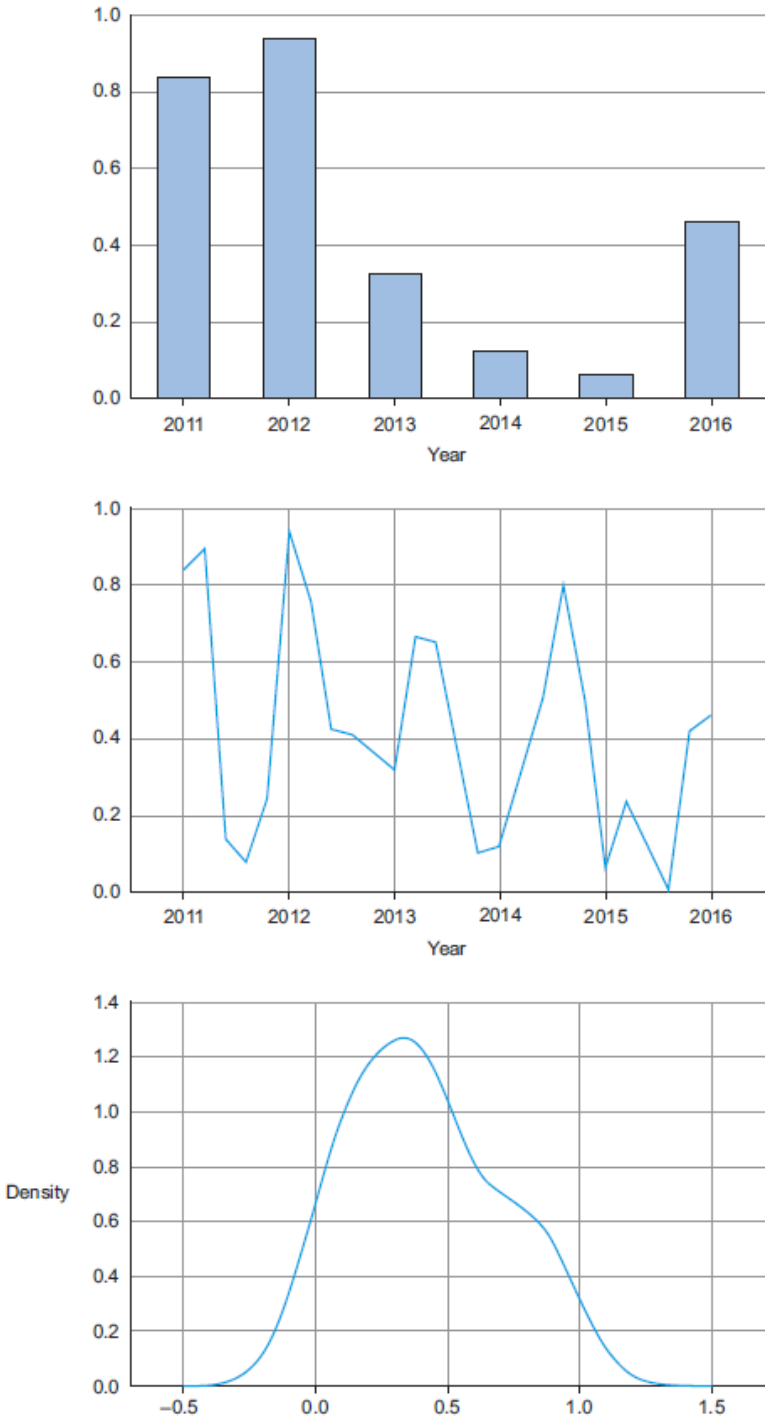
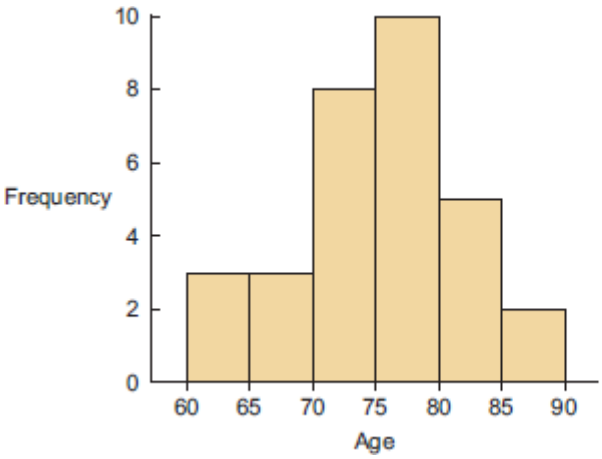
EXPLORATORY DATA ANALYSIS

During exploratory data analysis you take a deep dive into the data (see figure below). Information becomes much easier to grasp when shown in a picture, therefore you mainly use graphical techniques to gain an understanding of your data and the interactions between variables. The goal isn't to cleanse the data, but it's common that you'll still discover anomalies you missed before, forcing you to take a step back and fix them.



