Biometry for Clinical Research

Kishore Puthezhath

2022-01-11

Contents

Chapter 1

About

“Scientific etiquette demands that a field be defined before its study is begun”-R R Sokal. *Bios* means “life” and *metron* means “measure”.

Chapter 2

r Basics

```
# find out the working directory
getwd()
#> [1] "/Users/drkmenon/Sync/books/biometry"
# setwd in R

setwd("/Users/drkmenon/")

getwd()
#> [1] "/Users/drkmenon"

## change back
```

```
setwd("/Users/drkmenon/Sync/books/biometry")
getwd()
#> [1] "/Users/drkmenon/Sync/books/biometry"

# tab key will popout a list of things we may be looking for after entering
# first 1-2 alphabets

# save this as a project

# myrstats<-save.image("myrstats.rData")

# Essential packages
```

```
library(knitr)
library(tidyverse)
#> -- Attaching packages ------------------- tidyverse 1.3.1 --
#> v ggplot2 3.3.5     v purrr   0.3.4
#> v tibble  3.1.5     v dplyr   1.0.7
#> v tidyr   1.1.4     v stringr 1.4.0
#> v readr   2.0.2     v forcats 0.5.1
#> -- Conflicts ---------------------- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
library(epiR)
#> Loading required package: survival
#> Package epiR 2.0.39 is loaded
#> Type help(epi.about) for summary information
#> Type browseVignettes(package = 'epiR') to learn how to use epiR for applied epidemi
#>
library(epiDisplay)
#> Loading required package: foreign
#> Loading required package: MASS
#>
#> Attaching package: 'MASS'
#> The following object is masked from 'package:dplyr':
#>
#>     select
#> Loading required package: nnet
#>
#> Attaching package: 'epiDisplay'
#> The following object is masked from 'package:ggplot2':
#>
#>     alpha
library(survival)
library(survminer)
#> Loading required package: ggpubr
library(randomizeR)
#> Loading required package: plotrix
```

```
x<- 2
x
#> [1] 2
# x is an object containing variable 2

y<-"male"
y
#> [1] "male"

# y is an object containing variable "male"
# x is numeric/integer and "male" is factor/character
# factors can be grouped for analysis, characters cannot be
```

```
class(x)
#> [1] "numeric"
class(y)
#> [1] "character"



# Convert character to factor

z= as.factor(y)
class(z)
#> [1] "factor"
#concatenation: it is a string of interconnected things
x1<- c(1,2,3,4,5)
x1
#> [1] 1 2 3 4 5

# different methods of producing concatenated vectors

x1<-c(1:5)
x1<-seq(from=1, to=5, by=1)
x1
#> [1] 1 2 3 4 5

# we can also create repeated sequence in R

x2<-rep(1, times=5)
x2
#> [1] 1 1 1 1 1

x3<-rep(seq(from=2, to=6, by=0.05), times=5)
x3
#>   [1] 2.00 2.05 2.10 2.15 2.20 2.25 2.30 2.35 2.40 2.45 2.50
#>  [12] 2.55 2.60 2.65 2.70 2.75 2.80 2.85 2.90 2.95 3.00 3.05
#>  [23] 3.10 3.15 3.20 3.25 3.30 3.35 3.40 3.45 3.50 3.55 3.60
#>  [34] 3.65 3.70 3.75 3.80 3.85 3.90 3.95 4.00 4.05 4.10 4.15
#>  [45] 4.20 4.25 4.30 4.35 4.40 4.45 4.50 4.55 4.60 4.65 4.70
#>  [56] 4.75 4.80 4.85 4.90 4.95 5.00 5.05 5.10 5.15 5.20 5.25
#>  [67] 5.30 5.35 5.40 5.45 5.50 5.55 5.60 5.65 5.70 5.75 5.80
#>  [78] 5.85 5.90 5.95 6.00 2.00 2.05 2.10 2.15 2.20 2.25 2.30
#>  [89] 2.35 2.40 2.45 2.50 2.55 2.60 2.65 2.70 2.75 2.80 2.85
#> [100] 2.90 2.95 3.00 3.05 3.10 3.15 3.20 3.25 3.30 3.35 3.40
#> [111] 3.45 3.50 3.55 3.60 3.65 3.70 3.75 3.80 3.85 3.90 3.95
#> [122] 4.00 4.05 4.10 4.15 4.20 4.25 4.30 4.35 4.40 4.45 4.50
#> [133] 4.55 4.60 4.65 4.70 4.75 4.80 4.85 4.90 4.95 5.00 5.05
#> [144] 5.10 5.15 5.20 5.25 5.30 5.35 5.40 5.45 5.50 5.55 5.60
