

# CAS Data Visualization

Autorenprojekt  
Flight Fare Visualization



# Einleitung

## Flight Fare Visualization

*Täglich gibt es durchschnittlich 93'000 Flüge, die von mehr als 9'000 Flughäfen weltweit starten und landen. Rund um die Uhr befinden sich um den Globus zwischen 8'000 und 13'000 Flugzeuge gleichzeitig in der Luft [Reference Encyclopedia].*

Jedes Flugticket hat einen Preis, der nach Buchungsdatum und Zeitpunkt des Abfluges variiert. Das Ziel von Flight Fare Visualization (FFV): Preisaffinen Flugreisenden wird primär die Frage beantwortet, wann sie ihren Flug zum besten Preis buchen.

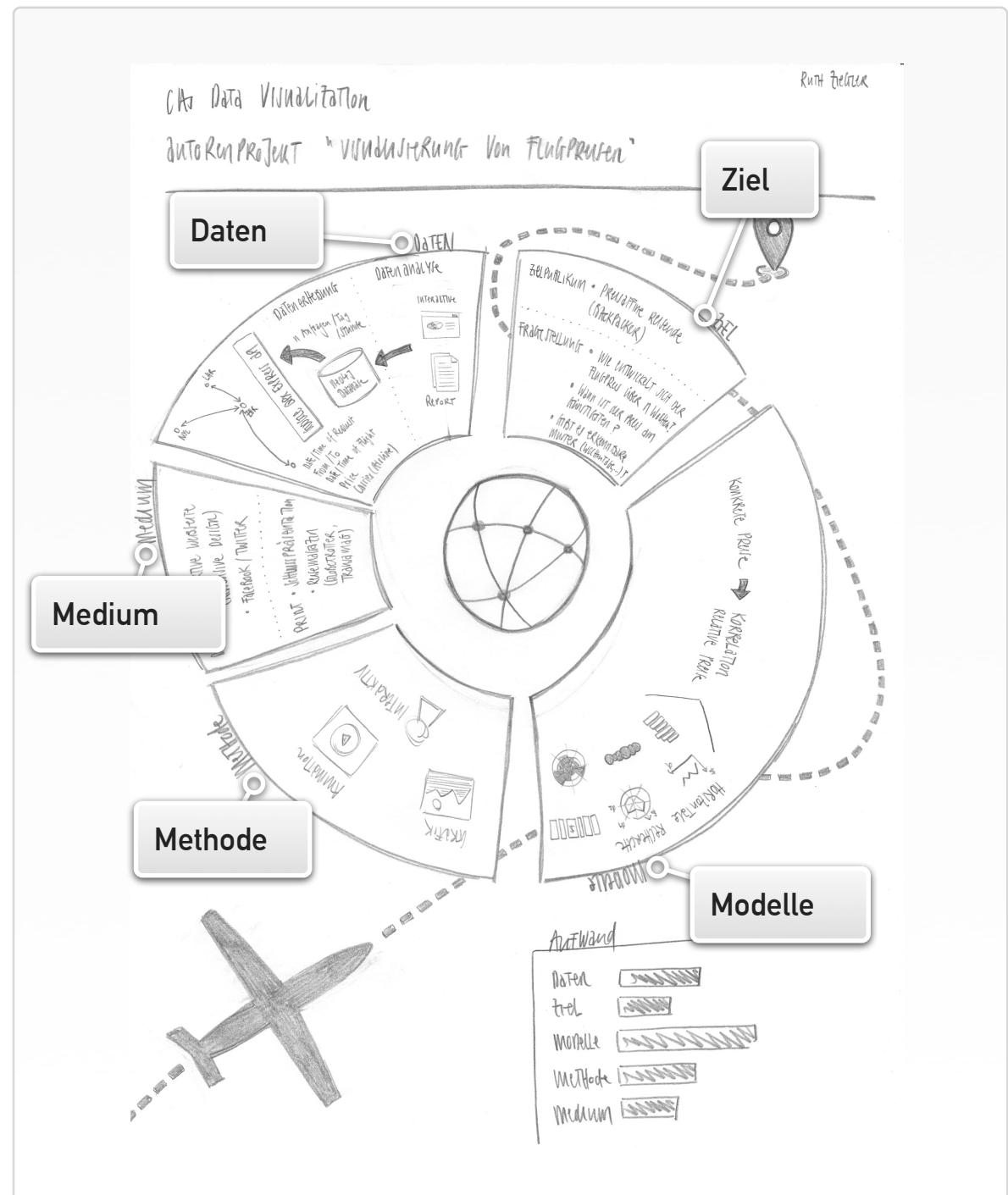
---

Die vorliegende Dokumentation beschreibt das Autorenprojekt "Flight Fare Visualization", das vom Frühling bis Herbst 2016 im CAS Data Visualization an der Hochschule für Künste HKB erarbeitet wurde.

**Das Endergebnis ist unter <http://ffv.mochila.ch> verfügbar.**

# Flight Fare Visualization

## Übersicht



# Flight Fare

# Visualization

## Fragestellung

Zu Beginn des Projektes bin ich mit einigen generellen Fragen gestartet. Im Laufe der Arbeit musste ich diese immer weiter schärfen. Der Fokus der einzelnen Fragen lag dabei immer beim Preis - ganz unter dem **Aspekt "Buchen zum besten Preis"**.

### Fragen zu Projektbeginn

- Wie entwickelt sich der Flugpreis über n Wochen?
- Wann ist der Preis am günstigsten?
- Gibt es erkennbare Muster (Wochentage, ...)?
- Welche Unterschiede gibt es zwischen den Destinationen?

### Fragen zu Projektende

Besonders während der Datenanalyse, aber auch in der visuellen Konzeption habe ich gemerkt, dass die Fragen präzise formuliert sein müssen. Nur so kann eine gezielte Visualisierungsform gefunden werden.

- An welchem Wochentag soll ich buchen?
- Wie viele Wochen vor dem Abflug soll ich buchen?
- An welchem Wochentag soll ich abfliegen?
- Wie entwickelt sich der Flugpreis über n Tage vor Abflug?
- Wie viel Geld kann ich durchschnittlich sparen, wenn ich zum richtigen Zeitpunkt buche?



**Daten**

*Im Unterrichts-Modul “Datenaufbereitung und -analyse” wurde eine Einführung in die Programmiersprache R und die Grundlagen der deskriptiven Statistik vermittelt. R dient der statistischen Datenanalyse und Visualisierung. Die Programmiersprache wurde ursprünglich von den Statistikern Ross Ihaka und Robert Gentleman an der Universität Auckland entwickelt und gilt heute als Standardsprache für statistische Problemstellungen [[Wikipedia](#)].*

R ist eine Open-Source Programmiersprache und besitzt eine aktive Community, die durch die R-Foundation gestützt wird. Mit sogenannten R-Packages kann die Funktionalität erweitert werden.

Da ich bereits fundierte Kenntnisse in anderen Programmiersprachen und Paradigmen habe (z.B. Java, Funktionale Programmierung, JavaScript), ist mir der Einstieg leicht gefallen. Insbesondere meine Erfahrungen mit der Datenbankabfangesprache SQL (Standard Query Language) halfen mir dabei, die Konzepte von Selektieren, Filtern und Gruppieren der Daten auch in komplexen Fällen anzuwenden. Die Einfachheit der Visualisierungsmöglichkeiten von R haben mich fasziniert. Sie haben mir geholfen, einen ersten Überblick über meine Daten zu verschaffen.

### Datenaufbereitung

Ein wichtiger Punkt der Datenaufbereitung ist das **Transformieren und Säubern der Ursprungsdaten**, damit diese anschliessend einfacher analysiert werden können. Hadley Wickham beschreibt diesen Prozess in „**Tidy Data**“ [[Journal of Statistical Software](#)] als „tidying data“. Dieser Prozess dient dazu, chaotische Ursprungsdaten in eine **standardisierte Repräsentation** zu überführen.

Wickham schlägt für die standardisierte Repräsentation der Daten drei Regeln vor. Werden diese eingehalten, spricht er von **Tidy Data**.

1. Jede **Variable** wird in einer eigenen Spalte abgebildet.
2. Jede **Beobachtung** wird in einer Zeile abgebildet.
3. Unterschiedliche Beobachtungsarten sind in unterschiedlichen Tabellen abzubilden.

## Deskriptive Statistik

Ziel der deskriptiven Statistik ist eine beschreibende Bestandesaufnahme, eine Informationsreduktion, eine Überprüfung von Regelmässigkeiten, ein Abschätzen von Stichprobenfehlern oder ein Aufdecken von Fehlschlüssen.

Die deskriptive Bestandesaufnahme dient dazu, **Häufigkeiten zu beschreiben** (z.B. Volkszählung).

Mittels Informationsreduktion wird eine **Komprimierung** der Daten auf einfache Kennzahlen erreicht. Dadurch kann ein Fokus auf einen zentralen Aspekt der Daten gelegt werden. Eine Reduktion kann aber unter Umständen auch wichtige Informationen vernachlässigen.

Durch den Einsatz von Statistik können **Zusammenhänge** von verschiedenen Variablen **geprüft** werden. Das Abschätzen von Stichprobenfehlern dient dazu festzustellen, ob die gesammelten **Daten einer Stichprobe verallgemeinert** werden können.

Mittels Statistik können schliesslich auch **Fehlschlüsse aufgedeckt** werden, wie beispielsweise eine Scheinkorrelation (z.B. Je mehr Feuerwehrleute einen Brand bekämpfen, umso grösser ist der Brandschaden).

Die Erkenntnisse aus dem Modul wurden in der Datenerhebung und -analyse für mein Autorenprojekt ausführlich angewendet.

# Daten

## Datenerhebung

Im Fokus der Datenerhebung stand das Sammeln der Flugdaten von einer Plattform, die Daten von verschiedenen Fluggesellschaften anbietet. Die gesammelten Flugpreise wurden für die weitere Verwendung im Projekt in einer Datenbank gespeichert.

Der Preis für einen Sitz im Flieger von A nach B wird durch unterschiedliche Kriterien beeinflusst:

- Abflugsort
- Zielort
- Nur Hinflug oder Hin- und Rückflug
- Direkt oder mit Zwischenstopp
- Klasse, wie Economy, Business oder First
- Anzahl Personen/Sitze
- Fluggesellschaft (Carrier)

Ein Flug mit den oben genannten Ausprägungen wird von verschiedenen Anbietern zu unterschiedlichen Preisen angeboten. Beispielsweise wird der Flug LX318 von Zürich nach London am 15.06.2016 um 09:50 von eBookers sowie Swiss angeboten. Die Preise können dabei unterschiedlich sein.

Um die Erhebung der Daten überschaubar zu gestalten und die Komplexität der Visualisierung zu begrenzen, habe ich mich für die Datenabfrage nach folgenden Kriterien entschieden:

1. **Abflugsort Zürich**

Es werden nur Preise zu Flügen mit Abflug-Flughafen Zürich erhoben.

2. **Einfach**

Es werden nur Preise zu einfachen Flügen (Hinflug) ermittelt.

3. **Einzelperson**

Der Flugpreis gilt immer für eine Person.

4. **Economy**

Ausschliesslich Preise der Economy-Klasse werden erhoben.

5. **Direktflüge**

Es werden nur Direktflüge berücksichtigt.

Unter Berücksichtigung dieser Abfragekriterien ergeben sich folgende mögliche Variablen für die Visualisierung:

1. **Destination**

Es werden zwanzig unterschiedliche Zielflughäfen abgefragt.

2. **Zeitpunkt des Abfluges**

Der Zeitpunkt des Abfluges lässt sich zusätzlich in folgende Variablen unterteilen:

- a. Datum
- b. Tageszeit
- c. Wochentag
- d. Kalenderwoche

3. **Zeitpunkt der Preisanfrage**

Auch der Zeitpunkt der Preisanfrage (request date) lässt sich nach Bedarf in dieselben Variablen wie unter Punkt 2. erwähnt unterteilen.

