Chapter 6 Descriptive Analysis

Descriptive analysis summarizes and helps you understand your data **numerically**. In marketing analytics, it is often your first "sanity check" before modeling or visualization.

In this chapter you will learn how to:

- inspect a dataset and its variables,
- create frequency tables and crosstabs,
- compute measures of central tendency and dispersion,
- and compute and interpret correlations.

For this chapter, we'll be using the `airlinesat_small` dataset from the `MKT4320BGSU`.

```
data(airlinesat_small)
```

6.1 Descriptive Analysis

A quick descriptive workflow often looks like:

1. **Confirm the data structure** (rows/columns, variable types)
2. **Look for missing values and odd ranges**
3. **Summarize key variables** (categorical → counts; numeric → mean/SD, etc.)
4. **Compare segments** (e.g., by treatment, gender, region, etc.)

6.1.1 Inspecting the dataset

The following functions are useful for inspecting the dataset.

- `str()` shows each variables' type, displays factor levels if present, and gives a compact preview of values for each variable
- `dim()` returns the number of rows and columsn to quickly tell you sample size and number of variables
- `names()` provides all column names, which helps with selecting variables for writing code or formulas
- `head()` gives the first six rows of the dataset to allow for visual inspection of raw values

```r
# Take a look at the data frame
str(airlinesat_small)
```

```
'data.frame':	1065 obs. of  13 variables:
 $ age           : num  30 55 56 43 44 40 39 41 33 51 ...
 $ country       : Factor w/ 5 levels "at","ch","de",..: 2 2 2 4 2 2 2 2 2 3 ...
 $ flight_class  : Factor w/ 3 levels "Business","Economy",..: 2 1 2 2 1 3 2 1 2 1 ...
 $ flight_latest : Factor w/ 6 levels "within the last 12 months",..: 4 3 5 3 6 5 6 3 3 4 .
 $ flight_purpose: Factor w/ 2 levels "Business","Leisure": 2 1 1 2 1 2 1 1 2 1 ...
 $ flight_type   : Factor w/ 2 levels "Domestic","International": 1 2 1 1 2 2 1 2 1 2 ...
 $ gender        : Factor w/ 2 levels "female","male": 2 2 1 1 1 2 2 2 2 2 ...
 $ language      : Factor w/ 3 levels "English","French",..: 2 1 1 2 1 3 2 2 2 3 ...
 $ nflights      : num  2 6 8 7 25 16 35 9 3 4 ...
 $ status        : Factor w/ 3 levels "Blue","Gold",..: 1 2 1 1 2 2 1 2 1 2 ...
 $ nps           : num  6 10 8 8 6 7 8 7 8 8 ...
 $ overall_sat   : num  2 6 2 4 2 4 4 4 4 3 ...
 $ reputation    : num  3 6 4 6 5 3 3 4 2 4 ...
```

```r
# Dimensions (rows, columns)
dim(airlinesat_small)
```

```
[1] 1065   13
```

```r
# Variable names
names(airlinesat_small)
```

```
 [1] "age"              "country"          "flight_class"     "flight_latest"    "flight_purpose"   "
 [7] "gender"           "language"         "nflights"         "status"           "nps"              "
[13] "reputation"
```

```r
# First few rows
head(airlinesat_small)
```

```
  age country flight_class            flight_latest flight_purpose   flight_type gender lar
1  30      ch      Economy within the last 6 months        Leisure      Domestic   male   F
2  55      ch     Business within the last 3 months       Business International   male  Er
3  56      ch      Economy    within the last month       Business      Domestic female  Er
4  43      fr      Economy within the last 3 months        Leisure      Domestic female   F
5  44      ch     Business      within the last week       Business International female  Er
6  40      ch        First    within the last month        Leisure International   male   G
  overall_sat reputation
1           2          3
2           6          6
3           2          4
4           4          6
5           2          5
6           4          3
```

6.1.2 Missing values (quick checks)

It is also important to check for missing values, because they can have an adverse effect on some analyses. The `is.na()` function is often used for this, which returns a `TRUE` if a value is `NA`. Ultimately, this will tell which values in the data are missing to help decide how to handle them.

```r
# Missing values by variable
colSums(is.na(airlinesat_small))
```

```
           age        country   flight_class  flight_latest flight_purpose    flight_type
             0              0              0              0              0              0
      nflights         status            nps    overall_sat     reputation
             0              0              0              0              0
```

```r
# Total missing values in the dataset
sum(is.na(airlinesat_small))
```

```
[1] 0
```

6.2 Frequency Tables

Frequency tables summarize **categorical** variables (and sometimes binned numeric variables).