#> [155] 5.65 5.70 5.75 5.80 5.85 5.90 5.95 6.00 2.00 2.05 2.10
#> [166] 2.15 2.20 2.25 2.30 2.35 2.40 2.45 2.50 2.55 2.60 2.65
#> [177] 2.70 2.75 2.80 2.85 2.90 2.95 3.00 3.05 3.10 3.15 3.20
#> [188] 3.25 3.30 3.35 3.40 3.45 3.50 3.55 3.60 3.65 3.70 3.75
#> [199] 3.80 3.85 3.90 3.95 4.00 4.05 4.10 4.15 4.20 4.25 4.30
#> [210] 4.35 4.40 4.45 4.50 4.55 4.60 4.65 4.70 4.75 4.80 4.85
#> [221] 4.90 4.95 5.00 5.05 5.10 5.15 5.20 5.25 5.30 5.35 5.40
#> [232] 5.45 5.50 5.55 5.60 5.65 5.70 5.75 5.80 5.85 5.90 5.95
#> [243] 6.00 2.00 2.05 2.10 2.15 2.20 2.25 2.30 2.35 2.40 2.45
#> [254] 2.50 2.55 2.60 2.65 2.70 2.75 2.80 2.85 2.90 2.95 3.00
#> [265] 3.05 3.10 3.15 3.20 3.25 3.30 3.35 3.40 3.45 3.50 3.55
#> [276] 3.60 3.65 3.70 3.75 3.80 3.85 3.90 3.95 4.00 4.05 4.10
#> [287] 4.15 4.20 4.25 4.30 4.35 4.40 4.45 4.50 4.55 4.60 4.65
#> [298] 4.70 4.75 4.80 4.85 4.90 4.95 5.00 5.05 5.10 5.15 5.20
#> [309] 5.25 5.30 5.35 5.40 5.45 5.50 5.55 5.60 5.65 5.70 5.75
#> [320] 5.80 5.85 5.90 5.95 6.00 2.00 2.05 2.10 2.15 2.20 2.25
#> [331] 2.30 2.35 2.40 2.45 2.50 2.55 2.60 2.65 2.70 2.75 2.80
#> [342] 2.85 2.90 2.95 3.00 3.05 3.10 3.15 3.20 3.25 3.30 3.35
#> [353] 3.40 3.45 3.50 3.55 3.60 3.65 3.70 3.75 3.80 3.85 3.90
#> [364] 3.95 4.00 4.05 4.10 4.15 4.20 4.25 4.30 4.35 4.40 4.45
#> [375] 4.50 4.55 4.60 4.65 4.70 4.75 4.80 4.85 4.90 4.95 5.00
#> [386] 5.05 5.10 5.15 5.20 5.25 5.30 5.35 5.40 5.45 5.50 5.55
#> [397] 5.60 5.65 5.70 5.75 5.80 5.85 5.90 5.95 6.00

# extracting elements from concatenated string

x1
#> [1] 1 2 3 4 5
x1[3]
#> [1] 3
x1[1:3]
#> [1] 1 2 3
x1[c(2,4)]
#> [1] 2 4
x1[-1]
#> [1] 2 3 4 5

# Matrix of elements
# Vector is a list of numbers/characters
# Matrix is an array of numbers/characters in raws and columns

m1<-matrix(c(1:20),nrow=5, byrow=T)
m1
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    2    3    4
#> [2,]    5    6    7    8
#> [3,]    9   10   11   12
#> [4,]   13   14   15   16
#> [5,]   17   18   19   20

# similar to vectors, matrix can be subsetted

m1[c(2,3), 2]
#> [1]  6 10

# data_frame is a type of matrix
# we can create data_frame in R or we can import

# Creating data_frame
# first we have to create concatenated strings of variable
name<-c(letters[1:10])
age<-seq(from=63, to=82, by=2)
type_surg<- c(0,1,0,0,1,1,1,0,0,0)

# R recognizes this as a number series, we have to covert this to factor
type_surg<-as.factor(type_surg)

pri_event<-c(0,0,0,0,0,1,1,0,0,1)
pri_event<-as.factor(pri_event)
time<-c(24,24,24,24,24,3,2,24,24,7)
test_data<- data.frame(name,age,type_surg,pri_event,time)
test_data
#>    name age type_surg pri_event time
#> 1     a  63         0         0   24
#> 2     b  65         1         0   24
#> 3     c  67         0         0   24
#> 4     d  69         0         0   24
#> 5     e  71         1         0   24
#> 6     f  73         1         1    3
#> 7     g  75         1         1    2
#> 8     h  77         0         0   24
#> 9     i  79         0         0   24
#> 10    j  81         0         1    7

# subsetting can be done. important to remember to specify the column as blank
# after a coma
ageo70<-test_data[age>70,]
ageo70
#>    name age type_surg pri_event time
#> 5      e  71         1         0   24
#> 6      f  73         1         1    3
#> 7      g  75         1         1    2
#> 8      h  77         0         0   24
#> 9      i  79         0         0   24
#> 10     j  81         0         1    7
```