4. **Carrier**

Alle verfügbaren Fluggesellschaften pro Destination werden erhoben.

5. **Agent**

Alle verfügbaren Anbieter werden erhoben.

## Ermittlung der Destinationen

Ich habe entschieden, dass Daten für **zwanzig Destinationen** erhoben werden: zehn innerhalb und zehn ausserhalb Europas.

Die Reiseziele habe ich nach folgenden Regeln ausgewählt:

1. Es werden die Listen von Flughäfen mit dem höchsten Passagieraufkommen in Europa, bzw. weltweit als Grundlage genommen [[Wikipedia EU](#)] [[Wikipedia Non-EU](#)].
2. Pro Land gibt es nur einen Flughafen.
3. Das Land darf nicht an die Schweiz angrenzen.
4. Es muss mindestens ein Direktflug angeboten werden.
5. Es werden die ersten fünf Treffer vom oberen Ende der Liste (vgl. Punkt 1) gemäss den Punkten 2 und 3 gewählt. Analog dazu werden die ersten fünf Treffer vom unteren Ende der Liste gewählt.
6. Die Regeln werden solange wiederholt, bis je zehn Destinationen ausgewählt sind.

## Ausgewählte Destinationen, geordnet nach Grösse des Passagieraufkommens

- |                    |                     |
|--------------------|---------------------|
| 1. London (LHR)    | 1. New York (JFK)   |
| 2. Istanbul (IST)  | 2. Peking (PEK)     |
| 3. Amsterdam (AMS) | 3. Dubai (DXB)      |
| 4. Madrid (MAD)    | 4. Tokyo (NRT)      |
| 5. Moskau (SVO)    | 5. Singapur (SIN)   |
| 6. Riga (RIX)      | 6. Bangkok (BKK)    |
| 7. Reykjavik (KEF) | 7. Seoul (ICN)      |
| 8. Belgrad (BEG)   | 8. Toronto (YYZ)    |
| 9. Malta (MLA)     | 9. Mumbai (BOM)     |
| 10. Rhodos (RHO)   | 10. Sao Paolo (GRU) |

Anmerkung: Anstelle von Atlanta (ATL) habe ich mich für New York (JFK) entschieden, da nach Atlanta nur sehr wenige Direktflüge vorhanden sind.

## Applikation zur Datenerhebung

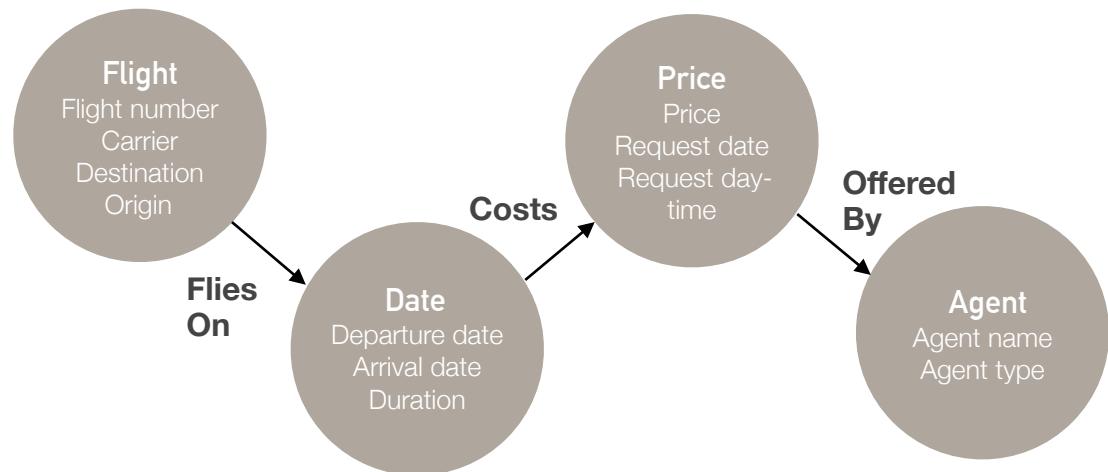
Für die Datenerhebung gemäss den vorgestellten Kriterien habe ich ein Java-Programm, genannt Crawler, entwickelt. Dieser fragt pro Tag viermal via Schnittstelle die Flugpreise für alle zwanzig Destinationen ab. Ich habe mich entschieden, die Preise jeweils drei Monate, also 90 Tage in die Zukunft abzufragen.

**Beispiel** Abfrage am Montag, 1. Juni 2016

**Resultat** Die Abfrage liefert die Flugpreise für alle Destinationen im Zeitraum vom 2. Juni bis zum 30. August 2016.

Ursprünglich wollte ich die Abfrage via Google Flight API durchführen. Google erlaubt jedoch nur 50 kostenfreie Abfragen pro Tag. Bei 20 Destinationen à 90 Tage und vierfacher Abfrage pro Tag wäre dies zu wenig gewesen. Aus diesem Grund suchte ich nach einer Alternative. Die Lösung ist die Verwendung des **Travel APIs von Skyscanner Business** [[Skyscanner Business](#)], die eine unbegrenzte Anzahl kostenfreier Abfragen zulässt.

Die Antworten von Skyscanner werden transformiert und in einer Datenbank (neo4j) in folgender Struktur gespeichert.



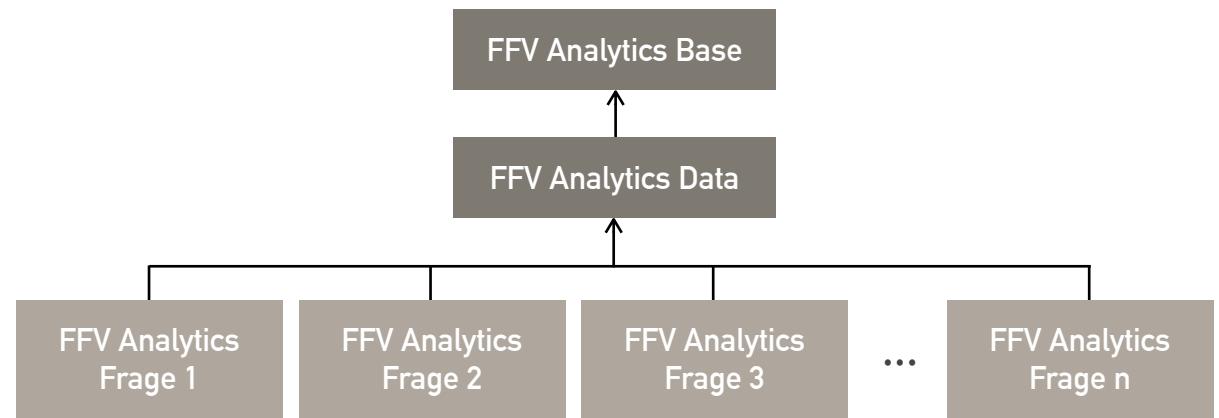
Pro Tag werden rund 28'000 Preise abgefragt (4x täglich rund 7'000). Die Daten werden vom Crawler seit dem 11. Mai 2016 gesammelt. Total sind per 13. September **33,21 Millionen Datensätze** gespeichert worden.

# Daten

## Datenanalyse

Die Datenanalyse basiert auf den über 33 Millionen Flugpreisen. Die Daten müssen so aufbereitet werden, damit diese für die Visualisierung benutzt werden können. Die Analyse der Daten wurde mit R gemacht.

Die R-Skripte habe ich folgendermassen strukturiert: In einem Base-Skript werden die Libraries geladen und Funktionen zur Verfügung gestellt. Das Data-Skript erzeugt eine Basis-Struktur der Daten. Diese Struktur wird anschliessend bei der Auswertung der einzelnen Fragen benutzt. Pro Fragestellung wird dann ein eigenes Skript erstellt.



Das Basis-Skript “FFV Analytics Data” liest die Daten ein, reichert diese mit den nötigen Berechnungen an und filtert. Ziel des Basis-Skripts ist die Datenaufbereitung als Tidy Data.

Als erstes habe ich die Zeitstempel (Datum und Uhrzeit) in Tag, Zeit sowie Wochentag aufgetrennt, da sonst keine Gruppierung nach Tag oder Wochentag

möglich ist. Dies gilt sowohl für das Anfragedatum (Request Date) wie auch das Abflugdatum (Departure Date).

Im weiteren habe ich eine neue Spalte berechnet, welche die **Delta-Tage zwischen dem Anfragedatum und dem Abflugdatum** berechnet. Diese ist für die Visualisierung zentral, da dadurch ein Vergleich zwischen den einzelnen Flügen erst möglich wird. Das folgende Beispiel verdeutlicht diese Aussage. Bei absolutem Anfrage- und Abflugdatum sind die einzelnen Flüge entsprechend verschoben. Werden jedoch die Anzahl Tage vor dem Abflug verwendet, so können die Flüge untereinander angeordnet werden.

Beispiel: Daten mit absolutem Anfrage-/Abflugdatum

10.06.	11.06.	12.06.	13.06.	14.06.	15.06.
11.06.	12.06.	13.06.	14.06.	15.06.	16.06.
12.06.	13.06.	14.06.	15.06.	15.06.	17.06.

■ Anfragedatum  
■ Abflugdatum

Beispiel: Daten mit relativem Anfragedatum (Delta-Tage)

5	4	3	2	1	15.06.
5	4	3	2	1	16.06.
5	4	3	2	1	17.06.

■ Anfragedatum  
■ Abflugdatum

Im Rahmen der Analyse wurde festgestellt, dass die unterschiedlichen Anfragezeiten (Vormittag, Nachmittag, Abend, Nacht) die Komplexität stark erhöht. Aus diesem Grund habe ich mich entschieden, die Anfragezeiten eines Request Days wegzulassen. Anstelle dessen habe ich jeweils den günstigsten Preis des Tages weiterverwendet.

Anschliessend wurden die **Daten auf komplette Serien** reduziert. Eine komplette Serie besteht dann, wenn für einen Flug die Preise für n Tage abgefragt worden sind.

