TECHACADEMY

Data Science Project Guide: Airbnb

TechAcademy e.V.

Winter Term 2021/22

Contents

1 Welcome!

In the first few chapters, we will introduce you to the basics of the `R` and `Python` tracks, and you will find helpful explanations to questions you might have at the beginning of your coding journey. There will be a quick introduction to the Data Science track so that you can get started with the project quickly. So let's start with the basics!

In all tracks, you will work on your project in small groups of fellow students. Such group work helps you finish the project faster and makes your results better. Our experience shows: The members' different backgrounds and discussing different opinions and ideas will produce the best results. Besides, it is more fun to work on a project together than to code alone.

The groups can consist of a maximum of three members. You can choose your two teammates yourself, and we won't interfere with your arrangements. All group members must complete the same difficulty level (beginner or advanced track) since the tasks are different for each track. We encourage you to collaborate with students from various departments. This interdisciplinarity allows you to get to know people from other departments, giving you a whole new perspective on the project and tasks.

It is essential to note that each person must submit the project individually for a certificate when submitting your project. However, each person's project can look identical to the other members of your group. You will get more information at our first Coding Meetup on **November 24, 2021**.

This Airbnb case study and its project guide were developed and written from scratch by TechAcademy's Data Science team. Lara Zaremba, Lukas Jürgensmeier, Annalisa Strauß, Karlo Lukic, and Inga Lasys developed the project in `R`, while Felix Schneider, Manuel Mair am Tinkhof, and Thilo Leitzbach developed it in `Python`. We would also like to thank Benjamin Lucht for his extensive input in making this project a reality.

2 What's Data Science and How Do I Do It?

Data Science is a multi-layered field in which the latest machine learning methods are only a tiny part. To finish your data science analysis, you'll need to complete many steps – from collecting to manipulating to exploring the data. And eventually, you will need to communicate your findings somehow.

But first things first. To analyze the data, you must first obtain it. You need to know where to get it and how to integrate it into your respective tools. The data is rarely available as it would be necessary for further processing. Familiarizing yourself with the information available, cleaning it up, and processing it into the desired formats that humans and machines can read are essential steps that often make up a large part of the data scientist's work.

Before you can analyze the obtained data, you must first select and master the right tool: the programming language. The most often used languages for Data Science are `R`, which was

explicitly developed for statistics, and `Python`, which is characterized by its additional versatility. The data scientist does not have to be a perfect software developer who masters every detail and programing paradigm. Still, the competent handling of syntax (writing code) and idiosyncrasies is essential for her.

There are some well-developed method collections, the so-called packages or libraries, which provide a lot of additional functionality to an elemental programing language. As a data scientist, you should also learn and master the use of these collections, especially when preparing the data. Once you have prepared your data, you can finally analyze it.

It is also crucial to know and understand the multitude of statistical approaches to choose the correct method for the problem at hand. The newest, best, and most beautiful neural network is not always the solution to everything! One step is still missing in the data science process: understanding and communicating your results. The results are often not spontaneously intuitive or sometimes even surprising. Here, the data scientist employs specific expertise and creativity, especially during data visualization.

2.1 What's R?

`R` is a programming language developed by statisticians in the early 90s to calculate and visualize statistical results. A lot has happened since then, and by now, `R` is one of the most widely used programming languages in Data Science. You don't have to compile your 'R' code, but you can use it interactively and dynamically. Such an approach makes it possible to quickly gain basic knowledge about existing data and display it graphically.

`R` offers much more than just programming. The language provides a complete ecosystem for solving statistical problems. A large number of packages and interfaces are available, which you can use to expand the basic functionality of the programing language to, say, create a COVID-Tracker application.

2.1.1 RStudio Cloud

Before you can use `R`, you usually have to install some separate programs locally on your computer. Typically, you first install a "raw" version of `R`. In theory, you can then start programming. However, it is challenging to carry out an entire project with a "raw" version of R. That's why there is RStudio, a free Integrated Development Environment (IDE) for `R`.

Such IDE includes many essential features that simplify programming with `R`. Among other things, an auto-completion of your code, a friendly user interface, and many expansion options. Think of R as your car's engine. And think of RStudio as your car's dashboard that shows fancy metrics, has a radio and allows you to adjust air-conditioning!

Experience has shown that installing `R` and RStudio locally on your computer takes some effort. Fortunately, RStudio also has a cloud solution that eliminates these steps: RStudio Cloud. You can edit your project in the same IDE in the browser without any prior installations on your computer. You can also easily switch your project from a private to a public project and give

your team an insight into your code via a link or by giving them access to the workspace directly. In this way, you can easily exchange ideas with your team.

We will introduce RStudio Cloud and unlock access to our workspace on our first Coding Meetup. Until then, focus on learning the "hard skills" of programming with the courses on DataCamp. That brings us to your curriculum in the next section!

2.1.2 Curriculum

The following list shows the required DataCamp courses for the Data Science with `R` Track at TechAcademy. As a beginner, please stick to the courses of the "beginner" program. Ambitious beginners can, of course, take the advanced courses afterward. However, it would be best if you worked through the courses in the order we listed them.

The same applies to the advanced courses. Here, too, you should finish the specified courses in the given order. Since it can, of course, happen that you have already mastered the topics of an advanced course, you can replace some courses. If you are convinced that the course does not add value to you, feel free to replace it with one of the courses in the "Exchange Pool" (see list below). However, you should not pursue an exchange course until you finish all chapters from the advanced course: "Intermediate R."

To receive the certificate, both beginners and advanced learners must complete at least two-thirds of the curriculum (6/9 courses). For the beginners, this means until – and including – the course "Data Visualization with ggplot2 (Part 1)" and for the advanced until –and including – "Supervised Learning in `R`: Classification." In addition, you should complete at least two-thirds of the project tasks. After completing the curriculum and the project's (minimal) requirements, you will receive your TechAcademy certificate!

R Fundamentals (Beginner Track)

1. Introduction to R (4h)
2. Intermediate R (6h)
3. Introduction to Importing Data in R (3h)
4. Cleaning Data in R (4h)
5. Data Manipulation with dplyr (4h)
6. Data Visualization with ggplot2 (Part1) (5h)
7. Exploratory Data Analysis in R (4h)
8. Correlation and Regression in R (4h)
9. Multiple and Logistic Regression in R (4h)

Machine Learning Fundamentals in R (Advanced Track)

1. Intermediate R (6h)
2. Introduction to Importing Data in R (3h)
3. Cleaning Data in R (4h)
4. Importing & Cleaning Data in R: Case Studies (4h)
5. Data Visualization with ggplot2 (Part1) (5h)
6. Supervised Learning in R: Classification (4h)
7. Supervised learning in R: Regression (4h)
8. Unsupervised Learning in R (4h)
9. Machine Learning with caret in R (4h)

Data Science R (Advanced Track) – Exchange Pool

- Data Visualization with ggplot2 (Part 2) (5h)
- Interactive Maps with leaflet in R (4h)
- Machine Learning in Tidyverse (5h)
- Writing Efficient R Code (4h)
- Support Vector Machines in R (4h)
- Supervised Learning in R: Case Studies (4h)
- Optimizing R Code with Rcpp (4h)

2.1.3 Helpful Links

- RStudio Cheat Sheets
- RMarkdown Explanation (to document your analyses)
- StackOverflow (forum for all kinds of coding questions)
- CrossValidated (Statistics and Data Science forum)

2.2 What's Python?

`Python` is a dynamic programming language. You can execute the code in the interpreter, so you do not have to compile the code first. This feature makes `Python` very easy and quick to use. The excellent usability, easy readability, and simple structuring were and still are core ideas in developing this programming language.