Chapter 3

Data handling

```
# create data_frame

# first we have to create concatenated strings of variable
name<-c(letters[1:10])
age<-seq(from=63, to=82, by=2)
type_surg<- c(0,1,0,0,1,1,1,0,0,0)

# R recognizes this as a number series, we have to covert this to factor
type_surg<-as.factor(type_surg)

pri_event<-c(0,0,0,0,0,1,1,0,0,1)
pri_event<-as.factor(pri_event)
time<-c(24,24,24,24,24,3,2,24,24,7)
test_data<- data.frame(name,age,type_surg,pri_event,time)
test_data
#>    name age type_surg pri_event time
#> 1     a  63         0         0   24
#> 2     b  65         1         0   24
#> 3     c  67         0         0   24
#> 4     d  69         0         0   24
#> 5     e  71         1         0   24
#> 6     f  73         1         1    3
#> 7     g  75         1         1    2
#> 8     h  77         0         0   24
#> 9     i  79         0         0   24
#> 10    j  81         0         1    7

#logic operations
```

```
# add new row to test_data2, cbind and rbind

new_data<-c("k", 63,0,0,24)
new_data
#> [1] "k"  "63" "0"  "0"  "24"

test_data2<-rbind(test_data,new_data)
test_data2
#>    name age type_surg pri_event time
#> 1     a  63         0         0   24
#> 2     b  65         1         0   24
#> 3     c  67         0         0   24
#> 4     d  69         0         0   24
#> 5     e  71         1         0   24
#> 6     f  73         1         1    3
#> 7     g  75         1         1    2
#> 8     h  77         0         0   24
#> 9     i  79         0         0   24
#> 10    j  81         0         1    7
#> 11    k  63         0         0   24

new_data2<-c(10:1)
test_data3<-cbind(test_data,new_data2)
test_data3
#>    name age type_surg pri_event time new_data2
#> 1     a  63         0         0   24        10
#> 2     b  65         1         0   24         9
#> 3     c  67         0         0   24         8
#> 4     d  69         0         0   24         7
#> 5     e  71         1         0   24         6
#> 6     f  73         1         1    3         5
#> 7     g  75         1         1    2         4
#> 8     h  77         0         0   24         3
#> 9     i  79         0         0   24         2
#> 10    j  81         0         1    7         1

# other logic operations, on the test_data in the r basics script

# is age >70?

typage<-age>70
typage[1:5]
#> [1] FALSE FALSE FALSE FALSE  TRUE

# get this answer as 0 and 1
```

```
typage2<-as.numeric(age>70)
typage2
#>  [1] 0 0 0 0 1 1 1 1 1 1

# multiple logic operations

oldtha<-age>70 & type_surg=="1"
oldtha
#>  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
#> [10] FALSE

# add this to our data as a new column

test_data
#>    name age type_surg pri_event time
#> 1     a  63         0         0   24
#> 2     b  65         1         0   24
#> 3     c  67         0         0   24
#> 4     d  69         0         0   24
#> 5     e  71         1         0   24
#> 6     f  73         1         1    3
#> 7     g  75         1         1    2
#> 8     h  77         0         0   24
#> 9     i  79         0         0   24
#> 10    j  81         0         1    7
test_data5<-cbind(test_data,oldtha)
test_data5
#>    name age type_surg pri_event time oldtha
#> 1     a  63         0         0   24  FALSE
#> 2     b  65         1         0   24  FALSE
#> 3     c  67         0         0   24  FALSE
#> 4     d  69         0         0   24  FALSE
#> 5     e  71         1         0   24   TRUE
#> 6     f  73         1         1    3   TRUE
#> 7     g  75         1         1    2   TRUE
#> 8     h  77         0         0   24  FALSE
#> 9     i  79         0         0   24  FALSE
#> 10    j  81         0         1    7  FALSE

# Clearing workspace inr

rm(list=ls())

# remember how to import dataset.
# new_datax<-read.csv(file.choose(), header = T)
# create table
# table()
```