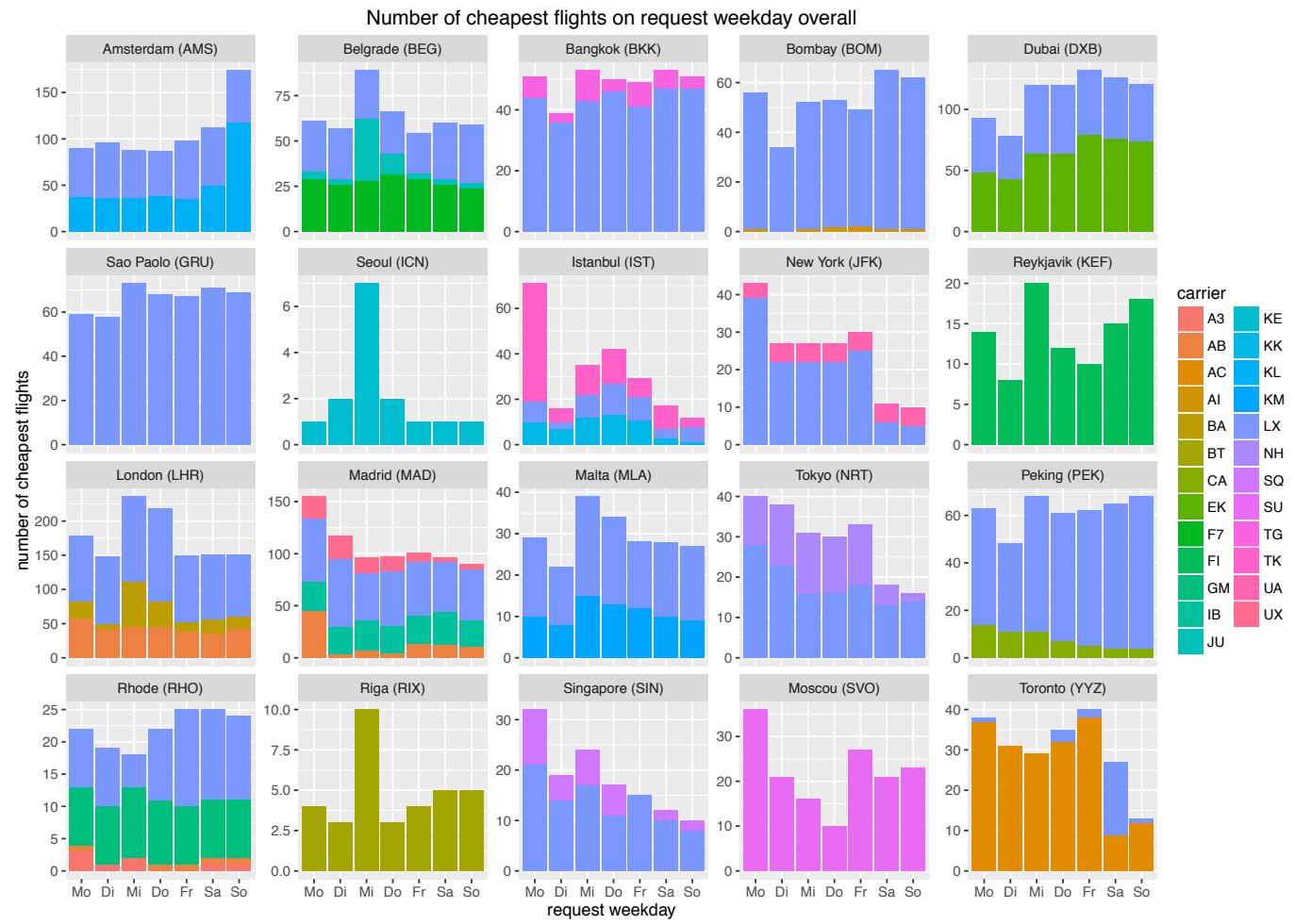
Da die Daten während der Analyse laufend weiter gesammelt wurden, habe ich die Angabe der Länge einer Serie in den Skripten variabel gestaltet. Im Laufe der Analyse habe ich dann gemerkt, dass nur ein Vielfaches von ganzen Wochen sinnvoll ist. Beispielsweise könnte die Serienlänge also den Wert 7, 14, 21, ... haben.

Die kompletten Serien waren anschliessend die Grundlage für alle weiteren Datenauswertungen. Es wurde schnell klar, dass die Formulierung der Frage sehr präzise sein muss, damit eine Auswertung in R umgesetzt werden kann.

Folgende Fragen wurden mit R ausgewertet:

- a. An welchem Wochentag soll ich buchen?
- b. An welchem Wochentag soll ich abfliegen?
- c. In welcher Woche vor dem Abflug ist es am günstigsten?
- d. Wieviel Geld kann ich durchschnittlich sparen, wenn ich zum richtigen Zeitpunkt fliege?

Die folgenden Seiten beschreiben die Details der einzelnen Auswertungen in R. Die Grafiken basieren alle auf **Daten vom 21. Juli 2016**, als Länge der Serie wurden **49 Tage** verwendet.



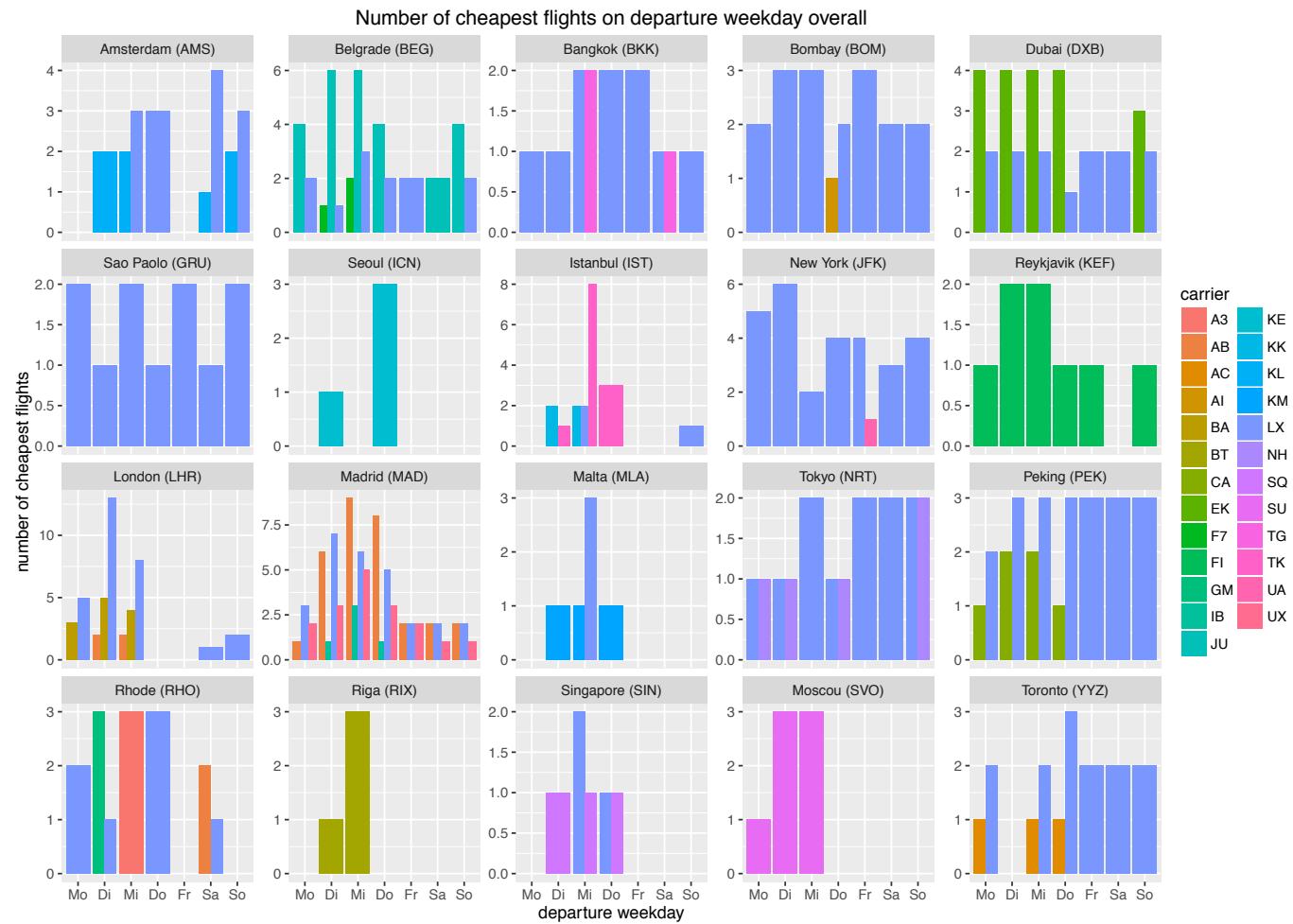
● ● ●

## a. An welchem Wochentag soll ich buchen?

Um diese Frage zu beantworten, werden die **Wochentage des Abfragedatums** gezählt. Es wird immer der Wochentag des günstigsten Preises eines Fluges in den entsprechenden "Topf" gelegt. Für jeden Wochentag gibt es einen Topf. Der beste Wochentag einer Destination ist derjenige Topf, der die meisten Einträge hat.

Aus den Daten der kompletten Serien werden zunächst pro eindeutigem Flug (Flugnummer plus Abflugdatum) die günstigsten Preise selektiert. Anschliessend erfolgt eine Gruppierung der Daten nach Destination, Carrier und Anfragedatum und es wird die Anzahl Einträge gezählt - dies entspricht dann einem Topf.

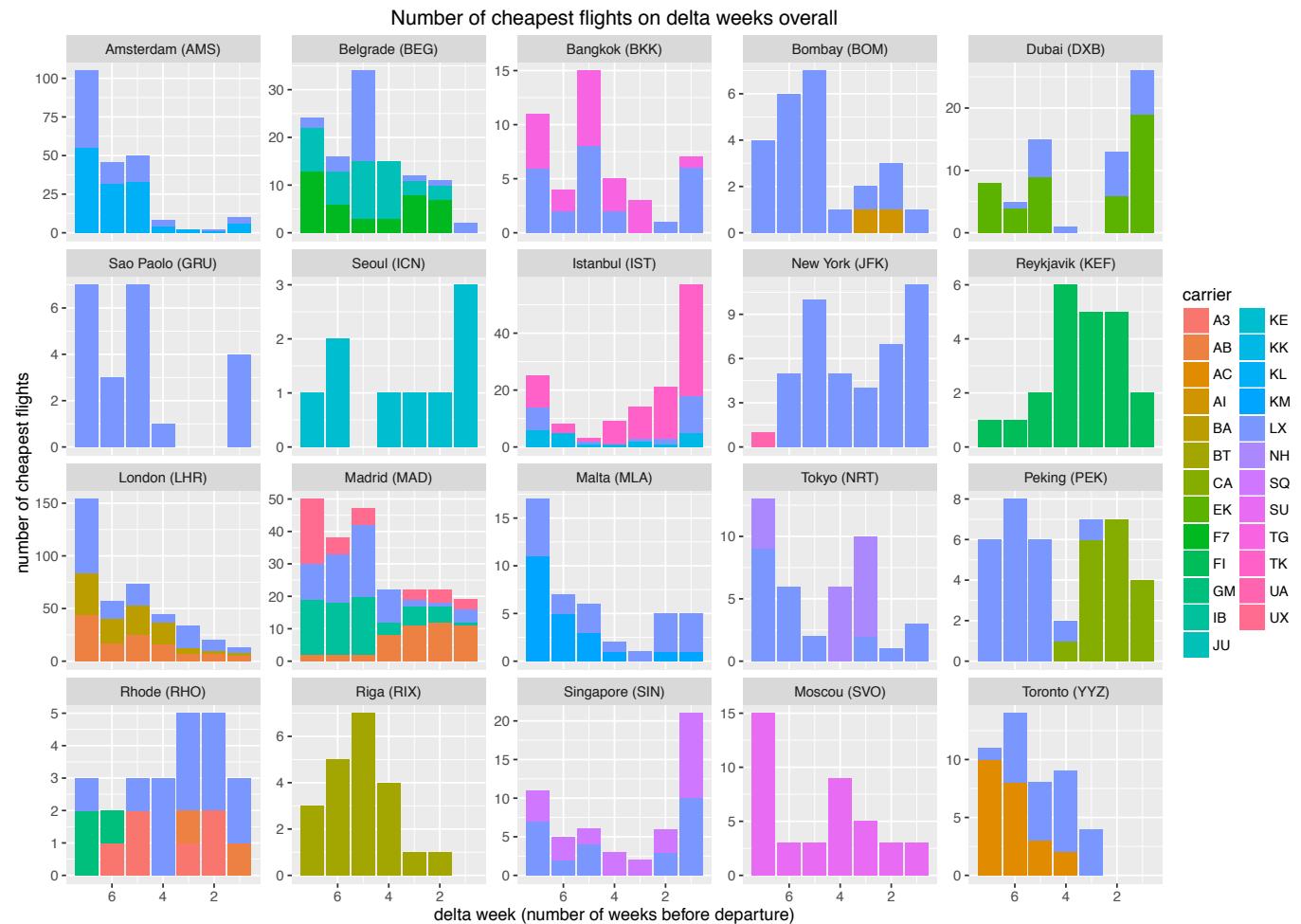
Es kann vorkommen, dass ein Preis für einen Flug bei einem Anbieter über mehrere Tage konstant ist. Sofern dieser Preis der günstigste ist, bedeutet dies, dass ein Flug mit demselben Preis zum selben Abflugtag mehrfach im Datenset vorkommen kann. Dies muss für die obige Fragestellung jedoch nicht speziell berücksichtigt werden, da hier die Gruppierung nach Anfragedatum stattfindet.



