



## AirBnB Projektleitfaden

TechAcademy e.V.

Wintersemester 2019/20



# Contents

<b>1 Wofür ist dieses Dokument gedacht?</b>	<b>5</b>
1.1 Um was geht das Projekt? . . . . .	5
1.2 Was ist das Ziel? . . . . .	6
<b>2 Coronavirus Projekt</b>	<b>9</b>
<b>3 Explorative Datenanalyse – Lerne den Datensatz kennen</b>	<b>17</b>
3.1 Datenimport, -bereinigung, -transformation und Lineplot (Calendardar) . . . . .	17
3.2 Visualisiere individuelle Airbnb-Angebote (listings) . . . . .	20
3.3 Merge zwei Datensätze anhand einer ID (listings, reviews) . . . . .	21
3.4 Geo-Daten mit Karten visualisieren (listings_reviews) . . . . .	22
<b>4 Preisvorhersage – Wende statistische Methoden an</b>	<b>27</b>
4.1 Untersuche die Korrelation zwischen den Variablen näher (train) . . . . .	28
4.2 Erste Vorhersagen mit einfachen Regressionsmodellen (train) . . . . .	29
4.3 Von Training zu Testen – Treffe Vorhersagen . . . . .	30
4.4 Wende fortgeschrittene Machine Learning-Algorithmen an . . . . .	32
<b>5 Noch Fragen?</b>	<b>35</b>
<b>6 Anhang</b>	<b>37</b>
6.1 Aufgaben-Checkliste für das Airbnb Data Science Projekt . . . . .	37
6.2 Beschreibung der Variablen in den Datensätzen . . . . .	39



# Chapter 1

## Wofür ist dieses Dokument gedacht?

Herzlich willkommen in dem Projekt-Leitfaden für dein TechAcademy *Data Science mit R* Projekt!

Diese Kurzbeschreibung des Projektes soll dir erste Anhaltspunkte dafür geben, wie du zu einem Ergebnis kommst. Dieses Dokument ist jedoch bewusst keine Schritt-für-Schritt Anleitung, wie du das Projekt durchführen sollst. Uns ist es wichtig, dass du dich in deinem Team selbst mit der Aufgabenstellung beschäftigst und eigene Wege findest, wie du zu einem Ergebnis kommst.

Da es aber besonders am Anfang nicht ganz offensichtlich sein kann, welche Schritte du durchlaufen sollst, geben wir dir mit diesem Dokument eine kleine Hilfestellung. Es wird sehr oft vorkommen, dass du nicht weiter weißt. Das ist ganz normal und gehört zu dem Lernprozess dazu. Du findest in unserem Handbuch Links zu sehr nützlichen Websites, wo deine Frage vermutlich schon einmal beantwortet wurde. Falls auch googlen dich nicht weiter bringt, stehen dir natürlich die Mentoren per Slack und bei unseren Coding Meetups persönlich zur Verfügung.

Am Schluss dieses Dokumentes findest du eine einseitige Übersicht aller Aufgaben in diesem Projekt. Sehe diese Auflistung als Orientierungshilfe, welche Aufgaben noch zu erledigen sind.

### 1.1 Um was geht das Projekt?

Die Sharing Economy ist in aller Munde: Uber verändert das Taxi-Geschäft grundlegend, in der ganzen Stadt stehen seit neuestem Scooter herum und im Urlaub bucht man sich ein Airbnb und übernachtet in einer fremden Wohnung. Nachdem wir uns im letzten Semester ausführlich mit einem

Leihfahrrad-Datensatz beschäftigt haben, schauen wir uns in diesem Semester das Geschäftsmodell Airbnb an. Genauer gesagt analysieren wir einen Teil eines sehr ausführlichen Datensatzes aller Airbnb Angebote in Berlin. Diese Daten wurden im November 2018 “gescraped”, also von der Airbnb Homepage ausgelesen. Darin findet ihr allerhand nützliche und unnütze Informationen zu den Angeboten.

Schon gespannt, das ganze selbst auszuprobieren? Analog zu einem typischen Data Science Workflow haben wir die Aufgaben in zwei Teile aufgeteilt. Als erstes lernst du in einer sogenannten Exploratory Data Analysis (EDA) den Datensatz genauer an und lernst die Variablen und deren Ausprägungen mit Grafiken kennen. Für Anfänger ist der Pflichtteil danach abgeschlossen – doch es lohnt sich, auch den Fortgeschrittenen Teil auszuprobieren. In diesem stellst du ein Modell auf, welches die Airbnb-Preise in Berlin möglichst akkurat vorhersagen kann. Ein Gefühl für dies bekommst du mit einem einfachen linearen Regressionsmodell, was du nach Belieben erweitern kannst. Doch alles der Reihe nach... Was genau ist EDA und was kann ich damit erreichen?

## 1.2 Was ist das Ziel?

### 1.2.1 Explorative Datenanalyse – Lerne den Datensatz kennen

Als ersten Schritt werden wir den Datensatz *deskriptiv* kennen lernen. Das heißt, wir nähern uns dem Ziel, indem wir die Daten *beschreiben*. Bei Data Science Projekten ist es sehr wichtig, sich zuallererst mit dem Datensatz vertraut zu machen. Welche Variablen sind in dem Datensatz enthalten und wie stehen sie im Verhältnis zueinander? Diese Fragen kann man sich sehr gut mit Grafiken beantworten.

Wir stellen dir dafür eine Reihe von strukturierten Aufgaben, die du nacheinander bearbeiten wirst. Anfänger, die bisher noch keine oder sehr wenige Statistik-Kenntnisse haben, können nach diesen Aufgaben aufhören – denn damit sind für Anfänger die Mindestvoraussetzungen erfüllt. Jedoch wird es gerade danach spannend. Versuche dich also auf jeden Fall trotzdem daran, wenn du noch etwas dazu lernen willst.

Für diesen Abschnitt ist es sinnvoll, die ersten DataCamp Kurse in deinem Curriculum absolviert zu haben. Insbesondere folgende Kurse helfen dir weiter bei der Explorativen Datenanalyse:

- Introduction to R
- Importing Data in R
- Data Visualization with ggplot2 (Part 1)
- Data Manipulation with dplyr in R
- Exploratory Data Analysis

### 1.2.2 Preisvorhersage – Wende statistische Methoden an

Dieser Teil ist vornehmlich für etwas fortgeschrittenere Teilnehmer vorgesehen. Wenn du jedoch als Anfänger gut durch den ersten Abschnitt gekommen bist, empfehlen wir dir ausdrücklich, auch diesen Teil zu bearbeiten. Statistische Modelle sind ein enorm wichtiger Teil des Themenbereiches Data Science. Nachdem wir den Datensatz kennen gelernt haben, können wir in diesem Schritt ein Modell entwickeln, mit dem wir die Airbnb Preise für einzelne Apartments zu unterschiedlichen Zeiten vorhersagen können. Cool, oder? Dein Ziel in diesem Abschnitt ist es, ein möglichst akkurate Modell dafür aufzustellen. Am Schluss schickst du uns deine Vorhersagen für einen kleinen Teil des Datensatzes, bei dem du den Preis noch nicht weißt. Wir vergleichen diese Vorhersagen dann mit den tatsächlichen Preisen und berechnen damit, wie akkurat dein Modell Preise vorhersagt. Das beste Modell gewinnt!

Für diesen Abschnitt empfehlen wir dir folgende DataCamp Kurse. Beachte jedoch, dass es noch viele weitere Kurse gibt, die dir eine fortgeschrittenere Lösungsmöglichkeit beibringen.

- Correlation and Regression
- Multiple and Logistic Regression
- Supervised Learning in R: Regression
- Unsupervised Learning in R
- Machine Learning Toolbox

Alles klar? Dann lasse uns gleich starten, nachdem du jetzt einen Überblick über die Aufgaben hast!

