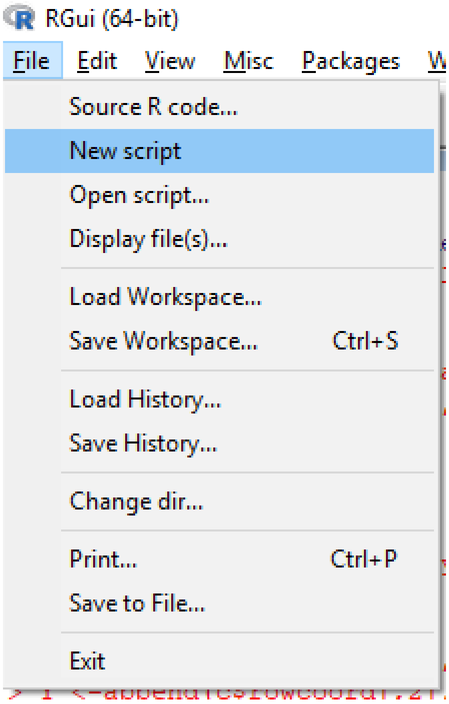
Lab 12. Advanced Analytics with Tableau

This chapter will cover the following topics:

- Running segmentation analysis
- Discovering the latent structure of the dataset
- Extracting the structure beneath discrete variables
- Data mining with tree-based models
- Identifying anomalies in data

Technical requirements

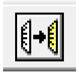
To follow the recipes from this lab, you'll need to have Tableau 2019.1 installed. You will also need to have R installed, as well as the `Rserve` library. To install the `Rserve` library, open a new R script by navigating to **`File`** | **`New script`**:



Run the following code:

```
install.packages('Rserve',repos='http://cran.us.r-project.org')
```

You can run the code by selecting it and clicking the **`Run`** button



or by pressing [*Ctrl]* + *[R*] on your keyboard.

Introduction

In this lab, you'll learn how to utilize joined capabilities of Tableau and R in order to perform some more advanced analytical tasks. This will enable us to dive below the surface and to discover hidden underlying patterns in our data.

We'll learn how to identify and interpret latent dimensions of our dataset, to group similar cases, and to detect and interpret unusual cases and anomalies in data.

> *["Hiding within those mounds of data is knowledge that could change the life of a patient, or change the world."*
>
> *--Atul Butte]*

Running segmentation analysis

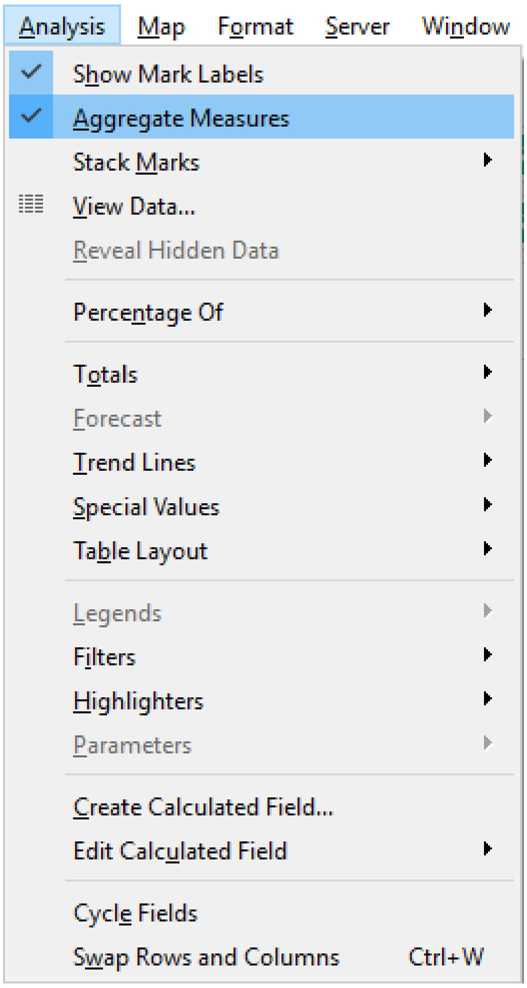
Cluster analysis is one of the most popular data analysis techniques. It allows us to find patterns or segments in our data, which we can then interpret in order to gain a more in-depth understanding of its underlying structure. Tableau has a built-in clustering functionality, which means we'll not be using R in this recipe---we'll perform the entire analysis through Tableau only.

Getting ready

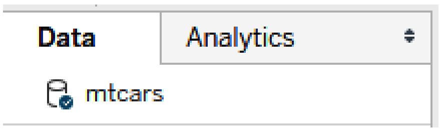
In this recipe, we'll be using the `mtcars.csv` dataset. It contains characteristics of various car models, such as the number of horsepowers, number of cylinders, miles per gallon, and so on. Before we dive into the recipe, make sure you have the dataset saved to your device and open Tableau and connect to it.

How to do it...

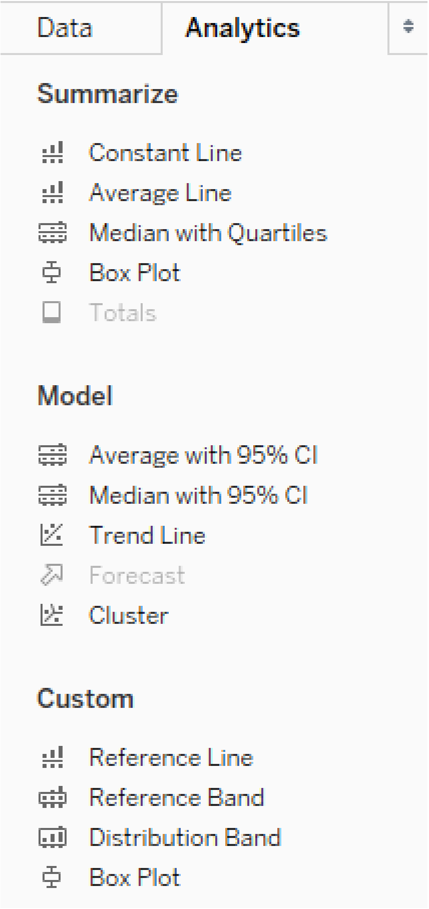
1. After connecting to the `mtcars.csv` dataset, open a new blank worksheet.
2. Drag and drop **`Qsec`** from **`Measures`** into the **`Rows`** shelf.
3. Drag and drop **`Mpg`** from **`Measures`** into the **`Columns`** shelf.
4. In the main menu toolbar, navigate to **`Analysis`** and, in the drop-down menu, deselect **`Aggregate Measures`**:



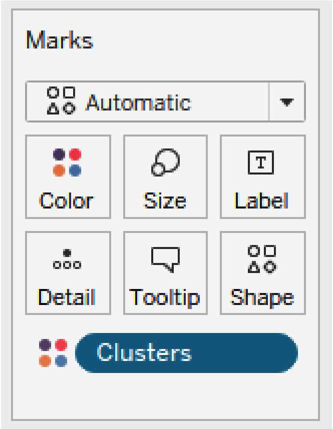
5. We've now produced a scatter plot of the car models. Let's add the clusters! In the `Data` pane, navigate to `Analytics` :



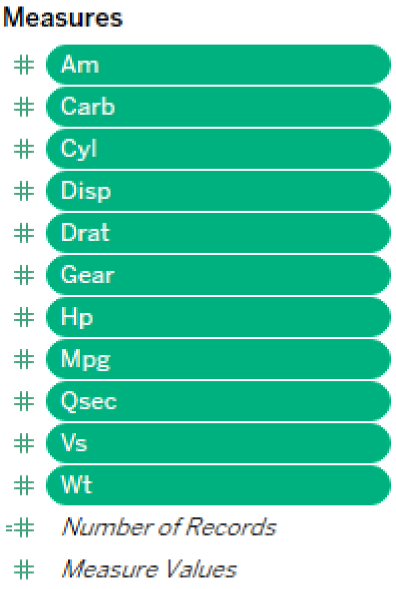
6. Drag and drop `Cluster` from `Model` into the view we have created:



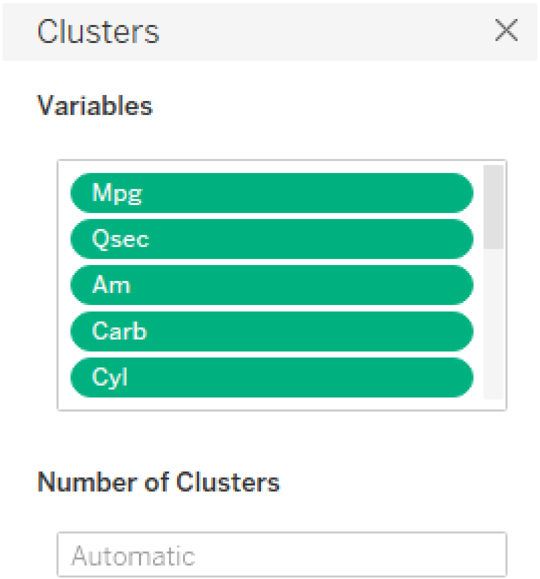
7. A dimension, `Clusters`, has been created and placed onto `Color` in the `Marks` card:



8. In the `Clusters` window that opened, we can see two measures that we previously placed into the scatter plot. Navigate back to the `Data` pane and, under `Measures`, select the remaining measures (`Am` through `Wt`), by clicking on them while holding the [*Ctrl*] key on your keyboard:



9. Drag and drop all of the selected measures into the `Clusters` window, under `Variables` :



Number of Clusters

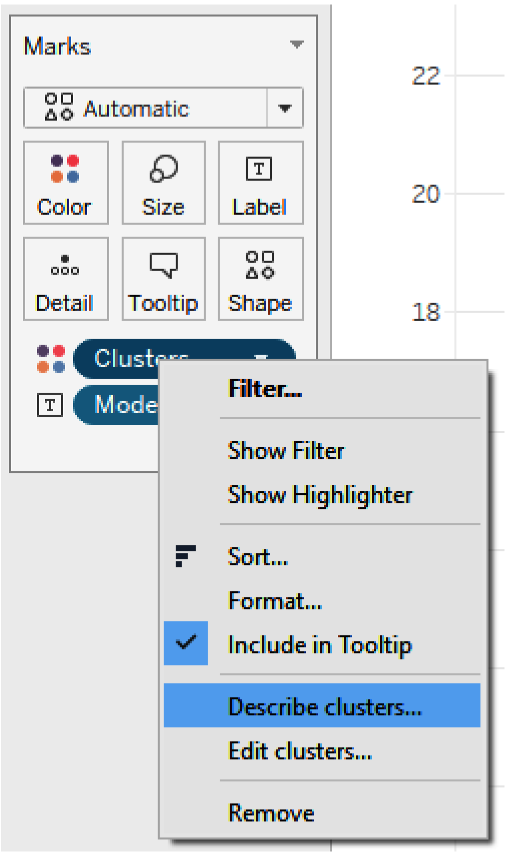
Automatic

10. Click on `X` to close the `Clusters` window.
11. Drag and drop `Model` from `Dimensions` onto `Label` in the `Marks` card:

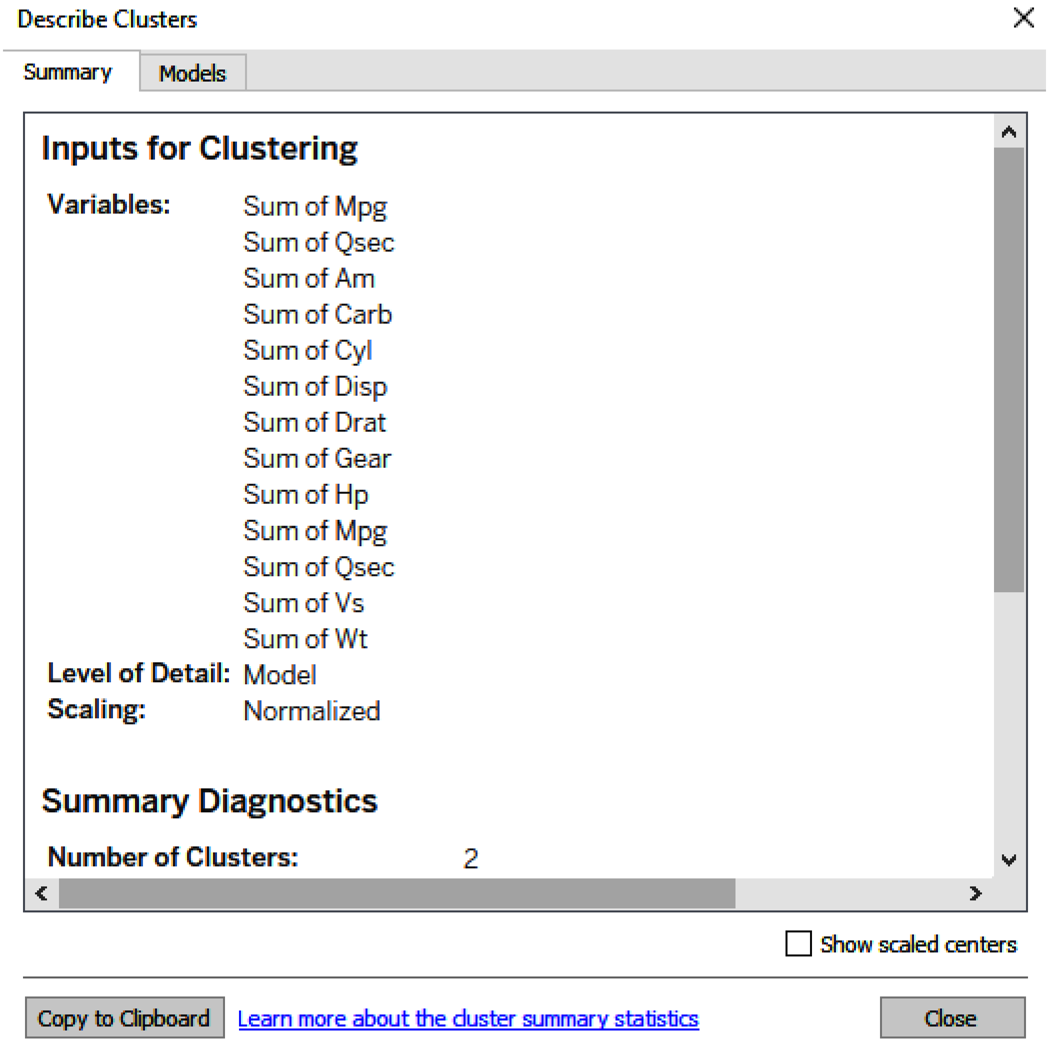
How it works...

Cluster analysis works by grouping cases (in our example, car models) into clusters based on a complex underlying pattern of scores on multiple measures. When setting up the cluster analysis, we entered all of the measures we wanted the cluster based on into the `Clusters` window, and Tableau automatically detected the optimal number of clusters. There are many clustering algorithms---the one implemented in Tableau is called k-means.

But this is just the beginning of the analysis. The next step is to interpret the clusters. We can use the `Describe Clusters` function to aid the interpretation. We'll access it by right-clicking on the `Clusters` pill in the `Marks` card and selecting `Describe Clusters` from the drop-down menu, as shown in the following screenshot:



This will launch the `Describe Clusters` window. Under `Summary`, we can see what variables entered into the cluster analysis, as well as `Summary Diagnostics`, which includes the description of the model itself, as well as that of the individual clusters. The `Summary` tab is shown in the following screenshot:



Inputs for Clustering

Variables: Sum of Mpg
Sum of Qsec
Sum of Am
Sum of Carb
Sum of Cyl
Sum of Disp
Sum of Drat
Sum of Gear
Sum of Hp
Sum of Mpg
Sum of Qsec
Sum of Vs
Sum of Wt

Level of Detail: Model

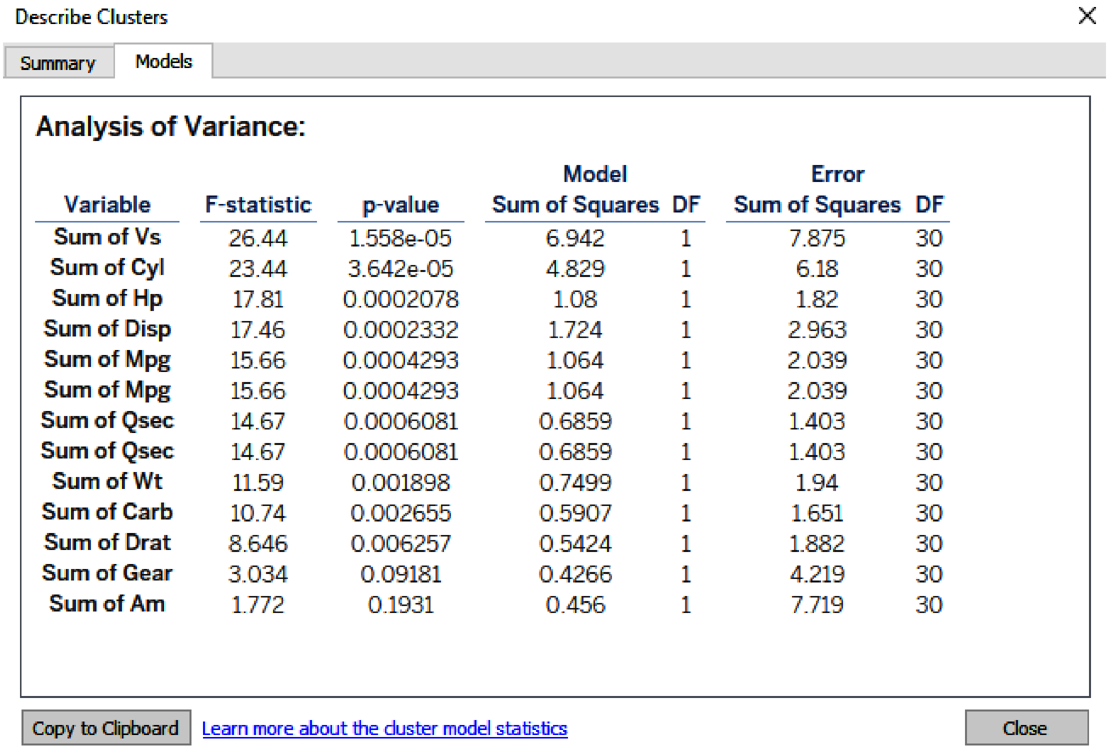
Scaling: Normalized

Summary Diagnostics

Number of Clusters: 2

□ Show scaled centers

If we navigate to the `Models` tab, we'll see the results of the `Analysis of Variance,` showing us which variables actually differentiate the clusters. The `Models` tab is shown in the following screenshot:



Analysis of Variance:

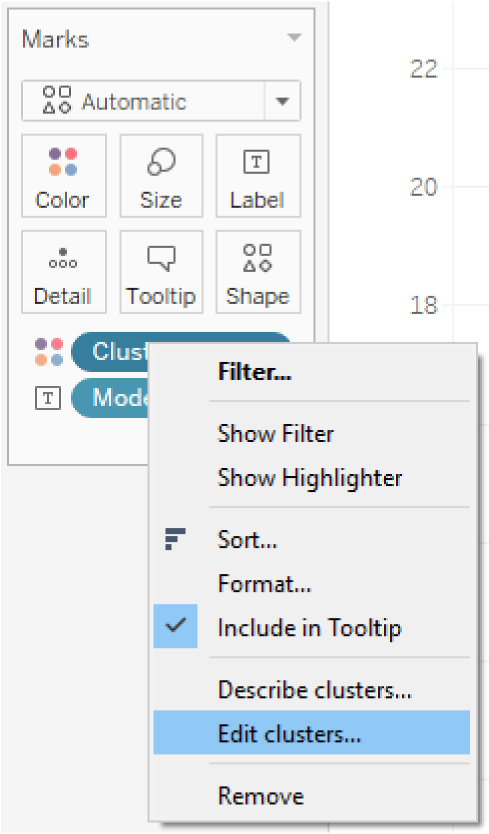
Variable	F-statistic	p-value	Model Sum of Squares	DF	Error Sum of Squares	DF
Sum of Vs	26.44	1.558e-05	6.942	1	7.875	30
Sum of Cyl	23.44	3.642e-05	4.829	1	6.18	30
Sum of Hp	17.81	0.0002078	1.08	1	1.82	30
Sum of Disp	17.46	0.0002332	1.724	1	2.963	30
Sum of Mpg	15.66	0.0004293	1.064	1	2.039	30
Sum of Mpg	15.66	0.0004293	1.064	1	2.039	30
Sum of Qsec	14.67	0.0006081	0.6859	1	1.403	30
Sum of Qsec	14.67	0.0006081	0.6859	1	1.403	30
Sum of Wt	11.59	0.001898	0.7499	1	1.94	30
Sum of Carb	10.74	0.002655	0.5907	1	1.651	30
Sum of Drat	8.646	0.006257	0.5424	1	1.882	30
Sum of Gear	3.034	0.09181	0.4266	1	4.219	30
Sum of Am	1.772	0.1931	0.456	1	7.719	30

In in the preceding screenshot, we can see that all of the variables, except for `Sum of Vs` and `Sum of Gear`, are significant by looking at their associated values in the `p-value` column. A `p-value` that is less than or equal to 0.05 is usually considered significant. To gain more insight into how these variables differentiate the clusters, we can compare the values of different variables for each of the clusters in the `Summary` tab, under `Summary Diagnostics`. We can also create new scatterplots by placing various measures into the `Rows` and `Columns` shelves, so we can gain a better sense of how the clusters are distributed.

