Data 607 - Tidyverse Assignment

Manolis Manoli –extended by Subhalaxmi Rout

The assignment is to present a use case for the tidyverse packages and sdemonstrate how to use one or more of the capabilities TidyVerse package with your selected dataset

1) Libraries and Data

Load needed libraries

```
# The easiest way to get all libraries is to load the whole tidyverse but we will load just the package

#library(tidyverse)

# Alternatively, loading all packages that we use:
library(readr)
library(lubridate)
library(dplyr)
library(knitr)
```

create my github path

```
urlRemote  <- "https://raw.githubusercontent.com/"
pathGithub <- "chilleundso/DATA607/master/Tidyverse/"
```

2) Readr

We start of by downloading our csv file from my Githib (originally from https://www.kaggle.com/jessemostipak/hotel-booking-demand) and turning it into a dataframe format:

```
#create HTML URL
fileNamecsv   <- "hotels.csv"
csv_URL <- paste0(urlRemote, pathGithub, fileNamecsv)

#We read the CSV
hotels_raw <- readr::read_csv(csv_URL)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   hotel = col_character(),
##   arrival_date_month = col_character(),
##   meal = col_character(),
##   country = col_character(),
##   market_segment = col_character(),
##   distribution_channel = col_character(),
##   reserved_room_type = col_character(),
##   assigned_room_type = col_character(),
##   deposit_type = col_character(),
##   agent = col_character(),
##   company = col_character(),
##   customer_type = col_character(),
##   reservation_status = col_character(),
##   reservation_status_date = col_character()
## )

## See spec(...) for full column specifications.
```

3) Dplyr::filter

We want to do some early filtering on the data to exclude some special cases from our data set:

```
#we exclude all data rows that have no weekend and no weekday stays:

hotels <- dplyr::filter(hotels_raw, stays_in_weekend_nights != 0 | stays_in_week_nights != 0 )
names(hotels)
```

```
##  [1] "hotel"                          "is_canceled"
##  [3] "lead_time"                      "arrival_date_year"
##  [5] "arrival_date_month"             "arrival_date_week_number"
##  [7] "arrival_date_day_of_month"      "stays_in_weekend_nights"
##  [9] "stays_in_week_nights"           "adults"
## [11] "children"                       "babies"
## [13] "meal"                           "country"
## [15] "market_segment"                 "distribution_channel"
## [17] "is_repeated_guest"              "previous_cancellations"
## [19] "previous_bookings_not_canceled" "reserved_room_type"
## [21] "assigned_room_type"             "booking_changes"
## [23] "deposit_type"                   "agent"
## [25] "company"                        "days_in_waiting_list"
## [27] "customer_type"                  "adr"
## [29] "required_car_parking_spaces"    "total_of_special_requests"
## [31] "reservation_status"             "reservation_status_date"
```

4) Lubridate

As we can see the dataframe has individual columns for the arrival year, month and day so we use lubridate to make an new arrival date column in date format and create a column that hows the check-out date based on adding days stayed in the hotel.

```
#lubridat lets us easily create a date object out of three columns that have year in yyyy, moonths in t
hotels$arrival_date <- paste(hotels$arrival_date_year  , hotels$arrival_date_month, hotels$arrival_date

#we can easily add days to the date to get a cehck-out date column (some )
hotels$checkout_date <- ymd(hotels$arrival_date) + days(hotels$stays_in_weekend_nights) + days(hotels$s
```

5) Dplyr::select

We want to have a look at just the columns we used and created in the above section so we use dplyr::select

```
kable(head(hotels %>%
  dplyr::select(arrival_date_year:arrival_date_day_of_month, arrival_date : checkout_date)))
```

arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	arrival_date	cl
2015	July	27	1	2015-07-01	2
2015	July	27	1	2015-07-01	2
2015	July	27	1	2015-07-01	2
2015	July	27	1	2015-07-01	2
2015	July	27	1	2015-07-01	2
2015	July	27	1	2015-07-01	2