You can use `Python` to program according to any paradigm, whereby structured and object-oriented programming is most straightforward due to the structure of the language. Still, functional or aspect-oriented programming is also possible. These options give users significant freedom to design projects the way they want and great space to write code that is difficult to

understand and confusing. For this reason, programmers developed specific standards based on the so-called `Python` Enhancement Proposals (PEP) over the decades.

2.2.1 Anaconda and Jupyter

Before you can use `Python`, you must install it on the computer. `Python` is already installed on Linux and Unix systems (such as macOS), but often it is an older version. Since there are differences in the handling of `Python` version 2 – which is no longer supported – and version 3, we decided to work with version 3.6 or higher.

One of the easiest ways to get Python and most of the best-known programming libraries is to install Anaconda. There are detailed explanations for installing all operating systems on the website of the provider.

With Anaconda installed, all you have to do is open the Anaconda Navigator, and you're ready to go. There are two ways to get started: Spyder or Jupyter. Spyder is the integrated development environment (IDE) for `Python` and offers all possibilities from syntax highlighting to debugging (links to tutorials below).

The other option is to use Jupyter or Jupyter notebooks. It is an internet technology-based interface for executing commands. The significant advantage of this is that you can quickly write shortcode pieces and try them out interactively without writing an entire executable program. Now you can get started!

If you have not worked with Jupyter before, we recommend that you complete this DataCamp course first. There you will get to know many tips and tricks that will make your workflow with Jupyter much easier.

To make your work and, above all, the collaboration more accessible, we are working with the Google Colab platform that contains a Jupyter environment with the necessary libraries. You can then import all the data required for the project with Google Drive. We will introduce this environment during our first Coding Meetup. Until then, focus on learning the "hard skills" of programming with your courses on DataCamp. This topic brings us to your curriculum in the next section!

2.2.2 Curriculum

The following list shows the required DataCamp courses for the Data Science with `Python` Track at TechAcademy. As a beginner, please stick to the courses of the "beginner" program. Ambitious beginners can, of course, take the advanced courses afterward. However, it would be best if you worked through the courses in the order we listed them.

The same applies to the advanced courses. Here, too, you should finish the specified courses in the given order. Since it can, of course, happen that you have already mastered the topics of an advanced course, you can replace some courses. If you are convinced that the course does not add value to you, feel free to replace it with one of the courses in the "Exchange Pool" (see list

below). However, you should not pursue an exchange course until you finish all chapters from the advanced course: "Intermediate Python."

To receive the certificate, both beginners and advanced learners must complete at least two-thirds of the curriculum (6/9 courses). For the beginners, this means until – and including – the course "Joining Data with pandas (4h)" and for the advanced until –and including – "Exploratory Data Analysis in Phyton (4h)." In addition, you should complete at least two-thirds of the project tasks. After completing the curriculum and the project's (minimal) requirements, you will receive your TechAcademy certificate!

Python Fundamentals (Beginner Track)

1. Introduction to Data Science in Python (4h)
2. Intermediate Python (4h)
3. Python for Data Science Toolbox (Part 1) (3h)
4. Introduction to Data Visualization with Matplotlib (4h)
5. Data Manipulation with pandas (4h)
6. Joining Data with pandas (4h)
7. Exploratory Data Analysis in Phyton (4h)
8. Introduction to DataCamp Projects (2h)
9. Introduction to Linear Modeling in Python (4h)

Data Science with Python (Advanced Track)

1. Intermediate Python (4h)
2. Python Data Science Toolbox (Part 1) (3h)
3. Python Data Science Toolbox (Part 2) (4h)
4. Cleaning Data in Python (4h)
5. Exploring the Bitcoin Cryptocurrency Market (3h)
6. Exploratory Data Analysis in Phyton (4h)
7. Introduction to Linear Modeling in Python (4h)
8. Supervised Learning with Scikit-Learn (4h)
9. Linear Classifiers in Python (4h)

Data Science with Python (Advanced Track) - Exchange Pool

- TV, Halftime Shows and the Big Game (4h)
- Interactive Data Visualization with Bokeh (4h)
- Time Series Analysis (4h)
- Machine Learning for Time Series Data in Python (4h)
- Advanced Deep Learning with Keras (4h)
- Data Visualization with Seaborn (4h)
- Web Scraping in Python (4h)
- Writing Efficient Python Code (4h)
- Unsupervised Learning in Python (4h)
- Writing Efficient Code with pandas (4h)
- Introduction to Deep Learning in Python (4h)
- ARIMA Models in Python (4h)

2.2.3 Helpful Links

Official Tutorials/Documentation:

- https://docs.python.org/3/tutorial/index.html

- https://jupyter.org/documentation

Further Explanations:

- https://pythonprogramming.net/
- https://automatetheboringstuff.com/
- https://www.reddit.com/r/learnpython
- https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook

2.3 Your Data Science Project

2.3.1 Coding Meetups and Requirements

Now that you have learned the theoretical foundation of Data Science in the DataCamp courses, you can put your skills into practice. We have put together a project for you based on real data sets. You can read about the details of this project in the following chapters of this project guide.

Of course, we will also describe the project and the tools that go with it. We will discuss everything you need to know during the first Coding Meetup, which will take place on **November 24, 2021**. After that, your work on the project will officially begin. You can find the exact project tasks together with further explanations and hints in the following chapters.

To receive your TechAcademy certificate, you must solve at least two-thirds of the "Exploratory Data Analysis" part of your Data Science project. We added the "Price Prediction – The Application of Statistical Models" part for the advanced participants. In addition, you should complete two-thirds (6/9 courses) of the respective curriculum on DataCamp, as mentioned. You can find more detailed information about the curriculum in the "Curriculum" section of the different programming languages above.

3 Introduction to Your Project

3.1 Purpose of the Project Guide

Welcome to the project guide for your TechAcademy Data Science project! This document will guide you through the different steps of your project and provide you with valuable hints along the way. However, the guide is not a detailed step-by-step manual. We think you should develop the skills of coming up with your way of solving different tasks. Such an approach is an excellent opportunity to apply the knowledge and tools you acquired through DataCamp.

Questions might come up, or you might not know how to solve a task right away - but do not worry - that is just a part of coding. In those cases, you can find helpful links in the introductory chapters, where someone might have answered your question(s). If not – and in the unlikely case that even Google cannot provide the answer – our TechAcademy mentors will help you via Slack or directly during the coding meetups. You are strongly encouraged to chat with your group's mentors!

At the end of the project guide, you will find an overview of the tasks you must complete, depending on your track (beginner vs. advanced). You can use that list to check which tasks you still need to complete or which assignments are relevant for your track.