6.2.1 One-way frequency table

To create a frequency table, use the `table()` function in R. Counts are often converted into proportions or rates to make results easier to interpret and compare.

In R, the `proportions()` function is used to convert frequency tables into proportions. Multiply the table by 100 to get percentages.

```r
# Frequency table for a categorical variable
table(airlinesat_small$gender)
```

```
female   male 
   280    785 
```

```r
# Add proportions (shares)
proportions(table(airlinesat_small$gender))
```

```
   female      male 
0.2629108 0.7370892 
```

```r
# Add percentages
100 * proportions(table(airlinesat_small$gender))
```

```
  female     male 
26.29108 73.70892 
```

6.3 Crosstabs

A crosstab is a frequency table for **two categorical variables**. Crosstabs are used frequently in marketing to compare segments (e.g., purchase by gender).

6.3.1 Base R

Base R does not do a great job of easily creating crosstabs and testing for independence of the two variables. Instead, a multistep process is required:

- Create the two-way frequency table using the `table(rowvar, colvar)` function and assign it to a separate object

- Display the two-way freq table by just using the table name

```r
# Counts
ct <- table(airlinesat_small$flight_class, airlinesat_small$gender)
ct
```

```
          female male
  Business     39  146
  Economy     239  626
  First         2   13
```

- Use the function `proportions(tablename, margin)` on the newly created object to get column, row, or total percentages
    - `proportions(tablename)` gives total percentages
    - `proportions(tablename, 1)` gives row percentages
    - `proportions(tablename, 2)` gives column percentages

```r
proportions(ct)             # Total proportions: what pprportion of all respondents
```

```
                female        male
  Business 0.036619718 0.137089202
  Economy  0.224413146 0.587793427
  First    0.001877934 0.012206573
```

```r
proportions(ct, margin=1)   # Row proportions: how are genders distributed within 
```

```
             female      male
  Business 0.2108108 0.7891892
  Economy  0.2763006 0.7236994
  First    0.1333333 0.8666667
```

```
proportions(ct, margin=2)   # Column proportions: how is th outcome distributed wi
```

```
               female        male
  Business 0.139285714 0.185987261
  Economy  0.853571429 0.797452229
  First    0.007142857 0.016560510
```

- Use the function `chisq.test(tablename)` on the newly created object to run the test of independence

```
chisq.test(ct)
```

```
Warning in chisq.test(ct): Chi-squared approximation may be incorrect
```

```

    Pearson's Chi-squared test

data:  ct
X-squared = 4.6912, df = 2, p-value = 0.09579
```

6.3.2 Using package *sjPlot*

The **sjPlot** package can print crosstabs with nicer formatting. Use the function `tab_xtab(var.row=, var.col=, show.col.prc=TRUE)` to get a standard crosstab with column percentages and the chi-square test of independence.

```r
tab_xtab(airlinesat_small$flight_class,
         airlinesat_small$gender,
         show.col.prc = TRUE)
```

flight_class	*gender*		***Total***
	female	male	
Business	39 13.9 %	146 18.6 %	185 17.4 %
Economy	239 85.4 %	626 79.7 %	865 81.2 %
First	2 0.7 %	13 1.7 %	15 1.4 %
Total	280 100 %	785 100 %	1065 100 %

χ^2=4.691 · df=2 · Cramer's V=0.066 · Fisher's p=0.095

6.4 Measures of Central Tendency and Dispersion

For **numeric** variables, the most common summaries are:

- Central tendency: mean, median, mode
- Dispersion: variance, standard deviation, IQR, range

6.4.1 Base R

Any individual summary statistic can be easily calculated using Base R with functions such as:

- `mean(var)` for mean
- `sd(var)` for standard deviation

- `quantile(var, .percentile)` for percentiles (e.g., '.50' would be median)