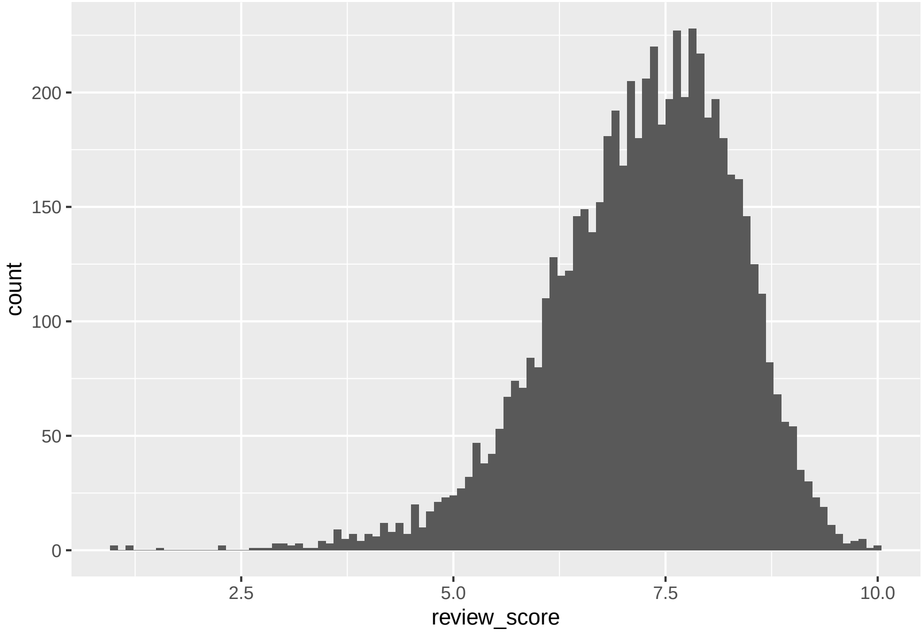
Chapter 4

Data Cleaning

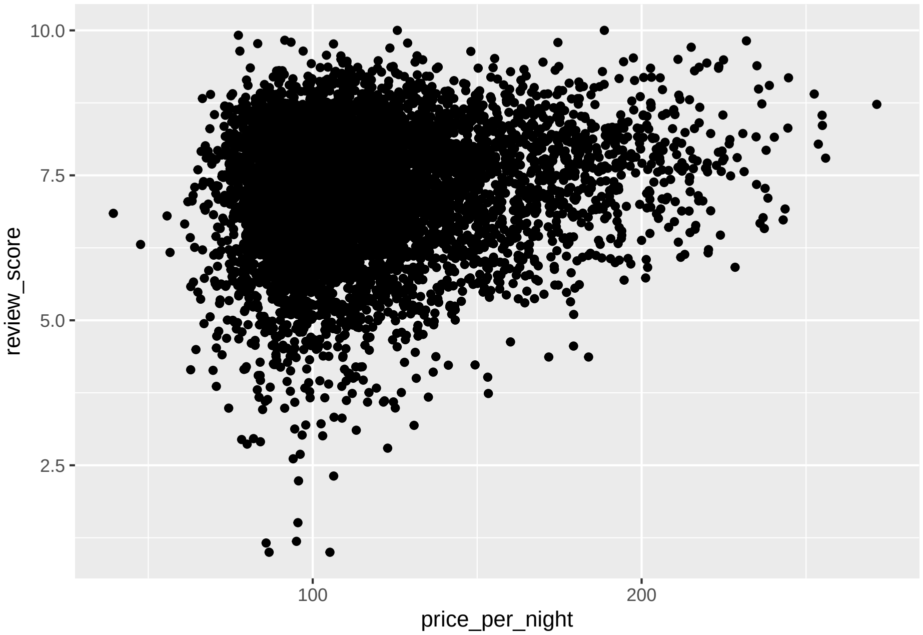
Main actions are select(), filter(), group_by(), mutate(), summarise(),full_join(), pivot_wide() and pivot_long(), spread(), map(), strsplit()

```
library(tidyverse)
#> -- Attaching packages ------------------- tidyverse 1.3.1 --
#> v ggplot2 3.3.5     v purrr   0.3.4
#> v tibble  3.1.5     v dplyr   1.0.7
#> v tidyr   1.1.4     v stringr 1.4.0
#> v readr   2.0.2     v forcats 0.5.1
#> -- Conflicts ---------------------- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
# Use read_csv/read_tsv insted of read.csv
# This will create tibble insted of data frame
booking= read_csv('data/bookings.csv')
#> Rows: 10000 Columns: 8
#> -- Column specification ------------------------------------
#> Delimiter: ","
#> chr (3): booker_id, checkin_day, status
#> dbl (4): property_id, room_nights, price_per_night, revi...
#> lgl (1): for_business
#> 
#> 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.
property=read_csv('data/properties.csv')
#> Rows: 4178 Columns: 5
#> -- Column specification ------------------------------------
#> Delimiter: ","
#> chr (3): destination, property_type, facilities
#> dbl (2): property_id, nr_rooms
```