From top to bottom, a bar chart, a line plot, and a distribution are some of the graphs used in exploratory analysis.

- The visualization techniques you use in this phase range from simple line graphs or histograms, as shown in below figure, to more complex diagrams such as Sankey and network graphs.
- Sometimes it's useful to compose a composite graph from simple graphs to get even more insight into the data Other times the graphs can be animated or made interactive to make it easier and, let's admit it, way more fun.



Example histogram: the number of people in the age-groups of 5-year intervals

The techniques we described in this phase are mainly visual, but in practice they're certainly not limited to visualization techniques. Tabulation, clustering, and other modeling techniques can also be a part of exploratory analysis. Even building simple models can be a part of this step.

BUILD THE MODELS

- With clean data in place and a good understanding of the content, you're ready to build models with the goal of making better predictions, classifying objects, or gaining an understanding of the system that you're modeling.
- This phase is much more focused than the exploratory analysis step, because you know what you're looking for and what you want the outcome to be.

Building a model is an iterative process. The way you build your model depends on whether you go with classic statistics or the somewhat more recent machine learning school, and the type of technique you want to use. Either way, most models consist of the following main steps:

- Selection of a modeling technique and variables to enter in the model
- Execution of the model
- Diagnosis and model comparison

Model and variable selection

You'll need to select the variables you want to include in your model and a modeling technique. You'll need to consider model performance and whether your project meets all the requirements to use your model, as well as other factors:

- Must the model be moved to a production environment and, if so, would it be easy to implement?
- How difficult is the maintenance on the model: how long will it remain relevant if left untouched?
- Does the model need to be easy to explain?

Model execution

- Once you've chosen a model, you'll need to implement it in code.
- Most programming languages, such as Python, already have libraries such as StatsModels or Scikit-learn. These packages use several of the most popular techniques.
- Coding a model is a nontrivial task in most cases, so having these libraries available can speed up the process. As you can see in the following code, it's fairly easy to use linear regression with StatsModels or Scikit-learn
- Doing this yourself would require much more effort even for the simple techniques. The following listing shows the execution of a linear prediction model.

```
import statsmodels.api as sm
import numpy as np
predictors = np.random.random(1000).reshape(500,2)
target = predictors.dot(np.array([0.4, 0.6])) + np.random.random(500)
lmRegModel = sm.OLS(target,predictors)
result = lmRegModel.fit()
result.summary()
```

Imports required Python modules.

Creates random data for predictors (x-values) and semi-random data for the target (y-values) of the model. We use predictors as input to create the target so we infer a correlation here.

Fits linear regression on data.

Shows model fit statistics.

Model diagnostics and model comparison

- You'll be building multiple models from which you then choose the best one based on multiple criteria. Working with a holdout sample helps you pick the best-performing model.
- A holdout sample is a part of the data you leave out of the model building so it can be used to evaluate the model afterward.
- The principle here is simple: the model should work on unseen data. You use only a fraction of your data to estimate the model and the other part, the holdout sample, is kept out of the equation.
- The model is then unleashed on the unseen data and error measures are calculated to evaluate it.
- Multiple error measures are available, and in figure we show the general idea on comparing models. The error measure used in the example is the mean square error.

Formula for mean square error

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

Mean square error is a simple measure: check for every prediction how far it was from the truth, square this error, and add up the error of every prediction.

	n	Size	Price	Predicted model 1	Predicted model 2	Error model 1	Error model 2
	1	10	3				
	2	15	5				
	3	18	6				
	4	14	5				
	…	…					
80% train	800	9	3				
	801	12	4	12	10	0	2
	802	13	4	12	10	1	3
	…						
	999	21	7	21	10	0	11
20% test	1000	10	4	12	10	–2	0
				Total		5861	110225

A holdout sample helps you compare models and ensures that you can generalize results to data that the model has not yet seen.

Above figure compares the performance of two models to predict the order size from the price. The first model is *size = 3 * price* and the second model is *size = 10*.

- To estimate the models, we use 800 randomly chosen observations out of 1,000 (or 80%), without showing the other 20% of data to the model.
- Once the model is trained, we predict the values for the other 20% of the variables based on those for which we already know the true value, and calculate the model error with an error measure.
- Then we choose the model with the lowest error. In this example we chose model 1 because it has the lowest total error.

Presenting findings and building applications

- Sometimes people get so excited about your work that you'll need to repeat it over and over again because they value the predictions of your models or the insights that you produced.
- This doesn't always mean that you have to redo all of your analysis all the time. Sometimes it's sufficient that you implement only the model scoring; other times you might build an application that automatically updates reports, Excel spreadsheets, or PowerPoint presentations. The last stage of the data science process is where your *soft skills* will be most useful, and yes, they're extremely important.