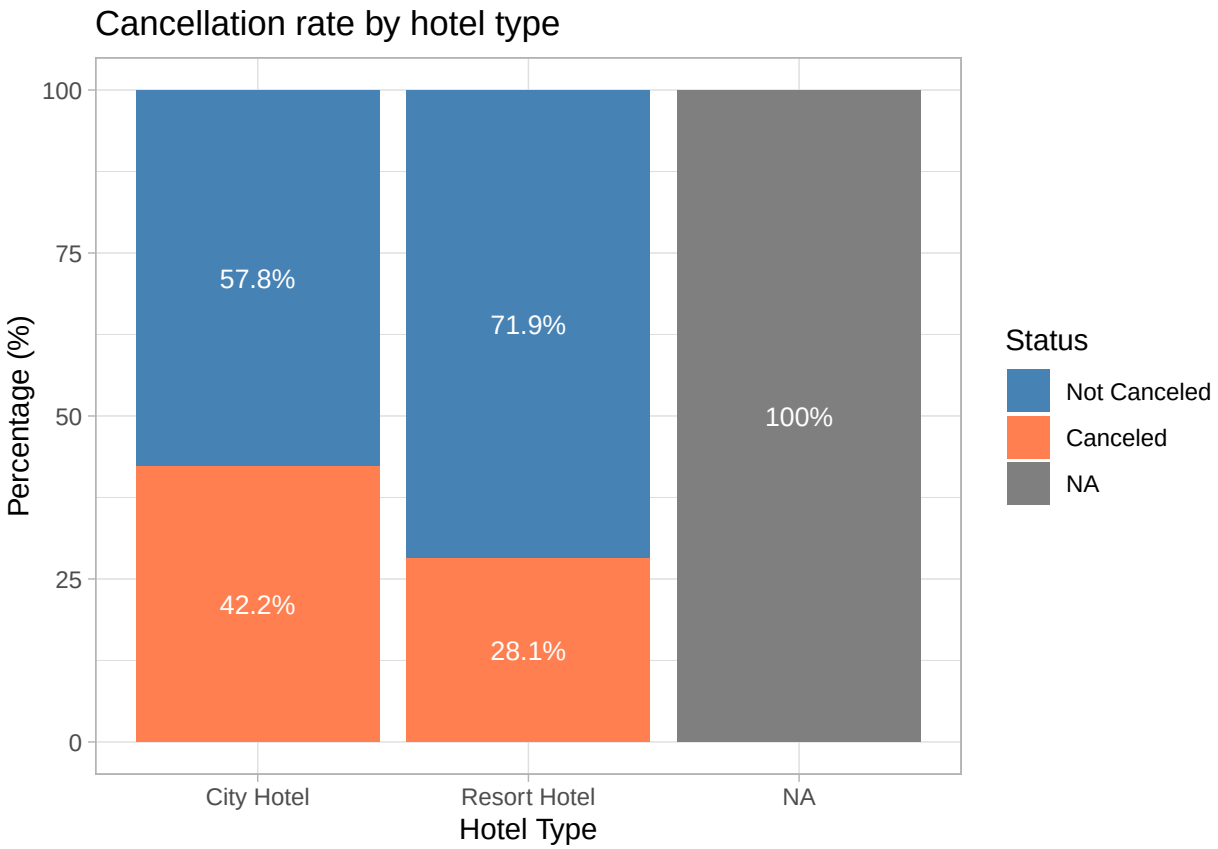
```
# require(ggmosaic)
x$is_canceled=as.factor(x$is_canceled)
if (require("ggmosaic", quietly = TRUE)) {
  ggplot(data=x) +
    geom_mosaic(aes(x=product(is_canceled, hotel), fill=hotel)) +
    theme_light()
} else {
  warning("ggmosaic not available. Using alternative visualization.")
}
```



Important note: In mosaic charts, the Y-axis labels (0/1) represent the categorical values, not the actual cancellation rate percentage. The visual proportions within each column indicate the relative cancellation rates, but for precise numerical values, see the alternative visualization below.

