Lab 11. Forecasting with Tableau

In this lab, we will cover the following recipes:

- Basic forecasting and statistical inference
- Forecasting on a dataset with outliers
- Using R within Tableau
- Forecasting based on multiple regression
- Regression with random forest
- Time series forecasting

Technical requirements

To follow the recipes from this lab, you will need to have Tableau 2019.1 installed. You will also need to install the latest version of R software for statistical computing. R software is free and can be downloaded from https://cran.r-project.org/.

In the following recipes, we will be using the `hormonal_response_to_excercise.csv` and `stock_prices.csv` datasets, which you can download from the following URLs:

- https://github.com/SlavenRB/Forecasting-with-Tableau/blob/master/hormonal_response_to_excercise.csv{.ulink}
- https://github.com/SlavenRB/Forecasting-with-Tableau/blob/master/stock-prices.txt Please make sure you have a local copy of the dataset saved to your device before we begin.

Introduction

In this lab, we will learn how to perform forecasting, using real-life data from health behavior research and from stock market prices. We are going to discover Tableau built-in functions for linear regression and learn how to correctly interpret the results of statistical tests. Also, this lab will cover the integration of R in Tableau. Using R functionality, we will be able to deal with slightly more complex datasets and perform more sophisticated forecasting.

> *["Prediction is difficult, especially if it's about the future."*
> *-- Niels Bohr]*

Basic forecasting and statistical inference

The aim of this recipe is to introduce a basic forecasting method that relies on linear regression. We are going to use a built-in Tableau facility for linear regression. Simply put, regression analysis helps us discover predictors of a variable that we are interested in. We model the relationship between potential predictors and our variable of interest. Once we establish the model of the relationship between predictors and our variable, we can use it for further predictions.

To perform for casting, we will use the `hormonal_response_to_excercise.csv` dataset. This dataset comes from a health behavior study that aimed to explore the factors influencing cortisol response while exerting the maximal, peak effort during physical exercise (the `Cortmax` variable in our dataset).

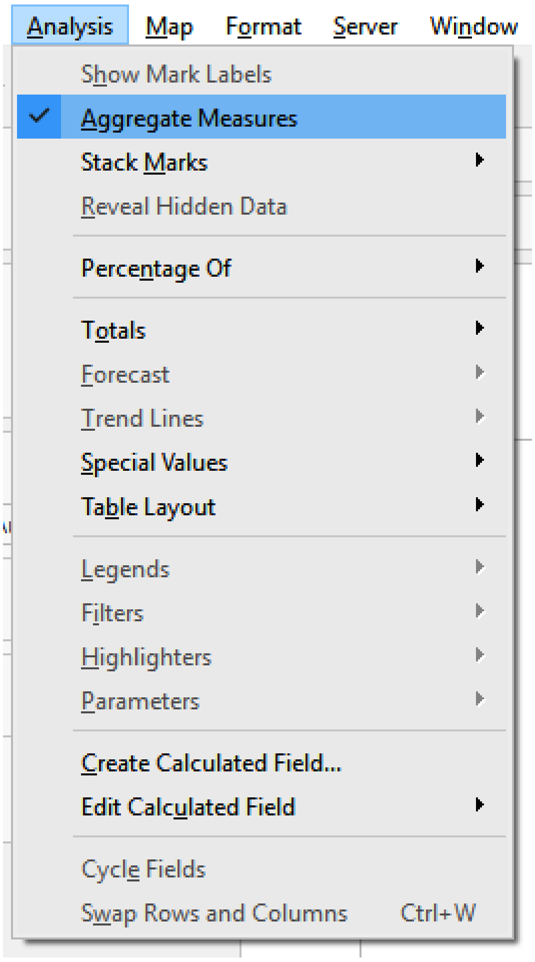
Our first task is to explore how effectively we can predict the level of cortisol response in the point of maximal effort during physical exercise, based on the cortisol level at rest (the `Cortrest` variable). So, in this example, our variable of interest (variable we are trying to predict) is `Cortmax` (cortisol level during maximal effort), while our predictor variable (the one that we are using to make a prediction) is `Cortrest` (cortisol level during rest). We will try to model the relationship between these two variables.

Getting ready

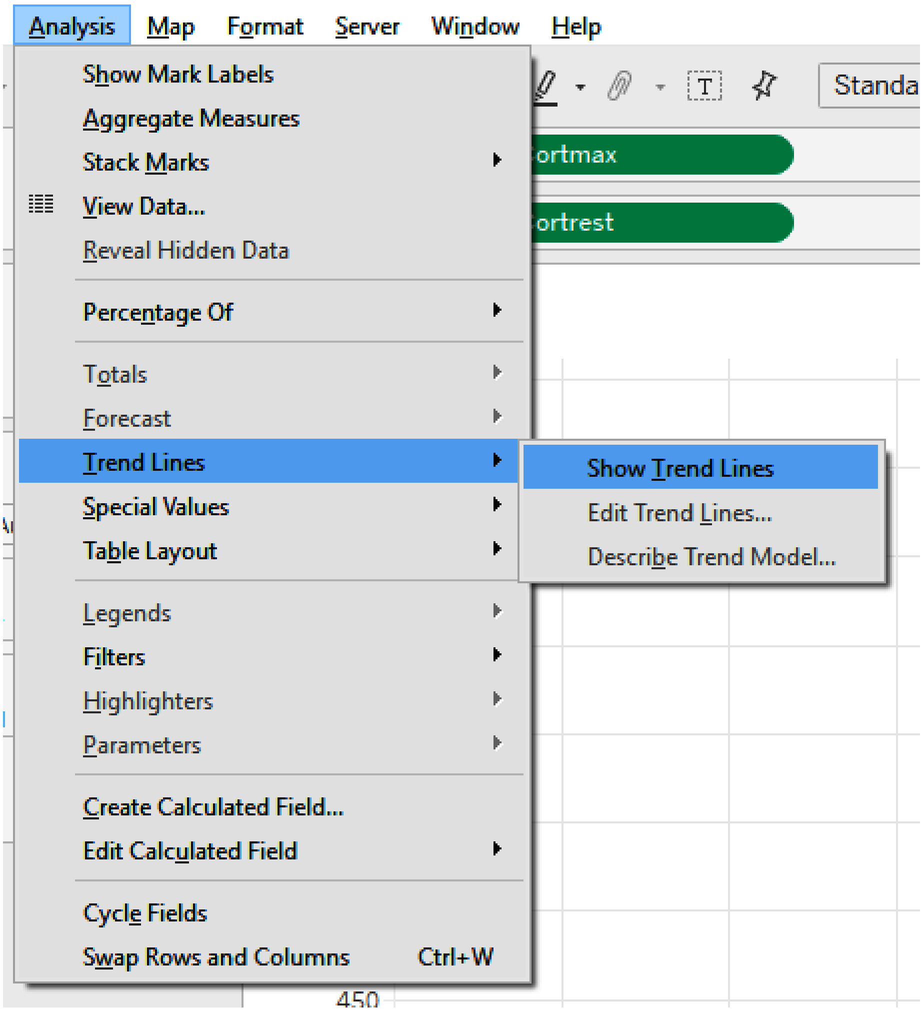
To perform the steps outlined in this recipe, you will need to connect to the `hormonal_response_to_excercise.csv` dataset.

How to do it...

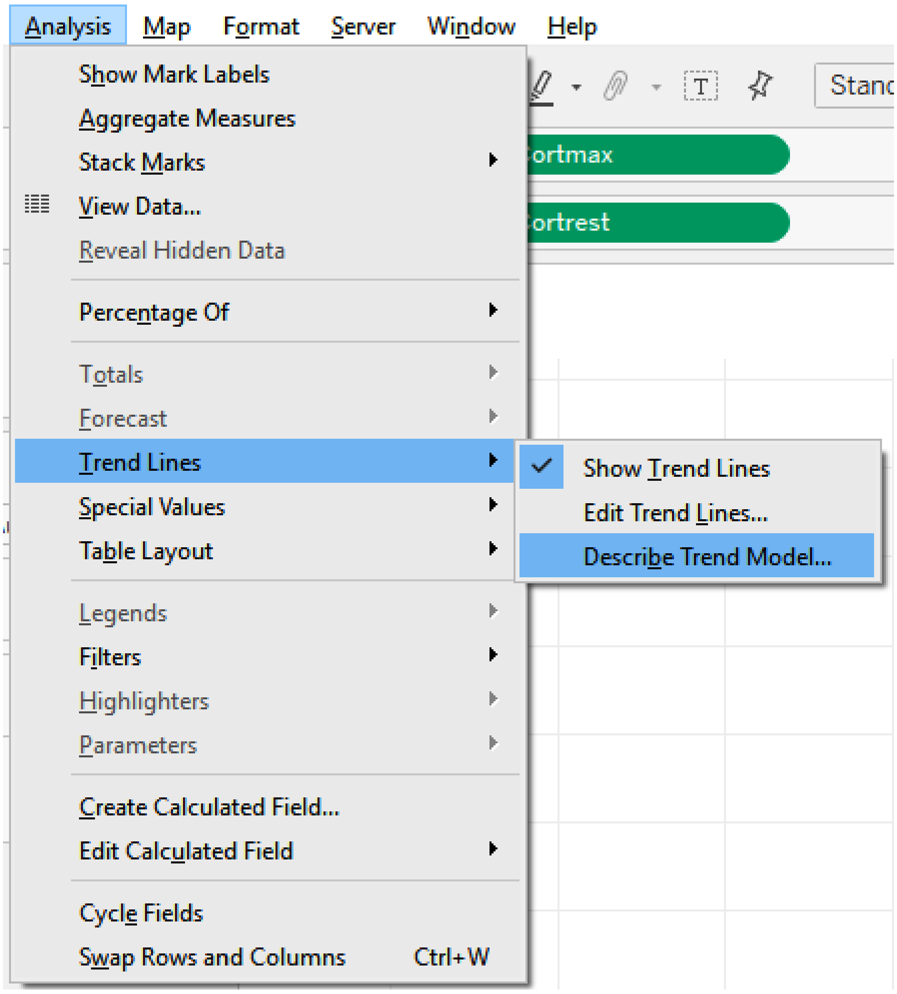
1. Open a blank worksheet and drag and drop **`Cortmax`** from **`Measures`** into the **`Columns`** shelf.
2. Drag and drop **`Cortrest`** from **`Measures`** to the **`Rows`** shelf.
3. In the main menu toolbar, navigate to **`Analysis`** and in the drop-down menu deselect **`Aggregate Measures`**:



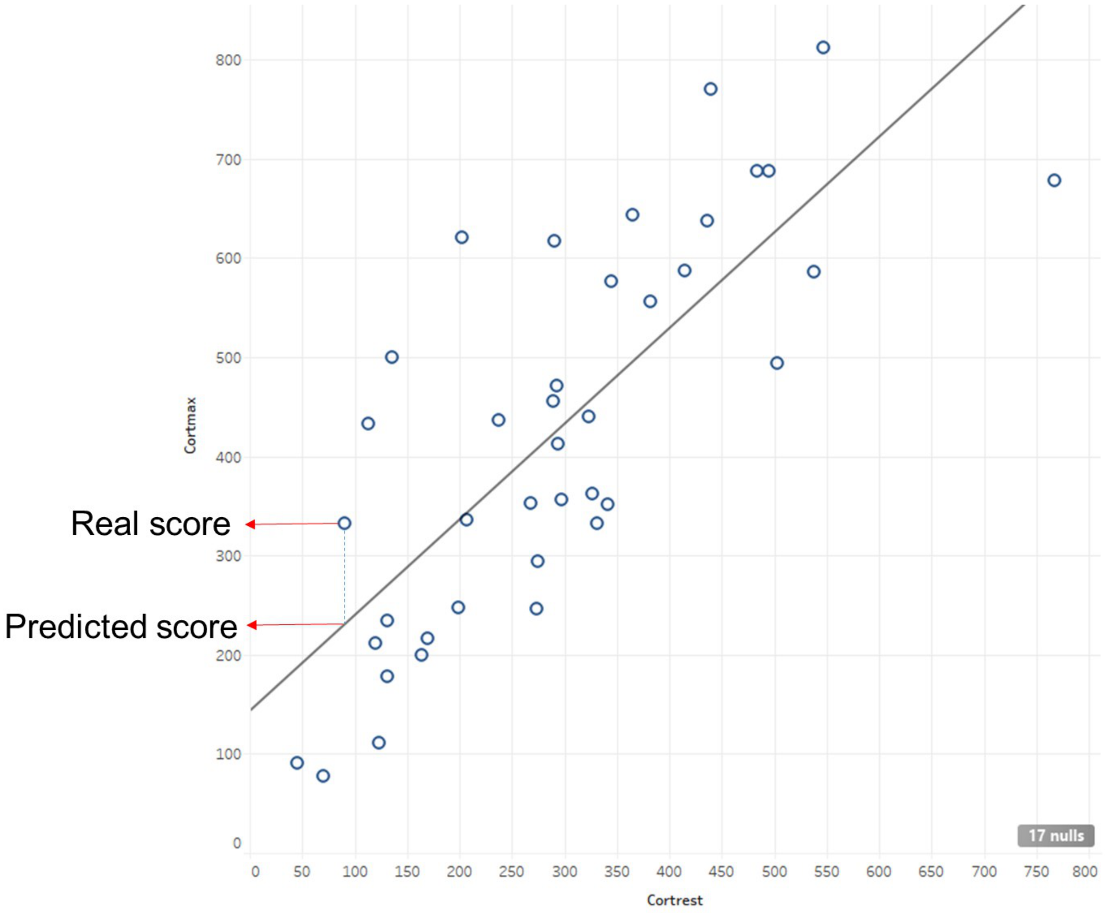
4. In the main menu toolbar, navigate to **`Analysis`** and in the drop-down menu, under **`Trend Lines`**, select **`Show Trend Lines`**:



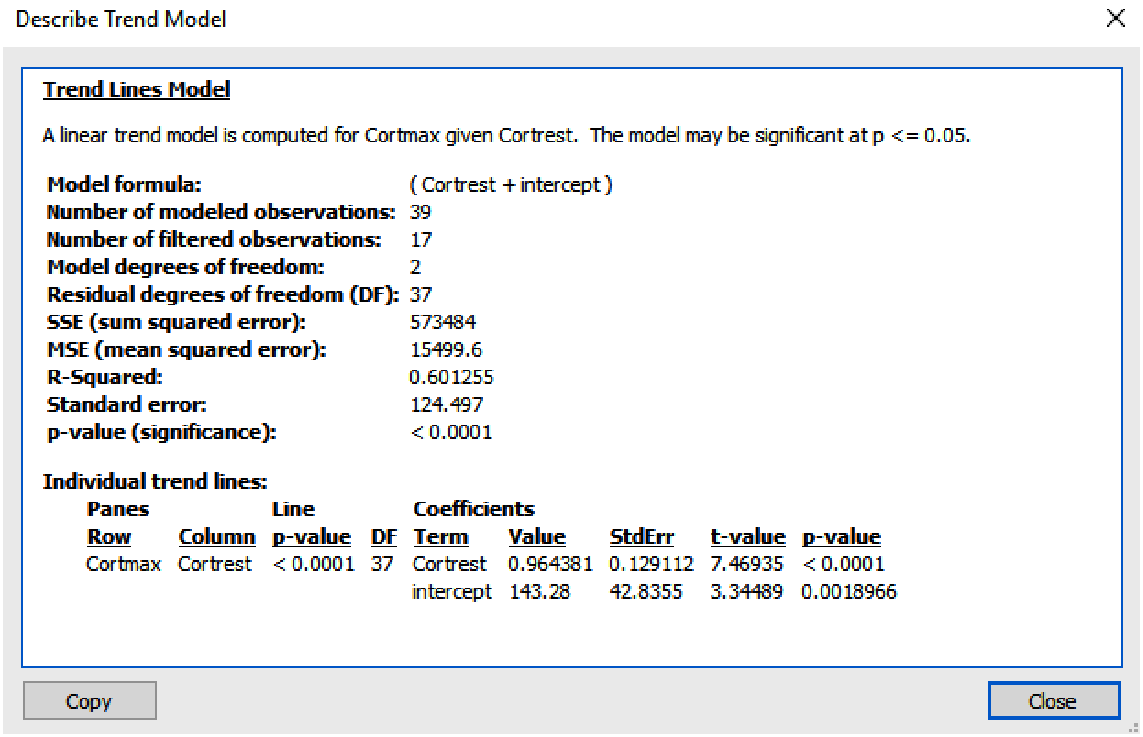
5. Once again, navigate to `Trend Lines` under `Analysis` in the main menu toolbar, and select `Describe Trend Model...`:



The following is the output:



In the following screenshot, we can see that we have successfully modeled the relationship between these two variables:



```
Trend Lines Model

A linear trend model is computed for Cortmax given Cortrest.  The model may be significant at p <= 0.05.

 Model formula:                     ( Cortrest + intercept )
 Number of modeled observations:    39
 Number of filtered observations:   17
 Model degrees of freedom:          2
 Residual degrees of freedom (DF):  37
 SSE (sum squared error):           573484
 MSE (mean squared error):          15499.6
 R-Squared:                         0.601255
 Standard error:                    124.497
 p-value (significance):            < 0.0001

Individual trend lines:
      Panes              Line           Coefficients
      Row      Column    p-value   DF   Term       Value      StdErr     t-value   p-value
      Cortmax  Cortrest  < 0.0001  37   Cortrest   0.964381   0.129112   7.46935   < 0.0001
                                        intercept  143.28     42.8355    3.34489   0.0018966
```

How it works...

We have created a regression model. Now, we are going to interpret the results, which tell us how successful our model is. But first, we need to get familiar with some basic statistics. In the broadest terms, the aim of each model is to represent real-life phenomena. All models differ in accuracy, or how well they depict reality. In statistics, we call the accuracy of the model its fit. The fit of a model is better if the difference between real data (that we measured) and predicted data (based on our model) is smaller.

We tried to predict cortisol level during maximum effort based on cortisol level during rest. Actual data points are represented with the circles, and the predictions that we made based on our model are vertically projected on the line (shown in the previous screenshot). As you can see, some circles lie almost on the line, some are above, and some are below the line. But the line is positioned so that these differences are minimized. If we want to estimate how good our model is, we should estimate the size of these differences. But, since the differences have both positive and negative values (some points are above and some are below the line), we can not just simply sum it up because they would cancel out. That's why we first need to square each difference and then sum it up. The result is [**sum squared error**] ([**SSE**])---that is a measure of our model's error, or how much it deviates from actual data. In order to estimate the goodness of our model's fit, we need to compare the size of that deviation with a benchmark. The most commonly used benchmark is the simple average of [*y*] value or the flat, horizontal line. Comparison of our model and baseline gives us R-squared. The bigger the R-squared is, the smaller the probability that we obtained it by chance. The conventionally accepted threshold of the probability is 0.05 and is denoted by the p-value (significance) in our output. If our p-value is smaller than the threshold, we can conclude that we have enough evidence to believe that our model is good enough. In our case, the p-value is much smaller than the mentioned threshold, so we can assume that the cortisol level during the maximum effort can be reasonably well predicted based on the cortisol level at rest. We can conclude that we have created a successful model of the relationship between these two variables. In the next lab, we are going to use it to create predictions.

There's more...

The regression that we presented in this example is linear because we assumed that the relationship between our variables was linear -- an assumption that turned out to be correct. However, other types of models are also available in Tableau. They can be accessed by navigating to `Analysis` | `Trend Lines` | `Edit Trend Lines...`. In the `Trend Lines Options` window, we can choose other models such as `Logarithmic`, `Exponential`, `Power`, or `Polynomial`. A good way to choose the model is to first plot the data and visually inspect it.

Forecasting on a dataset with outliers

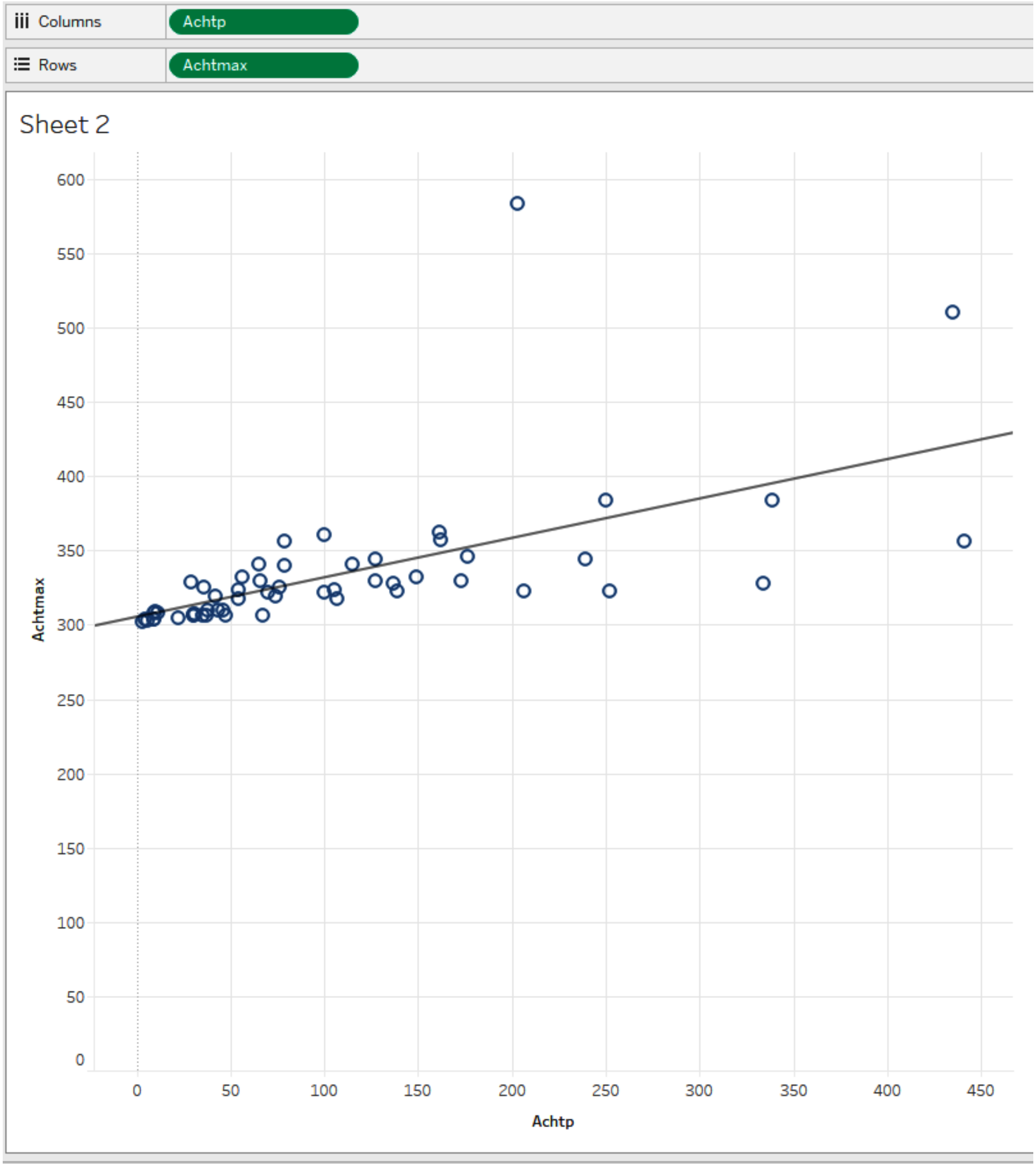
In this recipe, we are going to learn how to deal with outliers. Outliers are data points that are very unusual, atypical, and deviate from the trend present in the majority of the dataset. Outliers can be dangerous if not dealt with appropriately because they can significantly skew the results of an analysis. In this recipe, we will explore ways of detecting outliers in Tableau. We are going to perform a regression analysis and see how the regression line is affected by these cases.