## b. An welchem Wochentag soll ich abfliegen?

Gleich wie bei der vorhergehenden Frage werden die Einträge in den entsprechenden Töpfen gezählt. Diese Frage zählt allerdings den Wochentag des Abflugdatums.

Im Gegensatz zur Frage a. muss hier eine Mehrfachzählung vermieden werden. Wenn ein Flug mehrere Tage konstant den Minimalpreis besitzt, darf der Abflugtag nicht mehrmals berücksichtigt werden. Ansonsten würde die Auswertung verfälscht. Aus diesem Grund wird jeweils nur der erste Tag mit dem Minimalpreise berücksichtigt.



● ● ●

## c. In welcher Woche vor dem Abflug ist es am günstigsten?

Hier wird zunächst aus den Delta-Tagen (vgl. Seite 12) die entsprechende Woche vor dem Abflug berechnet. Dies kann quasi als Delta-Woche bezeichnet werden.

Analog den vorhergehenden Fragen kann es auch hier vorkommen, dass der günstigste Preis über mehrere Tage konstant ist. Entsprechend müssen auch hier Mehrfachzählungen vermieden werden. Folglich wird nur der erste Tag vom Minimalpreis pro Flug berücksichtigt.

Anschliessend wird wiederum pro Flug jeder günstigste Preis dem entsprechenden Topf zugeteilt. Es gibt pro Delta-Woche einen Topf. Pro Destination ist derjenige Topf mit den meisten Einträgen die beste Woche zum Buchen eines Fluges.

## **d. Wieviel Geld kann ich durchschnittlich sparen, wenn ich zum richtigen Zeitpunkt fliege?**

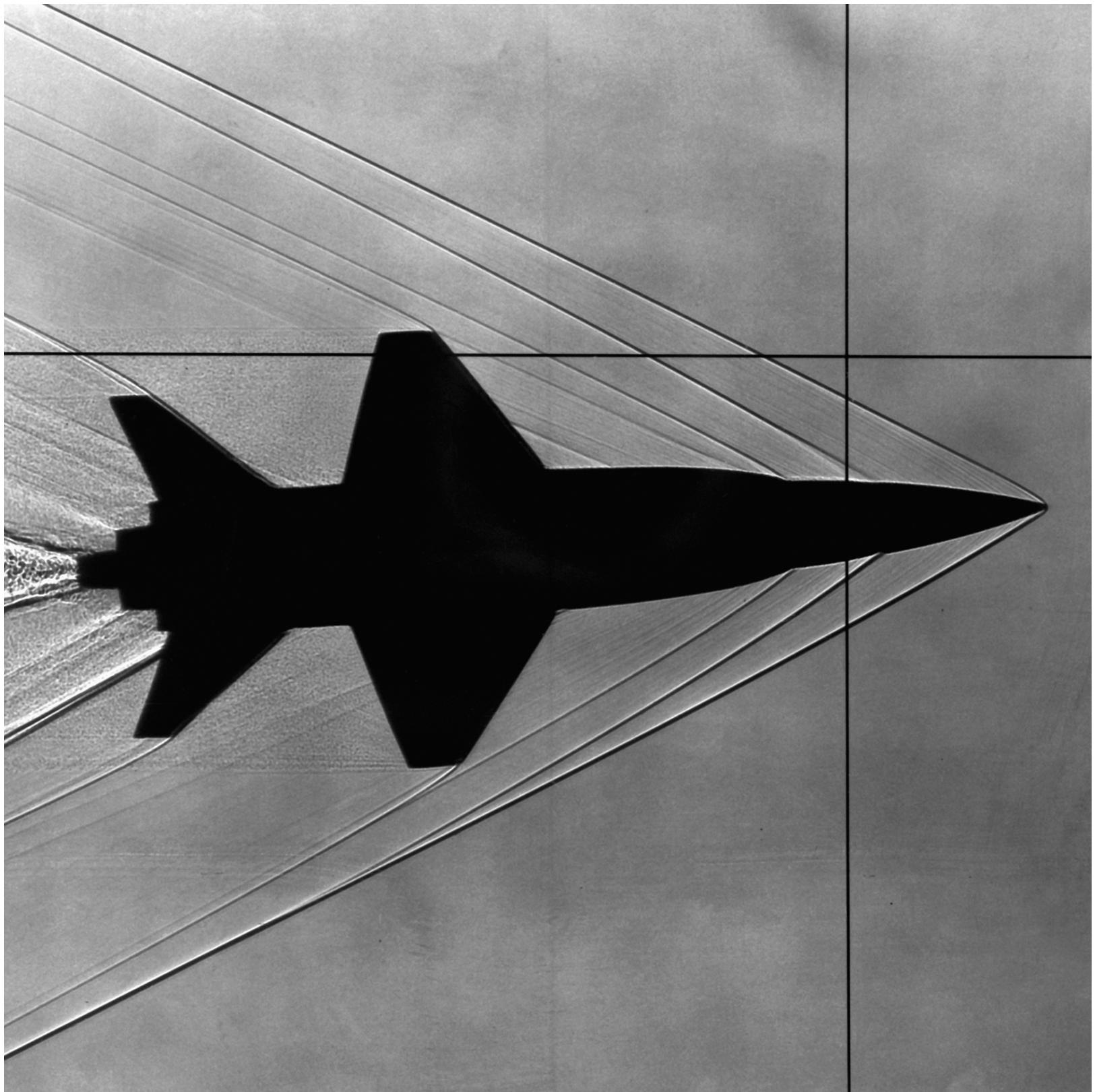
Für diese Fragestellung wurde die Preisdifferenz zwischen dem höchsten und niedrigsten Preis eines Fluges berechnet.

Die Preisersparnis errechnet sich folgendermassen:

Preisersparnis absolut = max. Preis - min. Preis

Preisersparnis prozentual =  $(\text{max. Preis} - \text{min. Preis}) / \text{max. Preis}$

Anschliessend wird der Durchschnitt aller Preisersparnisse von allen Flügen pro Destination berechnet (Mean).



## Visuelle Konzeption

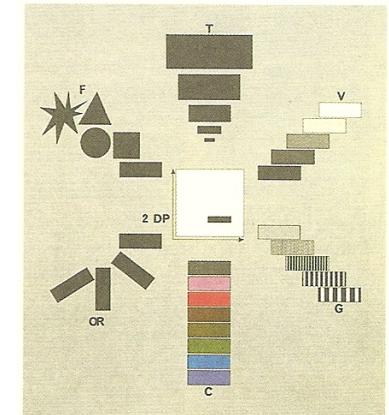
## **Im Modul “Visuelle Konzeption” wurden verschiedene Typen von Visualisierungen erarbeitet, verglichen und ausprobiert.**

Die Darstellungsmodelle können grob in drei Kategorien unterteilt werden:

- I. Menge**
- II. Ort**
- III. Zeit**

Gemäss Jaques Bertin [\[Wikipedia\]](#) gibt es sechs **visuelle Variablen** [\[Freie Universität Berlin\]](#), welche die Objekte einer Visualisierung prägen. Die einzelnen Variablen eignen sich für unterschiedliche Skalen und Eigenschaften.

- 1. Form**
- 2. Grösse**
- 3. Farbe**
- 4. Helligkeit**
- 5. Muster**
- 6. Richtung**



Variable	A	S	O	Q	darstellbares Skalierungsniveau
Größe	—	●	●	●	metrisch
Helligkeit	—	●	●	—	ordinal
Muster (Korn)	●	●	●	—	ordinal
Farbe	●	●	—	—	nominal
Richtung	●	●	—	—	nominal
Form	●	—	—	—	nominal

Erläuterung:  
A - assoziativ; S - selektiv; O - geordnet; Q - quantitativ  
● - graphische Variable besitzt diese Eigenschaft  
— - graphische Variable besitzt diese Eigenschaft nicht

	Points	Lines	Areas	Best to show
Shape	● ▲	possible, but too weird to show	cartogram	qualitative differences
Size	● ● ●	cartogram	cartogram	quantitative differences
Color Hue	● ● ●	cartogram	cartogram	qualitative differences
Color Value	● ● ●	cartogram	cartogram	quantitative differences
Color Intensity	● ● ●	cartogram	cartogram	qualitative differences
Texture	● ● ●	cartogram	cartogram	qualitative & quantitative differences

Zusätzlich zu den Darstellungsmodellen wurden die Visualisierungsaspekte beleuchtet.

### **1. Informationsdichte**

- Keine Banalitäten visualisieren, die Grafik muss einen Mehrwert liefern.
- Verhältnis Platz/Information muss auf das Format abgestimmt sein.

### **2. Korrekte Darstellung der Information**

- Achsen Skalierung, keine Manipulation durch Ausschnitte, regelmässige Zeitintervalle auf der Y-Achse.
- Prognosen klar kennzeichnen.
- Längen- und Flächenproportionalität berücksichtigen.

### **3. Reduktion aufs Wesentliche**

- Verzicht auf rein dekorative Elemente.
- Raster und Linien im Hintergrund, Einsatz als Strukturierung der Inhalte bzw. als Lesehilfe.

### **4. Beschriftung**

- Wenige Schriftarten und -größen verwenden.
- Beschriftung möglichst nahe an den Elementen in der Grafik platzieren.

### **5. Farbe**

- Kombination der Attribute Farbton, Deckkraft und Helligkeit.
- Mit möglichst wenig Farben auskommen, zu bunte Farben meiden.
- Unterschiedliche Farben machen Gruppen sichtbar.
- Rot-Grün-Blindheit beachten.

- Nachvollziehbarkeit, Abstufungen der Daten können visuell durch Farbe unterstützt werden.

## **6. Ordnung**

- Anordnung nach Wertgrößen ist häufig sinnvoll (z.B. bei Balken oder Pikogrammen).
- Kuchendiagramme: Grösstes Segment bei 12 Uhr, danach Anordnung in der Regel absteigend nach Grösse.

Dona M. Wong [[Dona Wong](#)] beschreibt diese Aspekte ebenfalls in ihrem Buch “The Wall Street Journal - Guide to Information Graphics” und liefert anschauliche Beispiele.

Die Inputs aus der Vorlesung “Visuelle Konzeption” sind in die vertikale Recherche eingeflossen. Gleichzeitig habe ich versucht, möglichst viele Darstellungsformen auszuprobieren. Interessant war dabei zu sehen, dass viele der oben beschriebenen Formen und Aspekte im eigenen Projekt zutreffen. Beispielsweise kann ein Kreisdiagramm oder eine Treemap nicht sinnvoll verwendet werden, wenn für den Betrachter kein “logisches Ganzes” visualisiert wird (vgl. [Kapitel Erkenntnisse](#)).