For a more interpretable view, we can use stacked bars showing the proportion of canceled vs non-canceled reservations per hotel:

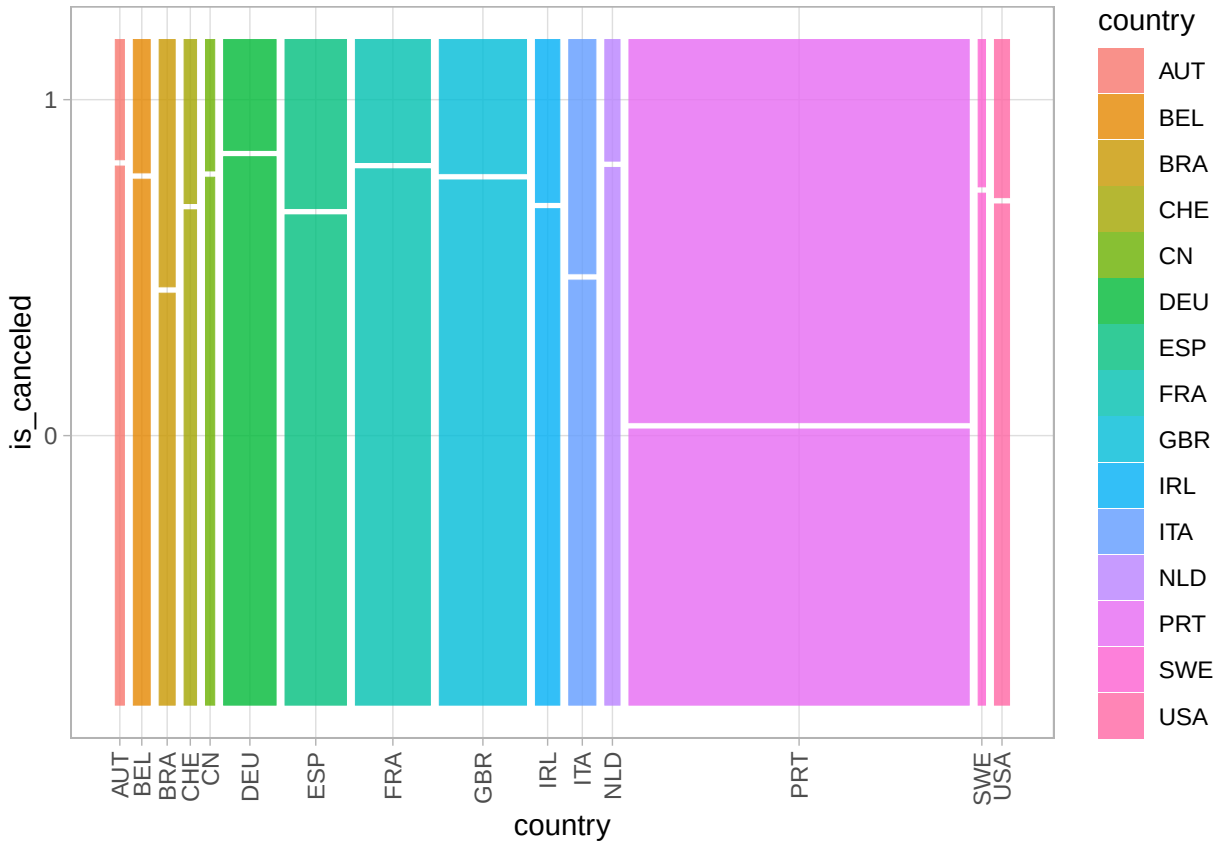
```
# Calculate cancellation rates per hotel
cancel_summary = x %>%
  group_by(hotel, is_canceled) %>%
  summarise(count = n(), .groups = 'drop') %>%
  group_by(hotel) %>%
  mutate(total = sum(count), pct = (count / total) * 100)

ggplot(data=cancel_summary, aes(x=hotel, y=pct, fill=is_canceled)) +
  geom_bar(stat="identity", position="stack") +
  scale_fill_manual(values=c("0"="steelblue", "1"="coral"),
                    labels=c("0"="Not Canceled", "1"="Canceled"),
                    name="Status") +
  labs(x="Hotel Type", y="Percentage (%)",
       title="Cancellation rate by hotel type") +
  theme_light() +
  geom_text(aes(label=paste0(round(pct, 1), "%")),
            position=position_stack(vjust=0.5), color="white", size=3.5)
```



This stacked bar chart shows the exact cancellation percentages for each hotel type, making it easier to interpret than the mosaic chart.