There's more...

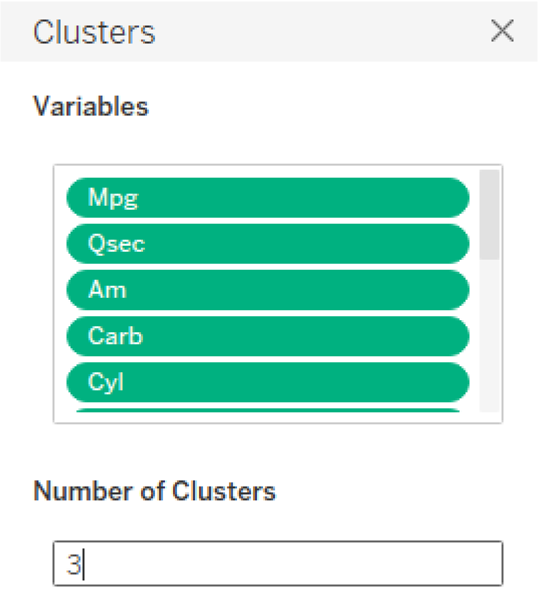
In the following example, we'll let the Tableau clustering algorithm automatically determine the number of clusters---in this case, two. However, it's also possible, and sometimes desirable, to set the number of clusters ourselves. Let's change the number of clusters by performing the following steps:

1. Right-click on the `Clusters` pill in the `Marks` card.
2. From the drop-down menu, select `Edit clusters...`:



This will launch the `Clusters` window that we already encountered when we were creating the initial clusters.

3. Under `Number of Clusters` , type in the desired number of clusters. This time, let's go with `3` :



Clusters

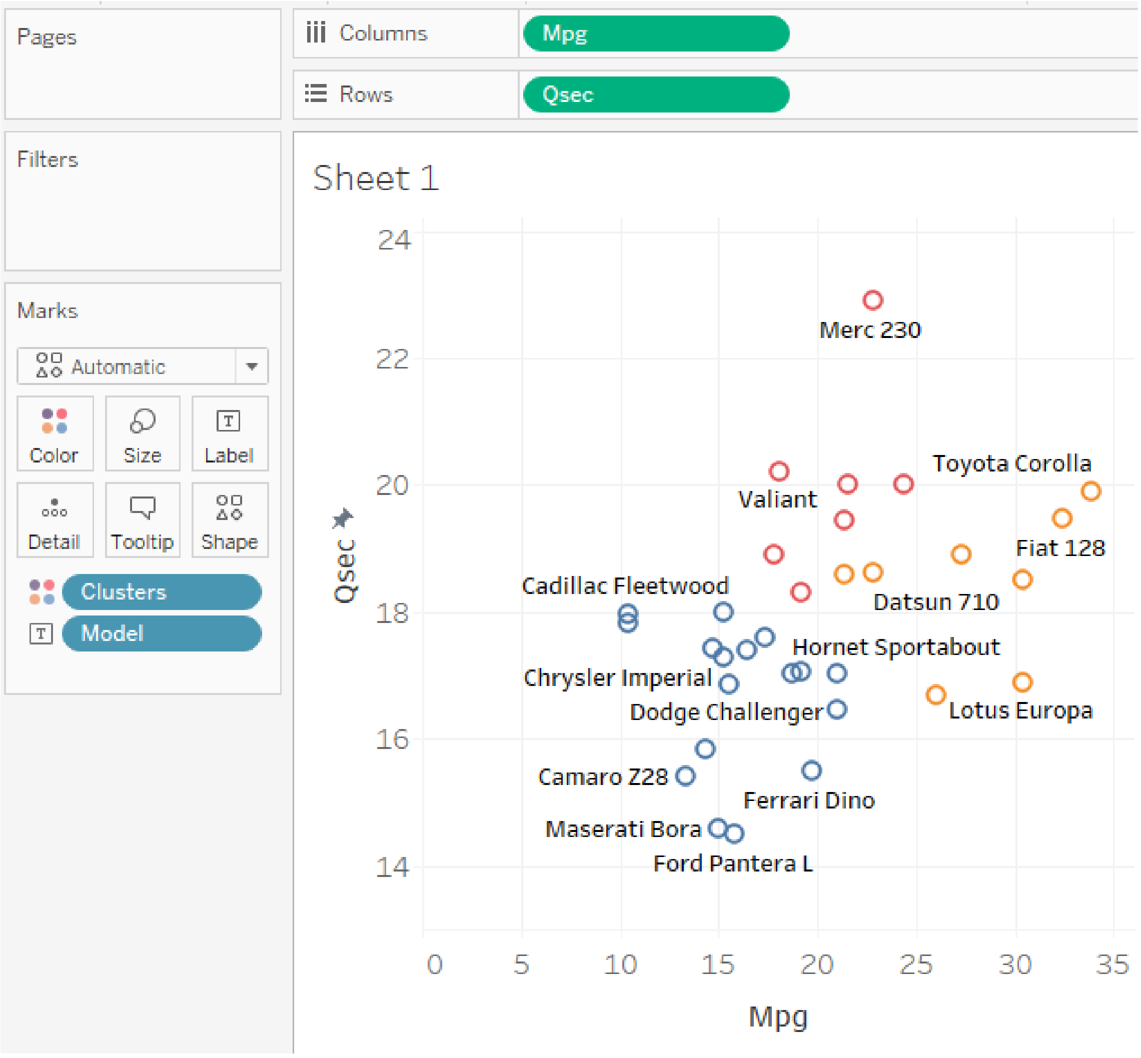
Variables

- Mpg
- Qsec
- Am
- Carb
- Cyl

Number of Clusters

3

4. Click on `x` to close the `Clusters` window. We've now created a three-cluster solution:



Note

Always be careful with setting the number of clusters and don't increase it unless you have a good reason to do so. Simpler solutions are usually better!

See also

- To learn more about k-means clustering, start here: https://en.wikipedia.org/wiki/K-means_clustering
- For more information about running cluster analysis in Tableau, feel free to visit the `Tableau Help` page on the topic: https://onlinehelp.tableau.com/current/pro/desktop/en-us/clustering.htm

Discovering the latent structure of the dataset

When dealing with complex topics, we usually end up with a dataset with a large number of variables. To find meaning in this kind of dataset is typically a tricky task. Luckily, there are some analytical techniques that can help us. One of those techniques is [**principal component analysis**] ([**PCA**]), which is a data reduction technique. Mathematical transformation in this analysis enables us to derive the most informative dimensions of our dataset. The mathematics underlying the analysis [**singular value decomposition**] ([**SVD**]) is somewhat complex, so we won't go into too much detail in this recipe. The basics of PCA can be described like this: you start with a dataset with many variables, then you simplify that dataset by turning your original variables into a smaller number of [**principal components**] in a way that guarantees that you'll preserve as much information as possible. Those components can be used for further analysis as well and provide insight into the underlying structure of our data, helping us make sense of our data and see the forest among the trees.

Getting ready

The dataset that we're going to use for this recipe comes from a psychological survey. A group of people was assessed using a psychological test. The result of the assessment is psychological profile composed of numerical scores representing following personality traits: anxiety, hostility, self-consciousness, assertiveness, sensation seeking, positive emotions, aesthetics, ideas, values, fairness, altruism, gentleness, competency, achievement, and discipline. Since simultaneously interpreting the results on various variables is a difficult task, we want to reduce the number of variables, but in a way that preserves the most information from the original dataset.

Note

For that task, we'll use the `prcomp` function that's built-in R.

How to do it...