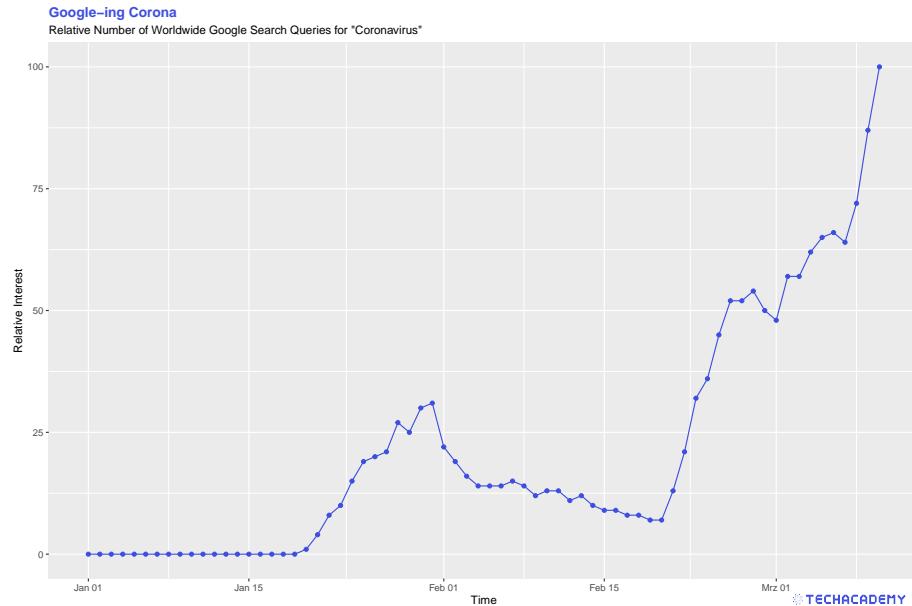


## Chapter 2

# Coronavirus Projekt

Blaaaa Test

```
> 'data.frame': 70 obs. of 5 variables:  
> $ date    : Factor w/ 70 levels "2020-01-01","2020-01-02",...: 1 2 3 4 5 6 7 8 9 10 ...  
> $ hits    : int  0 0 0 0 0 0 0 0 0 0 ...  
> $ geo     : Factor w/ 1 level "world": 1 1 1 1 1 1 1 1 1 1 ...  
> $ keyword: Factor w/ 1 level "coronavirus": 1 1 1 1 1 1 1 1 1 1 ...  
> $ gprop   : Factor w/ 1 level "web": 1 1 1 1 1 1 1 1 1 1 ...
```



```
> 'data.frame': 340 obs. of 8 variables:
```

```
> $ symbol  : Factor w/ 7 levels "^GDAXI","^GSPC",...: 2 2 2 2 2 2 2 ...  

> $ date    : Factor w/ 50 levels "2020-01-02","2020-01-03",...: 1 2 3 4 5 6 7 8 9 10 ...  

> $ open    : num 3245 3226 3218 3242 3239 ...  

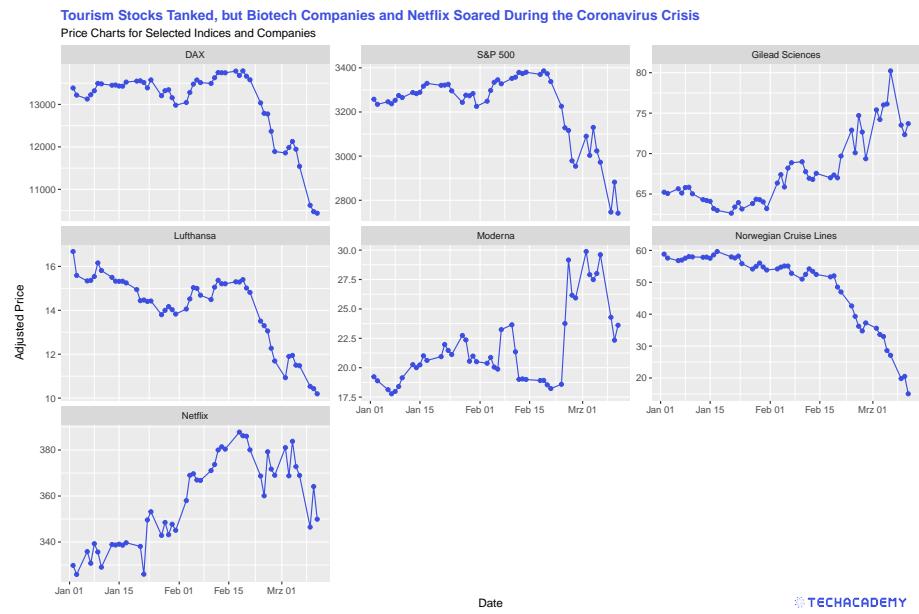
> $ high    : num 3258 3246 3247 3245 3267 ...  

> $ low     : num 3236 3222 3215 3232 3237 ...  

> $ close   : num 3258 3235 3246 3237 3253 ...  