Getting ready

For this recipe, we need the `hormonal_response_to_excercise.csv` dataset. We are going to use the `Achtp` and `Achtmax` variables. The `Achtp` variable is the level of adrenocorticotropic hormone at the beginning of the test, while `Achtmax` is the level of adrenocorticotropic hormone at the maximum effort during physical exertion.

How to do it...

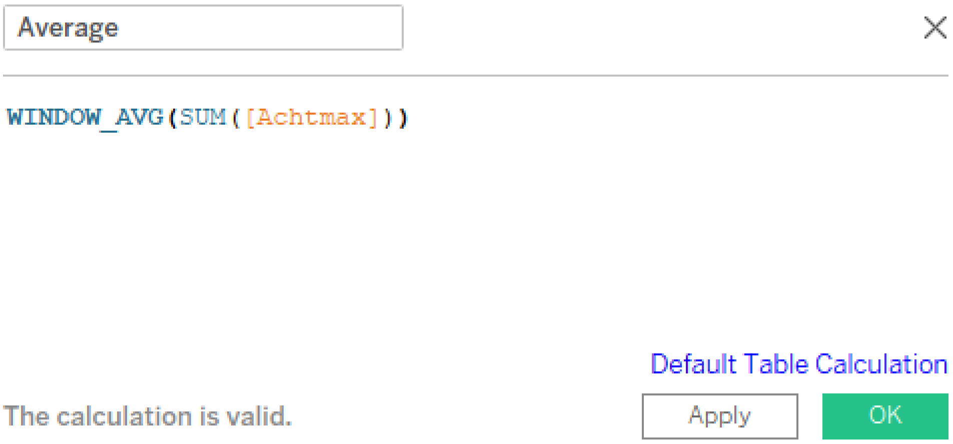
1. Drag and drop `Achtp` from `Measures` into the `Columns` shelf.
2. Drag and drop `Achtmax` from `Measures` into the `Rows` shelf.
3. In the main menu toolbar, in the `Analysis` drop-down menu, deselect `Aggregate Measures`.
4. In the main menu toolbar, in the `Analysis` drop-down menu, navigate to `Trend Lines` | `Show Trend Lines`:



5. Rename the sheet to `Outliers included`.
6. In the main menu navigate to **`Analysis`** | **`Create Calculated Field...`**.
7. Rename the calculated field from **`Calculation 1`** to `Average`, and in the formula space, type the following expression:

```
WINDOW_AVG(SUM([Achtmax]))
```

The calculation field shows the preceding expression in the following screenshot:



Average

WINDOW_AVG(SUM([Achtmax]))

The calculation is valid.

8. Click on `OK` to save and exit the calculated field editor window.
9. Repeat [*step 7*] to create another calculated field. Name the field `Lower` and in the formula space type the following expression:

```
 [Average] - 2.5*WINDOW_STDEV(SUM([Achtmax]))
```

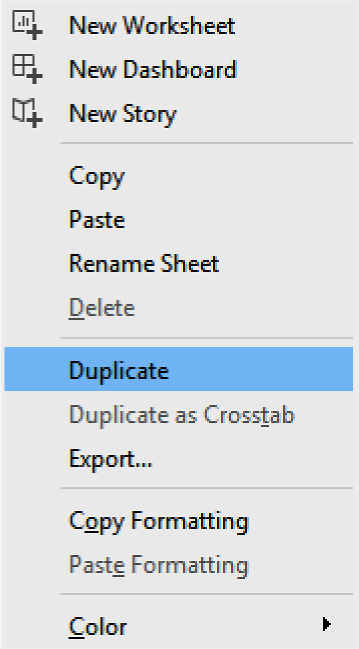
10. Save and exit by clicking `OK`.
11. Repeat [*step 7*] to create yet another calculated field. Name the field `Upper` and in the formula space, type the following expression:

```
[Average] + 2.5*WINDOW_STDEV(SUM([Achtmax]))
```

12. Save and exit by clicking on `OK`.
13. Repeat [*step 7*] one last time to create our final calculated field. Name this field `Outliers` and in the formula space, type the following expression:

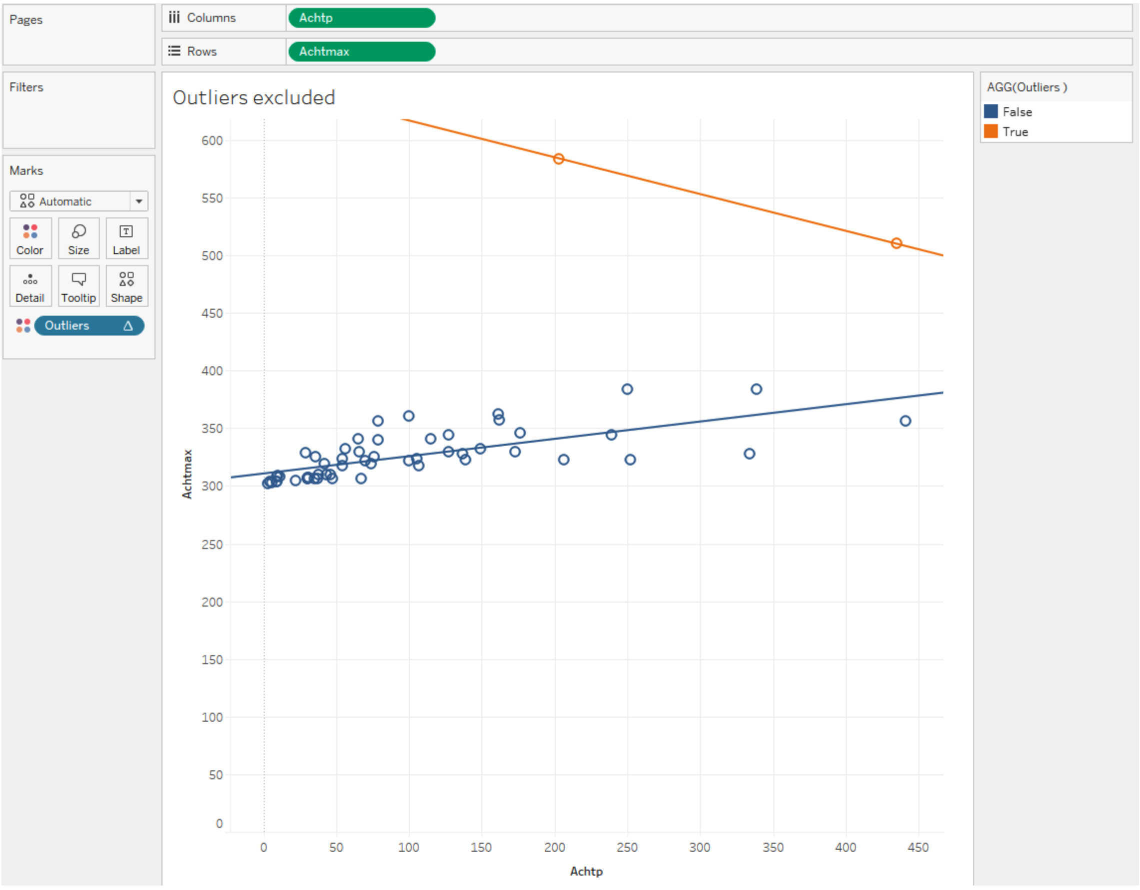
```
SUM([Achtmax])> [Upper] or SUM([Achtmax]) < [Lower]
```

14. Save and exit by clicking on `OK`.
15. Right-click on the **`Outliers included`** sheet tab at the bottom of the workspace and select **`Duplicate`** as seen in the following screenshot:

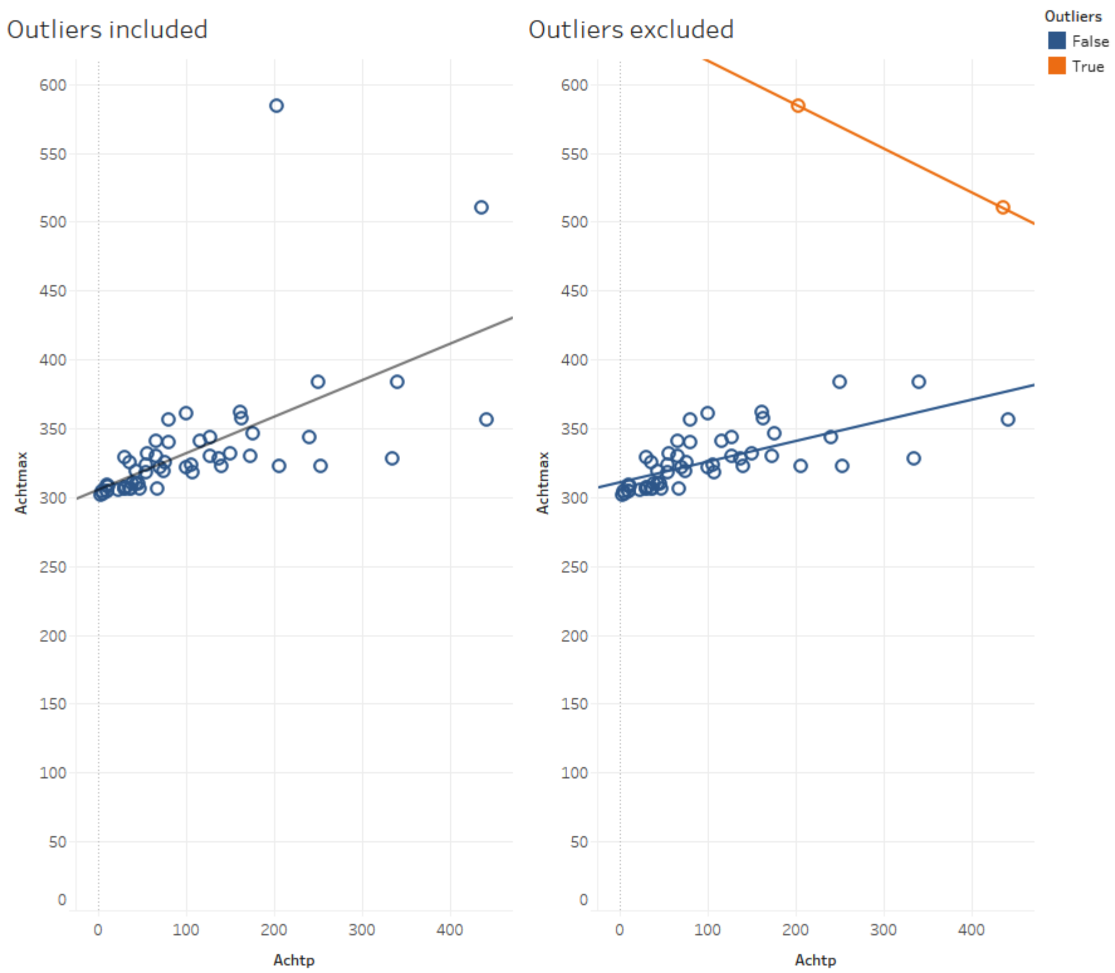


16. This will create an identical sheet named `Outliers included (2)`. Rename this sheet `Outliers excluded`.

17. Drag and drop `Outliers` from `Measures` to `Color` in the `Marks` card:



18. In the main menu toolbar, navigate to `Dashboard` | `New Dashboard`.
19. Drag and drop `Outliers included` sheet from the `Sheets` section of the `Dashboard` pane into the canvas.
20. Drag and drop the `Outliers excluded` sheet from the `Sheets` section of the `Dashboard` pane into the canvas, to the right of the `Outliers included` sheet in the chart:



How it works...