1. Launch R and open a new script by selecting **`File`** and then **`New script`**.
2. In the **`R Editor`** window, enter the following code and make sure to replace all of the file paths with the paths leading to the appropriate locations on your device. After pasting the paths, make sure to replace the backslashes with double backslashes, as seen in the following code:

```
pt <-read.table ("C:\\Users\\Slaven\\Desktop\\personality_ traits.csv", header=T,
sep=",")
pt.pc <- prcomp(pt, scale = TRUE)
X1 <- pt.pc$x [, 1]
X2 <- pt.pc$x [, 2]
X3 <- pt.pc$x [, 3]
X4 <- pt.pc$x [, 4]
X5 <- pt.pc$x [, 5]
scores <- cbind(pt, X1, X2, X3, X4, X5)
colnames(scores)[17] <- "X1"
colnames(scores)[18] <- "X2"
colnames(scores)[19] <- "X3"
colnames(scores)[20] <- "X4"
colnames(scores)[21] <- "X5"
write.csv (scores, "C:\\Users\\Slaven\\Desktop\\scores.csv")
loadings <- as.data.frame(pt.pc$rotation)
loadings <- loadings[2:16, 1:5]
loadings$trait <- row.names(loadings)
rownames(loadings) <- c()
path <-matrix(1,15,1)
up <- cbind(loadings,path)
zero <-matrix(0, 15, 6)
trait <- as.data.frame(loadings [,6])
down <-cbind (zero, trait)
colnames(down) <- c("PC1","PC2","PC3","PC4","PC5", "path", "trait")
pt.loadings <- rbind(up, down)
write.csv (pt.loadings, "C:\\Users\\Slaven\\Desktop\\pt.loadings.csv")
```

3. Select the entire code and click the **`Run`** icon

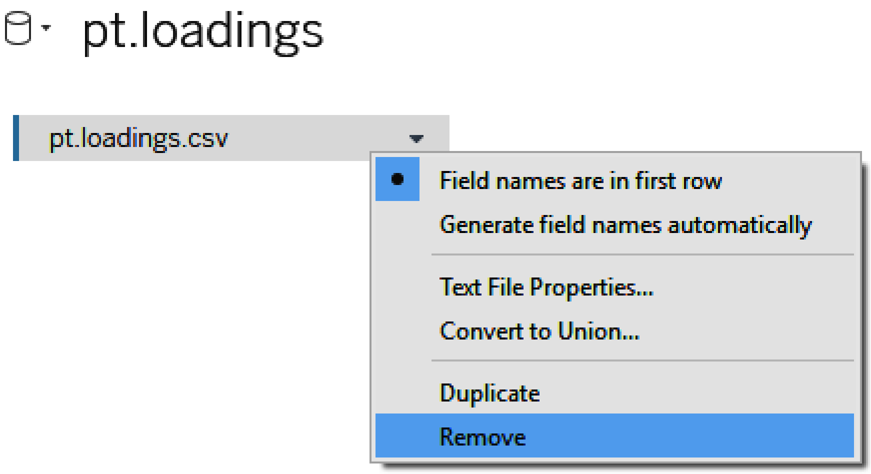
```
. Alternatively, use [*Ctrl*]+[*R*] to run
```

the code.

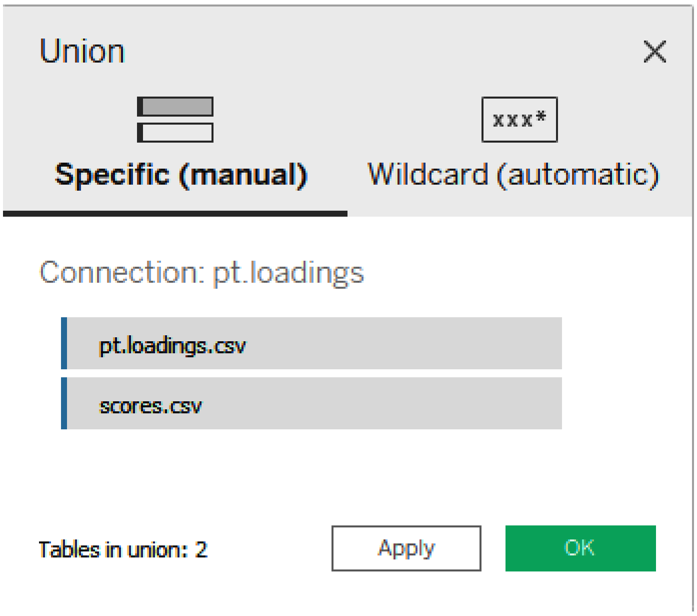
4. Two files have now been created at the location you specified: `pt.loadings.csv` and `scores.csv`. We'll now use them to create a visualization in Tableau. Launch Tableau.

5. From the **`Connect`** pane, select **`Text file`** and navigate to either of the files you have created---`pt.loadings.csv` or `scores.csv`. Select it and click **`Open`**.

6. In the **`Data Source`** page, remove the file you've connected to by right-clicking on it and selecting **`Remove`** from the drop-down menu:



7. From the **`Files`** pane, drag and drop **`New Union`** into the whitespace. The **`Union`** window will open.
8. Drag and drop `scores.csv` and `pt.loadings.csv` from the **`Files`** pane into the **`Union`** window, one by one. Click on **`Apply`**, and then click on **`OK`** to exit the window:



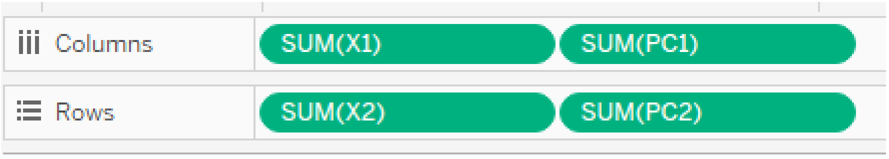
Connection: pt.loadings

pt.loadings.csv

scores.csv

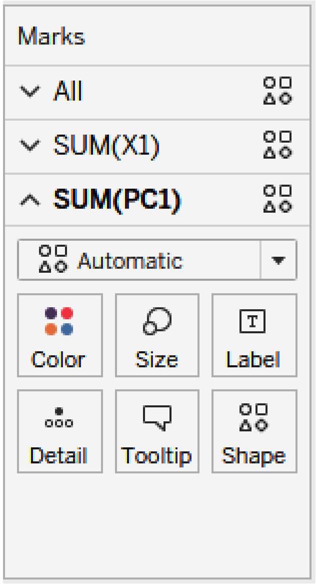
Tables in union: 2

9. Navigate to `Sheet 1` .
10. Drag and drop `X1` from `Measures` into the `Columns` shelf.
11. Drag and drop `X2` from `Measures` into the `Rows` shelf.
12. Drag and drop `PC1` from `Measures` into the `Columns` shelf, to the right of the `SUM(X1)` pill.
13. Drag and drop `PC2` from `Measures` into the `Rows` shelf, to the right of the `SUM(X2)` pill:

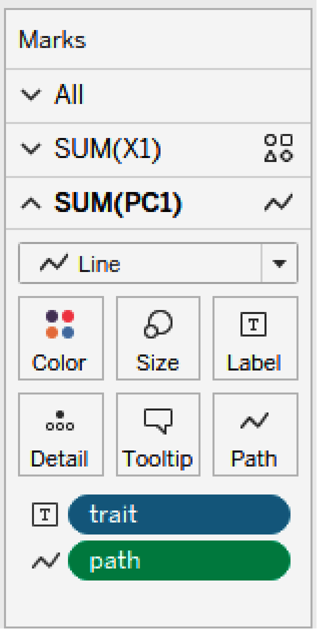


Columns	SUM(X1)	SUM(PC1)
Rows	SUM(X2)	SUM(PC2)

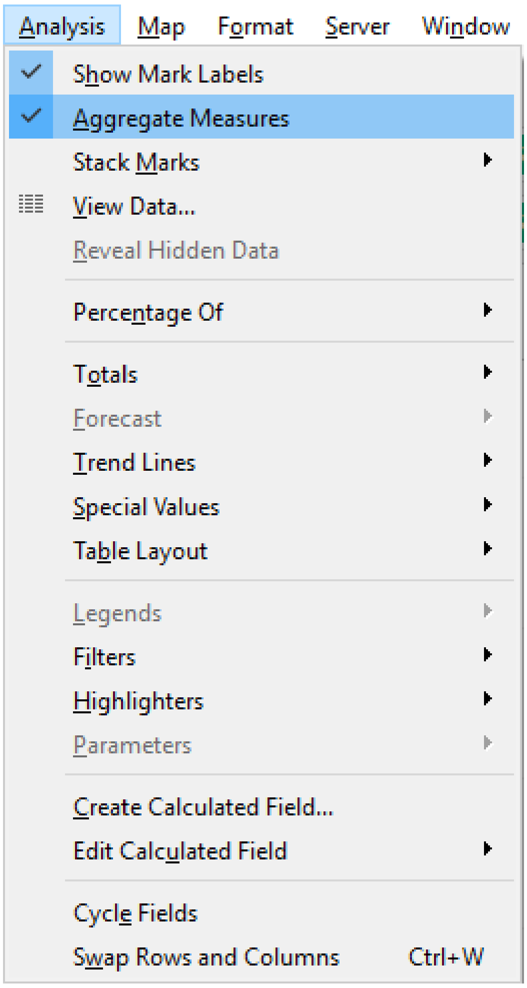
14. In the `Marks` card, click on the second `SUM(PC1)` chart in order to expand it:



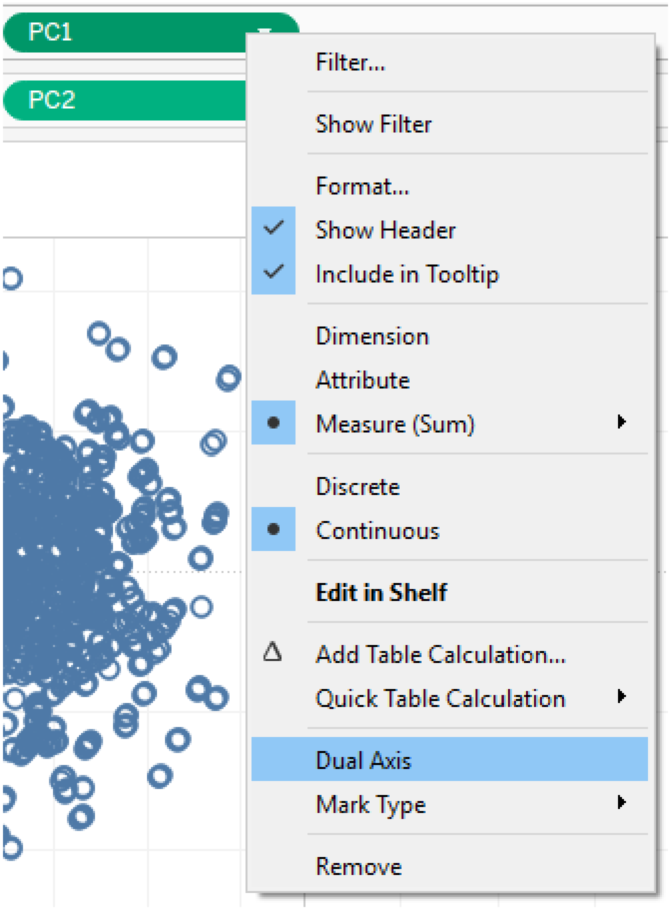
15. Click on the `Automatic` drop-down menu and change the mark type to `Line` .
16. Drag and drop `path` from `Measures` onto `Path` in the `Marks` card.
17. Drag and drop `trait` from `Dimensions` onto `Label` in the `Marks` card:



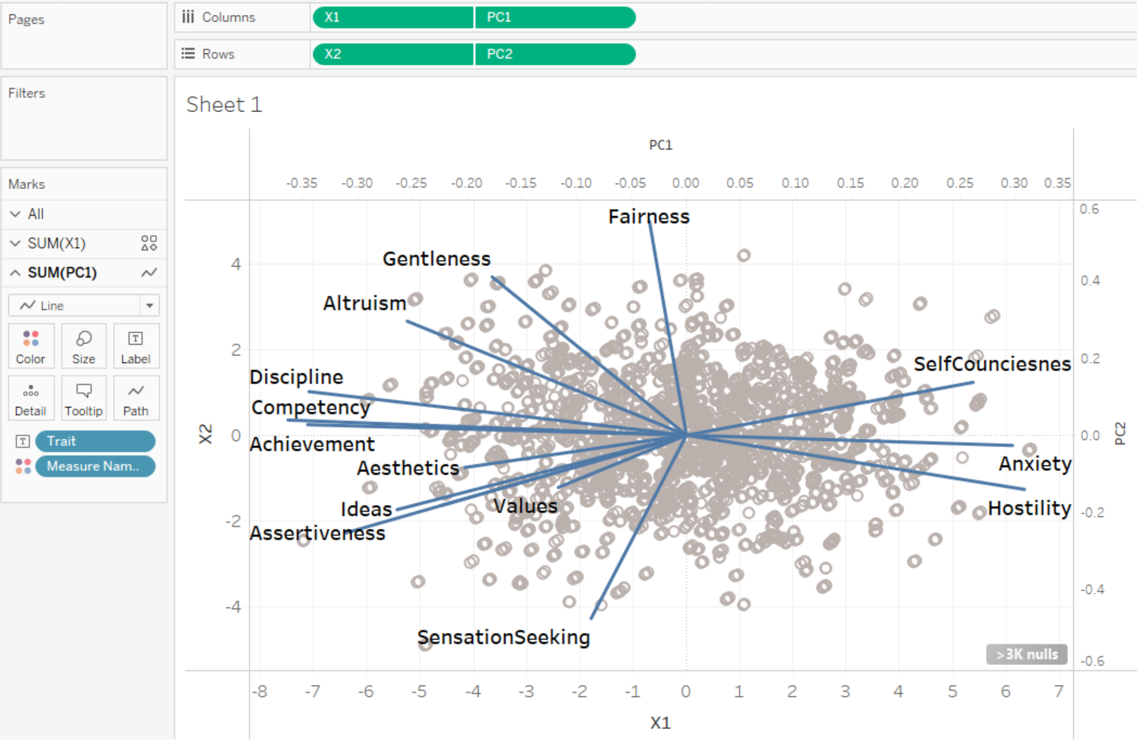
18. In the main menu toolbar, navigate to `Analysis` and, from the drop-down menu, deselect `Aggregate Measures` :



19. Right-click on the `PC1` pill in the `Columns` shelf and select `Dual Axis` from the drop-down menu:



20. Right-click on the `PC2` in the `Rows` shelf and select `Dual Axis` from the drop-down menu. We've finished our PCA and created a chart to visualizeit:



Note

In order to make the chart more appealing and easier to read, you can adjust the formatting in a way that emphasizes the most important elements. In the preceding screenshot, we've increased the label font and made it bold, while at the same time changing the color of circles from orange to gray and decreasing opacity. Feel free to explore the formatting options that work best for you!

How it works...

We've just performed a PCA and created a plot showing data points in a swarm and our original personality dimensions as lines radiating from the center of the chart. Values on [*x*] and [*y*]-axes are actually the first and the second principal components, respectively. By definition, the first two components explain the largest proportion of variability in the original data, which is why we use them in our chart. Points represent individual people (cases) from our original data file but, instead of showing scores on the different personality traits, we're charting the scores on the first two principal components. Through our R script, we wrote a new file, `scores.csv`, where we recorded the scores of individuals on each of the principal components. We saved those values (scores on principal components) as measures: X1, X2, and so on. On the other hand, the lines represent our original dimensions---personality traits. In the chart, we're showing their loadings on, or correlations with, the first two principal components. Those values are also saved as measures: PC1, PC2, and so on. We used them to interpret our principal components---the higher the loading, the more important that original dimension is for defining the principal component. We created this part of the chart using the `pt.loadings.csv` dataset, where we recorded loading of each trait on each of the principal components.

Based on the length and direction of lines, we can conclude that the first principal component, which carries the most information, differentiates people based on their emotional stability, ranging from emotionally stable (on the left-hand side), characterized by high **`Discipline`**, **`Competency`**, and **`Achievement`**, to emotionally unstable (on the right-hand side), characterized by high **`Self-consciousness`**, **`Anxiety`**, and **`Hostility`**.

The second principal component, which is shown on the [*y*]-axis, is defined by **`Fairness`** , **`Gentleness`** , and **`Altruism`** on one side, and **`Sensation-seeking`** and **`Assertiveness`** on the other side. This dimensions can be interpreted as speaking about cooperativeness versus competitiveness.

Note

Always be mindful of the fact that interpretation of the components goes beyond interpreting the data at hand and relies heavily on domain-specific knowledge about the topic of research.

There's more...

Visualizing the first and the second principal component is the usual way to go since these two components carry the most information. However, sometimes we might wish to see the third, fourth, and so on, principal component.

Note

A handy way to allow for this is to create a parameter that would enable users to switch between principal components they wish to see in the plot.

See also

- For more information on PCA, you may want to start here: https://en.wikipedia.org/wiki/Principal_component_analysis
- To find out more about the `prcomp` function that we used to implement the analysis, see the documentation: https://www.rdocumentation.org/packages/kazaam/versions/0.1-0/topics/prcomp
- Since the book is not meant to provide a comprehensive introduction to R language or data science reader is encouraged to refer the following links:
- https://www.statmethods.net/index.html
- https://www.r-bloggers.com{.ulink}
- https://www.datacamp.com{.ulink}

Extracting the structure beneath discrete variables

This recipe will guide you through the process of performing and visualizing the results of correspondence analysis. Correspondence analysis is a data reduction technique frequently used in brand image studies, but also in other types of research, because it allows us to neatly map brands on a map formed by brand attributes.

Getting ready

In this recipe, we'll use `telco_image.csv` , a dataset coming from a market research survey. Cell phone users were asked to rate the three biggest mobile network providers in their country on a list of attributes. For each attribute, they chose one brand they felt best fit that description. Before we begin, make sure you've saved the `telco_image.csv` dataset to your device.

How to do it...

1. Launch R, and open a new script by selecting **`File`** and then **`New script`** .
2. In the **`R Editor`** window, enter the following code. Make sure to replace all of the file paths with the paths leading to the appropriate locations on your device. After pasting the paths, make sure to replace the backslashes with double backslashes, as seen in the following code block:

```
install.packages('ca',repos='http://cran.us.r-project.org')
library(ca)
df <- read.table("C:\\Users\\Slaven\\Desktop\\telco_image.csv", header=T, sep=",")
n <-ncol(df)
blank <-matrix(NA, 1 ,n)
```

```
blank$brands <- c(colnames(df))
brands <- as.data.frame(blank$brands[-1])
names(brands) <- "labels"
labels <-df [ ,1]
labels <- as.data.frame(labels)
labels.df <- rbind(labels,brands)
type1 <-as.data.frame(rep("Brand", nrow(brands)))
names(type1) <- "Type"
type2 <-as.data.frame(rep("Feature", nrow(labels)))
names(type2) <- "Type"
type.df <- rbind(type2, type1)
num.df<- df[,-1]
c<-ca(num.df)
X <-append(c$rowcoord[,1],c$colcoord[,1], )
Y <-append(c$rowcoord[,2],c$colcoord[,2], )
axes <- data.frame(cbind(X,Y))
fin.data <- cbind(axes,type.df,labels.df)
write.csv(fin.data, "C:\\Users\\Slaven\\Desktop\\CA_input_data.csv")
```

3. Select the entire code and click on the `Run` icon

   ```
   . Alternatively, use [*Ctrl *]+ [*R*] to run
   ```

   the code.

4. A new file, `CA_input_data.csv`, has now been created at the location you specified. We'll now use it to create a visualization in Tableau. Launch Tableau.

5. From the `Connect` pane, select `Text file` and navigate to `CA_input_data.csv`, then select it and click on `Open`.

6. Navigate to `Sheet 1`.

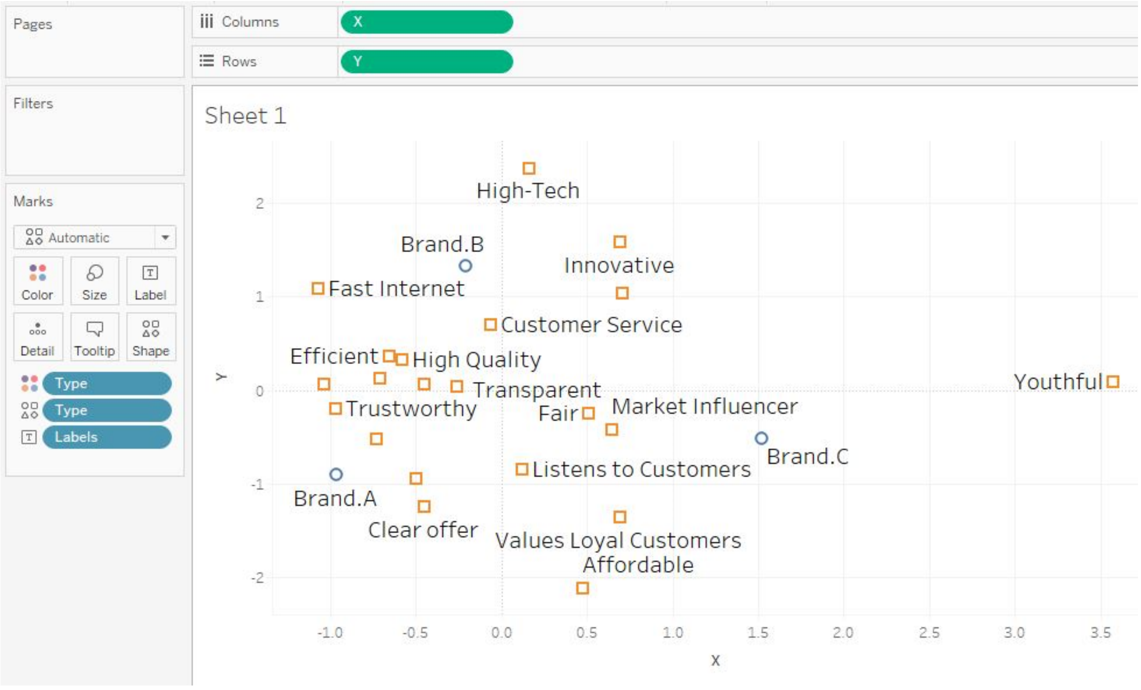
7. Drag and drop `X` from `Measures` into the `Columns` shelf.