3.2 What is this Project About?

Having worked with Spotify data last semester, we will focus on a different topic this semester: **Airbnb**. More precisely, you will analyze data about Airbnb offers in a city of your choice! You can choose from the following list of cities according to your preferences, but each group member should work on a different city:

Amsterdam, Boston, Edinburgh, Madrid, München, and Rom.

Your data was scraped and is hosted by the Inside Airbnb project. You will find all kinds of information in that data set – both valuable and worthless to your analysis. In the analogy of the typical data science workflow, we have split this project into two parts. First, you are going to learn how to perform an Exploratory Data Analysis (EDA). You will have a closer look at the data, transform it and then get to know the different variables and how they look in various visualizations.

Beginners will complete the project after this first part. Still, it would be beneficial for beginners to try and work on the second part too. In the next part of the project, advanced programmers will develop a model that predicts Airbnb prices in their city as accurately as possible. They will start with a linear regression algorithm which they can modify as pleased. Additionally, they will explore other possibilities of modeling and data prediction.

But first things first: What exactly is EDA, and what can you achieve with it?

3.3 Exploratory Data Analysis – Getting to Know the Data Set

As a first step, you will get to know the data set. This step means you will describe the data and answer questions like: What variables does the data set contains? And how are they related? For this purpose, you often use graphical tools like box plots or histograms.

We structured the first part of the project to let you know the data thoroughly by completing the given tasks one after the other. As a beginner, you can stop after this part because you will have fulfilled the necessary coding requirements for your TechAcademy certificate. However, if this first part inspires you to step up your "apartment-search game," we encourage you to work on the second part!

Since data science concepts are independent of specific programming languages, we will describe the general approach in this part of the text. Having understood the bigger picture and starting with the tasks, you will find language-specific tips and tricks in visually separated boxes.

If you participate in our `R`-Track, you will need to look at the boxes with the blue border. Likewise, you look at the yellow-bordered boxes if you code in `Python`. From time to time, it might be interesting to check out the other language – though you can code the exact solutions in both, they sometimes have a different approach to an identical problem. It makes sense that you complete the first few beginner chapters mentioned in Section 2. We recommend that you finish the courses at least until – and including – *Exploratory Data Analysis in [your programing language of choice].*

3.4 Prediction – Apply Statistical Methods

This part of the project is mainly for the advanced TechAcademy participants. If you are a beginner and have completed the first part and have some leftover motivation, we would love to see you complete the second part too! Statistical models are a significant part of data science, and this gives you a chance to get acquainted with such frameworks.

As you studied the data in part one, you should be familiar with its features, and you can start creating predictions about Airbnb prices – based on the information you learned about the apartments. After completing the second part, you will send us your predictions, and we will check how accurate your model was. The most precise model wins the race!

For this part of the project, we recommend the advanced courses mentioned in Section 2. Please note that even more DataCamp classes are available, so if you want to extend your skills further, feel free to complete more courses on the topics that interest you. We recommend that you finish the lessons at least until – and including – *Unsupervised Learning in Python* for the `Python` track and *Machine Learning Toolbox* for the `R` track.

Ready to get your hands dirty? After getting a first impression of what this project is about, let's get you started!

4 Exploratory Data Analysis

Before you can dive into the data, set up your programming environment. This will be the place where the magic happens – all your coding will take place there.

In your workspace on RStudio Cloud, we have already uploaded an "assignment" for you (Template Airbnb). When you create a new project within the workspace *Class of '21/22 | TechAcademy | Data Science with R*, your workspace will open up.
We've already made some arrangements for you: The data sets you will be working with throughout the project are already available in your working directory. We also created an RMarkdown file (a file that ends with `.Rmd` extension), with which you will be able to create a detailed report of your project. You can convert that file into an HTML document when you have finished coding the project in R. Open the file `Markdown_Airbnb.Rmd` and see for yourself!

We recommend using Google Colab for this project since it requires no particular setup, stores its notebooks to your Google Drive, and makes it easy for you to share them with your team members.
As an alternative to Google Colab, you might want to install Jupyter Notebook locally using the Anaconda distribution. Either way, when importing Airbnb data, you can use the links to the respective data files provided in the "Data-Links" document, which you will find in the TechAcademy drive.
We will give you a more detailed step-by-step demo during the first coding meetup.

4.1 Visualizing Apartment Availability

Imagine being a tourist on a budget. First things first, you might want to check for periods when apartments (or, Airbnbs) are available for booking on the market. Remember the basics of economics? A lower supply of a product on the market usually comes with higher prices (all else equal). Vice versa, a higher supply of a product might come with a lower price. That is precisely what you want if you are on a budget! Explore the data set looking for periods with high availability and find out later whether that basic economic idea holds in the Airbnb context.

4.1.1 Import, Clean and Transform the Data

In this chapter, you will apply the knowledge that you gained throughout the semester to real-world data. The exercises will cover typical problems that you usually encounter when analyzing data.

First things first, you will need to load the `calendar.csv` file into your coding environment: How is the data structured, and which variables does it contain? To get a peek at the data you'll be working with, output the first/last few rows. Also, compute some basic descriptive statistics (e.g., count, mean, and standard deviation) and determine the column's data types.

First, load the necessary packages/libraries that you intended to use for your analysis.
Then, load the data set from your folder structure into the workspace. Import the calendar data set into your workspace and name the object accordingly. Use, for example:
`calendar <- read_csv("calendar.csv")`
Now get an overview of the data set. How is it structured, and which variables does it contain? You can use the following functions, for example:
`head(calendar)`
`str(calendar)`
`summary(calendar)`

Import the pandas library to your notebook with `import pandas as pd`.
Use the same pandas method used to import CSV data on Datacamp to import CSV files from a web server: Just replace the file path with the URL of the CSV file. Find the links to all datasets here. Be aware that our csv data has been compressed to gzip format (.csv.gzip). This is no problem! You can still use the same method, but you will need to add an additional argument to the function, which is `compression="gzip"`.
...yes, the pandas library is extremely powerful!
Take a look at the first rows of your data frame and print some descriptive statistics with the pandas methods `.head()` and `.describe()`. Also, find out the data types for each column: Are values in the price column stored as numbers?

Visualization is one of the most helpful methods to get a feeling for the relation of the various variables of the data set. As already mentioned, this process is called *Exploratory Data Analysis* (EDA).

Before starting with that, we first have to clean the data so that the functions can process the data set correctly. You will have to do that quite often in the field of Data Science since the data very often comes in an unusable format. For example, in our case, the values in the price column are not stored as a numeric data type – this becomes an issue when performing math operations on such columns.

In order for `R` to process this, we need to remove a character from the observations and convert the variable into a numeric format. You can use the functions `str_remove_all()` or `gsub()` together with `as.numeric()` for this.
To make the next step easier, we still have to convert the logical variable `available` from a `character` to a `boolean` data type. Use, for example, the `ifelse()` function for this and replace "f" and "t" with the associated logical values `FALSE` and `TRUE`.
As a last step, transform the `date` variable into the `R` format `Date`. Remember the `as.numeric()` function? `R` has a similar function to convert variables to a date format.