In this recipe, we learned how to detect outliers. Outliers are extreme values that stand out from the other values in the sample. In order to detect outliers, we relied on a commonly used conventional rule---outliers are all values that deviate from the mean more than +/-2.5 standard deviations, which excludes around 1% of our sample.

In this example, we are able to see how outliers can influence statistical models. We can see that the model that includes outliers has a much steeper slope than the model that excludes outliers, meaning that they have pulled[** **]our linear model away from the majority of data, giving us skewed results. When interpreting results, we have to pay special attention to this (so-called leverage) effect. Otherwise, we risk declaring a statistical effect significant even when it does not actually exist.

Using R within Tableau

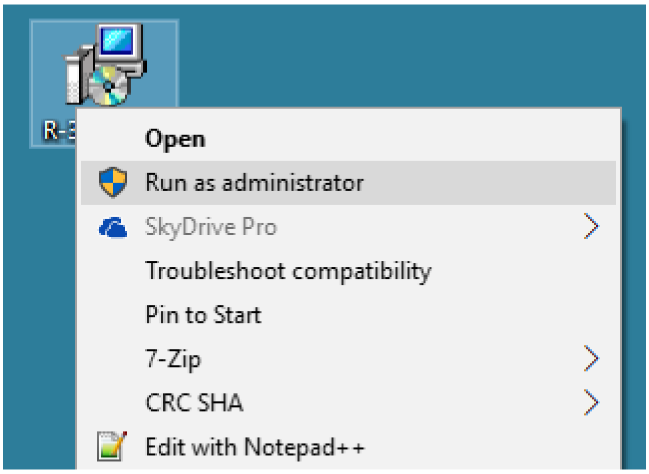
The main goal of this recipe is to demonstrate how R can be used in conjunction with Tableau. R is a popular statistical language that can be used to perform sophisticated statistical analysis and predictive analytics. R is open source and free. It is supported by a community of contributors who continually create new packages.

Getting ready

Make sure that you have installed the most recent version of R on your computer. If not, go to https://cran.r-project.org/ and download the latest version adequate for your operating system and follow the instructions for the installation. The default settings are just fine.

Note

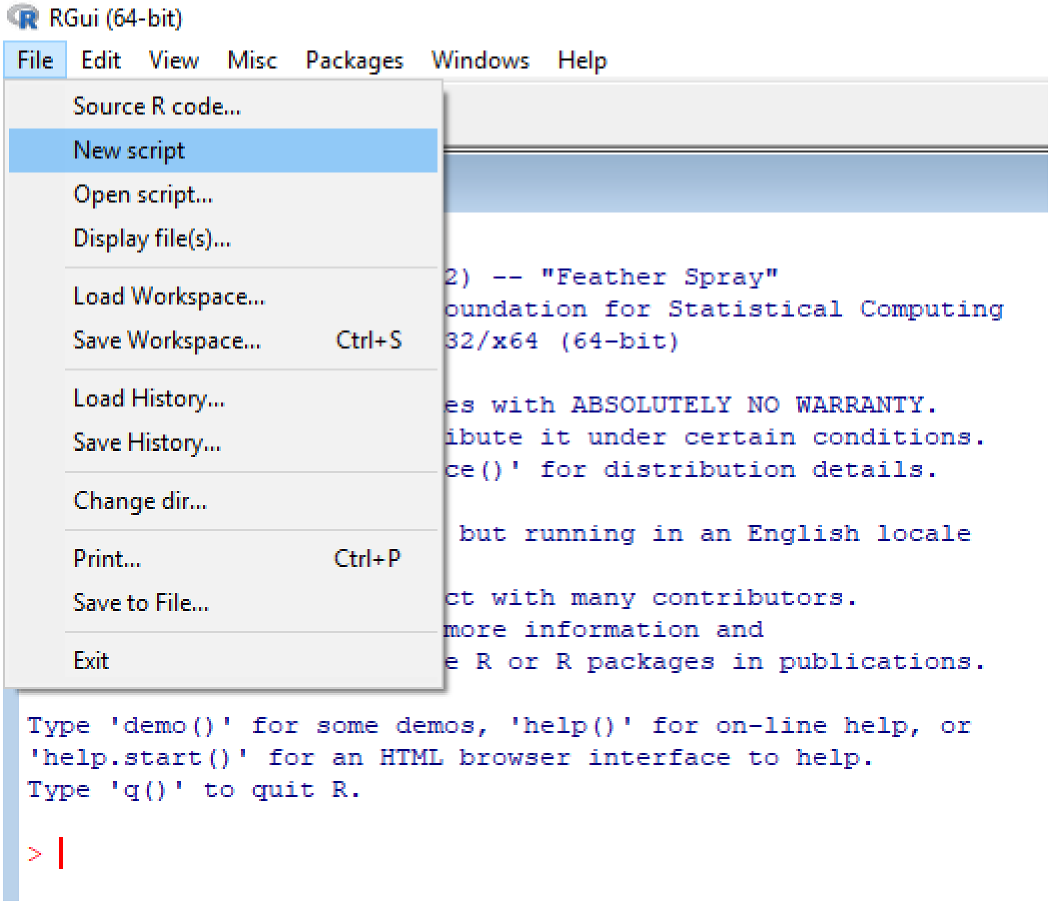
When installing R, it is important to run it as administrator (right-click on the `Setup` icon). Otherwise, you may encounter problems with the access to folders and having permissions to conduct specific tasks. Take a look at the following screenshot for better understanding:



How to do it...

1. a. Click on the icon

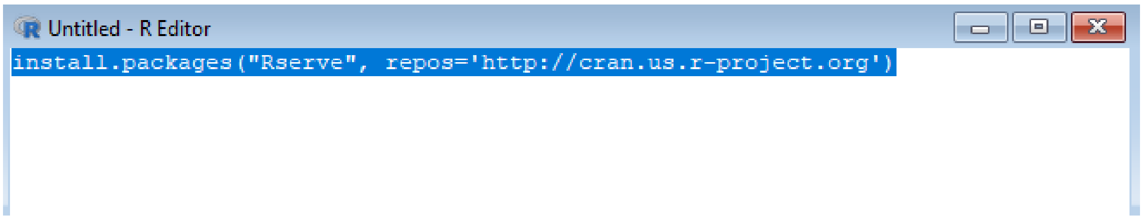
```
![]
(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAACcAAAAmCAYAAABH/4KQAAAAAXNSR0IArs

         at your desktop and run R as shown in the following screenshot:
```



```
2) -- "Feather Spray"
oundation for Statistical Computing
32/x64 (64-bit)

es with ABSOLUTELY NO WARRANTY.
ibute it under certain conditions.
ce()' for distribution details.

 but running in an English locale

ct with many contributors.
more information and
e R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

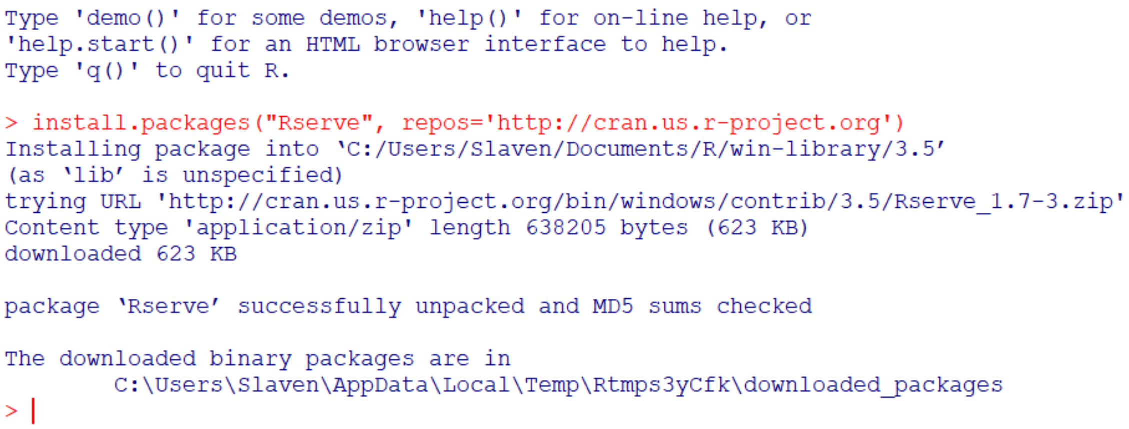
2. When the script is opened, type the following code `install.packages("Rserve", repos='http://cran.us.r-project.org')`, select it and click on

```
![]
(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAB4AAAAgCAYAAAAFQMh/AAAAAXNSR0IArs4c6QAAA

    or press [*Ctrl*] + [*R*]:
```



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

3. When the installation is completed you will receive the message that the `Rserve` package is successfully unpacked:



```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages("Rserve", repos='http://cran.us.r-project.org')
Installing package into 'C:/Users/Slaven/Documents/R/win-library/3.5'
(as 'lib' is unspecified)
trying URL 'http://cran.us.r-project.org/bin/windows/contrib/3.5/Rserve_1.7-3.zip'
Content type 'application/zip' length 638205 bytes (623 KB)
downloaded 623 KB