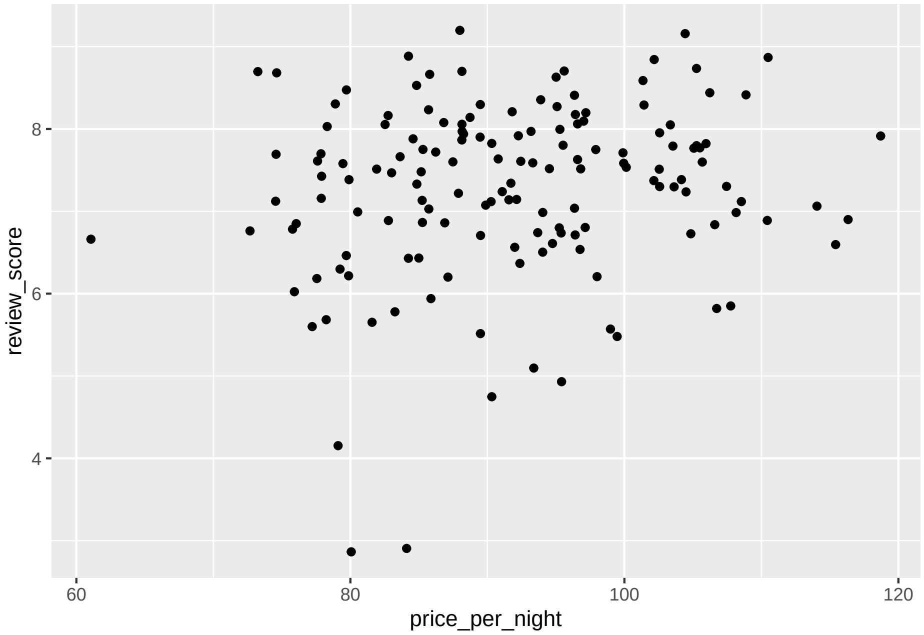
```
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.

booking
#> # A tibble: 10,000 x 8
#>    booker_id         property_id room_nights price_per_night
#>    <chr>                   <dbl>       <dbl>           <dbl>
#>  1 215934017ba98c09~        2668           4            91.5
#>  2 7f590fd6d318248a~        4656           5            107.
#>  3 10f0f138e8bb1015~        4563           6            87.0
#>  4 7b55021a4160dde6~        4088           7            92.4
#>  5 6694a79d158c7818~        2188           4            105.
#>  6 d0358740d5f15e85~        4171           2            110.
#>  7 944e568a0b511b91~        2907           4            116.
#>  8 95476c2ef6bb9e3c~        5141           4            111.
#>  9 df235631a4c281c0~        1696           1            106.
#> 10 ff610140227d40d2~        1901           7            82.3
#> # ... with 9,990 more rows, and 4 more variables:
#> #   checkin_day <chr>, for_business <lgl>, status <chr>,
#> #   review_score <dbl>
property
#> # A tibble: 4,178 x 5
#>    property_id destination property_type nr_rooms facilities
#>          <dbl> <chr>       <chr>            <dbl> <chr>
#>  1        2668 Brisbane    Hotel               32 airport s~
#>  2        4656 Brisbane    Hotel               39 on-site r~
#>  3        4563 Brisbane    Apartment            9 laundry
#>  4        4088 Brisbane    Apartment            9 kitchen,l~
#>  5        2188 Brisbane    Apartment            4 parking,k~
#>  6        4171 Brisbane    Apartment            5 kitchen,p~
#>  7        2907 Brisbane    Hotel               22 airport s~
#>  8        5141 Brisbane    Hotel               20 breakfast~
#>  9        1696 Brisbane    Apartment            5 free wifi~
#> 10        1901 Brisbane    Apartment           11 free wifi~
#> # ... with 4,168 more rows

#magrittr (pipe function, keyboard short cut: command+shift+m)
 # %>%
# select() and filter() functions help to extract data and study it

x=booking %>%
  select(review_score)
x
#> # A tibble: 10,000 x 1
#>    review_score
#>           <dbl>
#>  1        NA
#>  2        NA
#>  3         6.26
#>  4         5.95
#>  5         6.43
#>  6        NA