6) Dplyr::summarise/group_by/count/arrange

Lastly we want see how we can use summarise, group by and count to show some overview statistics:

```
#We want to see how the reservation status behaves with the deposit type:

hotels %>%
  dplyr::group_by(deposit_type) %>%
  dplyr::count(reservation_status)
```

```
## # A tibble: 9 x 3
## # Groups:   deposit_type [3]
##   deposit_type reservation_status     n
##   <chr>        <chr>              <int>
## 1 No Deposit   Canceled           28500
## 2 No Deposit   Check-Out          74267
## 3 No Deposit   No-Show             1159
## 4 Non Refund   Canceled           14460
## 5 Non Refund   Check-Out             93
## 6 Non Refund   No-Show               34
## 7 Refundable   Canceled              35
## 8 Refundable   Check-Out            126
## 9 Refundable   No-Show                1
```

```
#To demonstrate the powerful pipe operator in combination with some dplyr functions we look at the aver

hotels %>%
  dplyr::group_by(deposit_type) %>%
  dplyr::summarise(mean = mean(checkout_date-arrival_date), n = n()/nrow(hotels)) %>%
  dplyr::arrange(-mean)
```

```
## # A tibble: 3 x 3
##   deposit_type mean                 n
##   <chr>        <drtn>           <dbl>
## 1 Refundable   3.827160 days 0.00137
## 2 No Deposit   3.551229 days 0.876
## 3 Non Refund   2.712826 days 0.123
```

From the first table we can actually see a large amount of cancellations even within the non-refundable bookings.

The second table shows us that the refunable bookings are on average the longest stay (which makes sense since they are probably the most expensive ones) however there are only very few of them (about 0.1%) We can see that the next longest stays are from bookings without deposits which make up about 88% of bookings. Lastly, no refund bookings are on average the shortest.

7) Summary

dplyr has great functions to summarise, an access certain fields and pivot them around to show any desired permutation of the data

GitHub: https://github.com/chilleundso/DATA607/blob/master/Tidyverse/Data607_Tidyverse_Manolis.Rmd

https://rpubs.com/ManolisM/Data607_tidyverse

Extended by Subhalaxmi Rout

This is really clear explanations of tidyverse packages done by Manolis Manoli.

Below use `stringr`,`tibble`,`ggplot` and `tidyr` package with the `hotels` dataset.

```
library(stringr)
library(tibble)
library(ggplot2)
library(tidyr)
```

8) stringr()

Using stringr we can manupulate the strings in the dataset. From hotels dataset, select `market segment` column which have charecter datatype.

8.1) stringr: str_length(string)

A numeric vector giving number of characters (code points) in each element of the character vector. Missing string have missing length.

```
# select market
hotels_str <- hotels  %>% select(market_segment)
# unique values of market segment
market_segment <- unique(hotels_str$market_segment)
# length of each market segment
stringr::str_length(market_segment)
```

```
## [1]  6  9  9 13 13  6  9  8
```

8.2) stringr: str_subset(string, pattern, negate = FALSE)

Vectorised over string and pattern

```
# market segment which starts with "U"
stringr::str_subset(market_segment, "^U")
```

```
## [1] "Undefined"
```

```
# market segment which not includes "U"
stringr::str_subset(market_segment, "^U", negate = TRUE)
```

```
## [1] "Direct"        "Corporate"     "Online TA"     "Offline TA/TO"
## [5] "Complementary" "Groups"        "Aviation"
```

8.3) stringr: str_locate(string, pattern)

`str_locate`, an integer matrix. First column gives start postion of match, and second column gives end position.

```
# search position for "i"
stringr::str_locate(market_segment, "i")
```

```
##      start end
## [1,]     2   2
## [2,]    NA  NA
## [3,]     4   4
## [4,]     5   5
## [5,]    NA  NA
## [6,]    NA  NA
## [7,]     6   6
## [8,]     3   3
```

```
# search position for "o"
stringr::str_locate(market_segment, "o")
```

```
##      start end
## [1,]    NA  NA
## [2,]     2   2
## [3,]    NA  NA
## [4,]    NA  NA
## [5,]     2   2
## [6,]     3   3
## [7,]    NA  NA
## [8,]     7   7
```

8.4) stringr: str_sort(x, decreasing = FALSE, na_last = TRUE, locale = "en",numeric = FALSE, . . .)