To remove a character in a column's cells use pandas' `.replace()` method. Similarly, convert the data to a different type with `.astype()` (Hint: You might need to apply the replace method twice on the price column before converting).
Note that the column indicating apartment availability contains two values, namely `f` and `t`. Replace `f` and `t` with the associated boolean values `FALSE` and `TRUE`.

Our goal now is to show the number of Airbnb apartments available over the next year in a

simple line plot. However, this goal is still not possible since there is an entry for each apartment in the upcoming 365 days. And another related variable that shows us whether the apartment is available or not on each of those days is present too. The transformed data set should summarize this information and have only one entry for each day that tells us the aggregated number of available Airbnbs in the city.

The `R` package `dplyr` is the first choice for this type of transformation task. If you haven't heard of it yet, take the respective DataCamp course or the DataCamp course on Exploratory Data Analysis.
Use the `dplyr` functions `group_by()` and `summarize()` and save the resulting data set in a new data frame, which you can call `avail_by_date`. You can, of course, name the data frame however you wish.
If you need further help, you can also use the `dplyr` Cheat Sheet for this task, on which you can visually see such transformations.

To understand this step, remember that you converted the values in the `available` column to be of type `bool`. In Python, the boolean value `True` is equal to `1`, while `False` is equal to `0`. Thus, `True + True + False` is equal to `2` (you can try this out by executing `print(True + True + False)`). So, to get the number of available apartments, you could simply sum the (boolean) values in the `available` column!
But before doing so, make sure to group the data frame by date with `.groupby()` in combination with the aggregation method `.sum()` as you need to get the number of available apartments for each date.
The pandas Cheat Sheet might be of help here - feel free to make use of it.

4.1.2 365 Days Lineplot

We have finally transformed the data set into a useful format, so now it is time to actually visualize it. We are going to get started with a simple line plot that shows the number of available apartments over the next 365 days. This is what your result could look like:

For this visualization, you can use the `base` package that includes the function `plot()` or use the more extensive and very flexible graphics package `ggplot2`. We highly recommend getting comfortable with using the`ggplot2` package as it's one of the most used `R` packages overall.

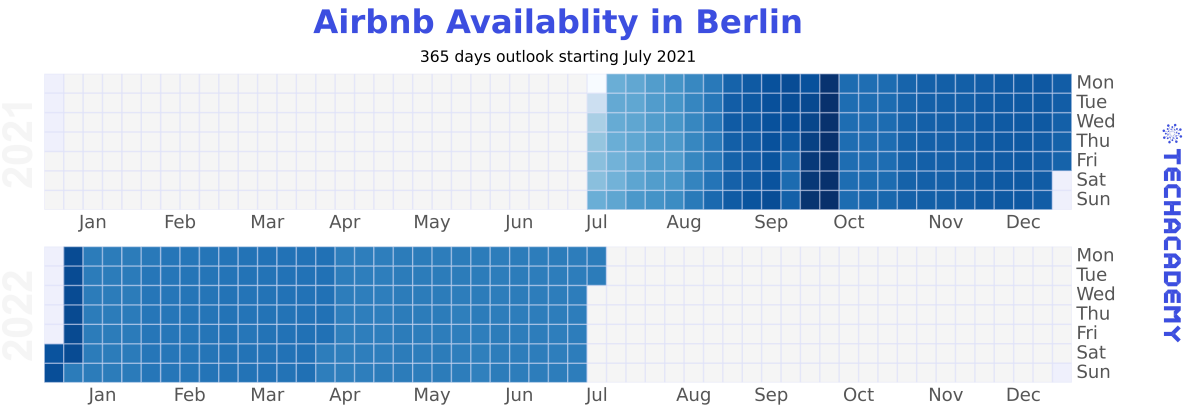
Having the data in the correct format, we can now start plotting a simple line graph. For this, you can use matplotlib's `plt.plot()`.

Looking at the line plot above, you can see a clear pattern for different dates. There are distinct drops in the availability of Airbnb apartments. **One task for your project is to think about a potential cause behind this pattern: Is there one?**

4.1.3 Calendar Plot

A line plot is good but let's be honest; it's nothing to write home about! How about using a calendar plot to get a quick overview of this year's most and least abundant Airbnb months? Let's see how this looks in both R...

… and Python:



There are country-specific holidays and seasons, after all!

Check out the calendR `R` package to visualize the availability of Airbnb listings in a calendar plot.

You can use the package calmap to visualize availability in an intuitive manner!

4.2 Correlation Coefficient and Plot

Now that you are confident that there are some distinct patterns in the data set, you might be willing to quantify your suspicion. For that purpose, calculate the correlation coefficient (r) between `availability` and `price` and plot both variables to confirm your intuition.

Regarding visualization, you could, for example, choose between:

a) line plot (boring as you already did it), or

b) scatter plot (exciting as it's something new for you).

Please **write a short sentence explaining what the correlation coefficient measures in this context.**

For example, this task could look like:

Answer: "The correlation coefficient shows …".

Have a look at `ggplot2` and plot a scatter plot with `geom_point()`. Use the stats function `cor(x,y)`, for example, to calculate the correlation coefficient between two variables.

You can use matplotlib's scatter plot for plotting and pandas's `.corr()` method to compute the correlation coefficient between two variables.

4.3 Visualize individual Airbnb Offers with the Listings Data Set

Can we use the mean price of a neighborhood to measure popularity? Perhaps… In the next section, you will compute the average price for each neighborhood, and since, as an (aspiring) Data Scientist, you love numbers and stats, you also want to compute the price variances for these neighborhoods.

Start by importing the `listings` dataset. First, you will need to clean the `price` column. After that, we can take a closer look at the price structure of the various neighborhoods.

Second, we would like to know the average price and corresponding standard deviation for each neighborhood. Please list the names of the different parts of the city and their `mean` and the `sd` for that purpose.

For this task, you can reuse `dplyr`-functions like `group_by()` or `summarize()`.

Once again, you can use the `.groupby()` method to group by neighborhood and apply an aggregation function for calculating the standard deviation. A very similar function exists for calculating the mean values. Filter both data frames by the price column.

Next, we now want to visually compare the price distribution for the, on average, most expensive district with the least expensive district. For this task, think about the different types of plots that you have explored in your courses and find the most suited plot type for this type of visualization. Once you have created your custom visualization, you will probably have to filter out some observations with extremely high prices (outliers) to get a more meaningful plot.

Your plot could end up looking like this:

In `ggplot2` graphics, you can easily filter in the plot specifications with `xlim()` or `ylim()`. You could use this resource to remind yourself of the different visualizations available in `R`.

One way to find the, on average, most expensive neighborhood is to sort the mean price column in descending order.
For plotting: The Matplotlib documentation is an excellent source of inspiration with many examples, including code that generated them.
Note: If you do not want to remove outliers, the plot could end up looking very strange. In that case, it is best to create two different plots, i.e., one for each neighborhood.

But wait! Until now, we focused much on prices. What about customer preferences? Next, we will use the average number of reviews to indicate the "hotness" of an area in your city!