#>  7         7.60
#>  8        NA
#>  9         6.97
#> 10        NA
#> # ... with 9,990 more rows

y=booking %>%
  filter(status=='stayed'&!is.na(review_score))
y
#> # A tibble: 6,183 x 8
#>    booker_id         property_id room_nights price_per_night
#>    <chr>                   <dbl>       <dbl>           <dbl>
#>  1 10f0f138e8bb1015~        4563           6            87.0
#>  2 7b55021a4160dde6~        4088           7            92.4
#>  3 6694a79d158c7818~        2188           4           105.
#>  4 944e568a0b511b91~        2907           4           116.
#>  5 df235631a4c281c0~        1696           1           106.
#>  6 5a1442f4c7237ec5~        2307           9            84.2
#>  7 39804a2e3fb2e4c6~        2907           6           112.
#>  8 e150e559405ef29b~        2870           4           127.
#>  9 4e9c7c21dfcf2758~        1674           5           102.
#> 10 4a2b8eaf63613548~        2885           5            86.3
#> # ... with 6,173 more rows, and 4 more variables:
#> #   checkin_day <chr>, for_business <lgl>, status <chr>,
#> #   review_score <dbl>

cheap=booking %>%
  select(review_score,room_nights) %>%
  filter(booking$price_per_night<80)
cheap
#> # A tibble: 434 x 2
#>    review_score room_nights
#>           <dbl>       <dbl>
#>  1         8.90           5
#>  2         5.87           6
#>  3        NA              4
#>  4        NA              7
#>  5         6.02           4
#>  6         9.64           6
#>  7        NA              3
#>  8        NA              4
#>  9         6.23           5
#> 10        NA              2
#> # ... with 424 more rows


booking %>%
  filter(checkin_day=='wed') %>%
  select(property_id,status) %>%
  head(2)
#> # A tibble: 2 x 2
#>   property_id status
#>         <dbl> <chr>
#> 1        4563 stayed
#> 2        5141 cancelled

#ggplot2: for plotting
# ggplot(aes()+geom_histogram()/geom_point()....
booking %>%
ggplot(
   aes(review_score)
)+geom_histogram(bins = 100)
#> Warning: Removed 3817 rows containing non-finite values
#> (stat_bin).
```



```
booking %>%
  ggplot(
    aes(price_per_night,review_score)
  )+geom_point()
#> Warning: Removed 3817 rows containing missing values
#> (geom_point).
```