Sort character vector to alphabetically.

```
# sort the string from A to Z
stringr::str_sort(market_segment, decreasing = FALSE)
```

```
## [1] "Aviation"      "Complementary" "Corporate"     "Direct"
## [5] "Groups"        "Offline TA/TO" "Online TA"     "Undefined"
```

```
# reverse the order
stringr::str_sort(market_segment, decreasing = TRUE)
```

```
## [1] "Undefined"     "Online TA"     "Offline TA/TO" "Groups"
## [5] "Direct"        "Corporate"     "Complementary" "Aviation"
```

9) tibble()

`tibble()` will convert a passed dataframe to a tibble

```
tibble::as_tibble(hotels)
```

```
## # A tibble: 118,675 x 34
##    hotel is_canceled lead_time arrival_date_ye~ arrival_date_mo~
##    <chr>       <dbl>     <dbl>            <dbl> <chr>
##  1 Reso~           0         7             2015 July
##  2 Reso~           0        13             2015 July
##  3 Reso~           0        14             2015 July
##  4 Reso~           0        14             2015 July
##  5 Reso~           0         0             2015 July
##  6 Reso~           0         9             2015 July
##  7 Reso~           1        85             2015 July
##  8 Reso~           1        75             2015 July
##  9 Reso~           1        23             2015 July
## 10 Reso~           0        35             2015 July
## # ... with 118,665 more rows, and 29 more variables:
## #   arrival_date_week_number <dbl>, arrival_date_day_of_month <dbl>,
## #   stays_in_weekend_nights <dbl>, stays_in_week_nights <dbl>, adults <dbl>,
## #   children <dbl>, babies <dbl>, meal <chr>, country <chr>,
## #   market_segment <chr>, distribution_channel <chr>, is_repeated_guest <dbl>,
## #   previous_cancellations <dbl>, previous_bookings_not_canceled <dbl>,
## #   reserved_room_type <chr>, assigned_room_type <chr>, booking_changes <dbl>,
## #   deposit_type <chr>, agent <chr>, company <chr>, days_in_waiting_list <dbl>,
## #   customer_type <chr>, adr <dbl>, required_car_parking_spaces <dbl>,
## #   total_of_special_requests <dbl>, reservation_status <chr>,
## #   reservation_status_date <chr>, arrival_date <date>, checkout_date <date>
```

Here we can see, below column name datatype and table shows in a structure way.

10) tidyr

Using tidyr package, easy to work with tidy data. There are many functions availabe in this package .I will use `Spread()`.

spread : takes two columns (a key-value pair) and spreads them in to multiple columns, making “long” data wider.

```
# getting from Dplyr::summarise/group_by/count/arrange
data <- hotels %>%
  dplyr::group_by(deposit_type) %>%
  dplyr::count(reservation_status)
```

```
# change the table structure, convert deposite_type from column to row
data_tidy <-  tidyr::spread(data, deposit_type, n)
tibble::as_tibble(data_tidy)
```