## **Visuelle Konzeption**

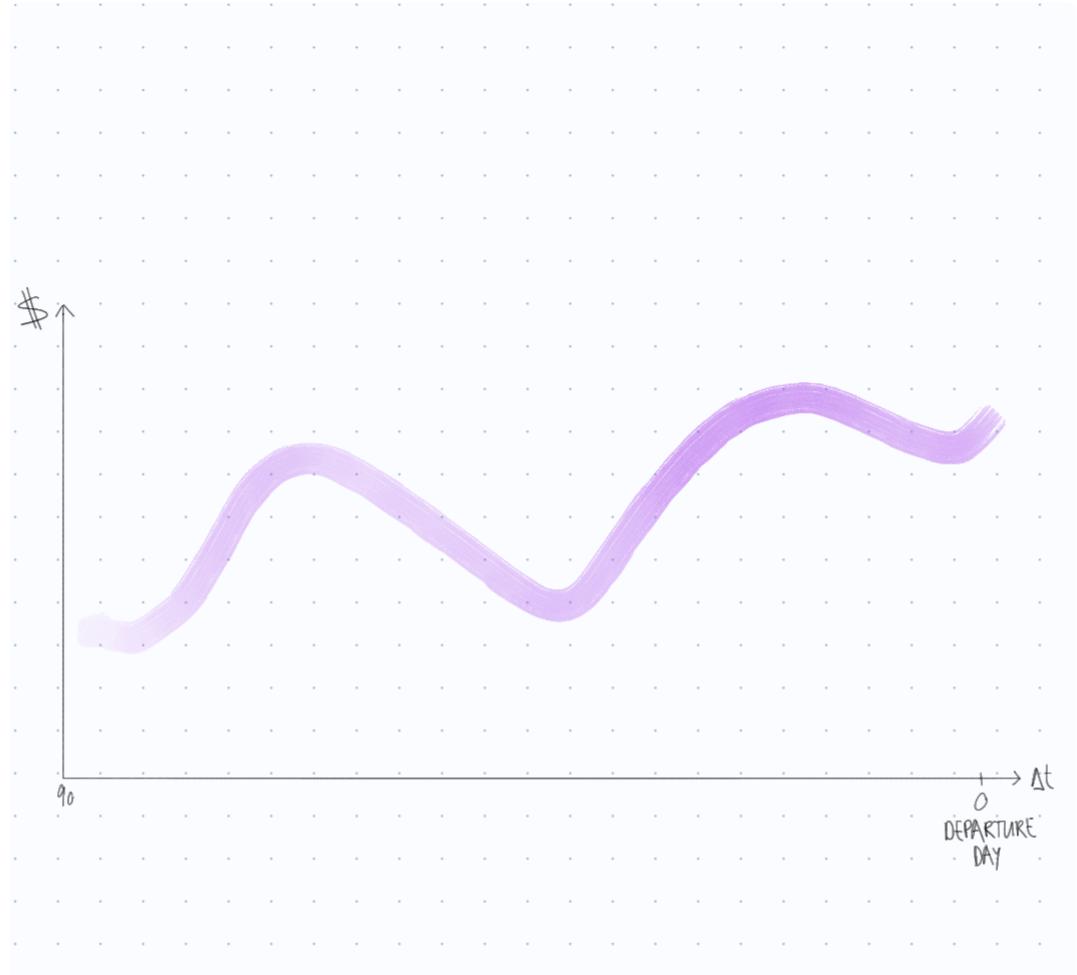
---

# **Konzeption/Paper Prototyping**

In der Konzeption der Grafik habe ich mit mehreren Visualisierungsformen verschiedene horizontale Prototypen erstellt. In dieser Phase des Projektes ging es mir darum, möglichst breit zu denken.

Als Ziel dieser Phase hatte ich die Selektion von einem bis zwei Prototypen, die ich im Coding mit meinen Daten verbinden wollte. Aufgrund der grossen Datenmenge war es schwierig und sehr zeitaufwändig in der visuellen Konzeption auf Papier die echten Daten zu verwenden.

Nachfolgend sind die Prototypen kurz beschrieben und mit Vor- und Nachteilen bewertet, anschliessend werden die Ergebnisse zusammengefasst.



## Liniendiagramm

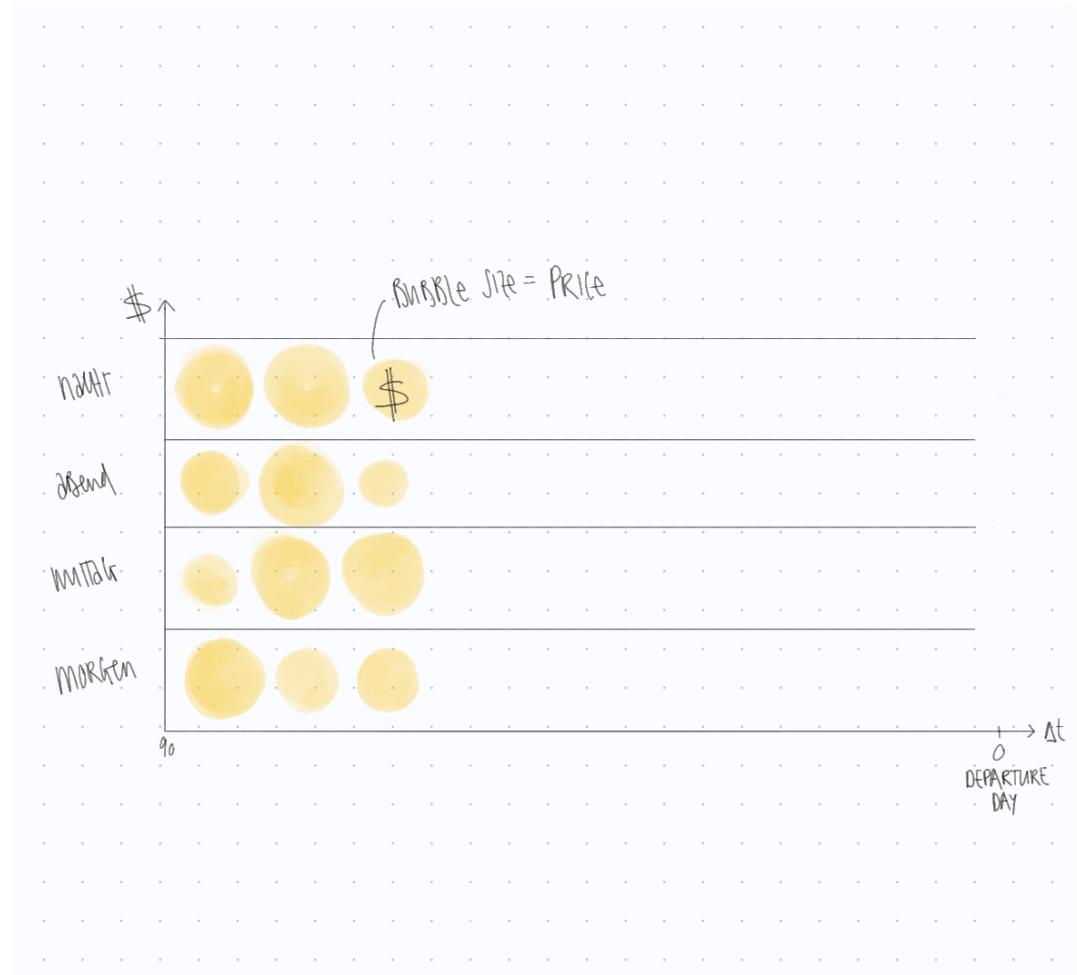
Eine einfache Darstellung ist das Liniendiagramm. Die Linie zeigt die Preisentwicklung über den Zeithorizont vor dem Abflug.

### Vorteil

Die Preisentwicklung über einen Zeithorizont in einem Liniendiagramm darzustellen ist geläufig.

### Nachteil

Die einzelnen Flüge können nicht sinnvoll überlagert werden. Multiples wären eine Möglichkeit dazu. Allerdings muss mit mehr als 100 Flügen pro Destination gerechnet werden, was eine Darstellung unüberschaubar macht.



## Bubble Chart

Im Bubble Chart wird die Preishöhe mit der Grösse der Bubbles visualisiert.

### Vorteil

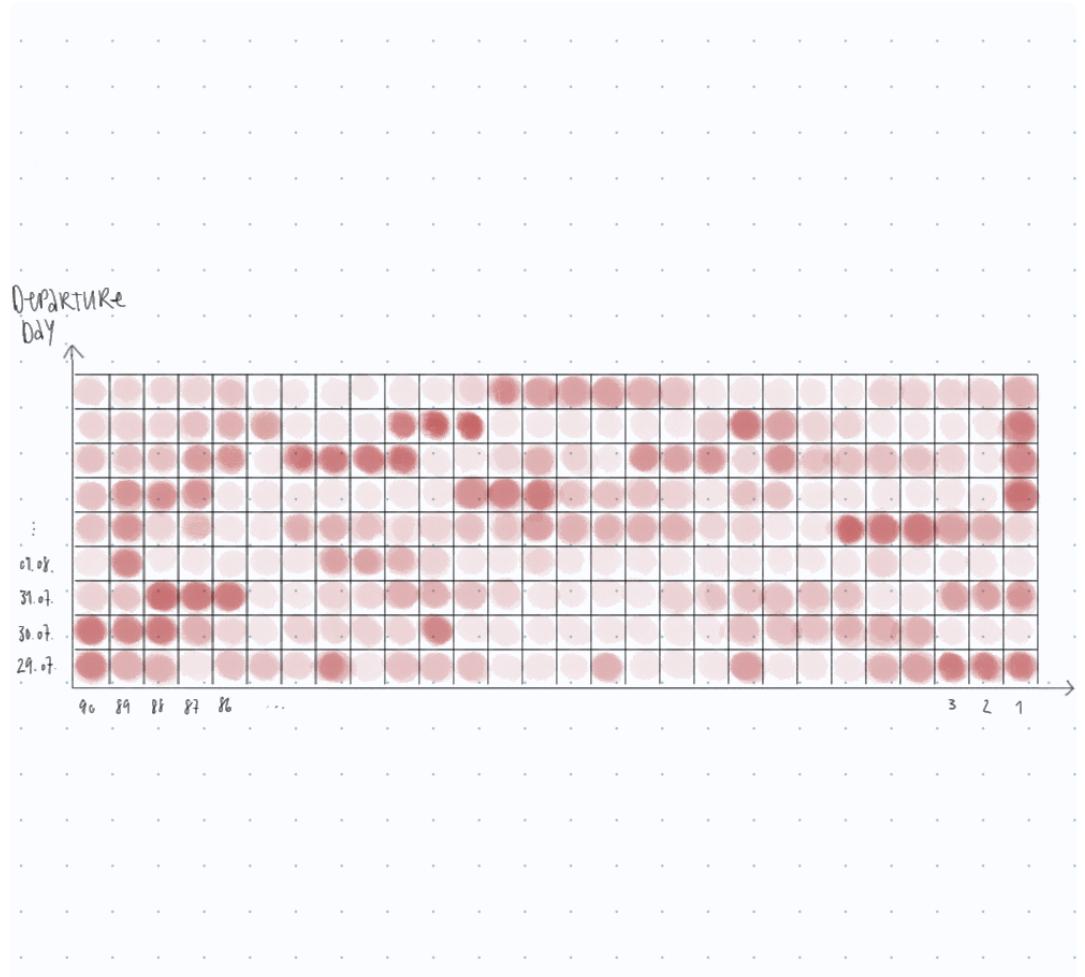
Da die Preisinformation in den Bubbles dargestellt wird, kann die y-Achse eine weitere Variable transportieren. Dies ist ein Vorteil gegenüber dem Liniendiagramm, wo die y-Achse die Höhe des Preises definiert.

Die y-Achse kann beispielsweise die einzelnen Flüge einer Destination darstellen.

### Nachteil

Die Vergleichbarkeit der Bubble-Größen ist schwierig, vor allem bei kleineren Preisdifferenzen in den Preisen.

Beispiel: Der Flugpreis am Tag x beträgt CHF 2'000.-, am nächsten Tag CHF 1'997.-, die Preisdifferenz von CHF 3.- bedeutet 1.5 Promille Unterschied.