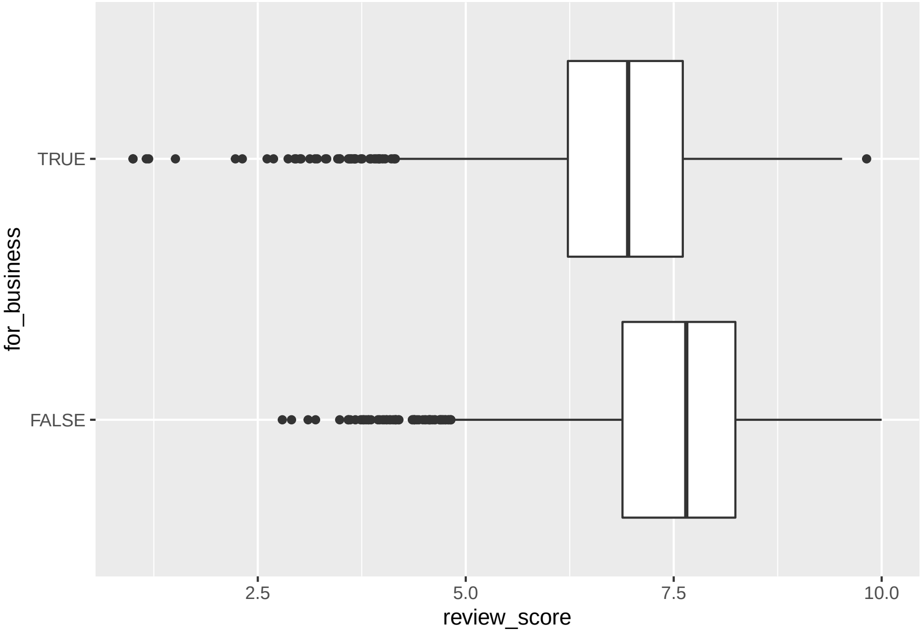
```
booking %>%
  filter(room_nights>7, status=='stayed') %>%
  select(price_per_night,review_score) %>%
  ggplot(
    aes(price_per_night,review_score)
  )+geom_point()
#> Warning: Removed 41 rows containing missing values
#> (geom_point).
```



```
#mutate

booking %>%
  mutate(centered_mean=price_per_night-mean(price_per_night)) %>%
  head(2)
#> # A tibble: 2 x 9
#>   booker_id           property_id room_nights price_per_night
#>   <chr>                     <dbl>       <dbl>           <dbl>
#> 1 215934017ba98c09f~         2668           4            91.5
#> 2 7f590fd6d318248a4~         4656           5           107.
#> # ... with 5 more variables: checkin_day <chr>,
#> #   for_business <lgl>, status <chr>, review_score <dbl>,
#> #   centered_mean <dbl>

# summarise: extracts the number of variables

booking %>%
  summarise(
    n()
    , n_miss=sum(is.na(review_score))
    ,mean_score=mean(review_score,na.rm=T))
#> # A tibble: 1 x 3
#>   `n()` n_miss mean_score
#>   <int>  <int>      <dbl>
#> 1 10000   3817       7.22
```

```
booking %>%
  summarise(
     n()
    , stayed_booking=sum(status=='stayed')
    , mean_total=mean(price_per_night*room_nights)

  )
#> # A tibble: 1 x 3
#>   `n()` stayed_booking mean_total
#>   <int>          <int>      <dbl>
#> 1 10000           7775       348.
#group by
booking %>%
  group_by(
    for_business
  ) %>%
  summarise(
  n=n()
, mean_review=mean(review_score,na.rm=T))
#> # A tibble: 2 x 3
#>   for_business     n mean_review
#>   <lgl>        <int>       <dbl>
#> 1 FALSE         6285        7.50
#> 2 TRUE          3715        6.85

mixed=booking %>%
  full_join(property) %>%
  count(destination,checkin_day) %>%
  pivot_wider(
    names_from = checkin_day,values_from = n
  )
#> Joining, by = "property_id"
mixed
#> # A tibble: 3 x 8
#>   destination   fri   mon   sat   sun   thu   tue   wed
#>   <chr>       <int> <int> <int> <int> <int> <int> <int>
#> 1 Amsterdam    1074   517   889   813   667   498   542
#> 2 Brisbane      162   133   114   153   162   148   128
#> 3 Tokyo         451   718   322   576   718   655   560

# make a long data form

long=mixed%>%
  pivot_longer(cols = 2:8, names_to = 'day',values_to = 'count')
long
```

```
#> # A tibble: 21 x 3
#>    destination day   count
#>    <chr>       <chr> <int>
#>  1 Amsterdam   fri    1074
#>  2 Amsterdam   mon     517
#>  3 Amsterdam   sat     889
#>  4 Amsterdam   sun     813
#>  5 Amsterdam   thu     667
#>  6 Amsterdam   tue     498
#>  7 Amsterdam   wed     542
#>  8 Brisbane    fri     162
#>  9 Brisbane    mon     133
#> 10 Brisbane    sat     114
#> # ... with 11 more rows

# make long data form

wide= long %>%
  pivot_wider(names_from = "day", values_from = "count")
wide
#> # A tibble: 3 x 8
#>   destination   fri   mon   sat   sun   thu   tue   wed
#>   <chr>       <int> <int> <int> <int> <int> <int> <int>
#> 1 Amsterdam    1074   517   889   813   667   498   542
#> 2 Brisbane      162   133   114   153   162   148   128
#> 3 Tokyo         451   718   322   576   718   655   560


# Boxplot with ggplot2
booking %>%
  ggplot(
    aes(
      review_score,for_business
    )
  )+geom_boxplot()