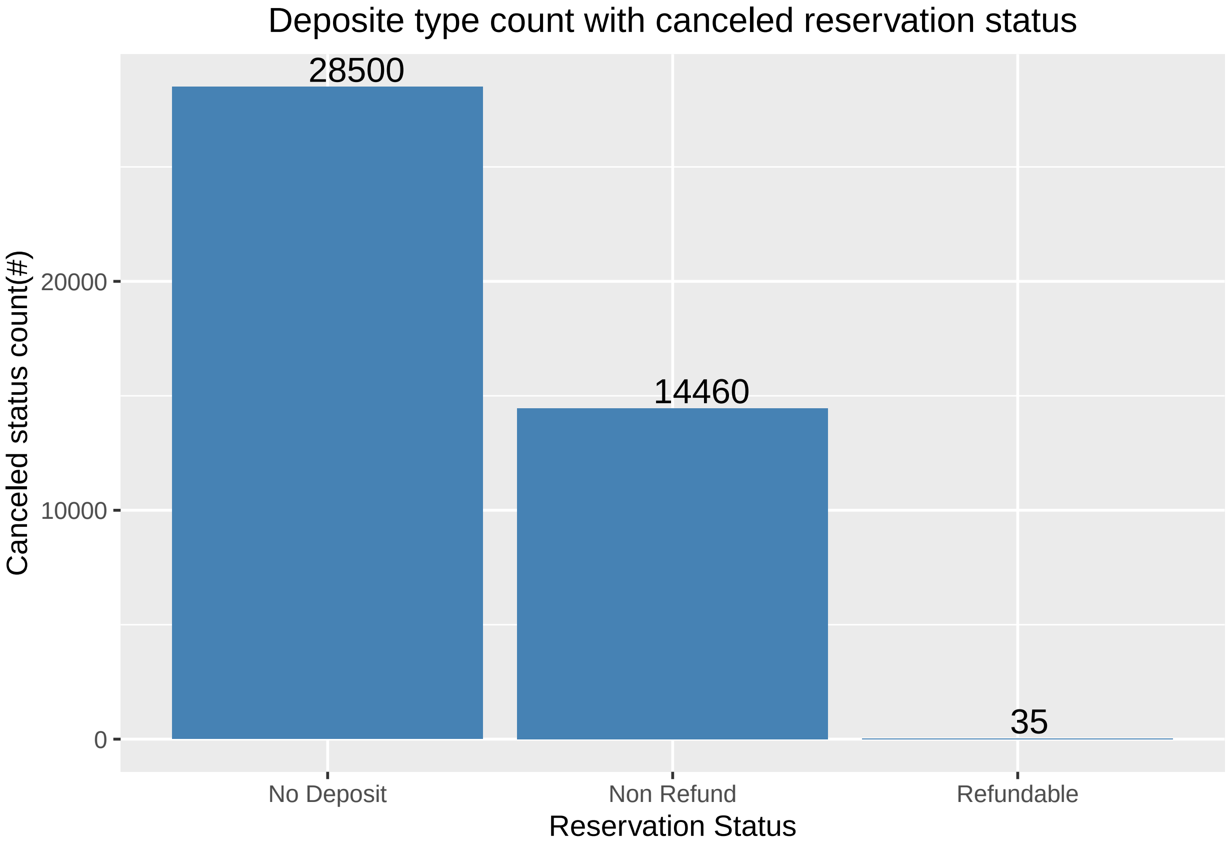
```
## # A tibble: 3 x 4
##   reservation_status `No Deposit` `Non Refund` Refundable
##   <chr>                     <int>        <int>      <int>
## 1 Canceled                  28500        14460         35
## 2 Check-Out                 74267           93        126
## 3 No-Show                    1159           34          1
```

11) ggplot()

ggplot() is a system for declaratively creating graphics for data. Visualizations made with ggplot() are easy to understand and contruct, thanks to an API that allows visualizations to be "built" via layering of graphics and other visual elements.