4.4 Analyzing Number of Reviews per Neighborhood

In the previous part, you got to know and visualize the `listings` data set. However, a critical piece of information is missing from this data that we could use to measure popularity. How about using the number of reviews?

We first start by computing the average number of reviews per apartment. Afterward, we will do the same for each neighborhood. Calculating the number of reviews for each apartment could lead to a vital feature for price prediction.

Fortunately, we have another data set for each city, `reviews`, that recorded the apartment ID and the `date` for each review. Our goal now is to count the number of reviews for each apartment and to keep them. Since we can also find the `ID` in the `listings` data set, we can use that variable to merge the two data sets.

First, load your city's `reviews` data set into your workspace and take a closer look at it with the familiar functions. Afterward, count the number of reviews per apartment.

This works with the `table()` function or with `group_by()` and `summarize()`. Note, however, that you still have to convert the result of the `table()` function into a `data.frame` format for further processing.

Count the number of reviews per apartament (i.e. per "listing_id") with the functions you're already familiar with at this point. You can use the groupby argument `as_index=False` to avoid setting the column with listing ids as index, which might make live more easy when we merge data frames in later steps.

To merge `reviews` with the `listings` data set, you have to rename the newly generated variables in the new data set. Name the apartment `ID` analogous to the `listings` data set `id` and the number of reviews `n_reviews`.

You can do this task with the `rename()` function from the `dplyr` package.

Assign the new column names as a list to the `.column` attribute.

If your data set looks similar to this, you can merge it with `listings`:

Table 1: Aggregate Number of Reviews

	id	n_reviews
1	2015	118
2	2695	6
3	3176	143
4	3309	25
5	7071	197
6	9991	6
7	14325	23
⋮	⋮	⋮

You can use the following function for merging:
`listings_reviews <- merge(dataset1, dataset2, by = ...)`

Merge with `listings_reviews=pd.merge(dataframe1,dataframe2,left_on=....`

Now, look at the new data set. Does each ID have exactly one entry with the number of ratings? Reduce the data frame to the columns listing id, neighborhood, and the number of reviews to better understand your data.

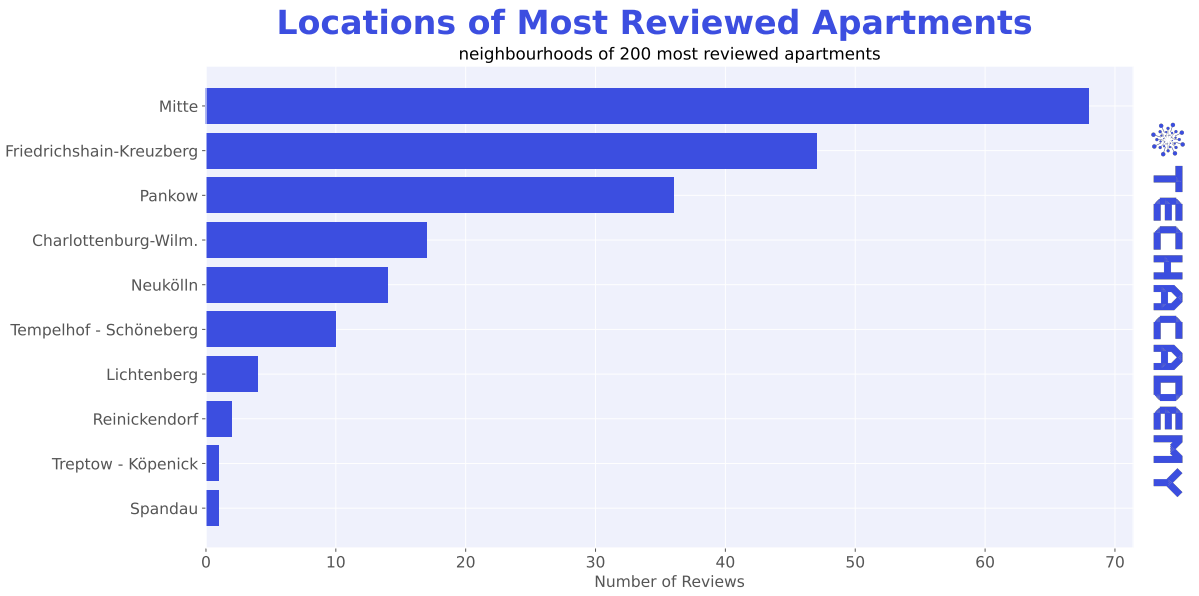
4.4.1 Your First Barplot

We now want to take a closer look at a minor part of the data set: What do the most reviewed apartments have in common? Is it the location?

Extract the 200 most reviewed apartments. One approach is to sort the data set in descending order according to `n_reviews` first and then extract the first 200 entries into a new data set.

Now you can easily visualize the districts with the 200 most frequently reviewed apartments. A bar plot is ideal for this. Feel free to try other types of visualizations best suited to answer this question.

This is what the visualization could look like:



The key commands that you can use for this within `ggplot()` are `geom_bar()` and `coord_flip()`.

You can plot bar plots with both pandas and matplotlib. If you chose to plot with matplotlib, you could use the `plt.barh()` method. Additionally you might want to use methods like `.set_yticks()`, `.set_yticklabels()` and `.invert_yaxis()`.
If Python throws you an `AttributeError`, then try working with the `plt.gca()` method.

4.5 Visualization with Maps

This part will be the final and most advanced part of the EDA and the most rewarding. Anyone who knows the Airbnb website has probably also seen their map showing the location of all apartments. We can do the same! The only difference is that our data gives us the option to show what interests us

To reduce complexity, use the data set filtered in the task before with the 200 most frequently reviewed apartments.

If you have not solved that previous task yet, select 200 listings according to other criteria or at random to complete this task.

Your map could look like this:

There are different ways of creating a map with R. Although we plotted our example using the `leaflet` package, the following tips will be about the `ggmap` package as it is easier to use.
Before you can download map material via an API interface, you have to define the corners of the map as coordinates.
First, define the height and width of the included coordinates. In the next step, you can specify the exact corners relative to the coordinates in the data set.

```
height <- max(...) - min(...)
width <- max(...) - min(...)
```

You can then define a vector, which you can call `berlin_borders`, as we did for Berlin's example. We can use such a vector to determine the values for the edges of the map. You can add a small safety margin to the respective minimum or maximum of the coordinates. Play around with the factors later to find a good section of the map.

```
berlin_borders <- c(bottom = min(listing_top200$latitude) - 0.1 * height,
top = max(listing_top200$latitude) + 0.1 * height,
left = min(listing_top200$longitude) - 0.1 * width,
right = max(listing_top200$longitude) + 0.1 * width)
```

Then, we can download the defined map section from the service provider Stamen Maps with the `get_stamenmap()` function and save it in an R-object. Now you just need to plot it!