In the case of cancellation by country for the countries with more tourists:

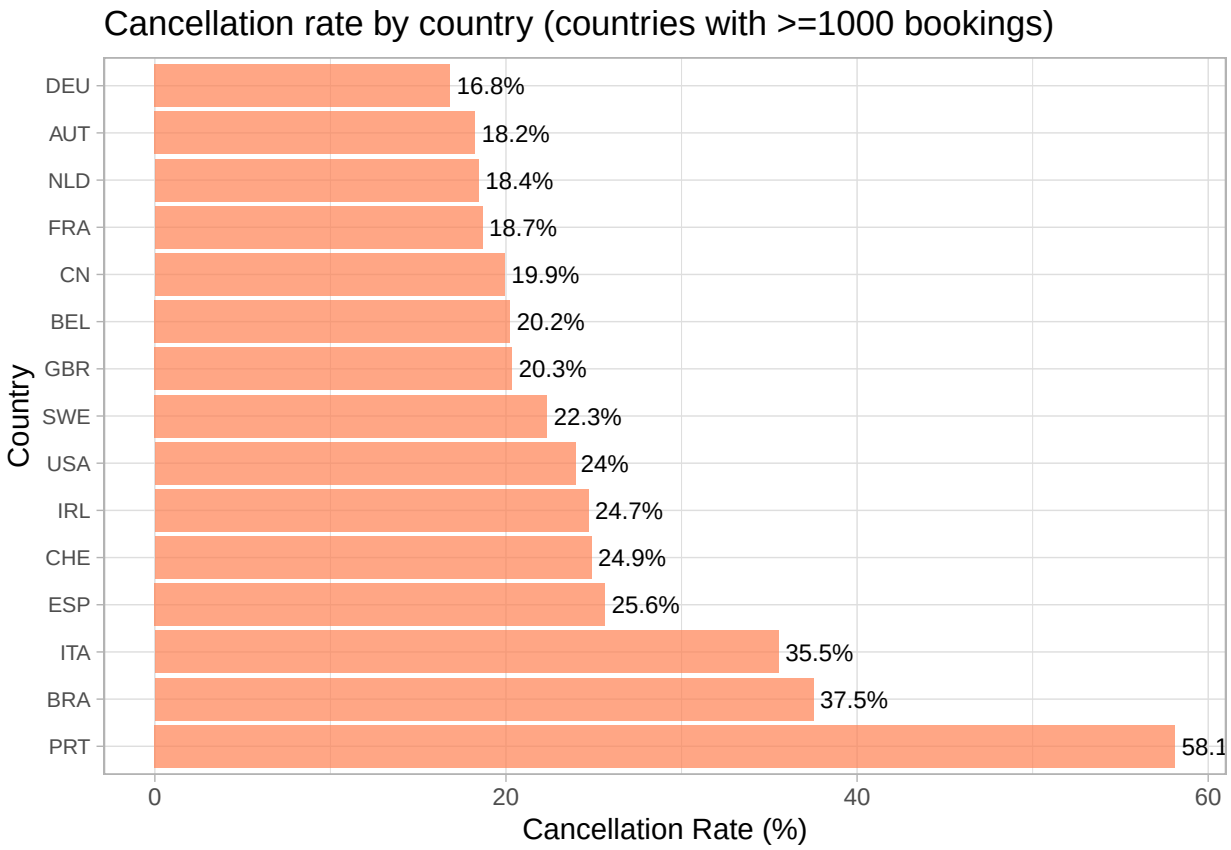
```
# at least 1000 bookings
xx = x %>% group_by(country) %>% mutate(pais=n()) %>% filter(pais>=1000)
xx$country=factor(xx$country)
if (require("ggmosaic", quietly = TRUE)) {
  ggplot(data=xx) +
    geom_mosaic(aes(x=product(is_canceled, country), fill=country)) +
    theme_light() +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
} else {
  warning("ggmosaic not available. Using alternative visualization.")
}
```



Note: As with the previous mosaic chart, the Y-axis labels (0/1) are categorical values, not percentages. For precise cancellation rates by country, see the alternative visualization below.