8. Drag and drop `Y` from `Measures` into the `Rows` shelf.

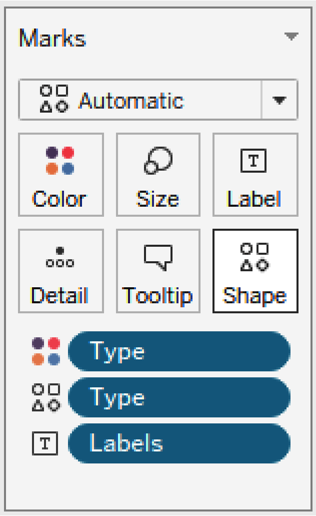
9. Drag and drop `Labels` from `Dimensions` onto `Label` in the `Marks` card.

10. Drag and drop `Type` from `Dimensions` onto `Color` in the `Marks` card.

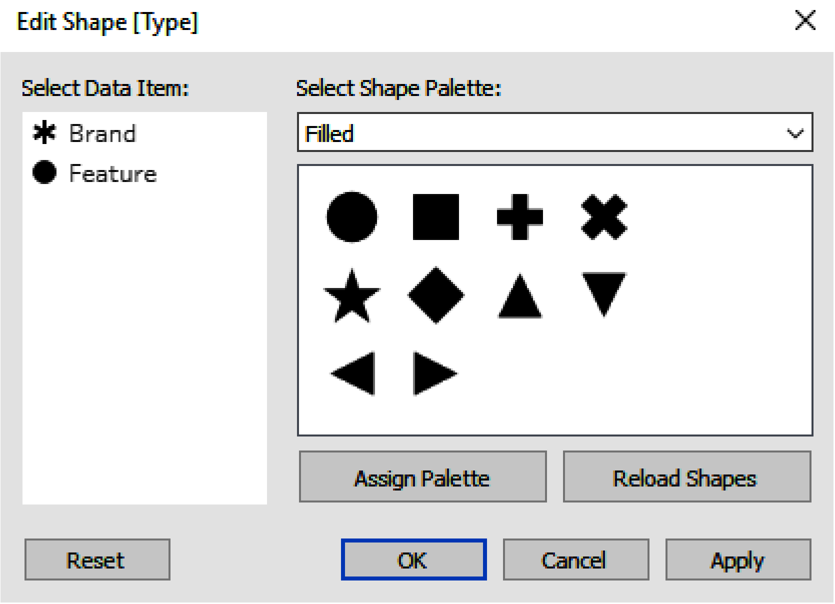
11. Drag and drop `Type` from `Dimensions` onto `Shape` in the `Marks` card:



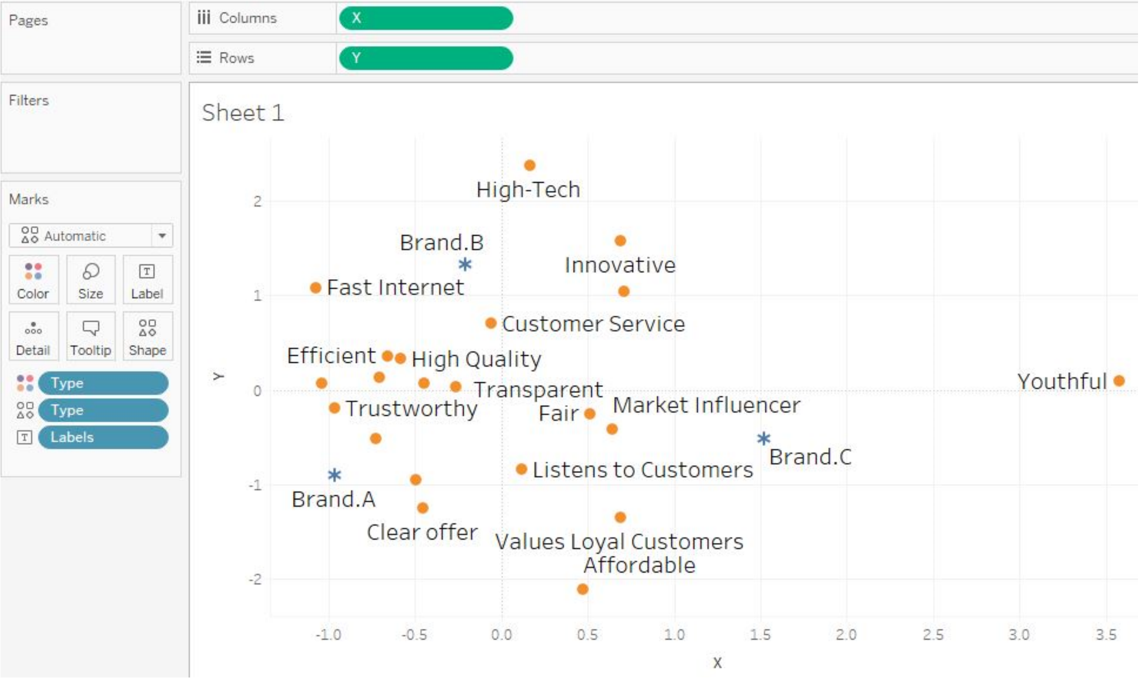
12. Click on the `Shape` button in the `Marks` card:



13. Assign the shapes you like to the brand and feature by clicking on the desired value (`Brand` or `Feature`) under `Select Data Item` , and choosing the desired shape from the `Select Shape Palette` drop-down menu. Let's choose the shapes shown in the following screenshot:



14. When you are done with choosing the shapes, click on `OK` to exit the `Edit Shape [Type]` window. Our correspondence analysis is done:



How it works...

Correspondence analysis works similarly to PCA, which we covered in the first recipe, Discovering latent structure of the dataset, of this lab. It reduces the number of dimensions that differentiate our cases (in this example, brands) so we have a clearer overview of how each of the brands is positioned. On the [*x*] and [*y*] axes, we plotted the first two dimensions extracted by correspondence analysis, which have the same function as the first two principal components in the principal components analysis.

Both the brands and the attributes describing them are plotted in the space created by the first two dimensions. We interpret the results, or read the map, by looking at the spatial relationship of attributes and brands. In our example, we see that **`Brand.B`** is perceived as **`High-Tech`** and **`Innovative`**, offering **`Fast Internet`** and a good choice of smartphones. On the other hand, it's on the opposite side of the spectrum from the attributes **`Affordable`**, **`Values Loyal Customers`**, and **`Clear offer`**, which are more representative of **`Brand.A`** and **`Brand.C`**. Additionally, **`Brand.A`** is also seen as **`Trustworthy`** and having professional staff, while **`Brand.C`** has an image of a **`Youthful`** and **`Fair`** brand that is a strong market influencer. Overall, the map helps us form a well-rounded image of how customers perceive each of the brands.

There's more...

Just as with the PCA, we can also use the positions of attributes on the plot to interpret the dimensions created by them. In our example, the [*y*]axis could be interpreted as ranging from affordable/simple offer to high-tech/innovative, while the [*x*] axis could be interpreted as being trustworthy/friendly/transparent on one pole, and youthful on the other.

See also

- If you're interested in gaining a more in-depth understanding of the correspondence analysis itself, you may want to start with this article: https://en.wikipedia.org/wiki/Correspondence_analysis
- For more information on the R `ca` package that we used to run the analysis, you can refer to this link: https://cran.r-project.org/web/packages/ca/index.html

Data mining with tree-based models

Classification of cases in groups is one of the most important tasks in data mining and analytics. In the previous recipe,

> *Clustering , we made clusters of similar cases based on measures by letting Tableau discover groups and patterns in the data. However, there are also situations where we already have a dimension labeling certain groups in our data, and we want to create a model that will predict group membership using other fields in our dataset. For this task, we can use a tree-based model. By the end of this lab, you'll have built a decision tree using a classification algorithm, while retaining a focus on a real-life business question.*

Getting ready

In this recipe, we'll be using the `new_or_used_car.csv` dataset. It contains data on people planning to buy a car in the next 12 months. We have some demographic data on them, such as age, gender, and income, and some data about the car they're currently driving---whether it was bought as new or used and its date of manufacture. Finally, we have a dimension, `Future Purchase`, which lets us know whether they're planning to buy a new or used car.

Before we dive into the recipe, please make sure you have the `new_or_used_car.csv` dataset saved to your device.

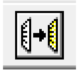
How to do it...