package 'Rserve' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Slaven\AppData\Local\Temp\Rtmps3yCfk\downloaded_packages
> |
```

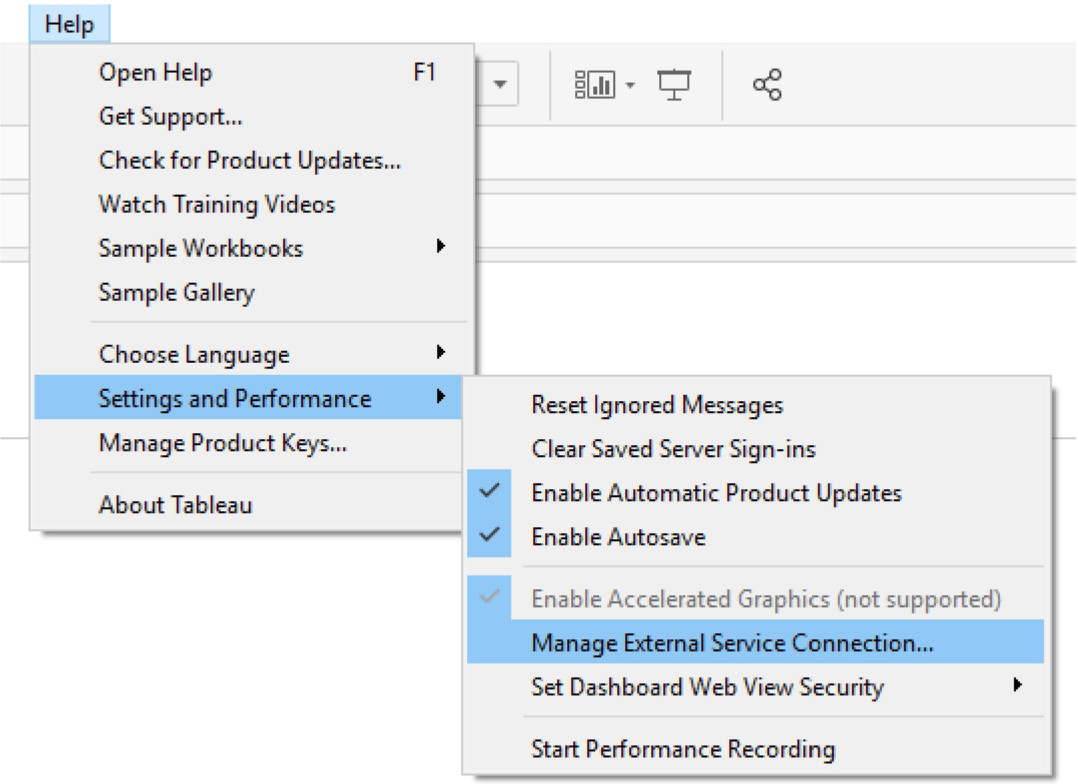
4. Get back to script and type the following code:

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

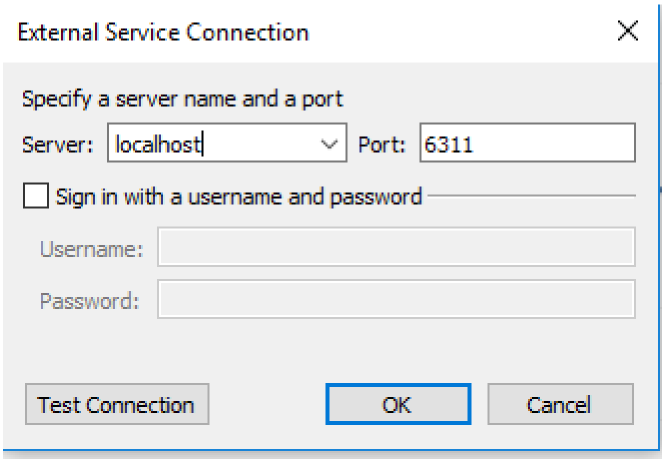
5. Select the code and click on

```
or press [*Ctrl*] + [*R*].
```

6. Open Tableau and in the main menu toolbar navigate to `Help` | `Settings and Performance` | `Manage External``Service Connection...` as shown in the following screenshot:


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 (not supported)
Manage External Service Connection...
Set Dashboard Web View Security
Start Performance Recording

7. In the `External Service Connection` dialog box, from the `Server` drop-down menu, choose `localhost.` The ``Port field should contain the value of `6311.` Click on the `Test Connection` button:



External Service Connection

Specify a server name and a port

Server: localhost Port: 6311

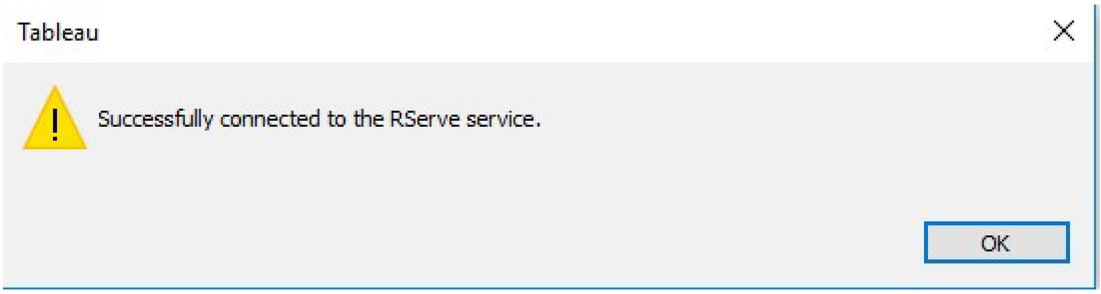
□ Sign in with a username and password

Username:

Password:

Test Connection OK Cancel

8. If you went through all of the steps outlined above you will get the message that the connection between Tableau and R is successful:



Tableau

Successfully connected to the RServe service.

9. Click on `OK` to exit the `External Service Connection` dialog box.

How it works...

In this recipe, we installed the R software on our computer. Also, we installed the `Rserve` package, loaded its library, and initialized it. Essentially, `Rserve` is a connector between R and Tableau Desktop.

After that, we configured and tested the `Rserve` connection. When all of this was done, we were ready to write R syntax in Tableau calculated field. We have the following four different scripts at our disposal:

- `SCRIPT_REAL` : This script returns real numbers
- `SCRIPT_INT` : This script returns integers
- `SCRIPT_STR` : This returns strings
- `SCRIPT_BOOL` : This script returns Boolean Within this script, we are allowed to write regular R syntax and the result of the calculation will be saved in a calculated field. This calculated field can be further used in the same manner as any other calculated field in Tableau.

Note

In order to use R functionality in a workbook, the reader needs to have an R Tableau connection. This holds for locally shared workbooks and for workbooks published on Tableau Server. As of 2019, Tableau supports `RSserve` connections, which means secure `Rserve` is being hosted remotely from Tableau Server, and the data is protected in transit.

There's more...

- A concrete example of performing analysis that requires writing R syntax will be covered in the next recipes, [*Forecasting based on multiple regression] and [*Regression with random forest], and other recipes in *Advanced Analytics with Tableau*. However, any of these chapters do not have the ambition to serve as an introduction in R language. Because of that, readers are encouraged to learn R in parallel by referring to the following links:
- https://www.statmethods.net/index.html
- https://www.r-bloggers.com/

However, readers without any previous experience in R should be able to perform all the recipes.

Forecasting based on multiple regression

In the first recipe of this lab, [*Basic forecasting and statistical inference*], we learned how to perform forecasting with simple linear regression. In this recipe, we will learn how to perform forecasting based on multiple regression. Multiple regression is a type of forecasting procedure in which we use more than one variable to predict the outcome variable that we are interested in. In this recipe, our goal is to predict the level of cortisol at the highest effort during the physical exercise, based on cortisol level at rest and cortisol level at the beginning of the test. In the dataset that we are going to use, we have some respondents with missing data for cortisol level during physical exertion. Our aim is to use the result of our regression analysis to approximate cortisol level for those respondents and use these predicted values for further analysis. For this recipe, we need to employ R functionality in Tableau.

Getting ready

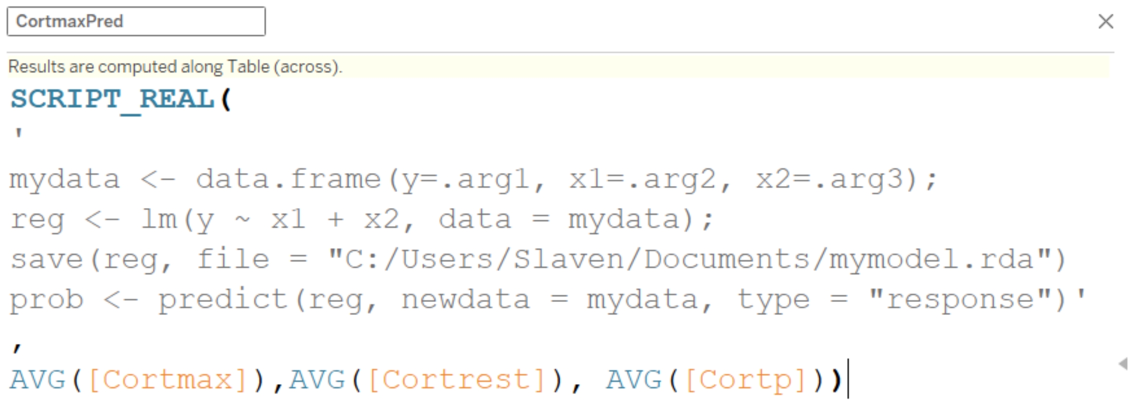
For this recipe, we will need the `hormonal_response_to_excercise.csv` dataset. Variables that we are going to use are `Cortmax` (the level of cortisol at the maximum level of physical excretion), `Cortrest` (cortisol level at rest), `Cortp` (the level of cortisol at the beginning of the test), and `Achtmax` (the level of adrenocorticotropic hormone at the beginning of the test). Before the start, make sure that R is installed on your machine, that `Rserve` is installed, loaded and initialized, and that connection between R and Tableau is configured. For detailed instruction on this, please refer to the [*Using R within Tableau*] recipe.

How to do it...

1. In the main menu toolbar, navigate to **`Analysis`** | **`Create Calculated Field...`** and type the following code:

```
SCRIPT_REAL('
mydata <- data.frame(y=.arg1, x1=.arg2, x2=.arg3);
reg <- lm(y ~ x1 + x2, data = mydata);
save(reg, file = "C:/Users/Slaven/Documents/mymodel.rda")
prob <- predict(reg, newdata = mydata, type = "response")'
,
AVG([Cortmax]),AVG([Cortrest]), AVG([Cortp]))
```

2. Name the field `CortmaxPred` as shown in the following screenshot:

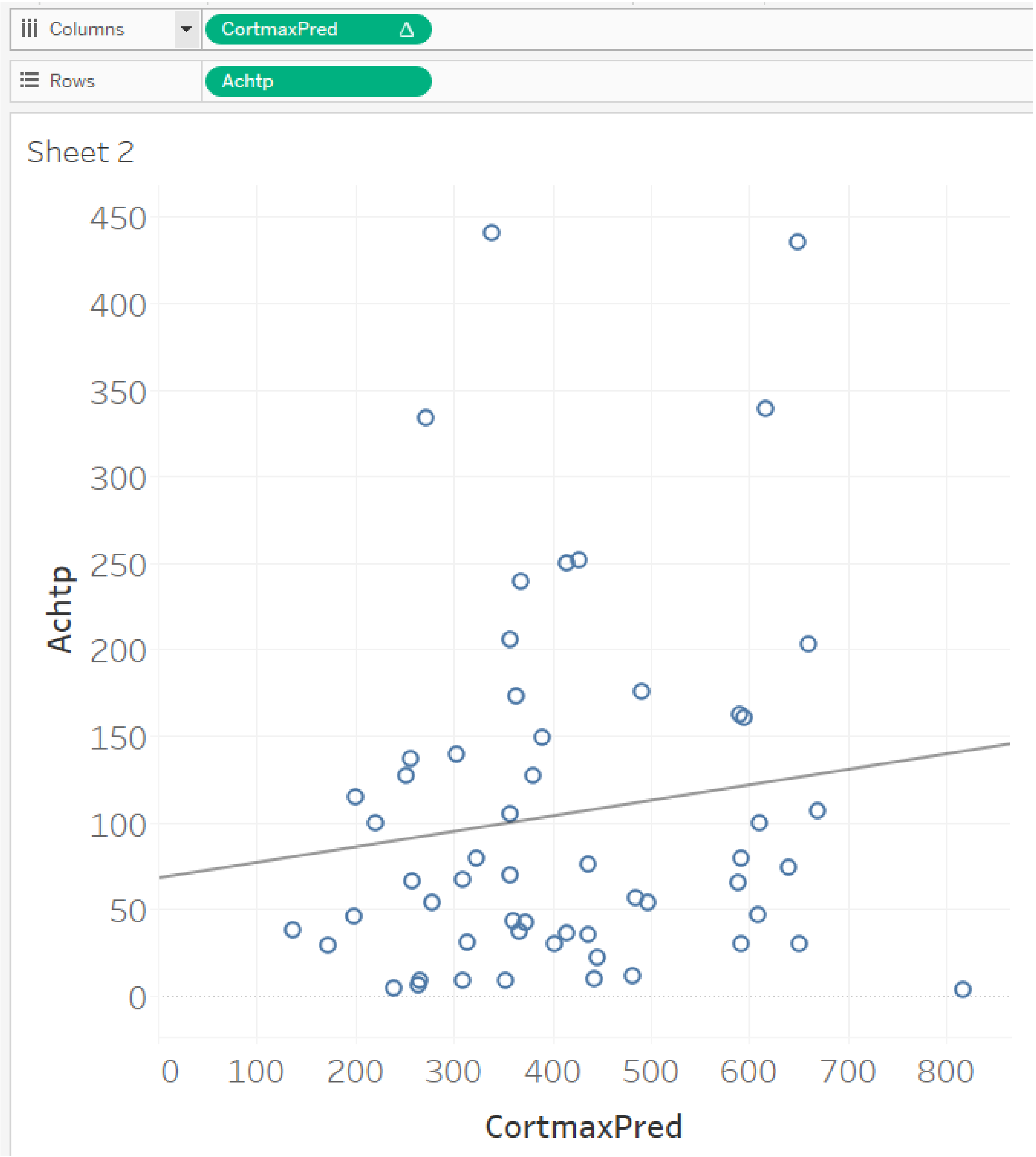


CortmaxPred

Results are computed along Table (across).