For a clearer comparison, we can calculate and visualize cancellation rates as percentages:

```
# Calculate cancellation rates per country (at least 1000 bookings)
country_cancel = xx %>%
  group_by(country, is_canceled) %>%
  summarise(count = n(), .groups = 'drop') %>%
  group_by(country) %>%
  mutate(total = sum(count)) %>%
  filter(is_canceled == 1) %>%
  mutate(cancel_rate = (count / total) * 100) %>%
  arrange(desc(cancel_rate))

# Order countries by cancellation rate
country_cancel$country = factor(country_cancel$country,
                                levels = country_cancel$country[order(country_cancel$cancel_rate, decr

ggplot(data=country_cancel, aes(x=country, y=cancel_rate)) +
  geom_bar(stat="identity", fill="coral", alpha=0.7) +
  geom_text(aes(label=paste0(round(cancel_rate, 1), "%")),
            hjust=-0.1, size=3) +
  coord_flip() +
  labs(x="Country", y="Cancellation Rate (%)",
       title="Cancellation rate by country (countries with >=1000 bookings)") +
  theme_light() +
  theme(axis.text.y = element_text(size=8))
```



It can be seen that the cancellation rate is much higher for local tourists (from Portugal, PRT), while it is much lower for the rest of the countries. The bar chart makes it easier to compare exact percentages and identify the countries with the highest cancellation rates.

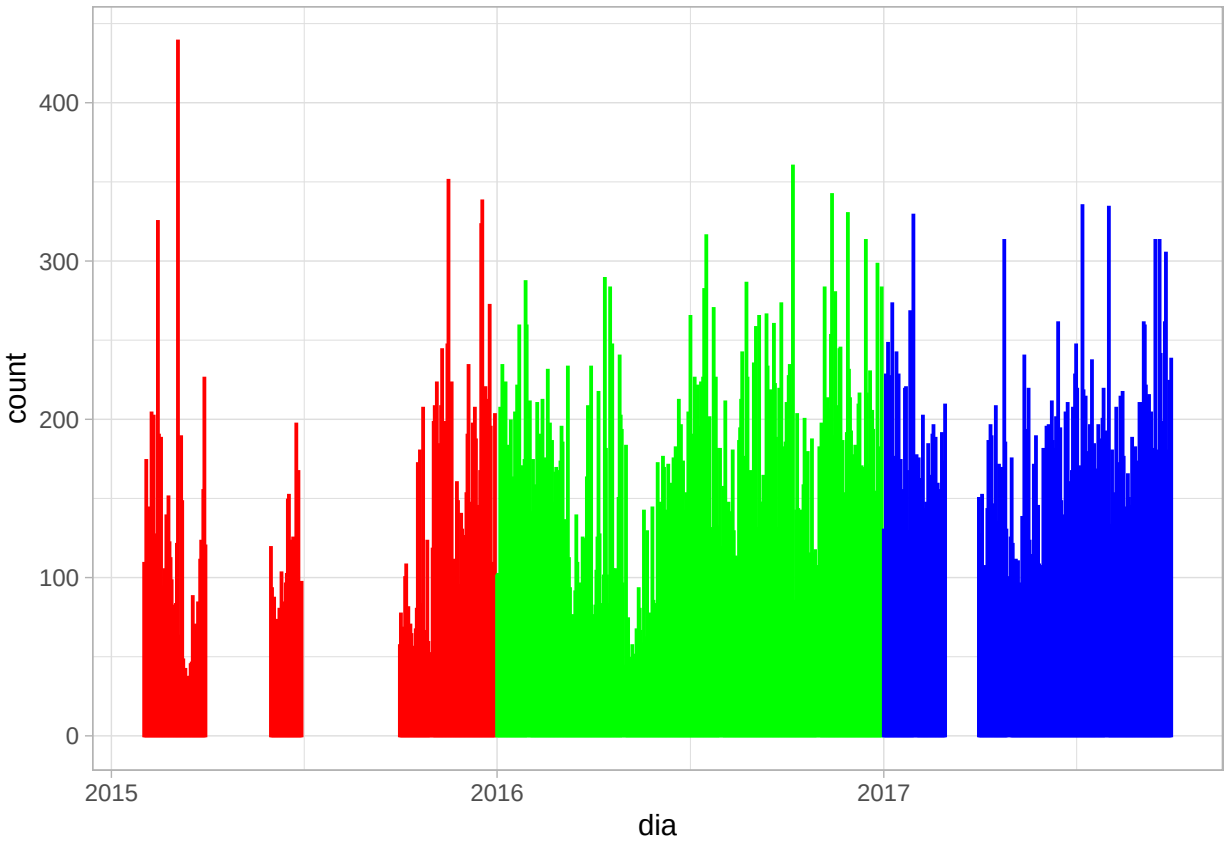
Finally, let's analyze the behavior of reservations relative to the arrival date. First, using the R lubridate package (a marvel for manipulating date and time data), we'll create a 'day' variable to determine the day of the week the hotel was checked in and analyze how many reservations there were each day:

```
# require(lubridate)
# Convert month names to numeric (robust date construction)
month_mapping = c("January"=1, "February"=2, "March"=3, "April"=4, "May"=5, "June"=6,
                  "July"=7, "August"=8, "September"=9, "October"=10, "November"=11, "December"=12)
x$month_num = month_mapping[x$arrival_date_month]
# Construct date using numeric components (more robust)
x$dia = ymd(paste(x$arrival_date_year, x$month_num, x$arrival_date_day_of_month, sep="-"))
```

```
## Warning: 1980 failed to parse.
```

```
ggplot(data=x,aes(x=dia,group=arrival_date_year,color=as.factor(arrival_date_year))) +
  geom_bar() + scale_color_manual(values=c("2015"="red","2016"="green","2017"="blue")) +
  theme_light() +
  theme(legend.position='none')
```

```
## Warning: Removed 1980 rows containing non-finite outside the scale range
## (`stat_count()`).
```