1. Launch R and open a new script by selecting **`File`** and then **`New script`**.
2. In the **`R Editor`** window, enter the following code. Make sure to replace all of the file paths with the paths leading to the appropriate locations on your device. After pasting the paths, make sure to replace the backslashes with double backslashes, as in the following code block:

```
install.packages('rpart',repos='http://cran.us.r-project.org')
library(rpart)
cars <- read.table("C:\\!Slaven\\6 KNJIGA\\4 Advanced analytics\\4 decision
tree\\new_or_used_car.csv", header=T, sep=",")
```

```
fit <- rpart(FuturePurchase ~ Age + Gender + Education + FamilyStatus + CurrentCar +
AgeOfCurrentCar + MunicipalityType, method="class", data=cars)
plot(fit, uniform=TRUE, main="Classification of new cars buyers")
text(fit, all=TRUE, cex=.8)
```

3. Run the entire block of code by selecting it clicking the `Run` button



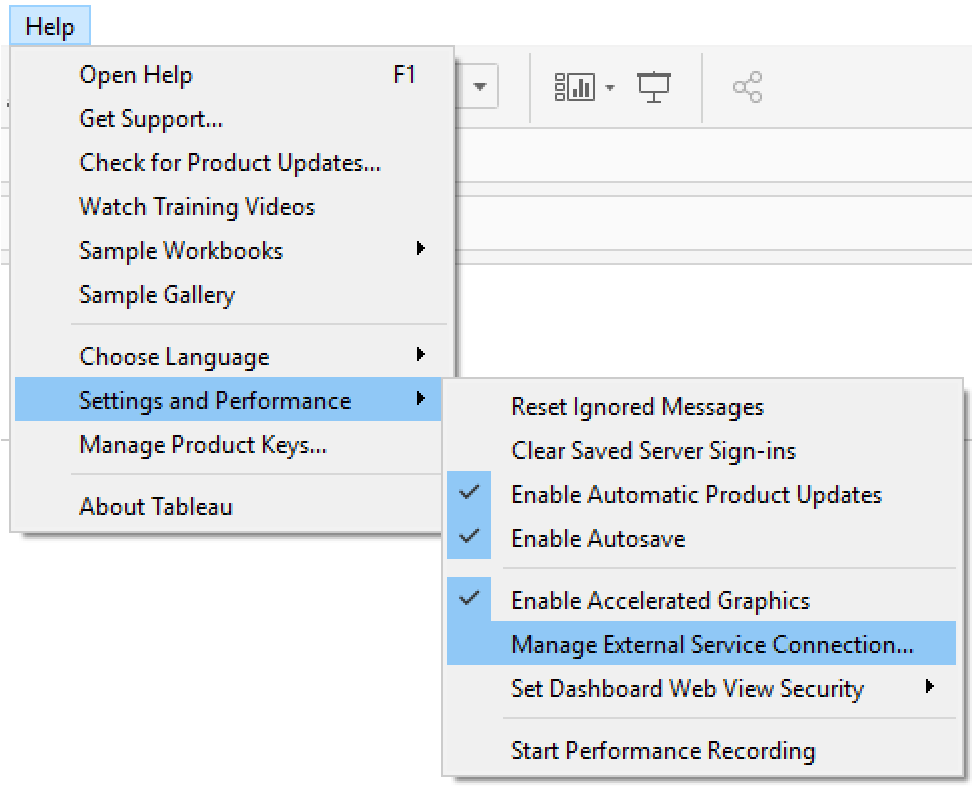
```
or by pressing [*Ctrl*] + [*R*] on your
```

keyboard.

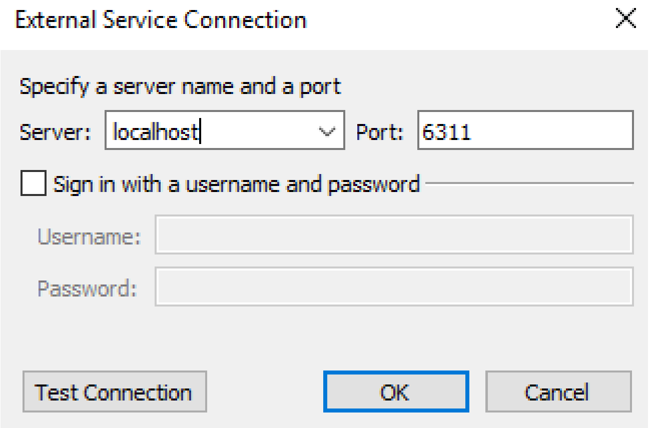
4. In order to make the connection with Tableau possible, call the `Rserve` library by running the following code:

```
library(Rserve)
Rserve()
```

5. Open Tableau and connect to the `new_or_used_car.csv` dataset.
6. In the main menu toolbar, navigate to `Help` and, from the drop-down menu, select `Settings and Performance`. From the additional drop-down menu that will open, select `Manage External Service Connection` ...:


Help
Open Help F1
Get Support...
Check for Product Updates...
Watch Training Videos
Sample Workbooks
Sample Gallery
Choose Language
Settings and Performance
Manage Product Keys...
About Tableau
Reset Ignored Messages
Clear Saved Server Sign-ins
✓ Enable Automatic Product Updates
✓ Enable Autosave
✓ Enable Accelerated Graphics
Manage External Service Connection...
Set Dashboard Web View Security
Start Performance Recording

7. In the **`Server`** field, enter `localhost`.
8. In the **`Port`** field, enter `6311`.
9. Click on the **`Test Connection`** button:



Specify a server name and a port

Server: localhost

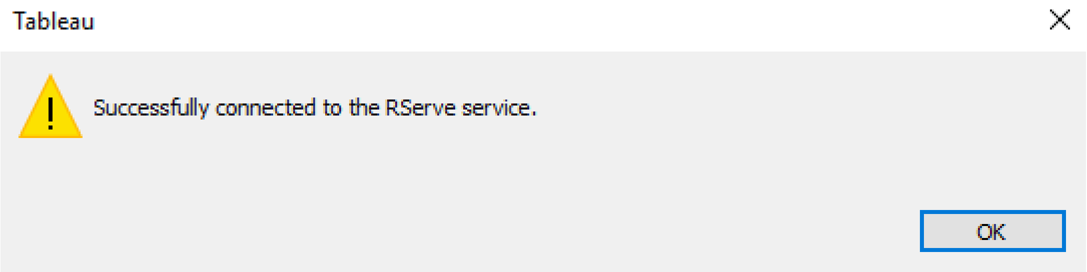
Port: 6311

☐ Sign in with a username and password

Username:

Password:

10. After receiving the message that you're successfully connected to the `RServe` service, click on **`OK`** to exit the notification, and then click on **`OK`** again to exit the **`External Service Connection`** window:



Successfully connected to the RServe service.

11. Open a new sheet.

12. Drag and drop **`Future Purchase`** from **`Dimensions`** into the **`Columns`** shelf.

13. Drag and drop **`Id`** from **`Dimensions`** onto **`Tooltip`** in the **`Marks`** card.

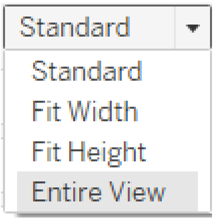
14. In the main menu toolbar, navigate to **`Analysis`** and, from the drop-down menu, select **`Create Calculated Field...`**.

15. Rename the new calculated field to

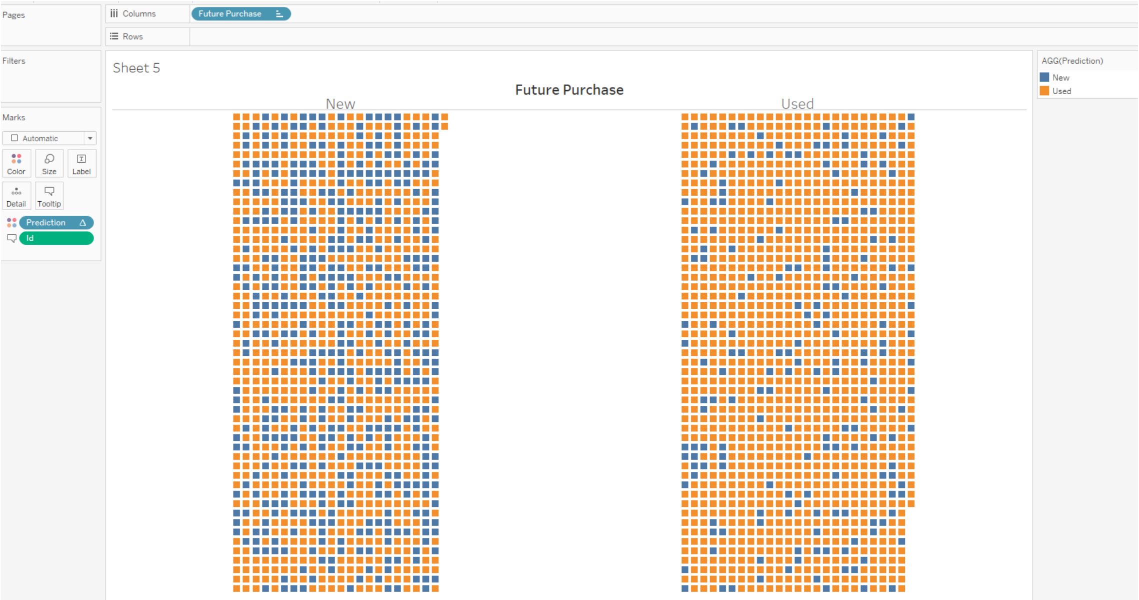
> ***`Prediction`***
>
> *`and enter the following code block:`*

```
SCRIPT_STR
('library(rpart);
fit = rpart(FuturePurchase ~ Age + Gender + Education + FamilyStatus + CurrentCar +
AgeOfCurrentCar + MunicipalityType,
method="class",
data.frame(FuturePurchase = .arg1,
Age =.arg2,
Gender =.arg3,
Education =.arg4,
FamilyStatus =.arg5,
CurrentCar =.arg6,
AgeOfCurrentCar =.arg7,
MunicipalityType=.arg8));
io<-predict(fit, type = "prob");colnames(io)[apply(io,1,which.max)]',
ATTR([Future Purchase]),
AVG([Age]),
ATTR([Gender]),
ATTR([Education]),
ATTR([Family Status]),
ATTR([Current Car]),
AVG([Age Of Current Car]),
ATTR([Municipality Type]))
```

16. Click `OK` to exit the calculated field window.
17. Drag and drop `Predictions` from `Measures` onto `Color` in the `Marks` card.
18. In main menu toolbar, go to `Analysis` and deselect `Aggregate Measures`.
19. In the drop-down menu, change `Standard` to `Entire Vie``w`:



In the following screenshot, we can see that we finished our chart showing the actual and predicted classification of customers:



How it works...