We are going to use yet another library, this one is called Folium. For this task you need to work with its documentation which you can find online.
Use folium's `Circle` to draw a circle for each apartment at its location coordinates. You will need to implement a `for`-loop to iterate over all 200 apartments.

```
# Initiate the map
m = folium.Map(
    location=[52.5, 13.4],  # Berlin Coordinates
    zoom_start=11,
    tiles='Stamen Toner'  # Map Style
)

# Use a for-loop to plot circles
for idx, row in df.iterrows() :
    # Your code here

m  # Displays the map
```

In addition to the coordinates, there is a lot of information about each listing in our data. This time, implement a pop-up window to display more details about each Airbnb apartment.

Your map should now look similar to this one:

This time, we created the plot with `leaflet`. Why don't you try to find information on the package online?
You can create a basic map with leaflet and add additional layers by following these simple steps:

```
m <- leaflet() %>%
  addTiles() %>%
  addMarkers(lng= ,
             lat= ,
             popup= )
```

Do `lng` and `lat` look familiar to you from a previous plot?
The `popup` is where the magic happens! Can you make the map show you listings ID, type, price, review score, and number of reviews per apartment?

Once again, you will need to loop through each apartment.
This time however, the body looks more like this:

```
folium.Marker(
    [longitude, latitude],
    popup="<i>some text here</i>",
    tooltip='click me!'
).add_to(m)
```

For some analyzes, it is easier if you don't just see points but their distributions on a map. For example, to see many apartments in a small space, you can display the apartment density. Why not plot a heatmap?

You are now familiar with two packages that can plot maps in `R`. If you want to use `ggmap` to create such a two-dimensional density plot, you can use the `geom_density2d()` and `stat_density2d()` packages on the map.
If you don't know precisely how to fill the individual arguments, you can always Google for more help.

No hints or tips here - it's up to you to study the documentation for `folium`'s heatmap and to come up with a (possibly) nice heatmap

4.6 Surprise Us! (With At Least One More Plot)

Be creative!

With the available data, you could do a ton of more stuff. You could try to find the perfect listing that suits your needs and plot an availability calendar for that listing, perfect for your holiday planning! You could further go into a more detailed price analysis or come up with other plots (e.g., distribution of room types per neighborhood, or even number of hosts who join Airbnb per month!). Also, compare the results with your teammates! How do things look like in other cities, compared to yours?

Congratulations – based on your work with fundamental data transformations and many visualizations, you now have a solid understanding of the Airbnb offers in your city! With this

step, you have now completed the first part of the project! If you are in the beginner group, you have now met your minimum requirements. Nevertheless, we strongly recommend that you also have a look at the following part. There you'll be developing methods to predict the price of an apartment! Sounds exciting?

5 Price Prediction – Application of Statistical Methods

In the previous part you got a feel for the data set. You now know which variables are included and a few characteristic properties of the data set. But so far, we have only visualized the data set. In this section we go a step further and use statistical methods to predict the price of individual Airbnb apartments as accurately as possible.

In order to be able to compare your models at the end, we use a uniform metric according to which we can check the price predictions for accuracy. In our case this is the Root Mean Square Error ($RMSE$), i.e. the root of the average squared difference between the predicted ($\hat{y}_i$) and actual value (y_i):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

The closer the $RMSE$ is to 0, the better your model predicts prices. In the following, your goal is therefore to reduce the $RMSE$ of your various models as much as possible through continuous improvements.

We use the Airbnb listings data set, which contains features about apartments including prices. When we train our model with the data, we need to estimate how well our model predicts prices for listings it has never seen before. Therefore we split our data set into two parts - a train and a test set.

Here is a brief description of what you need each of the data sets for:

- *train data* (70 %): You use this training data set to train your model. The model can learn the relationships between the variables based on the training data set that contains both, the variables needed to predict the prices and also the actual prices themselves.
- *test data* (30 %): With this test data set you can test how well your model predicts the price using data that has not been seen before. This will help you for example with recognizing under- or overfitting.

There are multiple ways to split/partition the data set in R. You could, e.g., use `createDataPartition()` function from {caret}, `sample.split()` function from {caTools} or — perhaps most intuitively — `split_train_test()` function from {healthcareai}.
Make sure, however, to include the `set.seed()` function in your R code before randomly dividing the data set into train/test parts. `set.seed()` function allows you to reproduce randomness each time you re-run your code. That means that you always get the same (random) result – a vital aspect of reproducibility within the scientific method.

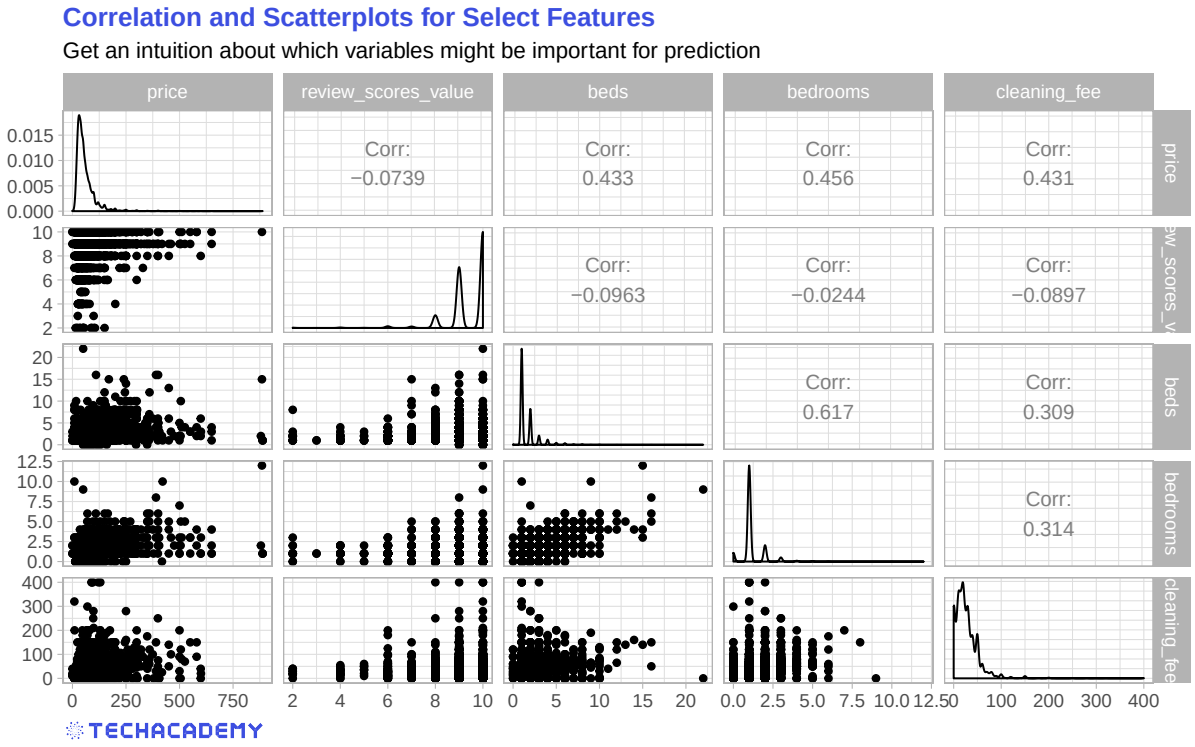
The sklearn library has a function that you can use to split the data set, correspondingly