```
SCRIPT_REAL(
'
mydata <- data.frame(y=.arg1, x1=.arg2, x2=.arg3);
reg <- lm(y ~ x1 + x2, data = mydata);
save(reg, file = "C:/Users/Slaven/Documents/mymodel.rda")
prob <- predict(reg, newdata = mydata, type = "response")'
,
AVG([Cortmax]),AVG([Cortrest]), AVG([Cortp]))
```

3. Drag and drop **`CortmaxPred`** from **`Measures`** to the **`Columns`** shelf.
4. Drag and drop **`Achtp`** from **`Measures`** to the **`Rows`** shelf.
5. In the main menu, go to **`Analysis`** and deselect **`Aggregate Measures`**.
6. In the same menu, choose **`Trend Lines`** and select **`Show`** **`` ``Trend Lines ``** and you will see the following output:



How it works...

In this recipe, we learned how to utilize R facility within Tableau. If you have some experience with R, you might notice that the syntax looks pretty standard with an exception at the beginning and at the end. Actually, the R syntax is wrapped, so that Tableau can be used to recognize it.

Also, it is important to notice that we need to assign arguments that are going to be used. `.arg1` represent the first variable from the left-hand side mentioned in the bottom line of the syntax (in our case, it is `Cortmax`), the `.arg2` variable represent the second one (`Cortrest`), and so on.

There's more...

In the syntax for this recipe, you might have noticed the following line of code:

```
save(reg, file = "C:/Users/Slaven/Documents/mymodel.rda")
```

This line created a `.rda` file on your hard drive, witch contains our regression model. Once you have saved the model, you can reuse it with another data set. Let's say we receive a new data set with data which contains information about cortisol level at the beginning of the test and at rest, but no information on the cortisol level at the point of maximum exertion. Thanks to our model, which predicts the level of cortisol at the point of maximum exertion, we can estimate its value for the new subjects. We just need to load our model by creating a calculated field with the following line of code:

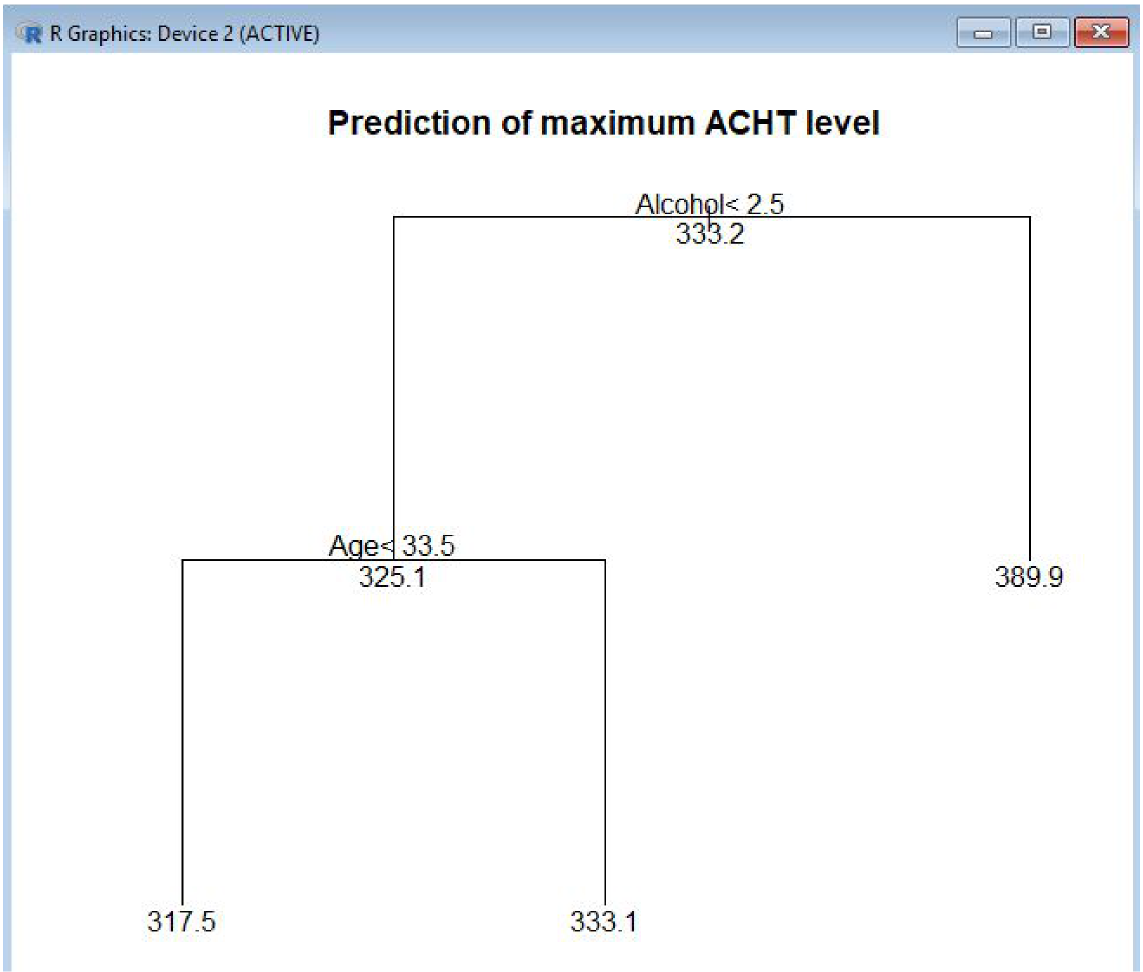
```
SCRIPT_REAL('
mydata <- data.frame(y=.arg1, x1=.arg2, x2=.arg3);
load ("C:/Users/Slaven/Documents/mymodel.rda")
prob <- predict(reg, newdata = mydata, type = "response")'
,
AVG([Cortmax]),AVG([Cortrest]), AVG([Cortp]))
```

The new calculated field will contain the predicted values of `Cortmax`. However, keep in mind that the new data set has to contain a field named the same as the variable we were predicting in the original data set (in this case, `Cortmax`) for the script to work, even if it is completely empty. Also, make sure that all the other fields are also named exactly the same as in the original data set, where the model was created.

Regression with random forest

In the previous recipe, [*Forecasting based on multiple regression*], we learned how to use multiple variables in order to predict the variable that we are interested in. Sometimes, we have a lot of variables and we are not sure which ones we should choose as predictors. Also, predictor variables can be related among themselves in different ways, which complicates the setup of the model and the interpretation of the results. In recent years, random forest algorithm has gained popularity among analysts and data scientists, as they provide a solution to these problems. The random forest algorithm is based on decision tree approach. This approach can be used to predict both discrete class membership (classification) and exact values of a continuous variable (regression). In this recipe, we will cover the latter. Regression-based on decision tree works by iteratively splitting cases in the dataset into increasingly homogeneous groups. Looping through all variables the algorithm searches for the one that splits cases into the groups. so that cases within each group are as similar as possible with regards to the predicted variable. This process continues, resulting in the three with more and more branches and data that is partitioned into smaller and smaller subsamples. Random forest is an enhanced version of the decision tree algorithm that builds many decision threes using randomly selected subsamples of both variables and cases.

Results obtained by each of the trees are compiled into a final single solution. Visualized in R, a random forest model looks like this:


R Graphics: Device 2 (ACTIVE)
Prediction of maximum ACHT level
Alcohol< 2.5
333.2
Age< 33.5
325.1
389.9
317.5
333.1

Note

It is not possible to create a tree chart like the one shown above in Tableau --- it is possible to create it directly in R though. You can learn how in the next lab,

> *Advanced Analytics with Tableau.*

We will use the `hormonal_response_to_excercise.csv` dataset. Our main task will be to predict the level of adrenocorticotropic hormone at the maximum level of physical exertion. In order to take into consideration different factors that can influence an ACHTspike during exercise, we are going to include the following variables in our model: cortisol level at rest, alcohol and tobacco consumption, height, age, and weight. This variable can be also interrelated: for example, smoking and drinking, height, and weight. Baseline cortisol may be also related to tobacco and alcohol consumption. For this reason, the random forest can make our life easier.

Getting ready

In order to perform this recipe make sure that you installed R, and that you have activated the `Rserve` package and connect it to Tableau (for detailed instruction see the recipe, [*Using R within Tableau*]). You will also need to connect to `hormonal_response_to_excercise.csv` and open a new blank worksheet.

How to do it...

1. Launch R, open a new script and type `install.packages("rpart", repos='http://cran.us.r-project.org')`.

2. Select the text, and click on

   ```
   or press [*Ctrl*] + [*R*].
   ```

3. When the installation process is completed, load the package by typing the following line:

```
library(rpart)
```

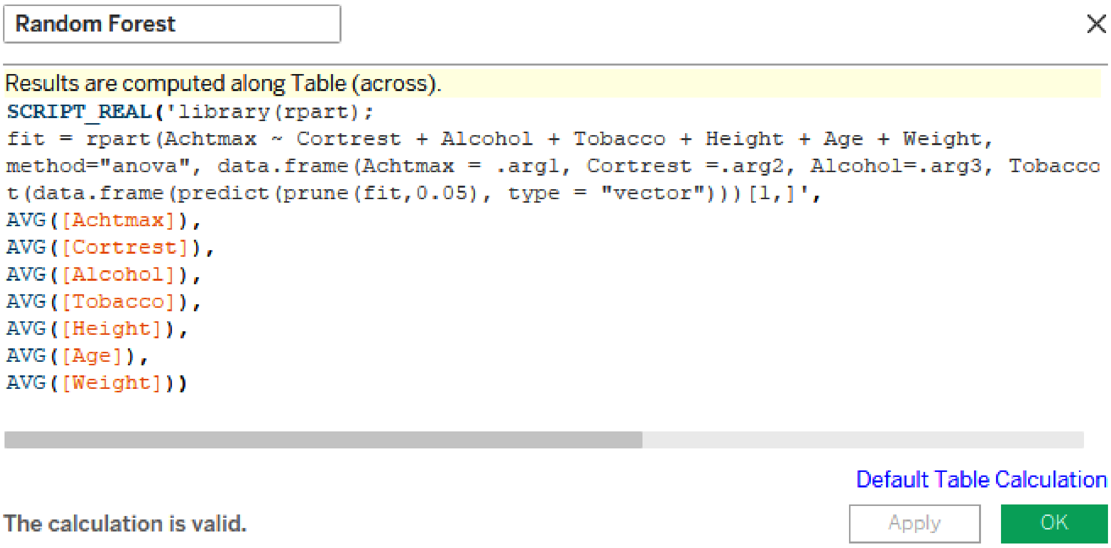
4. Select the text, and click on

   ```
   or press [*Ctrl*] + [*R*].
   ```

5. In Tableau, in the main menu toolbar, click on `Analysis` and then click on `Create Calculated Field...`.
6. Rename the field from `Calculation 1` to `Random Forest` and type the following expression into the formula space:

```
SCRIPT_REAL('library(rpart);
fit = rpart(Achtmax ~ Cortrest + Alcohol + Tobacco + Height + Age + Weight,
method="anova", data.frame(Achtmax = .arg1, Cortrest =.arg2, Alcohol=.arg3, Tobacco
=.arg4, Height =.arg5, Age =.arg6, Weight=.arg7));
t(data.frame(predict(prune(fit,0.05), type = "vector")))[1,]',
AVG([Achtmax]),
AVG([Cortrest]),
AVG([Alcohol]),
AVG([Tobacco]),
AVG([Height]),
AVG([Age]),
AVG([Weight]))
```

The preceding code is shown in the following screenshot:

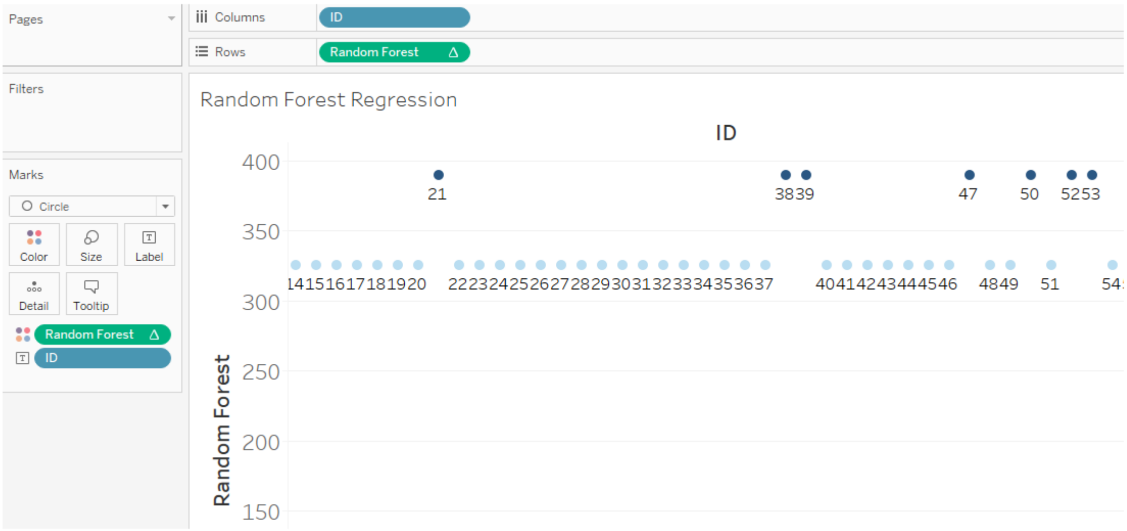


Results are computed along Table (across).