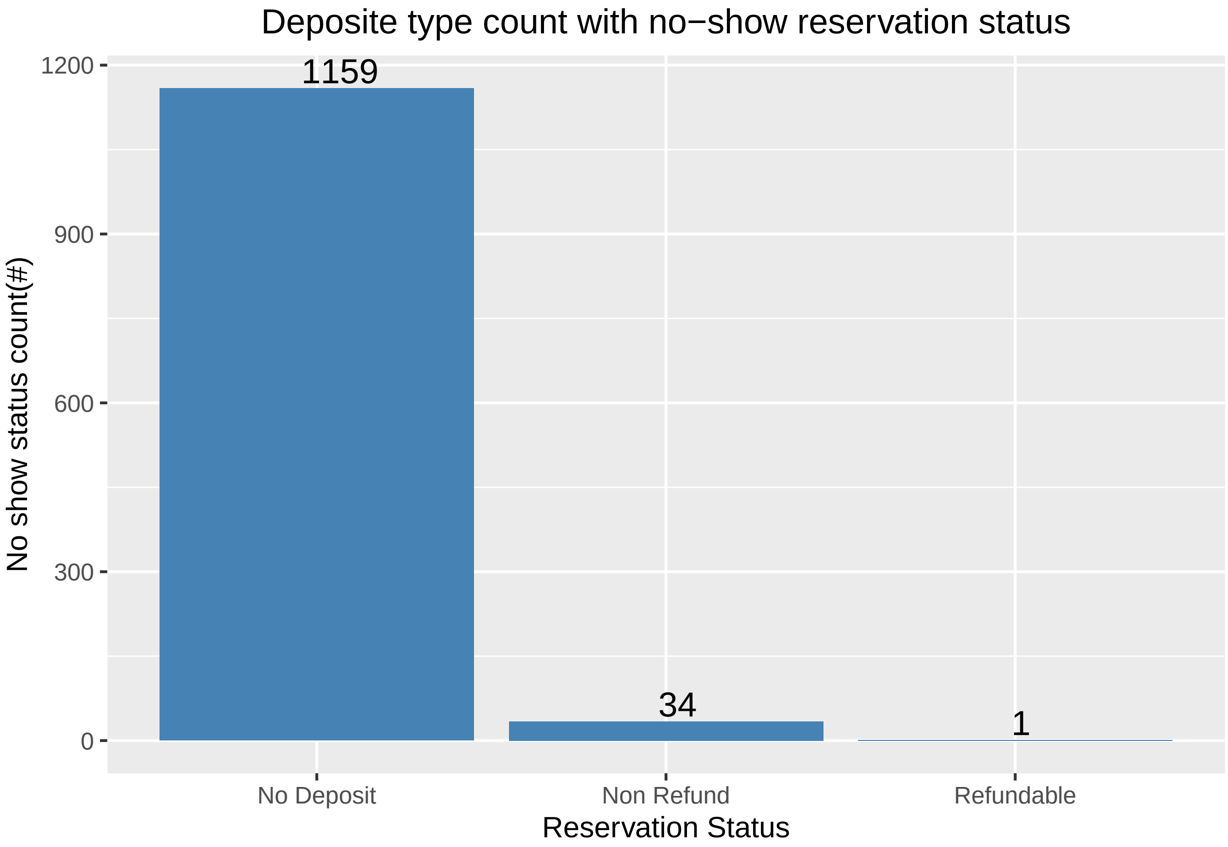
```
data_1 <- data %>% group_by(reservation_status) %>% filter(reservation_status == "Canceled")

ggplot(data = data_1, aes(x = data_1$deposit_type, y = data_1$n )) + geom_bar(stat = "identity", fill =
  xlab("Reservation Status") + ylab("Canceled status count(#)") +
  ggtitle("Deposite type count with canceled reservation status") + theme(plot.title = element_text(hju
  geom_text(aes(x = data_1$deposit_type,y = data_1$n,label=data_1$n), hjust=0.2,vjust = -0.2,color="bla
```