> $ volume  : num 3.46e+09 3.46e+09 3.67e+09 3.42e+09 3.72e+09 ...  

> $ adjusted: num 3258 3235 3246 3237 3253 ...
```



```
> 'data.frame': 20200 obs. of 7 variables:  

> $ id      : int 1 2 3 4 5 6 7 8 9 10 ...  

> $ province : Factor w/ 297 levels "Adams, IN","Alachua, FL",...: NA NA NA NA NA 20 11 ...  

> $ country  : Factor w/ 114 levels "Afghanistan",...: 106 57 97 75 66 23 7 7 7 21 ...  

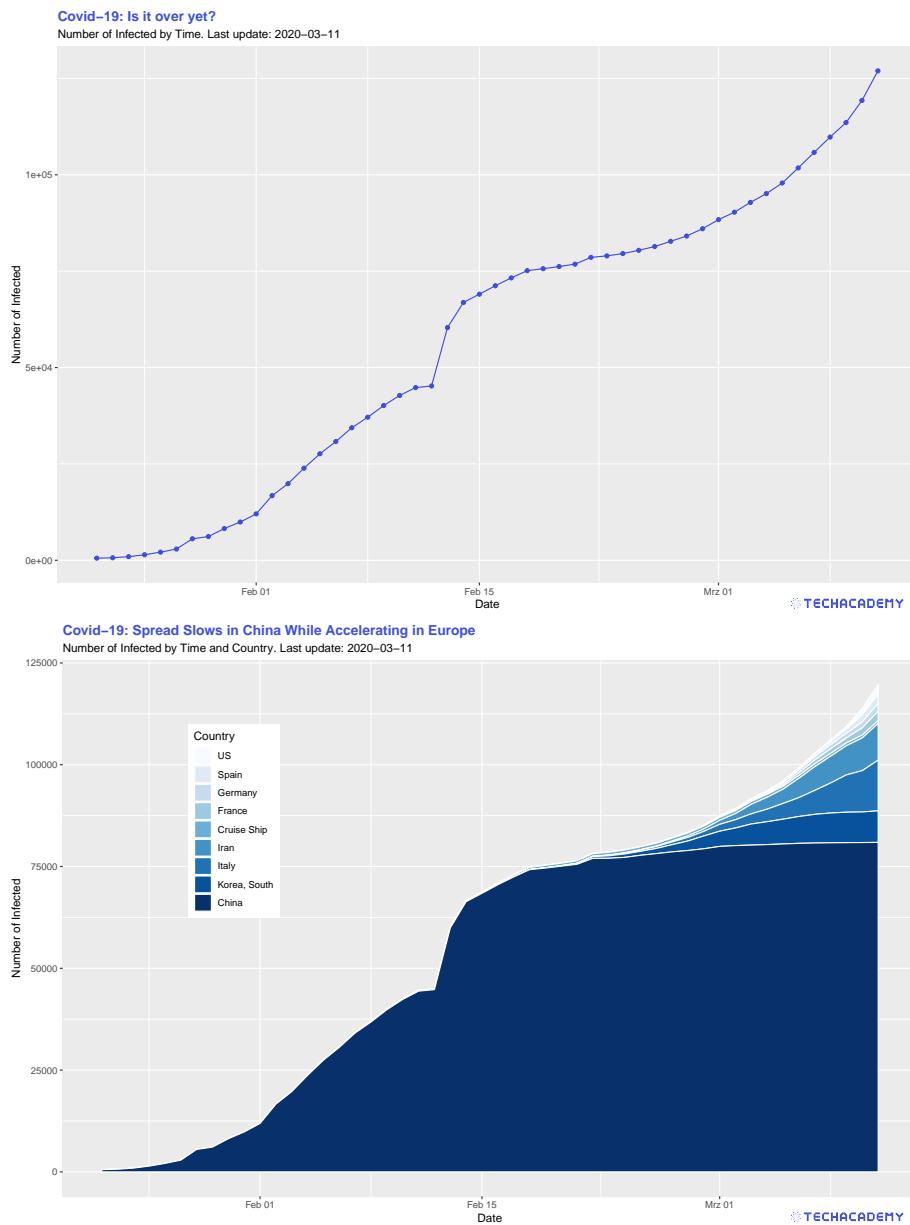
> $ latitude : num 15 36 1.28 28.17 2.5 ...  

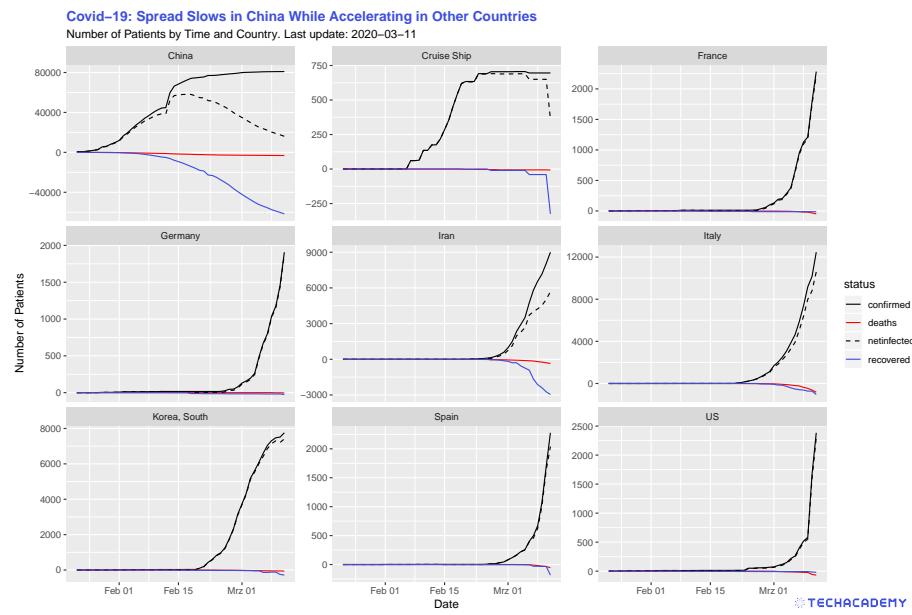
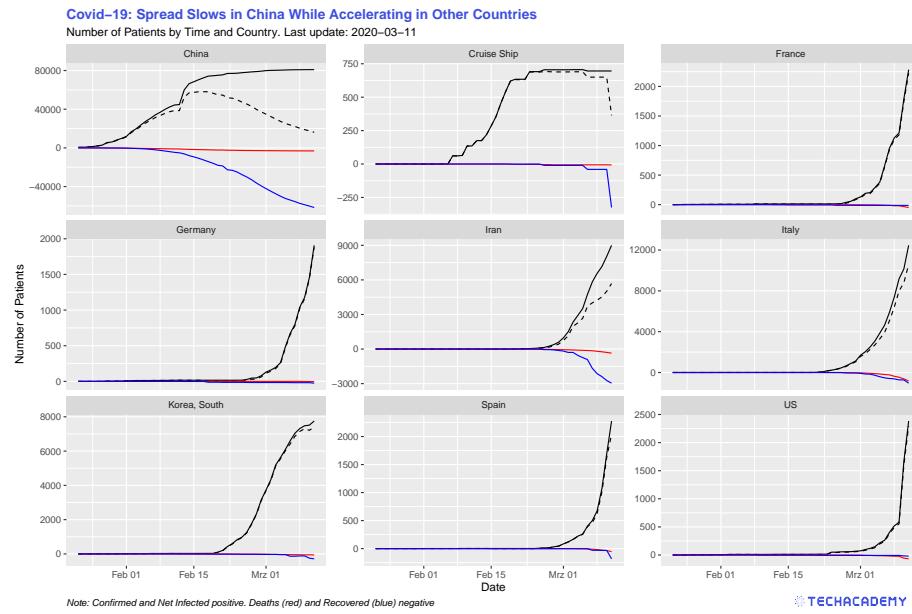
> $ longitude: num 101 138 103.8 84.2 112.5 ...  

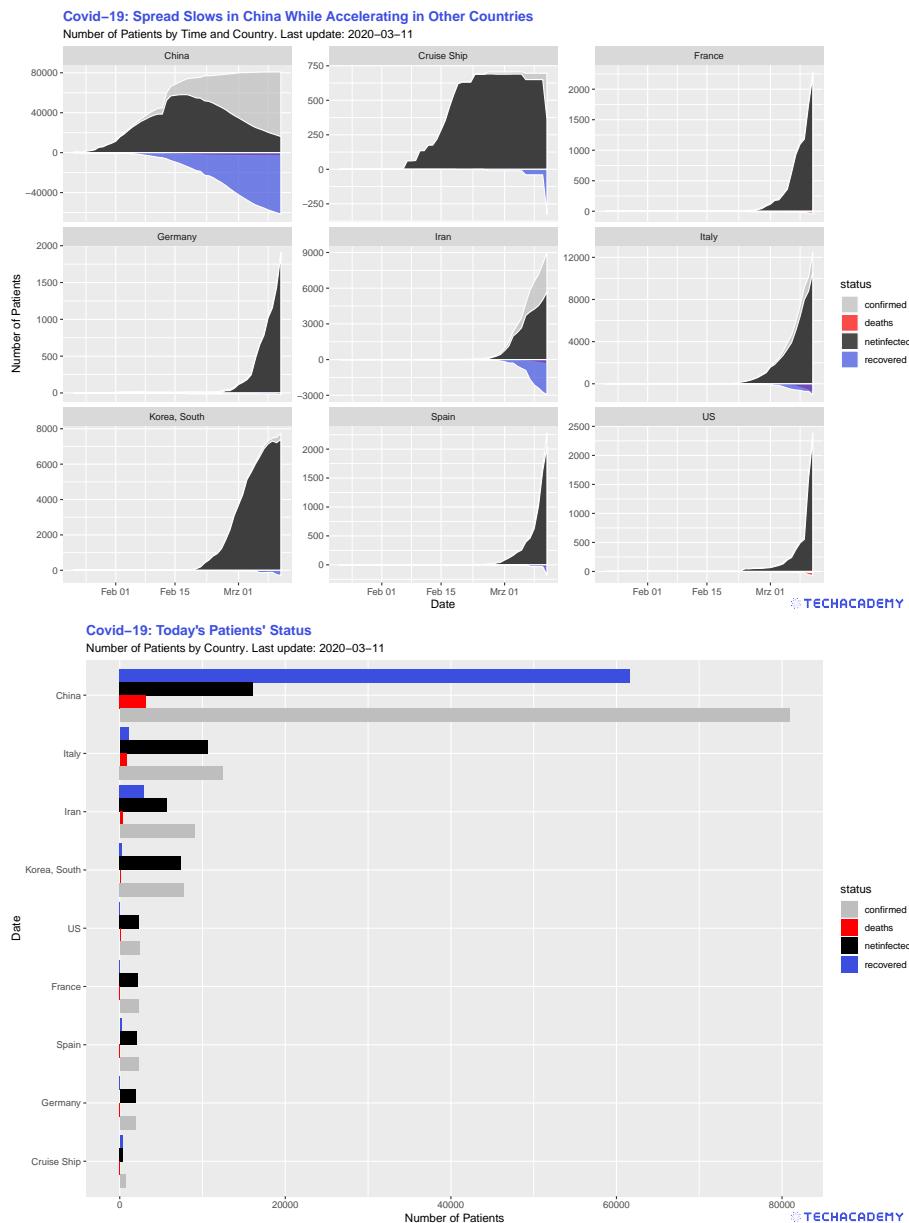
> $ time    : Factor w/ 50 levels "2020-01-22","2020-01-23",...: 1 1 1 1 1 1 1 1 1 1 ...  

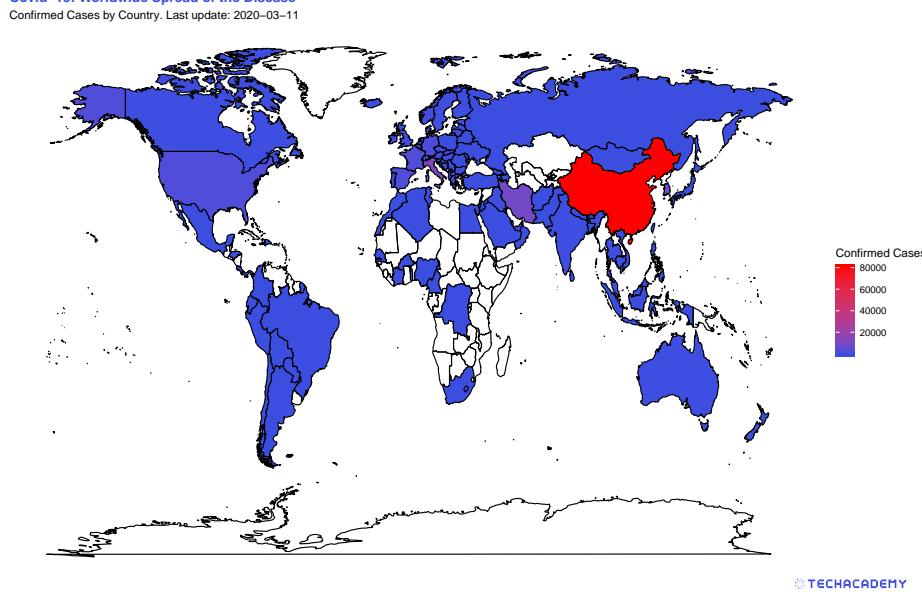
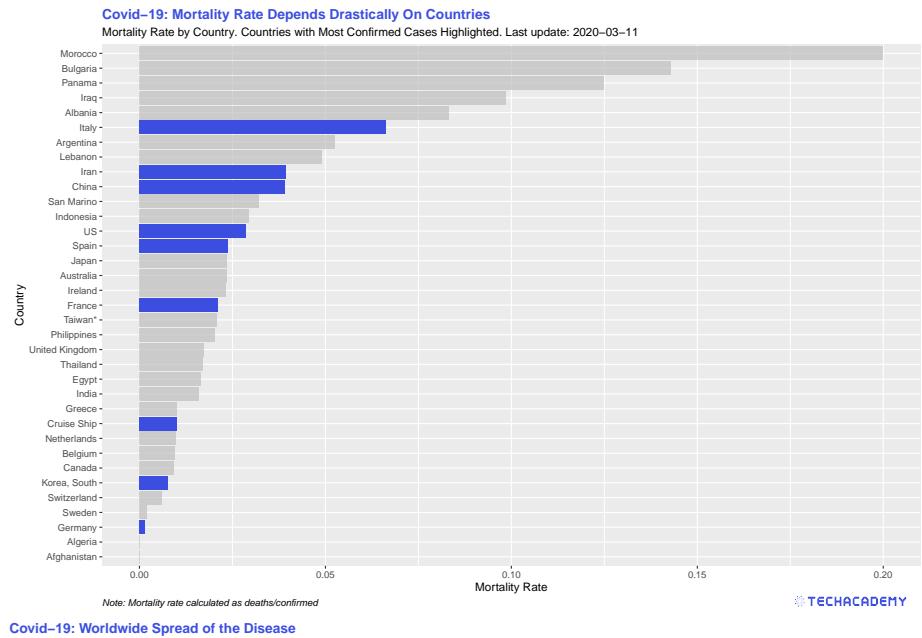
> $ confirmed: int 2 2 NA NA NA NA NA NA NA ...  
  

> [1] 0
```