```
SCRIPT_REAL('library(rpart);
fit = rpart(Achtmax ~ Cortrest + Alcohol + Tobacco + Height + Age + Weight,
method="anova", data.frame(Achtmax = .arg1, Cortrest =.arg2, Alcohol=.arg3, Tobacco
t(data.frame(predict(prune(fit,0.05), type = "vector")))[1,]',
AVG([Achtmax]),
AVG([Cortrest]),
AVG([Alcohol]),
AVG([Tobacco]),
AVG([Height]),
AVG([Age]),
AVG([Weight]))
```

The calculation is valid.

7. Click **OK** to exit the editor window and save the calculated field.
8. Drag and drop **Random Forest** from **Measures** into the **Rows** shelf.
9. Drag and drop the **ID** variable from **Dimensions** into the **Columns** shelf.
10. In the **Marks** card, change the mark type from **Automatic** to **Circle** using the drop-down menu.
11. Drag and drop **ID** from **Dimensions** to **Label** in the **Marks** card.
12. Drag and drop **Random Forest** from **Measures** to **Color** in the **Marks** card.
13. Rename the sheet to **Random Forest Regression**:



How it works...

In this recipe, we used the random forest algorithm to predict the level of ACHT during exercise. The subjects with high values of ACHT (389.86) are represented with dark blue circles at the top of the chart, while the rest of the respondents (with the average value of 325.14) are at the bottom of the chart (light blue cycles).

There's more...

In this recipe, we have created a model, and we can now save it and apply it to another dataset (for detailed explanation and instructions, see the recipe

> *Forecasting based on multiple regression). You might also want to try further evaluation the model by splitting the data into a training and a test set. It is a common practice in data modeling to use a part of the data set as the training set, on which the model is built (fitted). The model is then evaluated on the test set. If the fit that is observed for the training set is preserved in the test set, it is an indication that we did a good job. Otherwise, we need to reconsider our model.*

Time series forecasting

Tableau has excellent capabilities for dealing with time series data. One of them is time series forecasting -- extrapolating values for points in time that are outside our dataset, based on the time points in our dataset for which we have the recorded values.

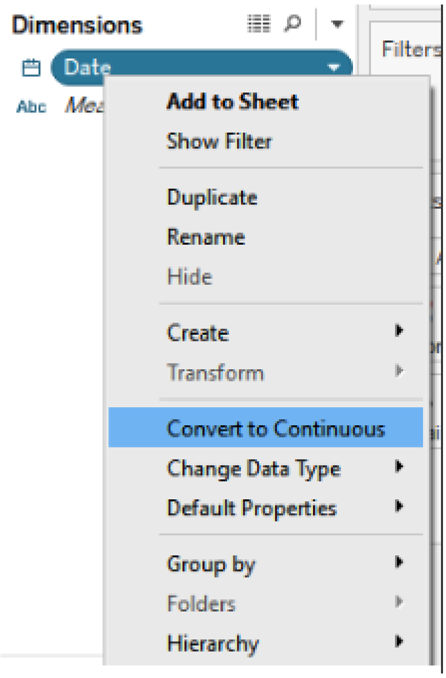
In this recipe, we will be using stock market prices of a soft drink company shares, stored in the `Stock_prices.csv` dataset.

Getting ready

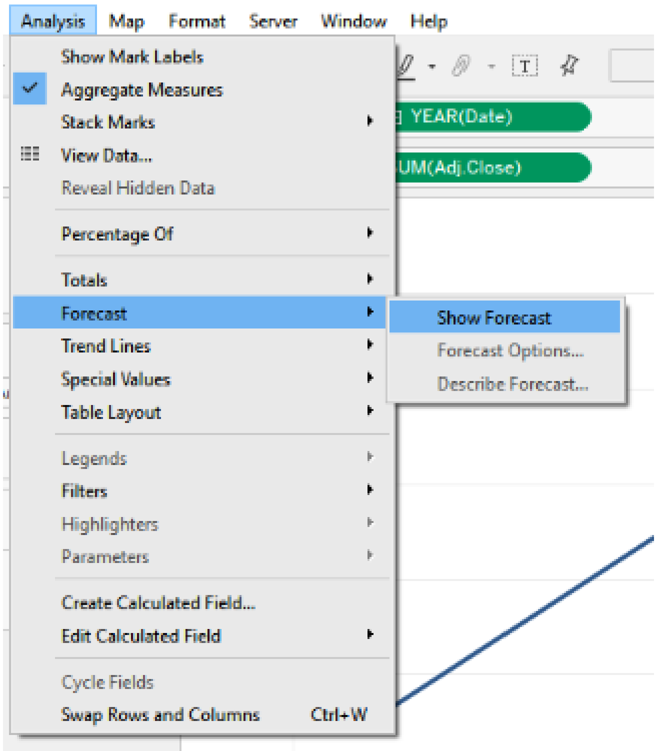
To perform the steps in this recipe, make sure you are connected to the `Stock_prices.csv` dataset and open a new blank worksheet.

How to do it...

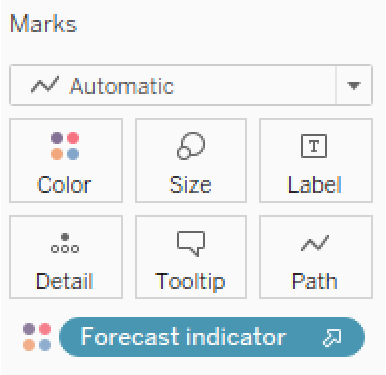
1. Right-click on the `Date` field under `Dimensions` and from the drop-down menu select `Convert to Continuous` :



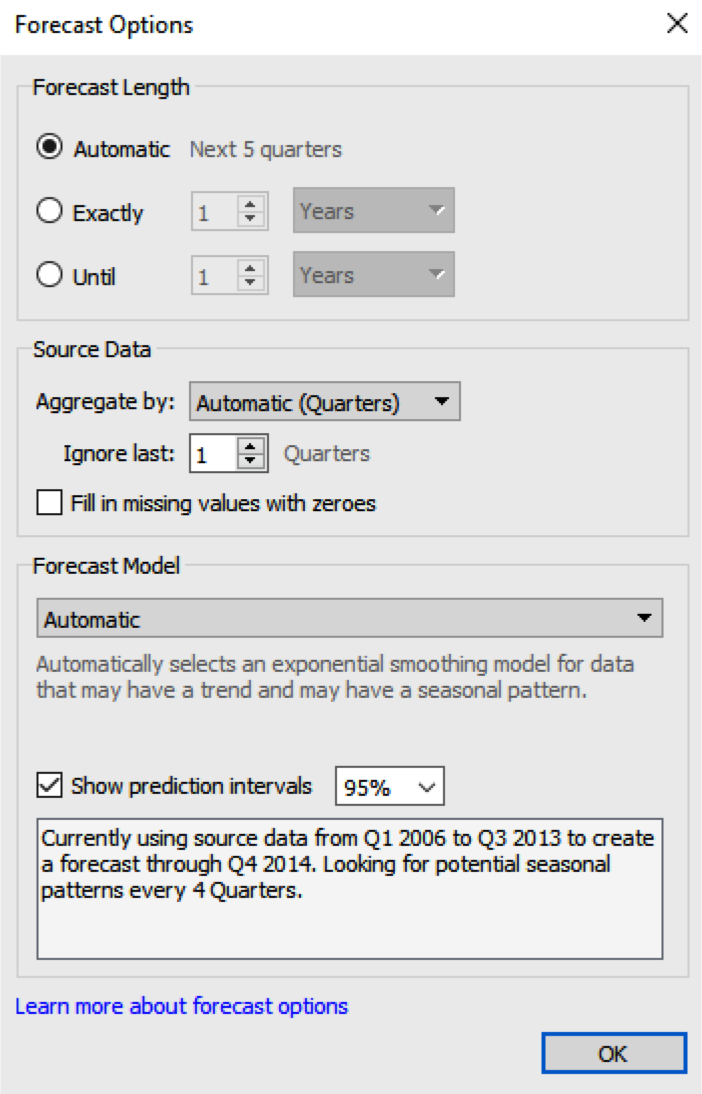
2. Drag and drop `Date` from `Dimensions` into the `Columns` shelf.
3. Drag and drop `Adj.Close` from `Measures` into the `Rows` shelf.
4. In the main menu toolbar, click on `Analysis` , navigate to `Forecast` and select `Show Forecast` :



5. Tableau will automatically place the `Forecast indicator` field in `Color` in the ```Mar``ks``` card. However, we can remove it from there:



6. Once again, navigate to `Analysis` in the main menu toolbar and under `Forecast` select `Forecast Options...` .
7. `Forecast Length` is automatically set to `Next 5 quarters` , but we can change it by selecting `Exactly` or `Until` and selecting the desired period. Try changing the `Forecast Length` period and notice how the shaded area around the line expands as you increase the `Forecast Length` . However, let's leave it at `5` quarters:



Forecast Options

Forecast Length

● Automatic Next 5 quarters

○ Exactly 1 Years

○ Until 1 Years

Source Data

Aggregate by: Automatic (Quarters)

Ignore last: 1 Quarters

□ Fill in missing values with zeroes

Forecast Model

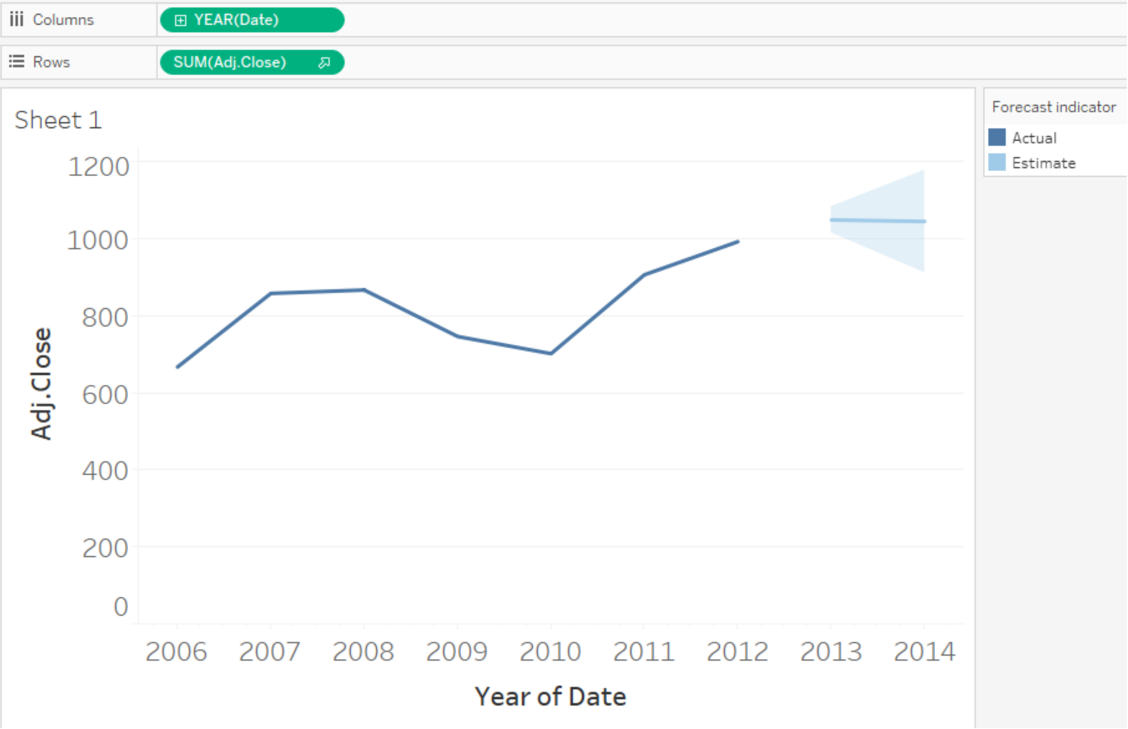
Automatic

Automatically selects an exponential smoothing model for data that may have a trend and may have a seasonal pattern.

☑ Show prediction intervals 95%

Currently using source data from Q1 2006 to Q3 2013 to create a forecast through Q4 2014. Looking for potential seasonal patterns every 4 Quarters.

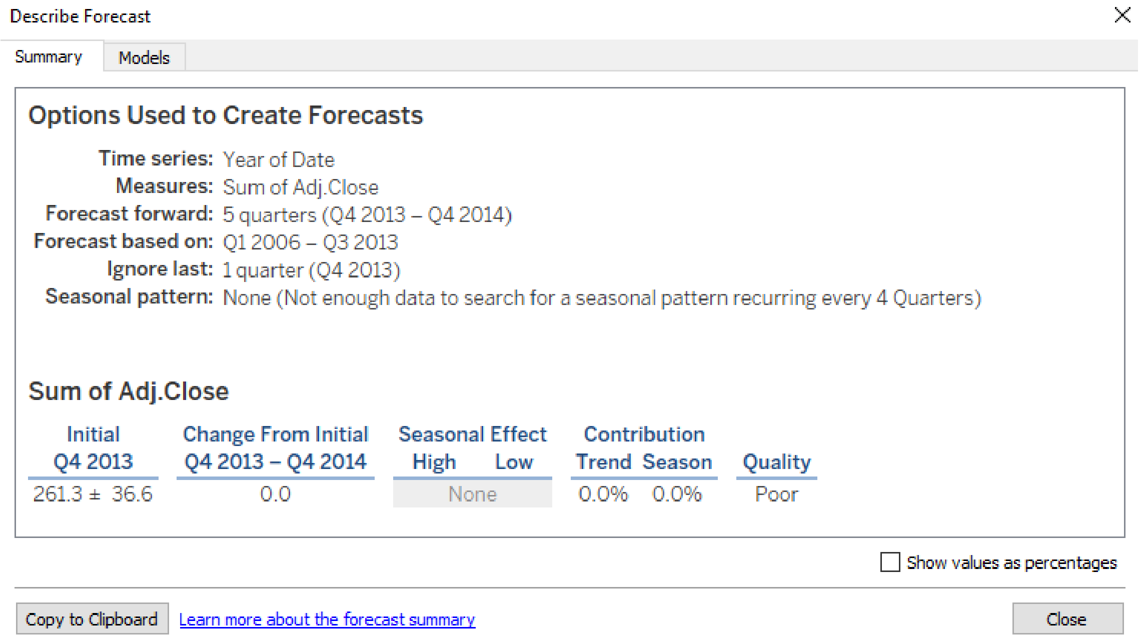
8. Click `OK` to exit the Forecast``Options window:



Columns: YEAR(Date)

Rows: SUM(Adj.Close)

9. Finally, under `Analysis` in the main menu toolbar, navigate to `Forecast` and select `Describe Forecast` to see the performance of the forecast we have created:



Options Used to Create Forecasts

Time series: Year of Date
Measures: Sum of Adj.Close
Forecast forward: 5 quarters (Q4 2013 – Q4 2014)
Forecast based on: Q1 2006 – Q3 2013
Ignore last: 1 quarter (Q4 2013)
Seasonal pattern: None (Not enough data to search for a seasonal pattern recurring every 4 Quarters)

Sum of Adj.Close

Initial	Change From Initial	Seasonal Effect		Contribution		
Q4 2013	Q4 2013 – Q4 2014	High	Low	Trend	Season	Quality
261.3 ± 36.6	0.0	None		0.0%	0.0%	Poor

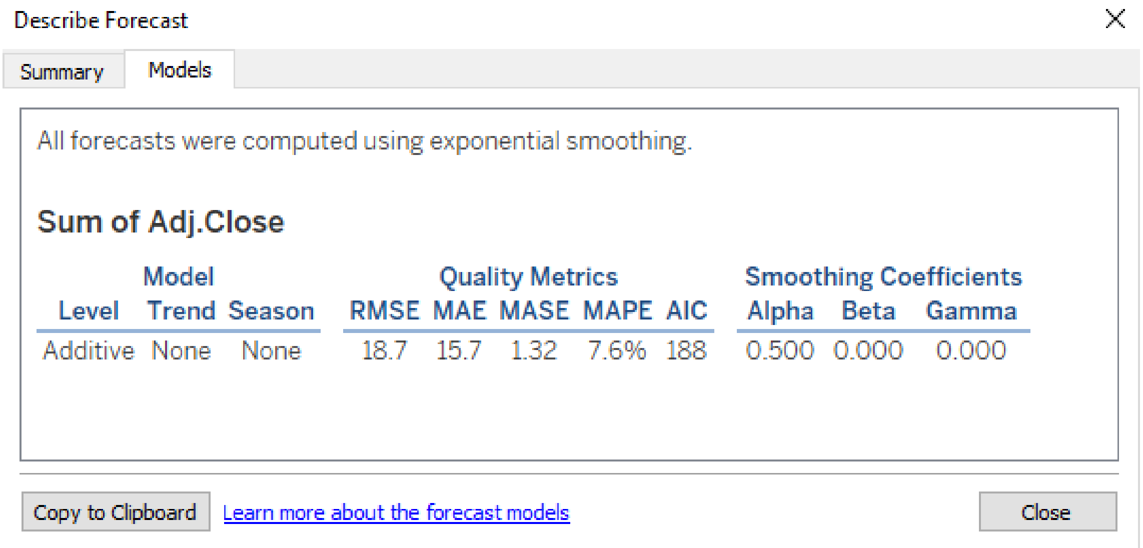
☐ Show values as percentages

How it works...

We have extrapolated the closing prices of the company's shares based on the prices we have recorded in our dataset. Let's explore the characteristics of the model we have created, by inspecting the content of the `Describe Forecast` window.

In the `Summary` tab, we can see some basic information describing our forecast. The most important of these is quality---it describes how well the forecast fits our actual data and can take values of `Good`, `Ok`, and `Poor`.

In the `Models` tab, we can see the role of the `Level`, `Trend`, and `Season` components of our model. We can also see the quality metrics that provide information about the statistical quality of our model---[**root mean squared error**] ([**RMSE**]), **mean absolute error **, [**mean absolute percentage error**] ([**MAPE**]), and [**Aikake Information Criterion**] ([**AIC**]). Finally, in the `Models` tab, we can get the information about smoothing coefficients that were used to weight the data points according to their how recent they are, so that forecast errors are minimized as shown in the following screenshot:



All forecasts were computed using exponential smoothing.

Sum of Adj.Close

Model			Quality Metrics					Smoothing Coefficients		
Level	Trend	Season	RMSE	MAE	MASE	MAPE	AIC	Alpha	Beta	Gamma
Additive	None	None	18.7	15.7	1.32	7.6%	188	0.500	0.000	0.000

Tableau allows us to choose the time period for which we wish to predict the values. We have chosen 5 quarters, but we can increase the period further. However, note that the precision of our prediction becomes smaller as our forecasting length period becomes longer. This is reflected by the prediction interval (the shaded area around the line), which becomes larger as we make our prediction period becomes longer.

There's more...

In the `Forecast Options` window, Tableau provides us with various option for customizing our forecast. We can choose how to aggregate our data under `Source Data`. By default, it is aggregated by `Quarters`, but in the drop-down menu, we can choose to aggregate across different time periods, such as `Year` or `Month`. Under `Data Source`, we can also choose how many time points to ignore. Finally, we can replace the null (missing) values with zeros if desired.

Under `Forecast Model`, we can change our model from `Automatic` to `Automatic without Seasonality`, or `Custom model`, in which case we can manually choose the trend and season.

Finally, we can select or deselect the box in front of `Show prediction intervals` in order to turn the prediction intervals on or off. We can also choose among the 90%, 95%, and 99% prediction intervals.