Decision trees utilize the approach of repetitively dividing cases into smaller, more homogeneous groups. Finally, we end up with a classification tree which can be effectively visualized and easily interpreted. In our example, we're trying to answer the question based on the data we have. The question that arises is: can we effectively predict who's going to buy a new car and who's going to buy a used car? As it turns out, we can. We've created a predictive model using the previous R script, and we recorded the predicted values for each case in the new field called `Prediction`. We've then created a visualization comparing the actual values of the `Future Purchase` dimension and the values we predicted. We can clearly see that, although there are some misclassifications, our model performs pretty well!

There's more

After having developed your model, you can save it as the `.rda` file at your computer and reuse it with a new dataset. Let's say you receive some more data about consumers, and you want to see how many of them you can expect to go for a new car---you can just feed that data into the model you've already developed and get an estimate.

See also

- If you're interested in learning more about tree-based models, you may want to start at the following link: https://en.wikipedia.org/wiki/Decision_tree_learning#General
- For more information about the `rpart` library we used in this recipe, start here: https://cran.r-project.org/web/packages/rpart/rpart.pdf

Identifying anomalies in data

When analyzing data we'll frequently encounter unusual cases, outliers, and anomalies. Those cases are different from the majority and they don't match the pattern that the rest of the cases fit in. Sometimes, we might want to identify them in order to remove them from the analysis, because they can skew our results. In other cases, we might be interested in analyzing them. Either way, it's very important to know how to deal with them properly. In *Forecasting with Tableau*, the *Forecasting on a dataset with outliers* recipe taught up how to deal with outliers on one dimension, which is relatively simple. But when we have more than one dimension, things get much more complicated. In this recipe, we'll learn how to deal with multidimensional outliers.

Getting ready

In our recipe, we'll use a dataset from a health study of blood pressure, `age_and_blood_pressure.csv` . It contains information on the age of participants and their blood pressure. Before we begin, make sure you have the dataset saved to your device.

How to do it...

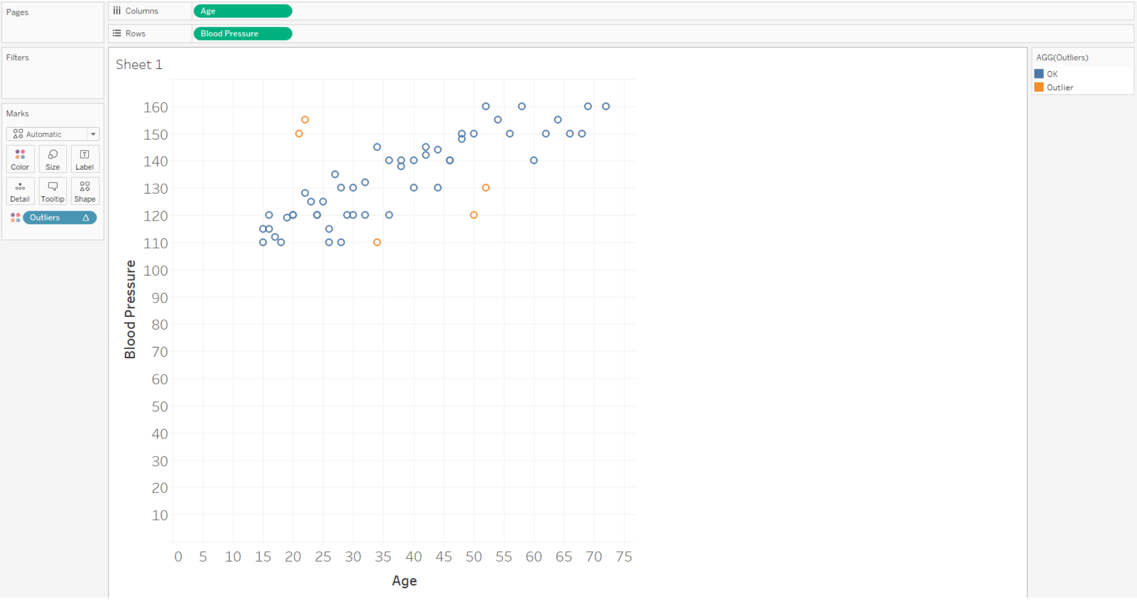
1. Start R, open a new script and run the following code block:

```
install.packages('mvoutlier',repos='http://cran.us.r-project.org')
library(Rserve)
Rserve()
```

2. Open Tableau and connect to the `age_and_blood_pressure.csv` file.
3. In the main menu toolbar, navigate to **`Help`** and, from the drop-down menu, select **`Settings and Performance`** . From the additional drop-down menu that will open, choose **`Manage External Service Connection...`** .
4. In the **`Server`** field, enter `localhost` .
5. In the **`Port`** field, enter `6311` .
6. Click on the **`Test Connection`** button.
7. After receiving a message that you're successfully connected to the `RServe` service, click on **`OK`** to close the notification, and then click on **`OK`** again to exit the **`External Service Connection`** window.
8. Drag and drop **`Age`** from **`Measures`** into the **`Columns`** shelf.
9. Drag and drop **`Blood Pressure`** from **`Measures`** into the **`Rows`** shelf.
10. In the main menu toolbar, navigate to **`Analysis`** and, from the drop-down menu, select **`Create Calculated Field...`** .
11. Rename the new calculated field `Outliers` and enter the following code:

```
IF SCRIPT_REAL("library(mvoutlier);sign2(cbind(.arg1, .arg2))$wfinal01", AVG([Age]),
AVG([Blood Pressure])) == 0 THEN "Outlier" ELSE "OK" END
```

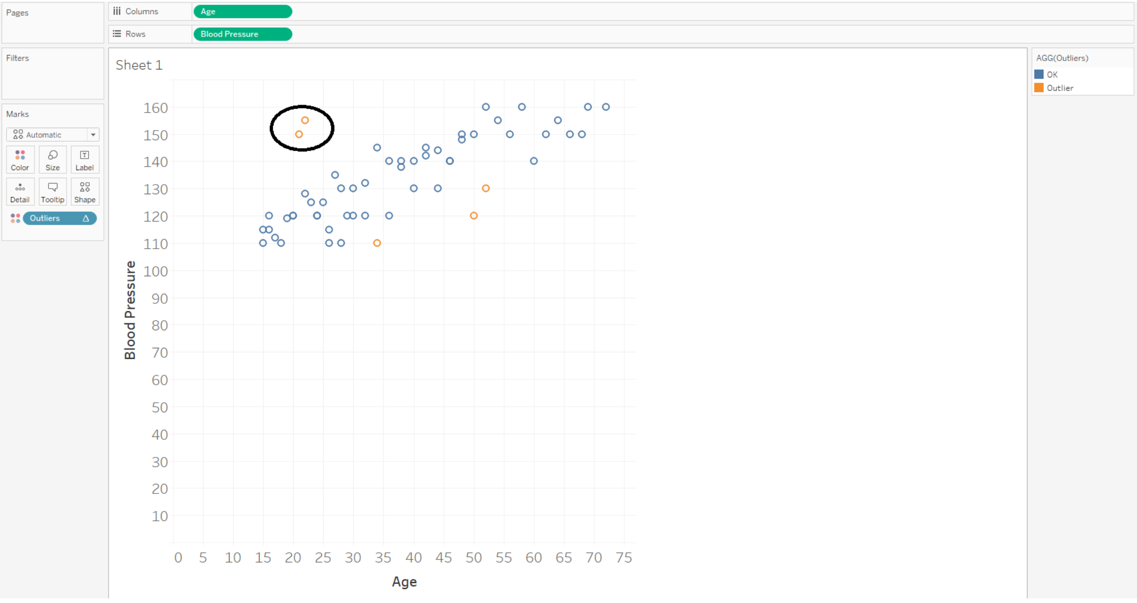
12. In the main menu toolbar, navigate to **`Analysis`** and, in the drop-down menu, deselect **`Aggregate Measures`** .
13. Drag and drop **`Outliers`** from **`Measures`** onto **`Color`** in the **`Marks`** card. Our chart clearly shows the multidimensional outliers now, as shown in the following screenshot:



How it works...

In this recipe, we created a scatter plot showing the multidimensional outliers. We did this because, sometimes, we can't simply isolate outliers on each dimension. Multivariate outliers are defined by their position on more than one variable, so it is important to observe them simultaneously. Certain cases don't have to be outliers on any one of the dimensions considered separately but are outliers if we observe their pattern of values on multiple dimensions.

In our example, we've flagged two people with a respectiveblood pressure of 150 and 155 as outliers, circled in the following screenshot:



However, these values of `Blood pressure` do not make them extreme---there are some subjects with much higher blood pressure. With reference to the preceding screenshot, we can see that other subjects who have blood pressure this high are also considerably older than the ones we marked---they are in their forties, or older, while our two flagged cases are in their early twenties! We can see that their peers have much lower blood pressure. So, while neither the values of `Blood pressure` nor the values of `Age` taken in isolation make this two cases extreme, when considered together, we see that these two subjects are unusual because they have unusually high blood pressure for their age. The opposite is true for other flagged cases---they are flagged because they have unusually low blood pressure for their (older) age. So, had we done a simple, one-dimensional, outlier analysis, we would have never detected these unusual cases.

There's more...

To keep our example clear and illustrative, we used a two-dimensional dataset (and two-dimensional outliers). However, you'll usually deal with datasets that have more than two dimensions. Nevertheless, the general approach you'll use to detect the multidimensional outliers will be the same. You can use the same syntax as the one we've used in this example, but simply add as many arguments as needed. The script will run in the same manner and detect the outliers.

However, when operating with more than two dimensions, you won't be able to visualize the multidimensional outliers on a scatter plot as easily, since the pattern of values that makes them extreme is spread across various dimensions. If you wish to represent them visually, you can check out the `uniplot` function (https://www.rdocumentation.org/packages/mvoutlier/versions/2.0.9/topics/uni.plot) from the `mvoutlier` package in R, which allows you to unfold the pattern of values into a series of unidimensional projections. Although it isn't as neat as Tableau visualizations, it'll provide some insight on how the outlier values can be interpreted.

See also

- To find out more about the `mvoutlier` package we used in this lab and the detection of multivariate outliers in general, visit this page: https://cran.r-project.org/web/packages/mvoutlier/index.html.