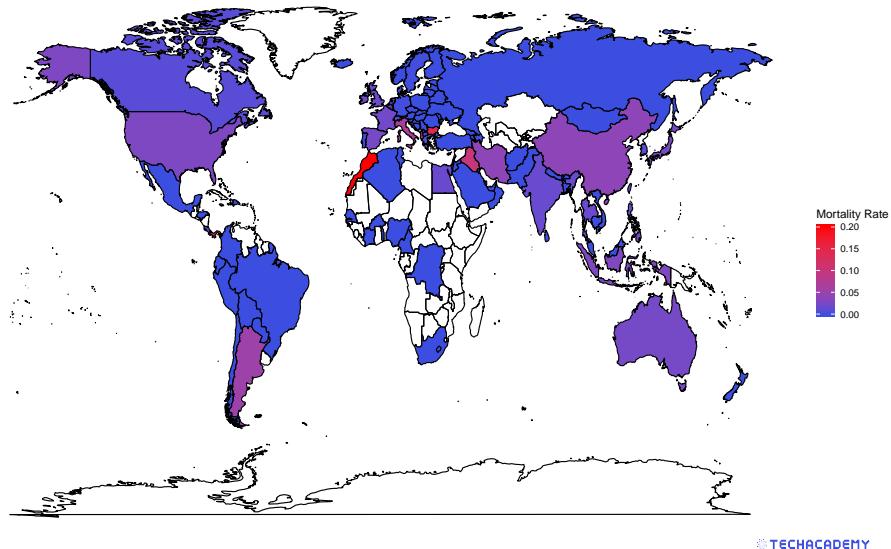








**Covid-19: Mortality Rate Varies Largely**  
Mortality Rate by Country. Last update: 2020-03-11



TECHACADEMY

**Covid-19: Spread of the Disease in Europe**  
Confirmed Cases by Country. Last update: 2020-03-11



TECHACADEMY



# Chapter 3

## Explorative Datenanalyse – Lerne den Datensatz kennen

Wir haben in deinem Workspace bei RStudio.cloud bereits ein sogenanntes Assignment (Template Airbnb) hochgeladen. Wenn du jetzt ein neues Projekt innerhalb des Workspaces “Class of ’19/20 | TechAcademy | Data Science with R” erstellst, öffnet sich dein eigener Workspace, in dem wir schon ein paar Vorbereitungen für die erfolgreiche Bearbeitung getroffen haben. So sind einerseits schon die benötigten Datensätze in deinem Working Directory abgelegt. Außerdem haben wir eine RMarkdown Datei erstellt, in der du strukturiert deine Aufgaben coden und letztendlich berichten kannst. Öffne also als ersten Schritt die Datei `Markdown_Airbnb.Rmd`.

### 3.1 Datenimport, -bereinigung, -transformation und Lineplot (Calendar)

In diesem Abschnitt wendest du die grundlegendsten Operationen an: Importieren, Bereinigen, Transformieren und am Schluss eine erste einfache Visualisierung des Datensatzes. Zuerst muss der Datensatz aus deiner Ordnerstruktur in den Workspace geladen werden. Importiere den `calendar` Datensatz in deinen Workspace und benenne das Objekt danach.

Verschaffe dir nun einen Überblick über den Datensatz. Wie ist dieser aufgebaut und welche Variablen sind darin enthalten? Was fällt dir dabei auf? Dafür kannst du zum Beispiel folgende Funktionen nutzen:

Bevor du dir überlegst, welche statistischen Methoden für die Vorhersage

## 18 CHAPTER 3. EXPLORATIVE DATENANALYSE – LERNE DEN DATENSATZ KENNEN

geeignet sein könnten, ist es immer nützlich sich die Zusammenhänge zwischen den Variablen visuell darzustellen. Wie bereits oben angesprochen nennt man diesen Prozess Exploratory Data Analysis (EDA).

Bevor wir damit starten können, müssen wir den Datensatz zuerst ein wenig bearbeiten, damit die Funktionen diesen verarbeiten können. Das klingt einfacher als gedacht — sehr oft sind die Daten in einem unbrauchbaren Format. In unserem Fall ist zum Beispiel die Preis-Variablen `price` als string-Variablen gespeichert. Damit R diese verarbeiten kann, müssen wir ein Zeichen aus den Beobachtungen entfernen und die Variable in ein `numeric` Format bringen. Dafür kannst du die Funktionen `str_remove_all()` oder `gsub()` in Kombination mit `as.numeric()` verwenden.

Damit der nächste Schritt leichter fällt, müssen wir noch die logische Variable `available` von einem `factor` zu einem `boolean` Datentyp konvertieren. Nutze dafür die `ifelse()` Funktion und ersetze jeweils “f” und “t” mit den dazugehörigen logischen Werten `FALSE` bzw. `TRUE`. Transformiere als letzten Schritt die Datumsvariable von in das R Datumsformat `date`. Erinnerst du dich an die `as.numeric()` Funktion? R besitzt eine ähnliche Funktion zur Konvertierung in Datums-Variablen.

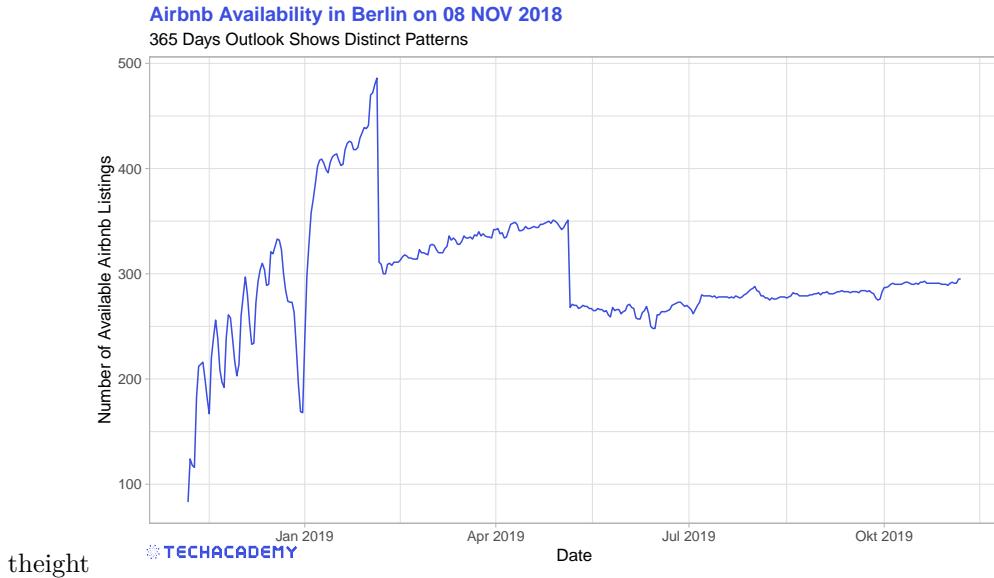
Unser Ziel ist es nun, die Anzahl der verfügbaren Airbnb Apartments über das nächste Jahr in einem einfachen Lineplot darzustellen. Jedoch ist unser Datensatz dafür noch nicht im richtigen Format – aktuell hat jedes Apartment einen Eintrag für jeden der kommenden 365 Tage und jeweils eine Variable, die uns anzeigt, ob das Apartment an diesem Tag verfügbar ist. Unser transformierter Datensatz soll jedoch diese Informationen zusammenfassen und nur einen einzigen Eintrag für jeden Tag im nächsten Jahr sowie die aggregierte Anzahl an verfügbaren Airbnbs in Berlin enthalten. Das R package `dplyr` ist die erste Wahl für diese Art von Transformations-Aufgaben. Falls du davon noch nichts gehört hast, absolviere den DataCamp Kurs dazu und/oder zu Exploratory Data Analysis. Nutze die `dplyr` Funktionen `group_by()` und `summarise()` dafür und speichere den daraus resultierenden Datensatz in einem neuen data frame ab, den du z.B. `avail_by_date` nennen kannst. Nutze für diese Aufgabe gerne das `dplyr` Cheat Sheet, auf welchem die Transformationen visuell dargestellt werden.

Jetzt haben wir den Datensatz in ein brauchbares Format gebracht und können die Ergebnisse visualisieren. Bevor du dich an das Coden machst, überlege dir, was du von diesem Plot erwartest und wie er aussehen könnte. Vergleiche diese Erwartungen dann mit dem tatsächlichen Plot und versuche die markanten Stellen mit deiner Intuition zu erklären. Plotte dafür einen einfachen Lineplot, welcher die Anzahl der verfügbaren Apartments über die nächsten 365 Tage zeichnet. Du kannst dafür das `base` Package mit der Funktion `plot()` oder gleich das umfangreiche und sehr flexibel einsetzbare Grafikpackage `ggplot2` verwenden.