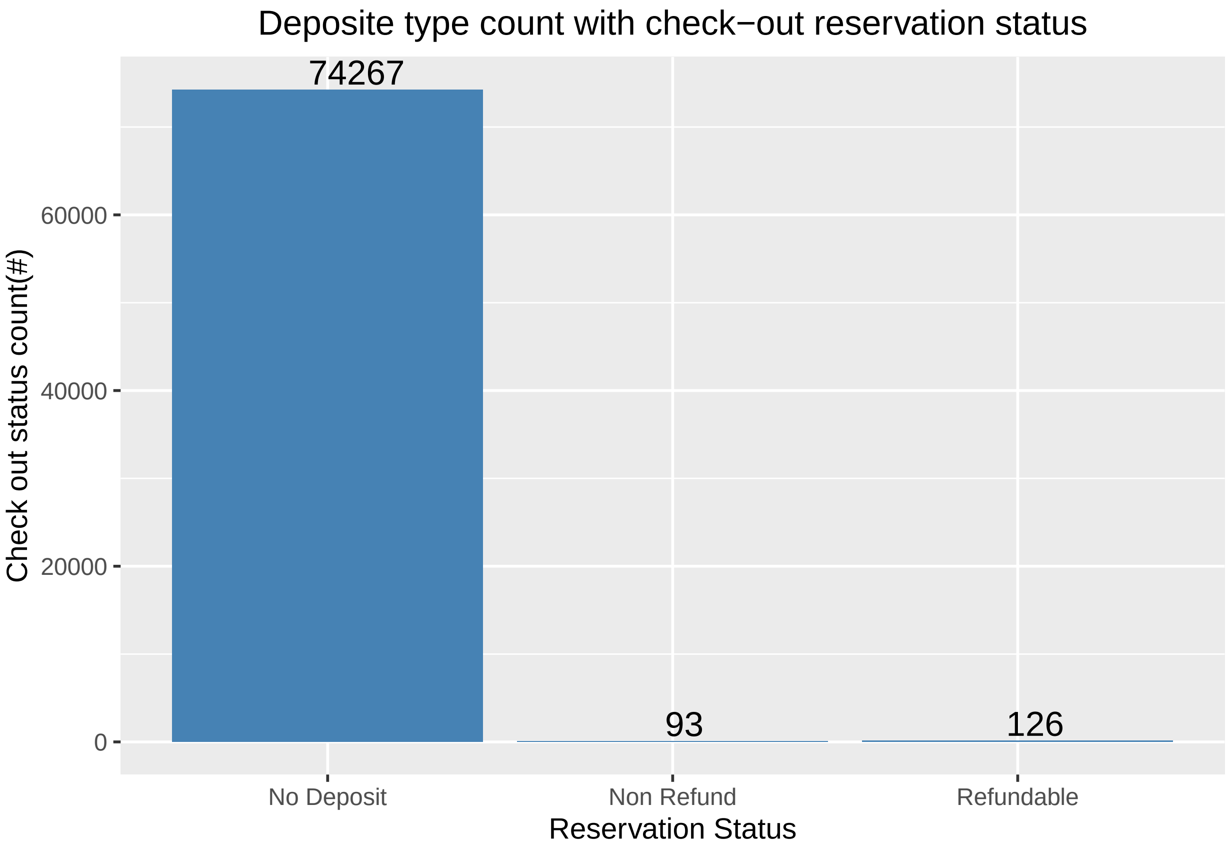
```
data_2 <- data %>% group_by(reservation_status) %>% filter(reservation_status == "No-Show")

ggplot(data = data_2, aes(x = data_2$deposit_type, y = data_2$n )) + geom_bar(stat = "identity", fill =
  xlab("Reservation Status") + ylab("No show status count(#)") +
  ggtitle("Deposite type count with no-show reservation status") + theme(plot.title = element_text(hjus
  geom_text(aes(x = data_2$deposit_type,y = data_2$n,label=data_2$n), hjust=0.2,vjust = -0.2,color="bla
```



```
data_3 <- data %>% group_by(reservation_status) %>% filter(reservation_status == "Check-Out")

ggplot(data = data_3, aes(x = data_3$deposit_type, y = data_3$n )) + geom_bar(stat = "identity", fill =
  xlab("Reservation Status") + ylab("Check out status count(#)") +
  ggtitle("Deposite type count with check-out reservation status") + theme(plot.title = element_text(hj
  geom_text(aes(x = data_3$deposit_type,y = data_3$n,label=data_3$n), hjust=0.2,vjust = -0.2,color="bla
```



Conclusion

Above sections 8,9,10, and 11 given examples of stringr, tidyr, tibble and ggplot package, using these packages, we can do a lot more data analysis. Stringr hepls manupulate string datatype, tidyr helps work with tidy data, tibble convert data to a daatframe, and ggplot helps in visualisation.

Reference

Github: https://github.com/SubhalaxmiRout002/tidyverse/blob/master/Data607_Tidyverse_Manolis_extended_by_Subhalaxmi_Rout.Rmd

RPuB: https://rpubs.com/Subhalaxmi/601580