For summary statistics except for standard deviation, the `summary(object)` function can be used, where object can be a single variable or an entire data frame

```
# Base R summary (min, quartiles, median, mean, max)
summary(airlinesat_small$age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  19.00   42.00   50.00   50.42   58.00  101.00 
```

```
summary(airlinesat_small)
```

```
      age          country    flight_class                    flight_latest  flight_purpose
 Min.   : 19.00   at:108   Business:185   within the last 12 months:139   Business:525    D
 1st Qu.: 42.00   ch: 66   Economy :865   within the last 2 days   : 57   Leisure :540    I
 Median : 50.00   de:695   First   : 15   within the last 3 months :296                  
 Mean   : 50.42   fr:  1                  within the last 6 months :187                  
 3rd Qu.: 58.00   us:195                  within the last month    :253                  
 Max.   :101.00                           within the last week     :133                  
    gender        language      nflights          status          nps          overall_sat  
 female:280   English:233   Min.   :  1.00   Blue  :677   Min.   : 1.00   Min.   :1.00   Mi
 male  :785   French : 10   1st Qu.:  4.00   Gold  :143   1st Qu.: 6.00   1st Qu.:3.00   1s
              German :822   Median :  8.00   Silver:245   Median : 8.00   Median :4.00   Me
                            Mean   : 13.42                Mean   : 7.52   Mean   :3.74   Me
                            3rd Qu.: 16.00                3rd Qu.: 9.00   3rd Qu.:5.00   3r
                            Max.   :457.00                Max.   :11.00   Max.   :7.00   Ma
```

```
# Mean and SD (remove missing values if present)
mean(airlinesat_small$age, na.rm = TRUE)
```

```
[1] 50.41972
```

```
sd(airlinesat_small$age, na.rm = TRUE)
```

```
[1] 12.27464
```

If you want a single, custom summary:

```
x <- airlinesat_small$age

c(n = sum(!is.na(x)),
  mean = mean(x, na.rm = TRUE),
  sd = sd(x, na.rm = TRUE),
  median = median(x, na.rm = TRUE),
  iqr = IQR(x, na.rm = TRUE),
  min = min(x, na.rm = TRUE),
  max = max(x, na.rm = TRUE))
```

```
         n       mean         sd     median        iqr        min        max
1065.00000   50.41972   12.27464   50.00000   16.00000   19.00000  101.00000
```

6.4.2 Using package *dplyr*

With **dplyr**, you can summarize many variables and/or do summaries by groups.

Overall summaries:

```
airlinesat_small %>%
  summarise(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            sd_age = sd(age, na.rm = TRUE),
            median_age = median(age, na.rm = TRUE),
            iqr_age = IQR(age, na.rm = TRUE),
            mean_nflights = mean(nflights, na.rm = TRUE),
            sd_nflights = sd(nflights, na.rm = TRUE))
```

```
     n mean_age   sd_age median_age iqr_age mean_nflights sd_nflights
1 1065 50.41972 12.27464         50      16      13.41878    20.22647
```

Group summaries (example: by `buy`):

```
airlinesat_small %>%
  group_by(status) %>%
  summarise(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            sd_age = sd(age, na.rm = TRUE),
            mean_nflights = mean(nflights, na.rm = TRUE),
            sd_nflights = sd(nflights, na.rm = TRUE))
```

```
# A tibble: 3 × 6
  status     n mean_age sd_age mean_nflights sd_nflights
  <fct>  <int>    <dbl>  <dbl>         <dbl>       <dbl>
1 Blue     677     50.6   13.3          8.23        19.2
2 Gold     143     53     10.0         24.6         20.9
3 Silver   245     48.3   10.0         21.2         17.2
```

6.4.3 Using package *vtable*

The **vtable** package can generate clean, compact descriptive tables with the `sumtable(data, vars=c(""))` function. For factor variables, it will provide the counts and percents of each level.

```
sumtable(airlinesat_small, c("age", "nflights", "status"),
         add.median=TRUE,    # 'add.median=TRUE' includes a 50th percentile column
         title=NA)
```

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 50	Pctl. 75	Max
age	1065	50	12	19	42	50	58	101
nflights	1065	13	20	1	4	8	16	457
status	1065							
… Blue	677	64%						
… Gold	143	13%						
… Silver	245	23%						

The `sumtable()` function can also provide summary statistics by a grouping variable.

```
sumtable(airlinesat_small, c("age", "nflights", "status"),
         add.median=TRUE,
         group="gender",
         title=NA)
```

gender	female				male			
Variable	**N**	**Mean**	**SD**	**Median**	**N**	**Mean**	**SD**	**Median**
age	280	51	13	51	785	50	12	50
nflights	280	9.3	12	5	785	15	22	9
status	280				785			
… Blue	225	80%			452	58%		
… Gold	16	6%			127	16%		
… Silver	39	14%			206	26%		

6.5 Correlation

Correlation measures the strength of a **linear relationship** between two numeric variables. The most common is Pearson correlation (the default).

6.5.1 Base R

Base R can easily provide a correlation matrix of many variables, and it can provide a correlation test between two variables at a time, but it cannot produce a correlation matrix with p-values.