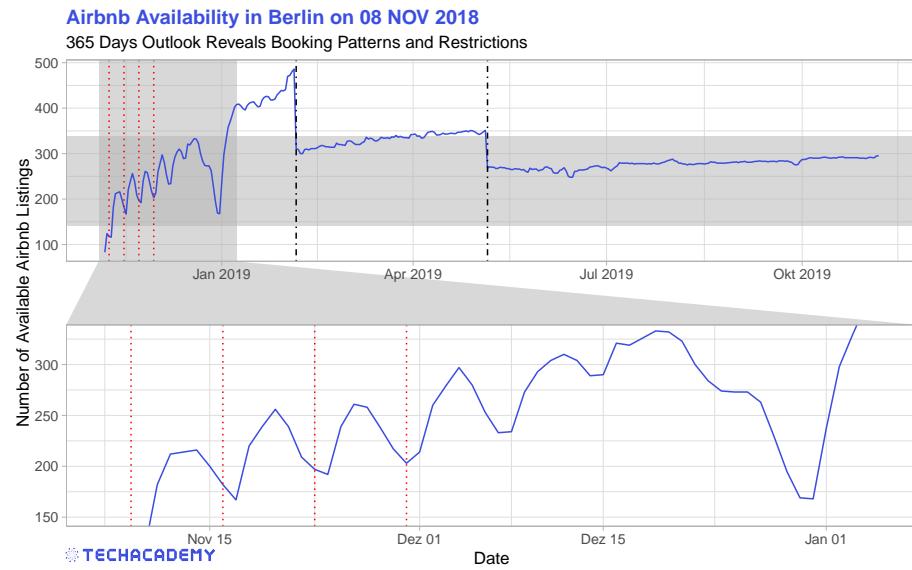
Starte einfach und verfeinere dann deinen Plot. So in etwa kann dein erstes

### 3.1. DATENIMPORT, -BEREINIGUNG, -TRANSFORMATION UND LINEPLOT (CALENDAR)19

Ergebnis ausschauen:



Überlege dir nun, was der Verlauf des Graphen bedeutet: Warum sind so viele regelmäßige Muster in der Verfügbarkeit? Und warum gibt es immer wieder deutliche und abrupte Verringerungen der Verfügbarkeit? Unten siehst du einen Beispielplot, wie man mit ggplot2 einen einfachen Plot verfeinern kann und somit die Aufmerksamkeit der Betrachter auf bestimmte Muster lenken kann. Halte dich jedoch nicht zu lange mit kleinen Feinheiten und Spielereien auf – falls du am Schluss noch Zeit hast, kannst du dich noch einmal damit befassen.



## 3.2 Visualisiere individuelle Airbnb-Angebote (listings)

Nachdem wir uns im ersten Schritt die Verfügbarkeit über die Zeit hinweg mithilfe des `calendar`-Datensatzes angesehen haben, möchten wir nun etwas mehr über die Preisstrukturen der zur Verfügung stehenden Apartments herausfinden. Dazu benötigen wir den `listings`-Datensatz, welchen wir wie gewohnt in den Workspace laden können. Beim genaueren Betrachten fällt auf, dass die Spalte `price` und `cleaning_fee` ein \$-Zeichen enthalten. Dies ist für uns zwar wichtig, damit wir wissen, um welche Währungen es sich handelt. Jedoch kann R damit nichts anfangen und weiß nicht, wie der String in eine Zahl umgewandelt werden soll.

Daher muss analog zur ersten Aufgabe für diese Spalten das Dollarzeichen gelöscht werden. Hinzu kommt, dass in manchen Spalten ein Komma als Tausender-Trennzeichen verwendet wird. Diese Kommas in der `price`-Spalte können auf die gleiche Weise entfernt werden. Da du die gleiche Vorgehensweise für die Spalten `price` und `cleaning_fee` benutzt, bietet sich hier ein `for-loop` an. Implementiere diese Schleife in einer selbst geschriebenen Funktion `clean_price()`, damit du dir besonders beim Vorhersageteil viele Zeilen Code sparst.

Da wir jetzt den Datensatz gesäubert haben, können wir uns die Preisstruktur der verschiedenen Stadtteile genauer anschauen. Zunächst möchten wir wissen, wie hoch der Durchschnittspreis und die zugehörige Standardabweichung für jeden Stadtteil ist. Erstelle dazu eine Liste mit den Namen der verschiedenen Stadtteile und dem `mean` sowie der `sd`. Verwende dafür zum Beispiel wieder die `dplyr`-Funktionen `group_by()` und `summarise()`.

Nun möchten wir die Verteilung der Preise im durchschnittlich teuersten und im günstigsten Stadtteil gegenüber stellen. Überlege dir dafür, welche Art von Diagramm du im Kurs kennengelernt hast und hier am meisten Sinn ergibt. Hast du das Diagramm erstellt, musst du vermutlich einen Teil der Ausreißer mit extrem hohen Preisen herausfiltern, um einen aussagekräftigen Plot zu erhalten. Das Filtern kannst du aber ganz einfach in den Plot-Spezifikationen mit `xlim` bzw. `ylim` durchführen.

Bei Airbnb können die verfügbaren Apartments nach Preis sortiert werden und dem Kunden in der entsprechenden Reihenfolge angezeigt werden. Eine Methode, um in dieser Rangliste weiter oben zu landen, ist es, einen günstigen Preis anzugeben und dafür eine höhere Reinigungsgebühr (Cleaning Fee) zu verlangen. Können wir dieses Verhalten in unserem Datensatz erkennen? Erstelle dafür eine zusätzliche Spalte im `data.frame` mit dem Namen `price_and_clean`, in der du beide einzelnen Preise addierst. Untersuche nun, wie sich die Preisverteilung in den beiden zuvor untersuchten Stadtteilen verändert. Stelle dazu beispielsweise den Preis sowie Preis + Reinigungsgebühr eines Stadtteils in einem Diagramm gegenüber. Was kannst du hier beobachten?

### 3.3 Merge zwei Datensätze anhand einer ID (listings, reviews)

Im vorherigen Teil hast du den `listings` Datensatz genauer kennen gelernt und visualisiert. Eine wichtige Information ist jedoch nicht in diesem Datensatz enthalten: Wie beliebt sind die einzelnen Apartments? Als Messgröße dafür verwenden wir die Anzahl der Bewertungen auf Airbnb. Diese Variable könnte später für die Preisvorhersage sehr wichtig werden. Zum Glück haben wir einen weiteren Datensatz `reviews`, indem zu jeder einzelnen Bewertung die Wohnungs-ID sowie das Datum gespeichert hat. Unser Ziel ist es jetzt, für jedes einzelne Apartment die Anzahl der Bewertungen zu zählen und diese in einem Datensatz abzuspeichern. Da wir auch im `listings` Datensatz die ID finden, können wir anhand dieser Variable die zwei Datensätze zusammenführen.

Lese zuerst den neuen `reviews` Datensatz in deinen Workspace ein und schaue ihn dir mit den bekannten Funktionen genauer an. Zähle nun die Anzahl der Bewertungen je Apartment. Tipp: Dies funktioniert sehr unkompliziert mit der `table()` Funktion oder mit `group_by()` und `summarise()`. Beachte jedoch, dass du das Ergebnis der `table()` Funktion noch in ein `data.frame`-Format für die weitere Verarbeitung bringen musst. Schaue dir nun den neuen Datensatz an. Hat jede ID genau einen Eintrag mit der Anzahl der Bewertungen?

Damit du die Datensätze mergen kannst, musst du die neu generierten Variablen in dem neuen Datensatz noch umbenennen. Nenne die Wohnungs-ID analog zu dem `listings` Datensatz `id`, sowie die Anzahl der Bewertungen `n_reviews`. Dies klappt einfach mit der Funktion `rename()` aus dem `dplyr` Package.

Table 3.1: Aggregate Number of Reviews

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

Wenn dein Datensatz so aussieht, kannst du ihn mit `listings` zusammenführen.

Schaue dir den neuen Datensatz an. Ist die neue Variable `n_reviews` im richtigen Datentyp?

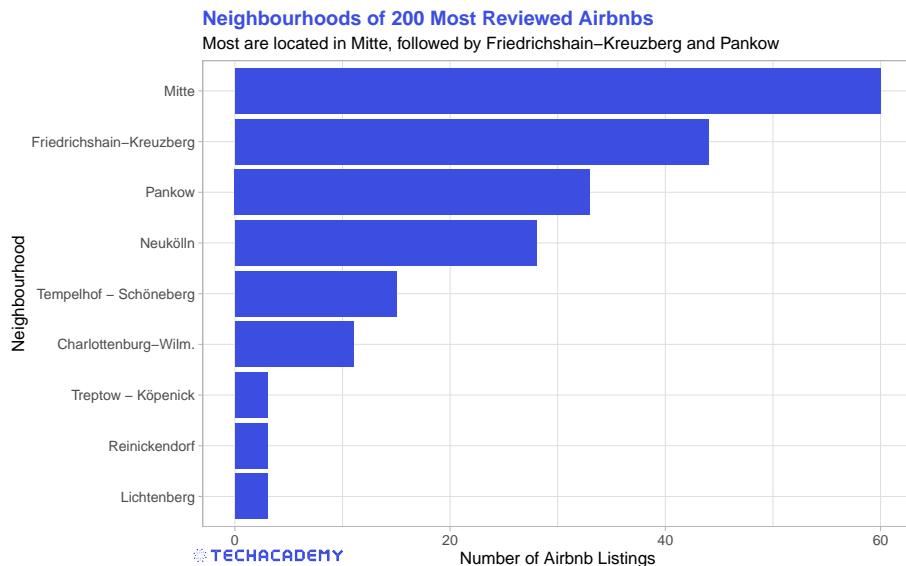
Wir wollen uns jetzt einen kleineren Teil des Datensatzes genauer anschauen: Was haben die beliebtesten Apartments gemeinsam? Als Indikator für die Be-

## 22CHAPTER 3. EXPLORATIVE DATENANALYSE – LERNE DEN DATENSATZ KENNEN

liebtheit eines Angebotes verwenden wir die vorher generierte Anzahl an Bewertungen `n_reviews`.

Extrahiere die 200 am meisten rezensierten Apartments. Eine Herangehensweise dafür ist es, den Datensatz erst in absteigender Reihenfolge nach `n_reviews` zu sortieren und dann die ersten 200 Einträge in einen neuen Datensatz zu extrahieren.