## Matrix

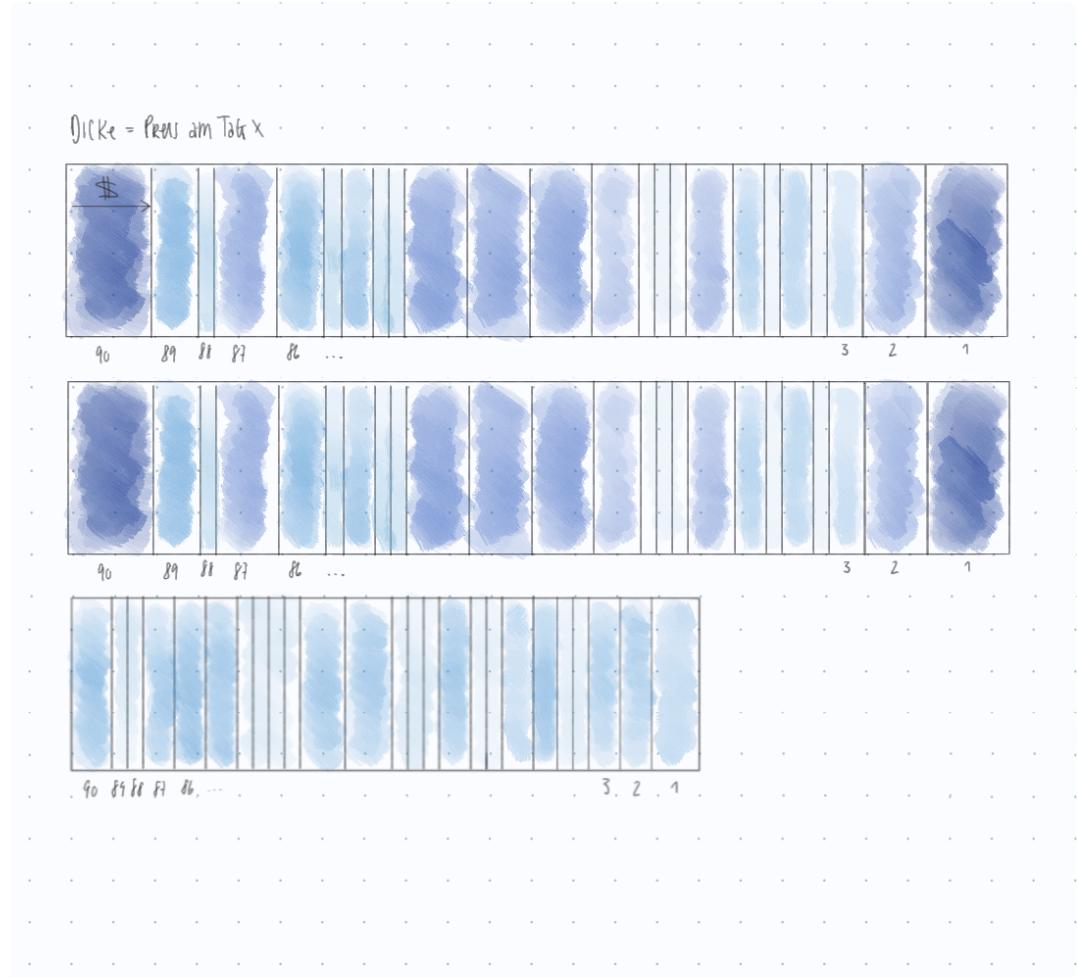
Die Matrix bietet im Kontext des Projektes eine ähnliche Darstellung wie das Bubble Chart. Die Preise werden jedoch über eine Farbkodierung dargestellt. Die Fläche ist so unabhängig vom Preis einheitlich gross.

### Vorteil

Eine Matrix erlaubt eine gute Vergleichbarkeit innerhalb eines Fluges (x-Achse) und zwischen den Flügen einer Destination (y-Achse) und ist einfach verständlich. Mit Hilfe von Multiples könnte ein Vergleich der unterschiedlichen Destinationen ermöglicht werden.

### Nachteil

Es sind keine genauen Preisinformationen im Chart ersichtlich.



## Stacked Bar Chart

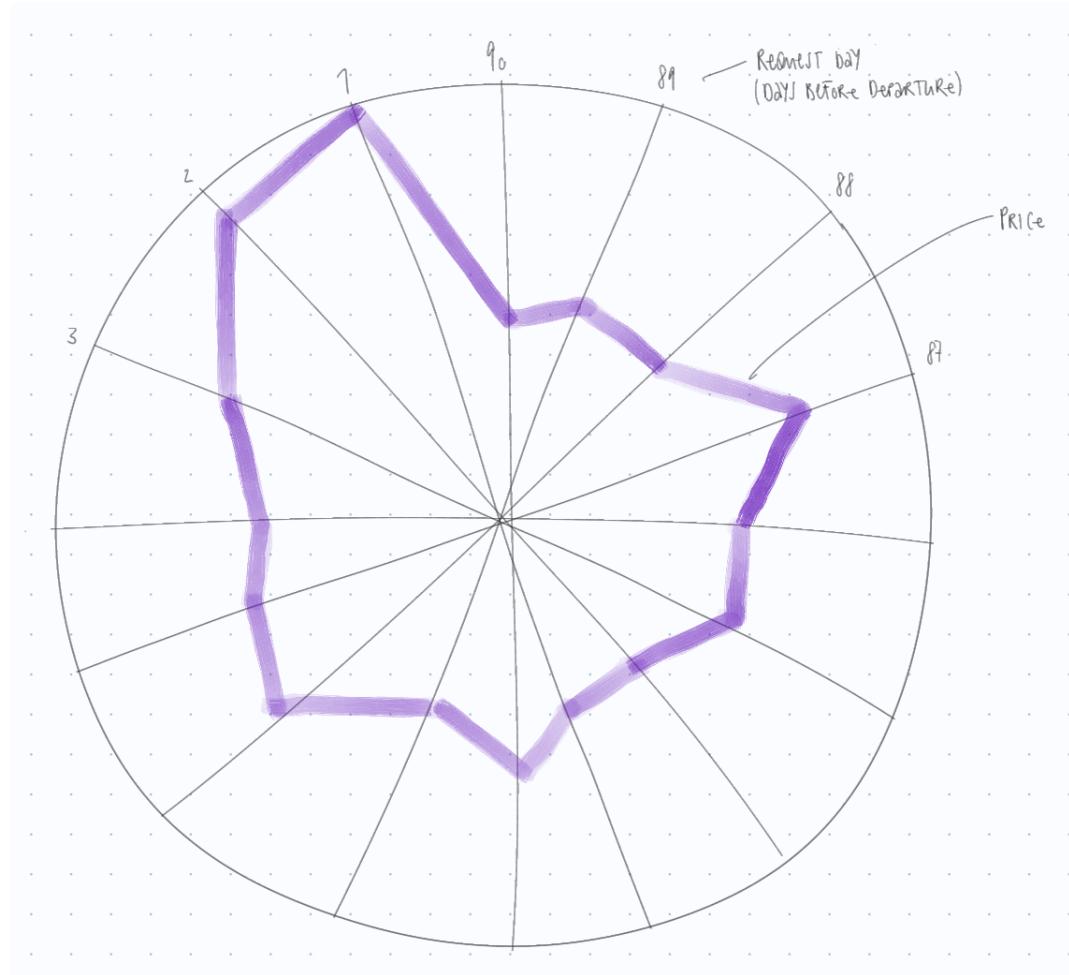
Die Preise werden in einem horizontalen Stacked Bar Chart dargestellt. Jeder Anfragetag vor Abflug ist entsprechend der Preishöhe mehr oder weniger breit.

### Vorteil

Die Grafik bietet eine kompakte Darstellung. Verschiedene Flüge können mit Multiples gut dargestellt werden.

### Nachteil

Die Vergleichbarkeit ist aufgrund der unterschiedlichen Balkenbreite eingeschränkt. Es gibt keinen Raster, an dem sich der Benutzer orientieren kann. Außerdem werden die Balken unterschiedlich lang, was die Vergleichbarkeit zusätzlich erschwert.



• •

## Spinnendiagramm/Rose Chart

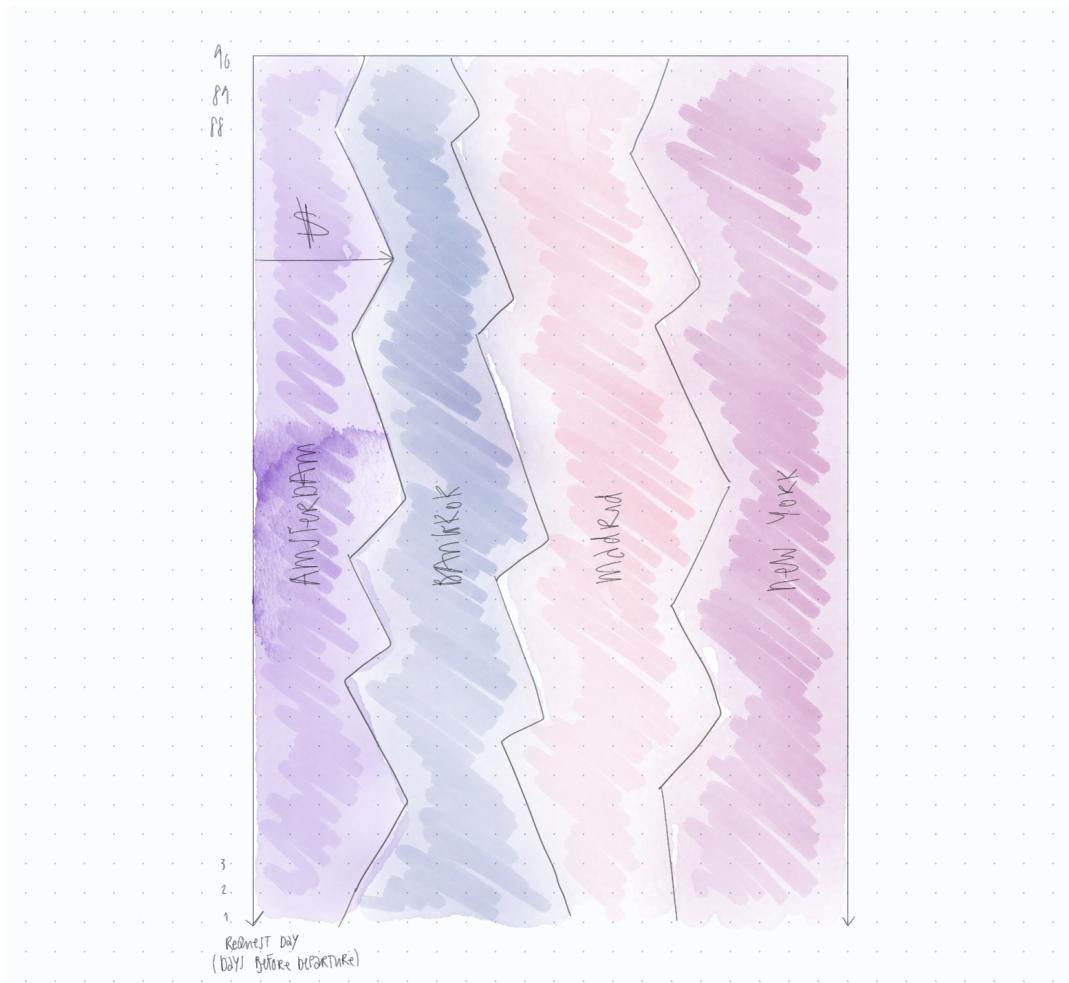
Die Zeitachse wird auf einem Kreis abgebildet, die einzelnen Abschnitte stellen je einen Anfragetag dar. Entweder kann der Abfragetag auf der Achse (vgl. erste Abbildung) oder zwischen zwei Achsen (vgl. zweite Abbildung) liegen.

### Vorteil

Das Spinnendiagramm bietet eine kompakte Darstellungsform.

### Nachteil

Da bis zu 90 Tage innerhalb von 360 Grad dargestellt werden sollen, wird ein Abschnitt sehr klein (4 Grad). Dies wird für den Betrachter schwierig zu lesen. Außerdem stellen die 90 Tage keine in sich abgeschlossene Zeitperiode dar, wie beispielsweise 24 Stunden pro Tag.