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(...)
```

Make sure to set the random seed parameter, such that the train and test data sets have the same values each time the method is called.

5.1 Examine Correlation Between the Variables (train)

How are the individual variables related to each other? In other words, to what extent do the variables in the data set correlate with one another? Finding this out is extremely important for deciding which model specification to use later. A good place to start is to create a correlation matrix:



Use the `cor()` function from the {base} R to create a correlation matrix. Select all numerical variables in your data set with the help of `sapply()` or {dplyr}'s `select()` and create a correlation matrix.
Alternatively, use the `ggcorrplot()` from the same-name {ggcorrplot} package.
Check out this article by James Marquez if you're curious about other ways to create a correlation plot in R.

A handy library for plotting correlation matrices is the seaborn library: `import seaborn as sns`. Use its pairplot method and pass on the dataframe with the selected columns to visualize distributions and correlations.
Additionally, you may want to plot a heatmap with `sns.heatmap()` which makes it even easier to see correlations.

Which of your examined variables correlates most with price and which seems to be more independent from price? You now have a first impression over which variables could be important for your model. So let's get to your first price prediction model!

5.2 First Predictions with Simple Regression Models (train)

Now you can make use of your statistical knowledge. You will need a method with which you can predict the price of an Airbnb apartment. A very simple first approach would be to use the average demand as the first prediction. However, this is almost certainly not the best prediction. In this case, your predicted price would be the same over all days and would ignore all factors that influence the price.

Ever heard of linear regression? That would be a much better approach. Now you can use your statistics skills. First set up a model with the dependent variable `price`. In the previous exercise you examined different variables. Now choose the variable with the highest correlation to price and use that as the only independent variable.

For example, your first regression model could look like this:

$$price_i = \beta_0 + \beta_1 bedrooms_i + \epsilon_i$$

In `R` you can implement a simple linear regression with the function `lm()`. You then get the model summary with the `summary()` function.

Define both dependent (`y_train`) and independent (`X_train`) variables and clean the data if necessary.
For the next step the X_train values need to be reshaped `.values.reshape(-1,1)`. Note: If you use more then one feature you don't have to reshape your data.
Import `LinearRegression()` from `sklear.linear_model` and train your model using `LinearRegression().fit()`.

Does your independent variable have a statistically significant impact on apartment price? Probably yes, because we selected the variable most correlated with price. However, if we stick to this very simplified model, we are making a typical mistake: the so-called Omitted Variable Bias (OVB). To put it simply, we neglect (in statistics jargon: "do not control for") variables that have a significant influence on the dependent variable. One could suspect that other influencing factors also play a large role in price setting. If we do not include them, the estimate of the effect of `bedrooms`is biased and thus hardly useful. In our case this is not a big problem for the time being, since we are not interested in causal effects, but rather in the best possible prediction. Your statistics professor would almost certainly object to such a model. Nonetheless, with just a single explanatory variable, this model will not necessarily predict the price well.

One possible solution is to simply include the omitted variables in the model – how practical that these are already included in the data set. So let's set up a somewhat more extensive model that includes one more variable:

$$price_i = \beta_0 + \beta_1 bedrooms_i + \beta_2 cleaning\_fee_i + \epsilon_i$$

Now compare the results of the two models. Does the larger model explain a higher proportion of the variance in price? In other words, which model shows the higher value for the R^2 measure?

You can easily include such LaTeX tables in your RMarkdown document with the `stargazer` package:

Table 2: Model Summary for Two Simple Linear Regression Models

	Dependent variable:	
	price	
	(1)	(2)
bedrooms	35.317***	28.715***
	(0.594)	(0.642)
cleaning_fee		0.576***
		(0.017)
Constant	18.272***	13.313***
	(0.790)	(0.864)
Observations	13,488	9,245
R^2	0.208	0.331
Adjusted R^2	0.208	0.331
Residual Std. Error	45.115 (df = 13486)	39.777 (df = 9242)
F Statistic	3,538.208*** (df = 1; 13486)	2,283.658*** (df = 2; 9242)
Note:		*p<0.1; **p<0.05; ***p<0.01

Above table has been created with `R`. In Python, when using `sklearn` we need to print all metrics and information independently: Find several metrics, such as R^2 in the `sklean.metrics` module while other model parameters (estimated coefficients, intercept, …) are simply stored as attributes in your trained model object.

5.3 From Training to Testing – Making Predictions

You have now trained your first model with the training data set. But how well does the model handle data that it has not seen yet? This is a very important test to evaluate the quality of your model.

Has your model only "memorized" the existing patterns in the training data set? Then the relationships from the training data set would not be transferable to the test data set. With so-called overfitting, the model was trained too closely to the training data set and therefore provides poor predictions for unknown data – for example the data in your test and validation data sets.

On the other hand, there is also the problem of underfitting: Your model has not sufficiently learned the actual relationships between the data and therefore makes poor predictions in the test data set. So it is important to find a balance between the two problems.

Now the distinction between training and test data sets becomes important. As a reminder: we use train data to train a model and the test data to ultimately test the quality of our model.

Now load the data set `test` in addition to the data set `train` that you have already used. In order to test your model on previously unseen data, you can apply the model to the test data set.

Use the `predict` function for this:

```
predicted_price <- predict(your_saved_model, your_test_data_frame)
```

You have now created a vector with all price predictions for the test data set. You can now compare this with the actual values for `price` from `test`.

In order to use a uniform comparison metric, please use the following function to measure your prediction accuracy:

```
# Function that returns Root Mean Squared Error while ignoring NAs
rmse <- function(actual, predicted) {
sqrt(mean((predicted - actual)^2, na.rm = TRUE))
}
```

After training, create a vector with price predictions applying `.predict(X_test)` on your model. Store your prediction in the variable `predicted_price`.

Finally, compare your predicted values with the test values:

```
from sklearn.metrics import mean_squared_error
# Function that returns Root Mean Squared Error while ignoring NaNs
rmse = mean_squared_error(y_test, predicted_price, squared=False)
```

Now compare both regression models. Does the larger model have better prediction accuracy, i.e. a lower $RMSE$? Now you have a benchmark for your more advanced models, which you can beat in the next part.

5.4 Apply Advanced Machine Learning Algorithms

Now that you have created and tested an initial prediction using a simple regression model, you can now apply more advanced methods. The goal is still to get the lowest possible $RMSE$ when applying the model to the test data set. Now look at at least one other algorithm and then see if that gives you a more accurate prediction (expressed as a lower $RMSE$). You can find inspiration for this in the advanced DataCamp courses, which are listed at the beginning of the project guide. There are no limits for you – you can refine the regression using certain methods (e.g. LASSO) or set up a random forest model or a neural network. It is usually a good idea to briefly recall the functionality of the respective algorithms and to consider whether this methodology makes sense in this case with a continuous prediction variable.

At this point, a disclaimer is appropriate: Our data set has a substantial part of missing observations (`NA`) for many variables. Some machine learning algorithms require a complete set of data with no missing values, while others can do well with a smaller number. If you get into trouble about the missing values, check whether you can impute the missing values. Which method is best for imputation depends heavily on your prediction algorithm.

You can also get a noticeable gain in predictive power by modifying existing variables or generating new variables from the data set ("feature engineering"). For example, we could imagine that the distance from an apartment to the city center has a significant impact on the price. However, this variable is not included in our data set. You can write a simple function that uses the two coordinate variables to calculate the distance to the center of Berlin and appends this to the data set as a new variable.

Always compare the $RMSE$ of your advanced models with each other, as well as with the benchmark regression model from before.

Congratulations! You've made it to the end of your TechAcademy Data Science project. After visualizing the data in the first part, you've also set up predictive models in this section. If you stuck around until this part and managed to code the better part of the exercises, you've definitely earned your certificate! We hope you had fun learning Data Science with this data set and you enjoyed it – at least the parts where you didn't get stuck forever because of some unexplainable coding error. Don't forget to send your project results and the prediction-csv file you just generated to our project submission email address before the deadline.

6 Exercise Checklist

This checklist should help you keep track of your exercises. Remember that you have to hand in satisfactory solutions to at least two-thirds of the exercises. If you're part of the beginner track, this refers to two-thirds of part A (EDA) only. If you're part of the advanced track, you have to hand in at least two-thirds of both individual parts A and B. Hence, you cannot hand in 100 percent of the first part and only 50 percent of the second one. You'll need more than 66% in each one for a certificate. After all, you're not that advanced if you only did half of it, right?

Part 1: Exploratory Data Analysis (Beginner + Advanced Tracks)

1. Visualize available apartments
    a. Load the `calendar` data set and get an overview of it
    b. Transform data in columns `price`, `available`, and `date`
    c. Show the number of available Airbnb apartments in a simple line plot
    d. Create a calendar plot to visualize the availability of Airbnb listings
2. Correlation between price and availability
    a. Calculate the Correlation Coefficient between `available` and `price`
    b. Visualize Correlation
3. Distribution of Prices
    a. Calculate Mean and Standard Deviation for each neighbourhood
    b. Visually compare the price distribution for the, on average, most expensive neighbourhood with the least expensive one
4. Analysis of Listing Reviews
    a. Compute the average number of reviews per apartment
    b. Merge with the `listings` data set to extract the location of listings
    c. Plot the location of 200 most reviewed apartments (e.g. with barplot)
5. Maps
    a. Map the 200 most frequently reviewed apartments
    b. Create a new map extending its functionality by adding a pop up window
    c. Create a heatmap
6. BE CREATIVE
    a. Surprise us with one or more plots of your choice!

Part 2: Price Prediction Using Statistical Methods (motivated Beginner + Advanced Tracks)

1. Visualize feature correlations in a correlation matrix/heatmap
2. Regression
    a. Simple regression model using one variable
    b. Improve your model using more features
3. Improve your model
    a. Train new models using more advanced methods

7 What's Next in Your Data Science Career?

At the end of your TechAcademy semester, you've successfully coded your way through a whole Data Science project. You liked what you did and would like bring your skills to the next level? Then this section provides you with many useful resources to deepen your knowledge in Data Science in general or Python and R in particular. The first section is useful for every aspiring Data Scientist, while the two following boxes introduce you to some language-specific resources. If you've come across some other useful materials that we didn't mention here, feel free to contact us – this list is far from complete!

7.1 Data Science in General

Version Control with Git

If you're serious about data science, you will need Git. Better learn it early and start enjoying and appreciating it before it's too late and you're pressured into learning it on the fly! Every project you do should be accompanied by a VCS (Version Control System) like Git. Regardless if you're working alone or with a big group of developers. No matter whether you write ten lines of code or a really complex program. With Git, you can keep track of all your changes. It's like a Dropbox/Google Drive for developers, but way better. Pro-Tip: Get free GitHub Pro as a Student with the GitHub Student Developer Pack. Besides all the perks of GitHub Pro, you'll also free access to many other great tools. See the respective tutorials on how to set up your Git workflow.

Advice for Non-Traditional Data Scientists

Important advice from Gordon Shotwell, a former lawyer, on what it takes to have a successful data science career coming from a non-computer science background. Extremely encouraging and helpful read on what you should and shouldn't do to reach that goal.

Learn from Great Data Scientists on Kaggle

Kaggle is a platform that hosts data science challenges. The great thing about it is that you can browse through many clever solutions to tricky machine learning tasks. And of course, you can also join the competition and measure your predictions with others. There are plenty of both Python and R notebooks.

7.2 R

Install R and RStudio Locally
RStudio.Cloud is great for getting started with R without having to worry about installing anything locally. Sooner or later you will have to install everything on your own computer. Here's a DataCamp tutorial on how to do that.

Version Control with Git
RStudio has a nice interface that lets you enjoy the perks of Git without ever having to touch the command line – sounds great, does it? Learn how to set up the Git & R workflow with Happy Git with R.

R Graph Gallery
Get inspiration to take your plotting to the next level. Includes code to reproduce the plots.

Follow the R Master Himself and the R Community
Hadley Wickham was and continues to be extremely influential on the development of R and its rise to one of the most popular data science languages. He's behind many tools that we taught you in this semester, especially the tidyverse (including great packages such as ggplot2 and dplyr). Follow him on Twitter to get great R advice and keep up to speed with everything new to R. Following the many people behind R (not only Hadley) is a great way for acquiring deeper understanding of the language and its developments.

Join the Campus useR Group in Frankfurt
There's a quite active R community in Frankfurt that meets once a month. It's open for students, professors, industry practitioners, journalists, and all people that love to use R. In those meetings, you'll hear about other's work, discuss new developments, and ask questions.

Listen to R Podcasts Another great way to easily keep up with new developments in the Data Science/R community. Check out Not So Standard Deviations or the R-Podcast

7.3 Python

Install Python Locally

Until now you've only programmed using JupyterHub on the TechAcademy Server. A next step would be to install Python and Jupyter locally on your computer. This link contains the necessary information on how to install the software on Windows, iOS or Linux.

Choosing the Right Editor

Using Jupyter is especially useful for short data analyses. But sometimes you want to write longer scripts in Python. In these cases, it is often more convenient to use a code editor instead of Jupyter. This tutorial highlights the positive aspects of such an editor and how to choose the right one for you. Pro-Tip: Also check out the other tutorials on Real Python and check out the Community Version of "PyCharm" which is the most common Python IDE (Integrated Development Environment.

Python Graph Gallery

Get inspiration to take your plotting to the next level. Includes code to reproduce the plots.

More Advanced Python Concepts

You know the basic data structures in Python like lists and dictionaries. What are the next steps to improve your knowledge? This website gives good explanations for slightly more advanced concepts which can be very useful from time to time.

A Deeper Understanding

If you want to get a deeper understanding of the Python programming language and into typical algorithms which are used in the field of Data Science, this free book can be a good starting point.

Writing Beautiful Python Code

"My code doesn't look nice, but it works!" This might work for yourself, but often you will work on code with other people. But even if you're just coding for yourself it's a good idea to follow the PEP8 style guide. It's a useful convention on how to structure and code in Python. You'll find useful resources for PEP8 here and here.

Listen to Python Podcasts

When you don't have time for books you can listen to Talk Python or the Python Podcast.