#> Warning: Removed 3817 rows containing non-finite values
#> (stat_boxplot).
```



```
# hash the property id
# we need to know map()function to do this. map(x,~.), where x = object and ~. is a fu
library(digest)

property %>%
  mutate(property_id=map_chr(property_id,digest))
#> # A tibble: 4,178 x 5
#>    property_id destination property_type nr_rooms facilities
#>    <chr>       <chr>       <chr>            <dbl> <chr>
#>  1 c5fe5a36c3~ Brisbane    Hotel               32 airport s~
#>  2 6abfc65c14~ Brisbane    Hotel               39 on-site r~
#>  3 8740143b90~ Brisbane    Apartment            9 laundry
#>  4 e30b95c1ec~ Brisbane    Apartment            9 kitchen,l~
#>  5 ab19240af8~ Brisbane    Apartment            4 parking,k~
#>  6 b2efd881c3~ Brisbane    Apartment            5 kitchen,p~
#>  7 d49c23b12c~ Brisbane    Hotel               22 airport s~
#>  8 1fd9f14595~ Brisbane    Hotel               20 breakfast~
#>  9 7319c32a43~ Brisbane    Apartment            5 free wifi~
#> 10 a38cc66d5f~ Brisbane    Apartment           11 free wifi~
#> # ... with 4,168 more rows

# list in data frame
# If we have a column with multiple strings, we can split it in to vectors using strsp
```

```
property %>%
  mutate(facilities=strsplit(facilities,","))
#> # A tibble: 4,178 x 5
#>    property_id destination property_type nr_rooms facilities
#>          <dbl> <chr>       <chr>            <dbl> <list>
#>  1        2668 Brisbane    Hotel               32 <chr [6]>
#>  2        4656 Brisbane    Hotel               39 <chr [7]>
#>  3        4563 Brisbane    Apartment            9 <chr [1]>
#>  4        4088 Brisbane    Apartment            9 <chr [3]>
#>  5        2188 Brisbane    Apartment            4 <chr [5]>
#>  6        4171 Brisbane    Apartment            5 <chr [6]>
#>  7        2907 Brisbane    Hotel               22 <chr [8]>
#>  8        5141 Brisbane    Hotel               20 <chr [8]>
#>  9        1696 Brisbane    Apartment            5 <chr [6]>
#> 10        1901 Brisbane    Apartment           11 <chr [8]>
#> # ... with 4,168 more rows
property$facilities[1]
#> [1] "airport shuttle,free wifi,garden,breakfast,pool,on-site restaurant"

# add a column with the number of facilities

property %>%
  mutate(facilities=strsplit(facilities,",")) %>%
  mutate(n_facility=map_int(facilities,length))
#> # A tibble: 4,178 x 6
#>    property_id destination property_type nr_rooms facilities
#>          <dbl> <chr>       <chr>            <dbl> <list>
#>  1        2668 Brisbane    Hotel               32 <chr [6]>
#>  2        4656 Brisbane    Hotel               39 <chr [7]>
#>  3        4563 Brisbane    Apartment            9 <chr [1]>
#>  4        4088 Brisbane    Apartment            9 <chr [3]>
#>  5        2188 Brisbane    Apartment            4 <chr [5]>
#>  6        4171 Brisbane    Apartment            5 <chr [6]>
#>  7        2907 Brisbane    Hotel               22 <chr [8]>
#>  8        5141 Brisbane    Hotel               20 <chr [8]>
#>  9        1696 Brisbane    Apartment            5 <chr [6]>
#> 10        1901 Brisbane    Apartment           11 <chr [8]>
#> # ... with 4,168 more rows, and 1 more variable:
#> #   n_facility <int>
```