# Flächendiagramm mit vertikaler Zeitachse

Für jeden Flug wird eine Fläche mit der Preisentwicklung dargestellt. In der Horizontalen können die einzelnen Flüge oder gar Destinationen nebeneinander dargestellt werden.

Vorteil

Ähnliche Preisentwicklungen führen zu ähnlichen Flächen, zudem ist das Flächendiagramm eine kompakte Darstellungsform.

## Nachteil

Die Vergleichbarkeit von Flügen mit unterschiedlichen Preisentwicklungen ist schwierig. Die Fläche muss in der Horizontalen ein Ganzes geben - ansonsten ist die Fläche nicht ausgefüllt. Die Flugpreise der einzelnen Destinationen summieren sich nicht zu einem logischen Ganzen.

# Visuelle Konzeption

## Erkenntnisse

Aus der visuellen Konzeption wurden folgende Punkte festgehalten, die in die Umsetzungsphase bzw. das Prototyping einfließen.

- Die Verwendung von Multiples bietet sich an, um die einzelnen Flugverbindungen zu vergleichen. Dabei ist eine Gruppierung der Verbindungen in die beiden Kategorien Europa und Overseas denkbar.  
*Nachtrag aus der Umsetzungsphase: Multiples zu verwenden um Destinationen zu vergleichen ist schwierig, da die Aussagekraft eines Vergleiches nicht sehr hoch ist. Dies ist der Fall, weil die Flüge je nach Destination eine unterschiedliche Flugdauer oder Carrier haben. Ausserdem besitzt jede Destination eine andere Anzahl Flüge, was einen direkten Vergleich zusätzlich erschwert.*
- Für eine **Visualisierung einer Zeitvariablen** bieten sich **lineare Darstellungen** an. Gegebenenfalls können diese jedoch auch zweidimensional erfolgen (vgl. **Bubble Chart** oder **Matrix**).
- Wenn die Daten absolut sind, kann ein Wert auch auf einem Kalender gezeigt werden. Bei relativen Daten, wie beispielsweise 56 Tage vor Abflug, ist eine Anordnung auf einem Kalender nicht sinnvoll.
- Um die **Vergleichbarkeit der Tage** zu erhalten, sollte auf einem **statischen Raster** gearbeitet werden. Dies ist umso wichtiger, wenn mit Multiples gearbeitet wird.
- Eine Visualisierung der Tage mittels Kreis (vgl. **Rose Chart**) scheint nicht sinnvoll, da es sich beim Beobachtungszeitraum von drei Monaten um keine ab-

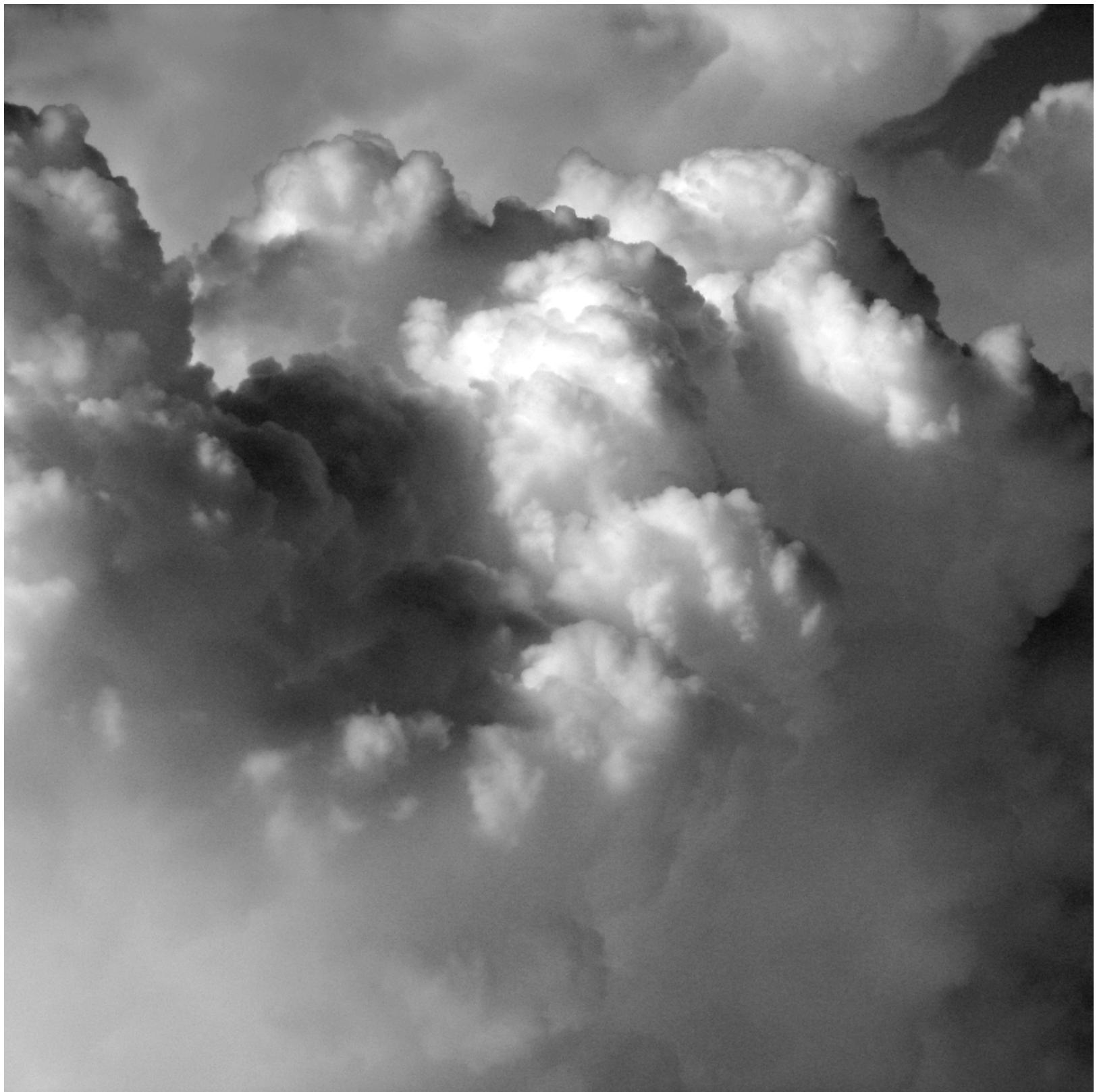
geschlossene "Zeitreihe" handelt, wie z.B. 24 Stunden für einen Tag oder 12 Monate für ein Jahr.

- Eine Visualisierung durch eine Zeitachse in Kombination mit Flächen (vgl. **Flächendiagramm**) scheint nicht zielführend, wenn kein Ganzes (100%) existiert. Analog verhält es sich beim Einsatz von TreeMaps.
- Eine Relativierung der Preisentwicklung durch Abbildung in unterschiedlichen Stufen (rot – blau mit unterschiedlicher Transparenz) schafft eine gute Vergleichbarkeit. Zudem erscheint mir eine flächenproportionale Darstellung eine erschwerete Vergleichbarkeit zu bieten (vgl. **Bubble Chart**).

*Nachtrag aus der Umsetzungsphase: Es wurde eine **einfarbige, sequentielle Skala** bevorzugt. Eine zweifarbige Skala hat die Leserlichkeit der Matrix unnötig erschwert.*

- Die Vergleichbarkeit von Flugstrecken könnte über die zurückgelegte Distanz eines Fluges, aber auch über die Dauer eines Fluges erreicht werden. Die Flugdauer ist unter Umständen sogar aussagekräftiger, da die Phasen "Start" und "Landung" bei jedem Flug gleich berücksichtigt werden (diese Aussage ist anhand der Daten zu überprüfen).
- Mögliche Piktogramme für die Verwendung in den Visualisierungen könnten das Dollar-Zeichen (Preis) oder ein Flugzeug-Symbol (Flug) sein.
- Der Einsatz einer typografischen Visualisierung kann für einen Vergleich qualitativer Merkmale der Destinationen, wie beispielsweise der Flugdauer geprüft werden.

**Aus der visuellen Konzeptionsphase wurden die Fragestellungen nochmals geschärft, damit eine fokussiertere Visualisierung möglich ist (vgl. Kapitel **Einleitung - Fragestellung**).**



**Coding**

*Im Unterrichts-Modul Programmierung wurden zwei Frameworks zur Datenvizualisierung betrachtet: Processing und D3.js (D3).*

Processing wurde mit dem Fokus als Einstieg für Nicht-Programmierer entwickelt. Processing [\[Wikipedia\]](#) hat sich vor allem auch in der Kunstszene etabliert, etwa für Installationen in Museen.

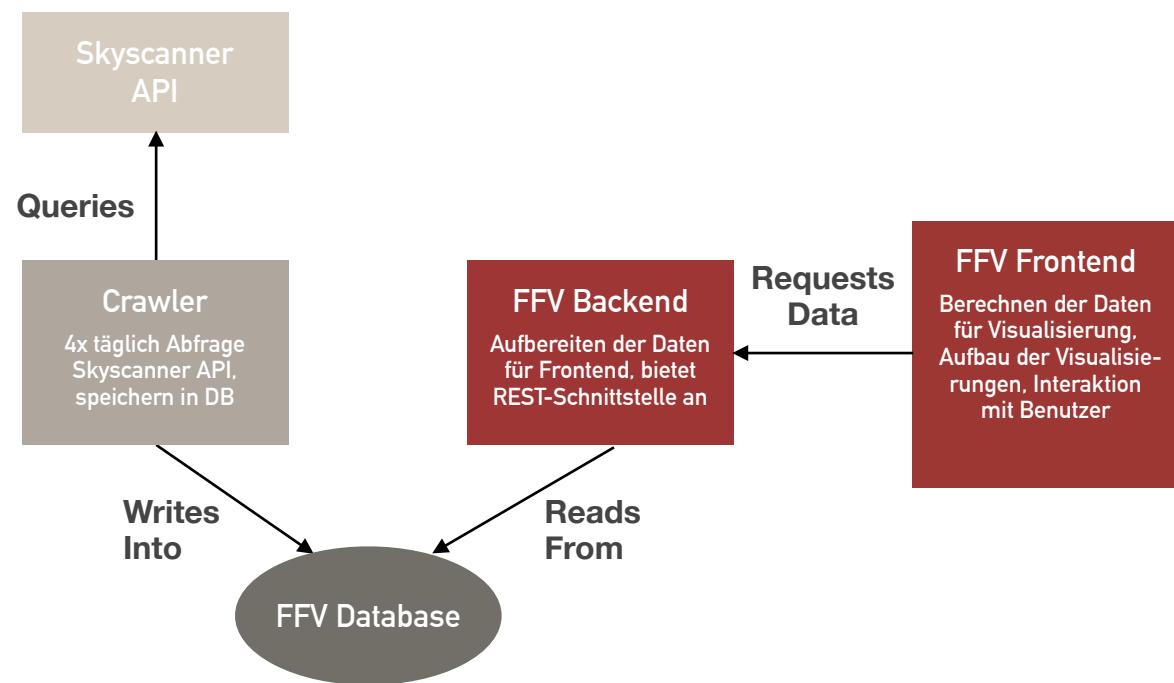
D3 ist eine reine Javascript-Library für die Visualisierung von Daten. D3 ist die Abkürzung für **Data Driven Documents**. D3 nutzt dazu die **Standards SVG, HTML5 und CSS**. Dadurch wird eine sehr breite Unterstützung in den Web-browsern erreicht. Dies hat sicherlich zum Erfolg von D3 [\[Blog ScribbleLive\]](#) beigetragen.