As described in the article, the data covers the period from July 1, 2015, to August 31, 2017. Some peaks can be observed that might be interesting to explain (what happened those days, i.e. 2015-12-05?). You can check Google Trends to get some insights:

https://trends.google.es/trends/explore?date=2015-01-01%202017-12-31&q=lisboa,algarve&hl=es

```
max(table(x$dia))
```

```
## [1] 439
```

```
which.max(table(x$dia))
```

```
## 2015-03-05
##         33
```

With the computed day 'dia', along with the variables 'stays_in_week' and 'weekend_nights', we can try to manually categorize the trip type according to the following criteria (this is arbitrary, clearly improvable):

1) if 'stays_in_weekend_nights' is zero => work trip
2) if 'stays_in_week_nights' is zero or one and in this case the entry is on Friday => weekend
3) if 'stays_in_week_nights' is five and 'stays_in_weekend_nights' is three (that is, from Saturday or Sunday to Saturday or Sunday) => week holiday package
4) if 'stays_in_weekend_nights' is one or two and 'stays_in_week_days' is five or less => work + rest
5) the rest of combinations => holidays

```
# require(lubridate)
x$tipo=ifelse(x$stays_in_weekend_nights==0, "work",
       ifelse(x$stays_in_week_nights==0, "weekend",
       ifelse(x$stays_in_week_nights==1 & wday(x$dia)==6, "weekend",
       ifelse(x$stays_in_week_nights==5 &
              (x$stays_in_weekend_nights==3 |
```

```
            x$stays_in_weekend_nights==4), "package",
       ifelse(x$stays_in_week_nights<=5 &
              x$stays_in_weekend_nights<3, "work+rest",
       "rest")))))
```

One way to refine this classification would be to look at the number of adults, children, and infants to decide whether it is a business traveler or a family. The possibilities are endless: you can enrich the dataset with geographic data (distance between countries), demographic data, economic data (per capita income), weather data (in both Portugal and the country of origin), etc.

NOTE: This is a good example of using ChatGPT or other generative AI to ask interesting questions about the proposed dataset. The following paper describes the potential uses of generative AI in the different phases of creating a data visualization for storytelling:

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10891192

Save cleaned data

After completing the data cleansing process, we save the cleaned dataset to a CSV file so it can be used directly in the visualization script (Component 2 of the PAC), ensuring consistency between the exploratory analysis and the final visualization:

```
# Save cleaned data for use in visualization script
write.csv(x, "hotel_bookings_clean.csv", row.names = FALSE)
cat("Cleaned data saved to hotel_bookings_clean.csv\n")
```

```
## Cleaned data saved to hotel_bookings_clean.csv
```

```
cat("Original rows:", nrow(read.csv("hotel_bookings.csv")), "\n")
```

```
## Original rows: 119390
```

```
cat("Cleaned rows:", nrow(x), "\n")
```

```
## Cleaned rows: 117395
```