Verwende nun wieder `ggplot2`, um zu visualisieren, in welchen Stadtteilen die 200 am häufigsten rezensierten Apartments liegen. Ein Barplot bietet sich dafür an. Versuche gerne auch andere Arten von Plots, mit denen sich diese Fragestellung am besten beantworten lassen kann.



### 3.4 Geo-Daten mit Karten visualisieren (`listings_reviews`)

Wer die Internetseite von Airbnb kennt, hat bestimmt auch die Funktion gesehen, die Apartments auf einer Karte anzeigen zu lassen. Das gleiche können wir auch! Mit dem Unterschied, dass unsere Daten uns noch mehr Möglichkeiten geben anzusehen, was uns wirklich interessiert!

Wie kommen wir praktisch zu unserer Karte? Es gibt einige unterschiedliche packages, die Karten in R zeichnen können. Die folgenden Tipps beziehen sich auf `ggmap`.

Bevor du über eine API-Schnittstelle Kartenmaterial downloaden kannst, musst du die Ecken der Karte als Koordinaten definieren. Nutze zur Komplexitätsreduktion den in der dritten Aufgabe gefilterten kleineren Datensatz mit den 200

### 3.4. GEO-DATEN MIT KARTEN VISUALISIEREN (LISTINGS\_REVIEWS)23

am häufigsten rezensierten Apartments für diese Aufgabe.

Definiere zuerst die Höhe und Breite der enthaltenen Koordinaten.

Damit kannst du im nächsten Schritt die genauen Ecken relativ zu den Koordinaten im Datensatz festlegen. Dies erledigst du in einem Vektor, den du zum Beispiel `berlin_borders` nennen kannst. In diesem sind jeweils Werte für die Kanten der Karte definiert. Zu den jeweiligen Minima, bzw. Maxima der Koordinaten kannst du noch einen kleinen Sicherheitsabstand hinzufügen. Spiele später etwas mit den Faktoren herum, um einen guten Kartenausschnitt zu finden.

Danach downloaden wir den definierten Kartenausschnitt vom Dienstleister Stamen Maps mit der Funktion `get_stamenmap()` und speichern diesen in einem Objekt.

Jetzt können wir anfangen, Objekte auf der Karte zu platzieren. Damit die Karte nicht unübersichtlich wird, sollten nicht zu viele Apartments angezeigt werden. Praktisch ist dafür, dass wir in der letzten Aufgabe bereits eine Vorauswahl (Top 200) getroffen haben, die wir jetzt weiter benutzen können.

Plotte die 200 meistbewerteten Listings (aus Aufgabe 3) auf der Karte! Wenn du Aufgabe 3 noch nicht gelöst hast, wähle einfach 200 Listings nach anderen Kriterien oder zufällig aus, um die Aufgabe zu lösen. Das sollte dann in etwa folgendermaßen aussehen:

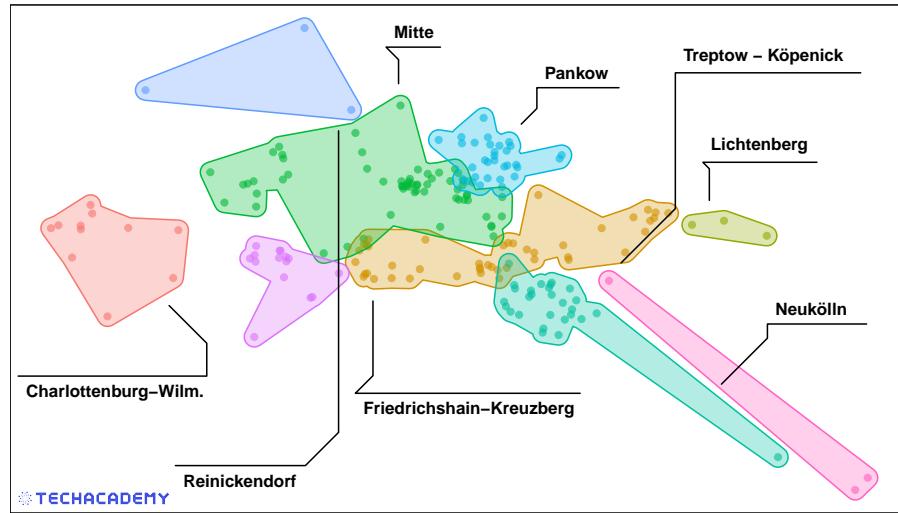


In unseren Daten gibt es über die GPS-Koordinaten hinaus noch viele Informationen zu jedem Listing. Plotte die Apartments diesmal in unterschiedlichen Farben. Nutze als Unterscheidungsmerkmal hierfür die Stadtteile (auch um leicht zu sehen, ob die Zuordnung funktioniert). Unter diesem Absatz findest du ein Beispiel, wie ein etwas fortgeschrittenes Plot dazu aussehen kann. Dieser Plot wurde mit `ggmaps` und dem zusätzlichen Package `concaveman` sowie dessen Funktion `geom_mark_hull()` erstellt, welches Polygone um ein Cluster von Koordinaten zeichnet. Es ist aber ausreichend, wenn du den gleichen Plot wie

## 24 CHAPTER 3. EXPLORATIVE DATENANALYSE – LERNE DEN DATENSATZ KENNEN

zuvor, nur mit unterschiedlichen Farben umsetzt.

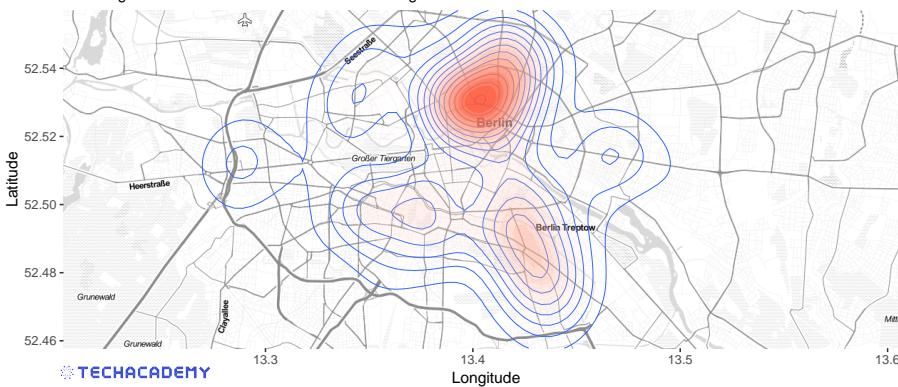
**Most Popular Airbnbs in Berlin**  
The 200 most rated apartments by neighbourhood



Für einige Analysen ist es leichter, wenn man nicht einfach nur Punkte, sondern dessen Verteilungen auf einer Karte sieht. Um zum Beispiel zu erkennen, wo sich viele Apartments auf wenig Raum befinden, kannst du dir die Apartment-Dichte anzeigen lassen. Erstelle einen solchen zweidimensionalen Dichteplot mit `geom_density2d()` sowie `stat_density2d()` auf der Karte! Falls du nicht genau weißt, wie die einzelnen Argumente befüllt werden sollen, google dich zu einer Lösung.

**Density of Most Popular Airbnbs in Berlin**

Two Large Clusters in Mitte/Pankow and Kreuzberg



Herzlichen Glückwunsch – Du hast jetzt dank viel Arbeit mit grundlegenden Daten-Transformationen und vieler Visualisierungen ein gutes Grundverständnis des Airbnb-Angebots in Berlin. Damit hast du den ersten Teil des Projektes

### **3.4. GEO-DATEN MIT KARTEN VISUALISIEREN (LISTINGS\_REVIEWS)25**

erfolgreich abgeschlossen! Wenn du in der Anfänger-Gruppe bist, sind deine Mindestvoraussetzungen hiermit erfüllt. Wir empfehlen dir aber trotzdem dringend, dich auch mit dem folgenden Teil auseinanderzusetzen. Denn wir entwickeln jetzt Methoden, um die Zukunft vorauszusagen! Klingt spannend, oder?



## Chapter 4

# Preisvorhersage – Wende statistische Methoden an

Im vorherigen Teil hast du ein Gefühl für den Datensatz bekommen. Du weißt jetzt, welche Variablen enthalten sind und kennst ein paar charakteristische Eigenschaften des Datensatzes. Noch haben wir den Datensatz aber nur visualisiert. In diesem Abschnitt gehen wir einen Schritt weiter und wenden statistische Methoden an, um den Preis von einzelnen Airbnb Apartments möglichst genau vorherzusagen.

Um dein Modell am Schluss vergleichen zu können, verwenden wir eine einheitliche Metrik, nach der wir die Preisvorhersagen auf Genauigkeit überprüfen können. In unserem Fall ist dies der Root Mean Squared Error (*RMSE*), also die Wurzel der durchschnittlich quadrierten Differenz zwischen dem vorhergesagten ( $\hat{y}_i$ ) und tatsächlichen Wert ( $y_i$ ):

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

Je näher der *RMSE* an 0 ist, desto besser sagt dein Modell die Preise vorher. Dein Ziel ist es im Folgenden also den *RMSE* deiner verschiedenen Modelle durch kontinuierliche Verbesserungen möglichst weit zu reduzieren.

Wir nutzen für den folgenden Teil drei verschiedene Datensätze, welche du im Unterordner *Full Data Set* findest. Diese basieren auf einem deutlich umfangreicherem Datensatz, der insgesamt 96 Variablen für jedes Apartment enthält. Wir haben bereits den Test/Train/Validation split der Daten vorgenommen, damit jede Gruppe mit der gleichen Aufteilung arbeitet. Hier eine kurze Beschreibung, wofür du die Datensätze benötigst:

- **train.csv** (60 %): Diesen Trainings-Datensatz verwendest du, um dein Modell zu trainieren. Das Modell lernt also die Zusammenhänge zwischen den Variablen dadurch kennen.