To get a correlation matrix, use the `cor()` function with a dataframe of the variables desired or using indexing on variable names. By default, it uses all observations, which can create `NA` values if any missing values exists. Therefore, the preference is to add the option `use = "pairwise.complete.obs"` to only calculate the correlation on non-missing values for each pair of variables.

```
mycorr_df <- airlinesat_small %>%
  select(age, nflights, nps, overall_sat, reputation)
cor(mycorr_df, use = "pairwise.complete.obs")
```

```
                     age    nflights         nps overall_sat  reputation
age          1.00000000 -0.11576301  0.09867319  0.05903446  0.06082991
nflights    -0.11576301  1.00000000 -0.08949782 -0.05366975 -0.06364290
nps          0.09867319 -0.08949782  1.00000000  0.29961310  0.50712230
overall_sat  0.05903446 -0.05366975  0.29961310  1.00000000  0.17748688
reputation   0.06082991 -0.06364290  0.50712230  0.17748688  1.00000000
```

```r
cor(airlinesat_small[,c("age", "nflights", "nps", "overall_sat", "reputation")],
    use = "pairwise.complete.obs")
```

```
                     age    nflights         nps overall_sat  reputation
age          1.00000000 -0.11576301  0.09867319  0.05903446  0.06082991
nflights    -0.11576301  1.00000000 -0.08949782 -0.05366975 -0.06364290
nps          0.09867319 -0.08949782  1.00000000  0.29961310  0.50712230
overall_sat  0.05903446 -0.05366975  0.29961310  1.00000000  0.17748688
reputation   0.06082991 -0.06364290  0.50712230  0.17748688  1.00000000
```

To get the correlation test for any one pair of variables, use the `cor.test(var1, var2)` function. By default, it includes only observations that are non-missing in both variables.

```r
cor.test(airlinesat_small$age, airlinesat_small$nflights)
```

```

    Pearson's product-moment correlation

data:  airlinesat_small$age and airlinesat_small$nflights
t = -3.7998, df = 1063, p-value = 0.000153
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.17461941 -0.05608231
sample estimates:
      cor
-0.115763
```

6.5.2 Using package *Hmisc*

The **Hmisc** package is useful for correlation matrices with p-values using the `rcorr()` function. This function expects a matrix (or data frame coerced to matrix).

```
rcorr(as.matrix(mycorr_df))
```

```
              age nflights   nps overall_sat reputation
age          1.00    -0.12  0.10        0.06       0.06
nflights    -0.12     1.00 -0.09       -0.05      -0.06
nps          0.10    -0.09  1.00        0.30       0.51
overall_sat  0.06    -0.05  0.30        1.00       0.18
reputation   0.06    -0.06  0.51        0.18       1.00

n= 1065


P
            age    nflights nps    overall_sat reputation
age                0.0002   0.0013 0.0541      0.0472
nflights    0.0002          0.0035 0.0800      0.0378
nps         0.0013 0.0035          0.0000      0.0000
overall_sat 0.0541 0.0800   0.0000             0.0000
reputation  0.0472 0.0378   0.0000 0.0000
```

6.5.3 Using package *sjPlot*

The **sjPlot** package can produce formatted correlation tables for reports using the `tab_corr()` function. By default, it uses listwise (or casewise) deletion, which removes an entire case (or row) if any value is `NA`, which may result in major data loss. Use the option `na.deletion = "pairwise")` to prevent this.

```
tab_corr(mycorr_df,
         triangle = "lower",
         show.p = TRUE,
         na.deletion = "pairwise")
```

	age	*nflights*	*nps*	*overall_sat*	*reputation*
age					
nflights	-0.116***				
nps	0.099**	-0.089**			
overall_sat	0.059	-0.054	0.300***		
reputation	0.061*	-0.064*	0.507***	0.177***	

Computed correlation used pearson-method with pairwise-deletion.

6.6 Why descriptive analysis matters

Descriptive statistics help you to:

- detect unusual values,
- understand typical behavior,
- compare groups,
- and check whether results are plausible.

They also guide decisions about which models or visualizations are appropriate.

6.7 What's next

In the next chapter, we move from **numbers** to **visuals**. You will learn how to create effective data visualizations using a small amount of Base R, but primarily `ggplot2`. Visualizations allow you to see patterns, distributions, and relationships that are often difficult to detect from tables alone.

Together, descriptive statistics and visualization form the foundation of **exploratory data analysis**, which prepares you for modeling and inference later in the course.