- **test.csv** (30 %): Mit diesem Test-Datensatz kannst du testen, wie gut dein Modell den Preis mit Hilfe von bisher nicht gesehenen Daten vorher sagt. Dabei erkennst du zum Beispiel under-/overfitting.
- **val.csv** (10 %): In diesem Validation-Datensatz haben wir die Variable `price` entfernt. Du wendest am Schluss dein bestes Modell darauf an und schickst uns deine Vorhersagen für `price`. Wir vergleichen diese dann mit den tatsächlichen (nur uns bekannten) Werten mit Hilfe des *RMSE* und küren nach Projektabgabe das beste Modell über alle Gruppen hinweg.

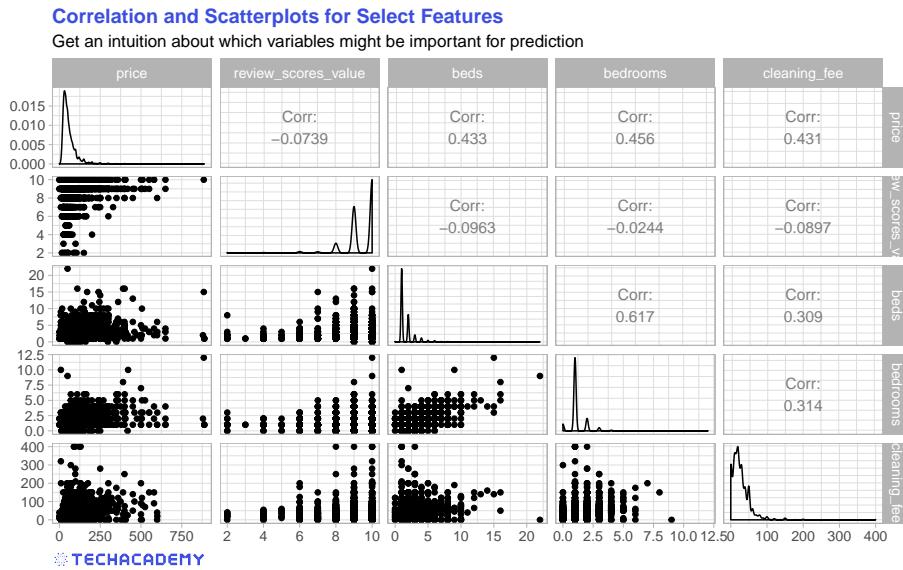
Beachte, dass in diesen drei Datensätzen wieder einige Bereinigungen notwendig sind. So sind zum Beispiel alle Preis-Variablen mit einem \$-Zeichen versehen. Wir müssen diese entfernen, damit R die Variablen als numerisch interpretieren kann und wir diese für die folgenden Modelle verwenden können. Behalte immer im Hinterkopf, dass du alle Transformationen auf alle drei Datensätze anwenden musst, da du sonst dein trainiertes Modell nicht auf den Test- sowie Validierungsdatensatz anwenden kannst.

## 4.1 Untersuche die Korrelation zwischen den Variablen näher (train)

Lade zuerst den `train.csv` Datensatz aus dem Ordner *Full Data Set* in deinen Workspace. Schaue dir jetzt die Variablennamen und die ersten Einträge an um zu entscheiden, welche Daten für die Vorhersage des Preises nützlich sein können. Wähle diese (beschränke dich zuerst auf nicht mehr als 20 Variablen) plus `price` aus und speichere diese in einem neuen `data.frame`.

Wie stehen einzelne Variablen miteinander in Verbindung? Sprich inwiefern korrelieren die Variablen des Datensatzes miteinander? Das herauszufinden ist enorm wichtig für die Entscheidung, welches Modell du später anwenden kannst. Ein guter Anfang ist es, eine Korrelationsmatrix zu erstellen. Ein Teil dafür ist die Funktion `cor()` aus dem `base` package. Selektiere alle numerische Variablen in deinem Datensatzes mit Hilfe von `sapply()` und erstelle eine Korrelationsmatrix.

Einen sehr praktischen Plot zur Visualisierung von Zusammenhängen zwischen vielen Variablen liefert das Package `GGally` mit der Funktion `ggpairs()`. Wähle die vier Variablen (und `price`) aus, die deiner Meinung nach am meisten den Preis beeinflussen und erstelle einen `ggpairs`-Plot. Beachte hierbei, dass der Plot schnell unlesbar wird und lange zum Erstellen braucht, sobald du deutlich mehr als fünf Variablen plottest.



Welche deiner untersuchten Variablen korreliert am meisten mit dem Preis und welche scheinen eher unabhängig vom Preis zu sein? Du hast jetzt einen ersten Eindruck, welche Variablen für dein Modell wichtig werden könnten. Kommen wir also zu deinem ersten Preis-Vorhersage-Modell!

## 4.2 Erste Vorhersagen mit einfachen Regressionsmodellen (train)

Jetzt kannst du dich mit deinen Statistik-Kenntnissen austoben: Du benötigst jetzt ein Verfahren, wie du den Preis eines Airbnb Apartments an einem bestimmten Tag vorhersagen kannst.

Eine erste sehr einfache Herangehensweise wäre den Durchschnitt der Nachfrage als erste Vorhersage zu verwenden. Ziemlich sicher ist das jedoch nicht die beste Vorhersage. Deine vorhergesagter Preis wäre in diesem Fall über alle Tage gleich und würde alle Faktoren, die den Preis beeinflussen, außer Acht lassen.

Schon einmal was von einer linearen Regression gehört? Das wäre ein deutlich besserer Ansatz. Jetzt kannst du deine Statistik-Skills ausspielen. Stelle zuerst ein Modell mit der abhängigen Variable `price` auf. In der vorherigen Aufgabe hast du unterschiedliche Variablen untersucht. Suche dir jetzt diejenige Variable mit der höchsten Korrelation zum Preis aus und verwende diese als einzige unabhängige Variable.

Dein erstes Regressionsmodell könnte zum Beispiel so aussehen:

$$price = \beta_0 + \beta_1 bedrooms + \epsilon$$

In R kannst du eine einfache lineare Regression mit der Funktion `lm()` imple-

mentieren. Die Ergebnisse davon gibst du dann mit der *summary()* Funktion aus.

Hat deine unabhängige Variable einen statistisch signifikanten Einfluss auf den Apartment-Preis? Vermutlich ja, denn wir haben als einzige die am höchsten zum Preis korrelierte Variable ausgewählt. Wenn wir jedoch bei diesem sehr vereinfachten Modell bleiben, begehen wir einen typischen Fehler: Den sogenannten Omitted Variable Bias (OVB). Grob vereinfacht gesprochen vernachlässigen wir (im Statistik-Jargon: kontrollieren nicht für) Variablen, die einen signifikanten Einfluss auf die abhängige Variable haben. Man könnte vermuten, dass andere Einflussfaktoren auch eine große Rolle bei der Preisbildung haben. Wenn wir diese also nicht mit aufnehmen, ist die Schätzung des Effektes von *bedrooms* verzerrt und damit schlecht zu gebrauchen. In diesem Fall ist das vorerst kein großes Problem für uns, da wir nicht an kausalen Effekten, sondern ausschließlich an einer möglichst guten Vorhersage interessiert sind. Deinem Statistik-Prof würden bei so einem Modell ziemlich sicher die Haare zu Berge stehen. Nichtsdestotrotz wird dieses Modell mit nur einer einzigen erklärenden Variable den Preis nicht unbedingt gut vorhersagen.

Eine Lösungsmöglichkeit ist, die vernachlässigten Variablen einfach mit in das Modell aufzunehmen — wie praktisch, dass diese auch schon in dem Datensatz enthalten sind. Stellen wir also ein etwas umfangreicheres Modell auf, das die noch eine weitere Variable mit aufnimmt:

$$price = \beta_0 + \beta_1 bedrooms + \beta_2 cleaning\_fee + \epsilon$$

Vergleiche nun die Ergebnisse der beiden Modelle. Erklärt das umfangreichere Modell einen höheren Anteil der Varianz im Preis? Sprich welches Modell weist dem höheren Wert für das Bestimmtheitsmaß  $R^2$  aus? Tipp: Solche LaTeX-Tabellen kannst du einfach mit dem **stargazer** Package in dein RMarkdown Dokument mit aufnehmen.

### 4.3 Von Training zu Testen – Treffe Vorhersagen

Jetzt hast du dein erstes Modell mit dem Trainings-Datensatz *trainiert*. Doch wie gut geht das Modell mit Daten um, die es noch nicht gesehen hat? Das ist ein sehr wichtiger Test um die Qualität deines Modells zu bewerten.

Hat dein Modell nur die vorhandenen Muster im Trainings-Datensatz „auswendig“ gelernt? Dann wären die Zusammenhänge aus dem Trainings-Datensatz nicht übertragbar auf den Test-Datensatz. Beim sogenannten Overfitting hat das Modell zu nah am Trainings-Datensatz gelernt und liefert deshalb schlechte Vorhersagen bei unbekannten Daten — zum Beispiel in deinem Test- und Validierungs-Datensatz.

Auf der andern Seite gibt es auch das Problem Underfitting: Dein Modell hat die tatsächlichen Zusammenhänge der Daten nicht ausreichend gelernt und sagt deshalb in dem Test-Datensatz schlecht voraus. Es gilt also, die goldene

Table 4.1: Model Summary for Two Simple Linear Regression Models

	<i>Dependent variable:</i>	
	price	
	(1)	(2)
bedrooms	35.317*** (0.594)	28.715*** (0.642)
cleaning_fee		0.576*** (0.017)
Constant	18.272*** (0.790)	13.313*** (0.864)
Observations	13,488	9,245
R <sup>2</sup>	0.208	0.331
Adjusted R <sup>2</sup>	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

Mitte zwischen den beiden Problemen zu finden.

Jetzt wird die Unterscheidung zwischen Trainings- und Testdatensatz wichtig. Zur Erinnerung: wir nutzen `train`, um ein Modell zu **trainieren** und `test`, um die Qualität unseres Modells letztendlich zu **testen**.

Lade nun zusätzlich zu dem Datensatz `train`, den du bereits vorher verwendet hast, den Datensatz `test`. Um nun dein Modell an bisher ungesehenen Daten zu testen, kannst du das Modell auf den `test` Datensatz anwenden. Nutze dafür die Funktion `predict`:

Damit hast du einen Vektor mit allen Preisvorhersagen für den `test` Datensatz erstellt. Diesen kannst du jetzt mit den tatsächlichen Werten für `price` aus `test` vergleichen. Um eine einheitliche Vergleichsmetrik zu verwenden, nutze bitte folgende Funktion zur Messung deiner Vorhersagegenauigkeit.

Vergleiche nun beide Regressionsmodelle. Hat das umfangreichere Modell bessere Vorhersagegenauigkeit, also einen niedrigeren *RMSE*? Jetzt hast du einen Benckmark für deine fortgeschritteneren Modelle, den es im nächsten Teil zu schlagen gilt.

## 4.4 Wende fortgeschrittene Machine Learning-Algorithmen an

Nachdem du jetzt eine erste Vorhersage mit Hilfe eines einfachen Regressionsmodells erstellt und getestet hast, kannst du dich jetzt an fortgeschrittenere Methoden herantasten. Das Ziel ist immer noch, einen möglichst niedrigen *RMSE* beim Anwenden des Modells auf dem `test` Datensatz zu erhalten. Suche dir jetzt mindestens einen anderen Algorithmus heraus und überprüfe letztendlich, ob du dadurch eine akkurate Vorhersage (ausgedrückt durch niedrigeren *RMSE*) erhältst. Inspirationen dazu findest du bei den fortgeschrittenen DataCamp-Kursen, welche am Anfang des Leitfadens aufgelistet sind. Dir sind dabei keine Grenzen gesetzt – du kannst die Regression durch bestimmte Verfahren verfeinern (z.B. LASSO) oder gleich ein Random Forest Modell oder ein Neuronales Netzwerk aufstellen. Es ist meistens eine gute Idee, sich kurz die Funktionsweise der jeweiligen Algorithmen in Erinnerung zu rufen und zu überlegen, ob diese Methodik in diesem Fall bei einer kontinuierlichen Vorhersagevariable Sinn macht.

An dieser Stelle ist ein Hinweis angebracht: Unser Datensatz hat über viele Variablen teilweise einen substantiellen Teil an fehlenden Beobachtungen (`NA`). Einige Machine Learning Algorithmen verlangen einen vollständigen Datensatz ohne Missing Values, während andere mit einer kleineren Anzahl gut zurecht kommen. Überprüfe also zuerst, ob du die Missing Values durch ein bestimmtes Verfahren imputieren<sup>1</sup> kannst. Welche Methode dafür am besten geeignet ist, hängt stark von deinem Vorhersage-Algorithmus ab.

Zudem kannst du einen spürbaren Zugewinn an Vorhersagekraft erhalten, indem du bestehende Variablen modifizierst oder neue Variablen aus dem Datensatz generierst (“feature engineering”). Zum Beispiel könnten wir uns vorstellen, dass die Distanz eines Apartments zum Stadtzentrum einen deutlichen Einfluss auf den Preis hat. Diese Variable ist jedoch nicht in unserem Datensatz enthalten. Du kannst jedoch eine einfache Funktion schreiben, die mit Hilfe der zwei Koordinaten-Variablen die Distanz zum Zentrum Berlins berechnet und diese als neue Variable an den Datensatz anhängt.

Vergleiche immer den *RMSE* deiner fortgeschrittenen Modelle untereinander, sowie im Vergleich zu dem Benchmark Regressionsmodell.

Du hast dein bestes Modell gefunden? Dann wende wie oben die Funktion `predict()` mit deinem Gewinnermodell an – dieses mal jedoch auf den Validierungs-Datensatz `val`.

Schicke in Anhang deiner Projektabgabe einen .csv Datensatz in folgendem Format mit den beiden einzigen Variablen `id` und `predicted_price` mit.

---

<sup>1</sup> “imputieren” bedeutet, dass du deine ‘NA’ durch einen Wert ersetzt, der auf Basis der restlichen Werte oder den anderen Variablen berechnet wird. So kannst du zum Beispiel den fehlenden Wert durch eine Regression auf die restlichen Variablen vorhersagen

#### 4.4. WENDE FORTGESCHRITTENE MACHINE LEARNING-ALGORITHMEN AN33

Table 4.2: Submission Format

id	predicted_price
2015	113
2695	9
3176	137
28089647	25
5012107	78

Dies erreichst du, indem du die beiden Vektoren `id` und `predicted_price` aneinanderfügst und als .csv Datei abspeicherst.

Damit hast du alle Aufgaben bearbeitet! Wir hoffen du hattest Spaß beim Programmieren und hast einige spannende Methoden in R gelernt. Vergesse nicht, dein PDF Dokument und die eben generierte .csv Datei vor der Deadline an unsere Projektabgabe-Email-Adresse zu schicken.



## **Chapter 5**

# **Noch Fragen?**

Du kommst nicht weiter? Willst dein Modell noch weiter verbessern, weißt aber nicht genau wie? Oder dir fällt gerade nicht ein, wie man etwas bestimmtes im Code umsetzt? Schaue dir noch einmal unser Handbuch an. Dort findest du hilfreiche Hinweise, wie du in diesem Fall weiter verfahren und wo du nach einer Lösung für deine Fragestellung suchen kannst.



# Chapter 6

## Anhang

### 6.1 Aufgaben-Checkliste für das Airbnb Data Science Projekt

Exploratory Data Analysis (Anfänger + Fortgeschrittene)

1. Visualisieren der verfügbaren Apartments (calendar dataset):
  - 1.1 Wandle die Werte t/f zu `true` bzw. `false` um
  - 1.2 Aggregiere die Daten nach dem Datum
  - 1.3 Plotte die Anzahl der verfügbaren Apartments über die Zeit
2. Visualisieren des `listings` Datensatz
  - 2.1 Bereinige und transformiere die Spalten `price` und `cleaning_fee` zu `numeric`-Werten
  - 2.2 Bestimme mean und standard deviation des Apartment-Preises nach Stadtteil
  - 2.3 Visualisiere die Preisverteilung des teuersten und günstigsten Stadtteils
  - 2.4 Visualisiere, ob die `cleaning_fee` benutzt wird um die Preise versteckt zu erhöhen
3. Merging des `listings` und `review` Datensatzes
  - 3.1 Aggregiere die Reviews anhand der ID
  - 3.2 Merge `listings` und den aggregierten Reviews Datensatz
  - 3.3 Filtere Apartments und plotte die Anzahl dieser in einem Barplot nach Stadtteil

4. Visualisiere die Apartments auf einer Karte

4.1 Zeiche einen Punkt für jedes zur Verfügung stehende Apartment auf eine Karte

4.2 Färbe die Punkte unterschiedlich, je nachdem in welchem Stadtteil sie sind

4.3 Erstelle eine 2d-Density Plot der Apartments auf einer Karte

**Preisvorhersage mit statistischen Verfahren (Fortgeschrittene und ambitionierte Anfänger)**

1. Visualisiere die Korrelationen der Features mit einer Heatmap

2. Regression

2.1 Einfaches Regressionsmodel mit einer erklärenden Variable

2.2 Regressionsmodel mit zwei oder mehr Features

3. Testen der Modelle

3.1 Benutze den Testdatensatz um Preise für die Apartments vorherzusagen

3.2 Vergleiche deine Prognosen mit den tatsächlichen Preisen mit Hilfe des RMSE

3.3 Implementiere einen fortgeschrittenen Algorithmus und sende uns dein bestes Ergebnis zu

## 6.2 Beschreibung der Variablen in den Datensätzen

Name	Beschreibung
id	Identifikationsnummer
price	Preis
neighbourhood_group_cleansed	Stadtviertel
latitude	Breitengrad
longitude	Längengrad
bathrooms	Anzahl an Badezimmern
availability_30	Verfügbarkeit in den nächsten 30 Tagen (Tage)
availability_365	Verfügbarkeit im nächsten Jahr (Tage)
beds	Anzahl an Betten
bedrooms	Anzahl Schlafzimmer
review_scores_value	Bewertungspunktzahl
bed_type	Beschreibung zum Bett
minimum_nights	minimale Anzahl an Übernachtungen
maximum_nights	maximale Anzahl an Übernachtungen
property_type	Art des Angebotes: (Apartment, Zimmer,...)
is_business_travel_ready	für Geschäftsreisen geeignet?
cleaning_fee	Reinigungsgebühr
room_type	Art des Zimmers
square_feet	Größe in square feet
guests_included	Sind Gäste im Preis inbegriffen?