Mit Processing hatte ich bereits in der Vergangenheit erste Erfahrungen gesammelt. Das Tool hat für mich eher im Bereich der künstlerischen Visualisierungen seine Stärken. Mit p5.js (p5) existiert mittlerweile zwar eine Implementation für das Web. Die Funktionalität von Processing ist jedoch noch nicht komplett in p5 [\[Site-point\]](#) umgesetzt worden.

Da ich für das Autorenprojekt eine Webseite erstellen wollte, entschied ich mich für die Umsetzung mit D3. Trotzdem habe ich auch einige der Prototypen in Processing erstellt.

In der Coding-Phase wurde einerseits die Webseite entwickelt. In einem ersten Schritt erstellte und testete ich jedoch anhand der Echtdaten erste Prototypen.

In der effektiven Umsetzungsphase erfolgte schliesslich die Implementation der Webseite implementiert. Diese besteht aus einem Frontend und einem Backend. Das Frontend ist für die Aufbereitung der Visualisierung verantwortlich. Dazu werden die Daten via REST-Schnittstelle vom Backend angefragt. Das Backend liest die Daten von der Datenbank und bereitet diese in die optimale Struktur für das Frontend auf.



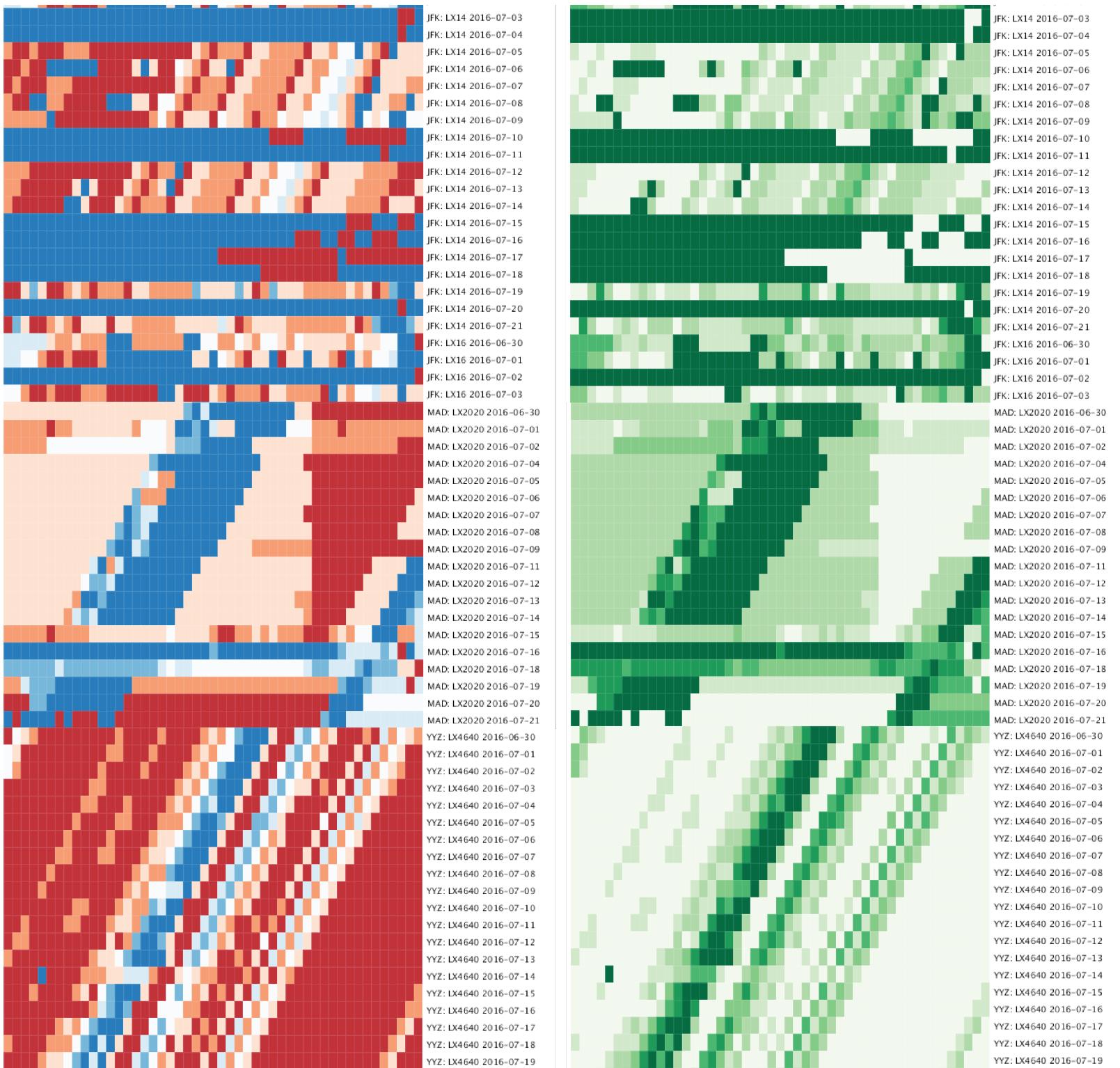
# Coding

## Prototyping

*Beim Prototyping ging es darum, die Ideen aus der visuellen Konzeption anhand der Echtdaten zu prüfen. Die Datenanalyse war soweit fortgeschritten, dass diese für erste Prototypen verwendet werden konnte.*

Die Prototypen wurden in Processing wie auch D3 erstellt. Zudem konnte ich in dieser Phase auch horizontale Prototypen mit wenig Aufwand ausprobieren und testen.

Die Erkenntnisse aus dem Testing der Prototypen, sowie die in dieser Phase erlernten Fertigkeiten, wurden anschliessend in der Umsetzung benötigt.



Der erste Processing-Prototyp hat die Matrix aus der visuellen Konzeption aufgegriffen. Es wurde mit verschiedenen Parametern, wie Farbskalen oder der Grösse der Zellen experimentiert.

## Fazit

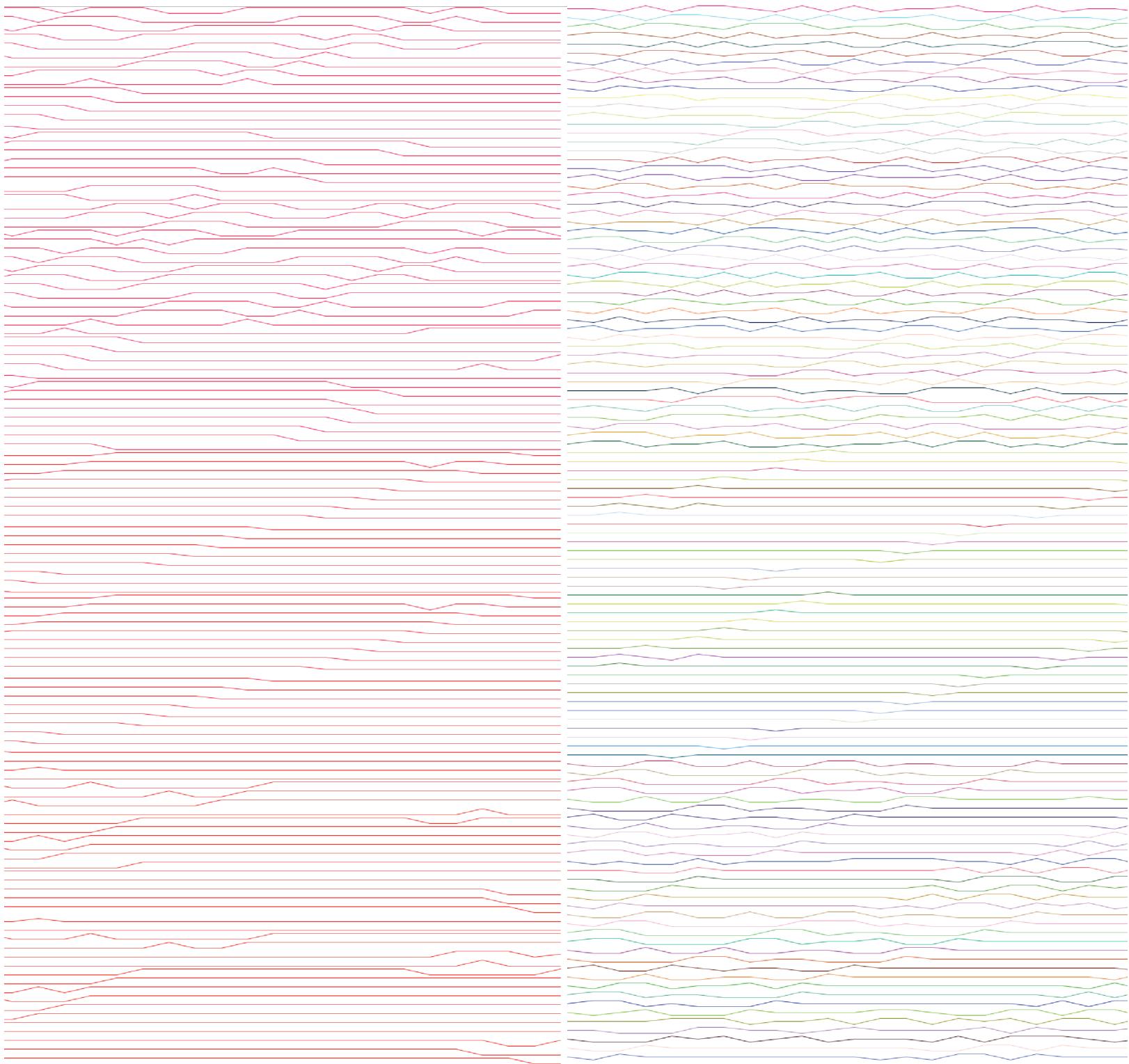
Eine sequentielle Farbskala ist einfacher zu lesen. Die Grösse der Zellen muss nicht zwingend quadratisch sein.



In diesem Prototypen habe ich nur den minimalen und maximalen Preis auf einem Zeitstrahl visualisiert. Wenn ein Preis über mehrere Tage konstant tief oder hoch ist, wird er mehrmals dargestellt. Die Strickdicke visualisiert die Höhe des Preises.

## Fazit

Eine schwierige Darstellungsform, wenn sich die Preise über eine längere Zeit konstant auf dem minimalen oder maximalen Preisniveau befinden.



Dieser Processing-Sketch zeigt den Preisverlauf aller Flüge. Da die Preise der Flüge unterschiedlich sind, wurden diese normalisiert. Wenn ein Preis ansteigt, wird +1 angezeigt. Bei keiner Preisänderung wird 0 verwendet, -1 wird angezeigt wenn der Preis fällt.

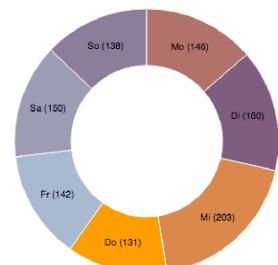
In der linken Grafik ist der absolute Preisverlauf ersichtlich. Hier wird die Normalisierung immer zum ersten Tag berechnet. Rechts ist der relative Preisverlauf abgebildet, welcher die Normalisierung jeweils zum Vortag darstellt.

### Fazit

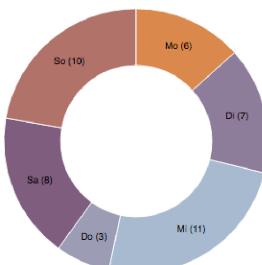
Schwere Nachvollziehbarkeit, zu abstrakt für den Betrachter.

## Flight Fare Visualization: Zurich - London

An welchem Wochentag soll ich buchen?

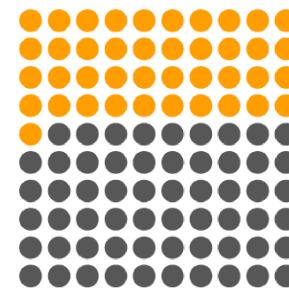


An welchem Wochentag soll ich abfliegen?

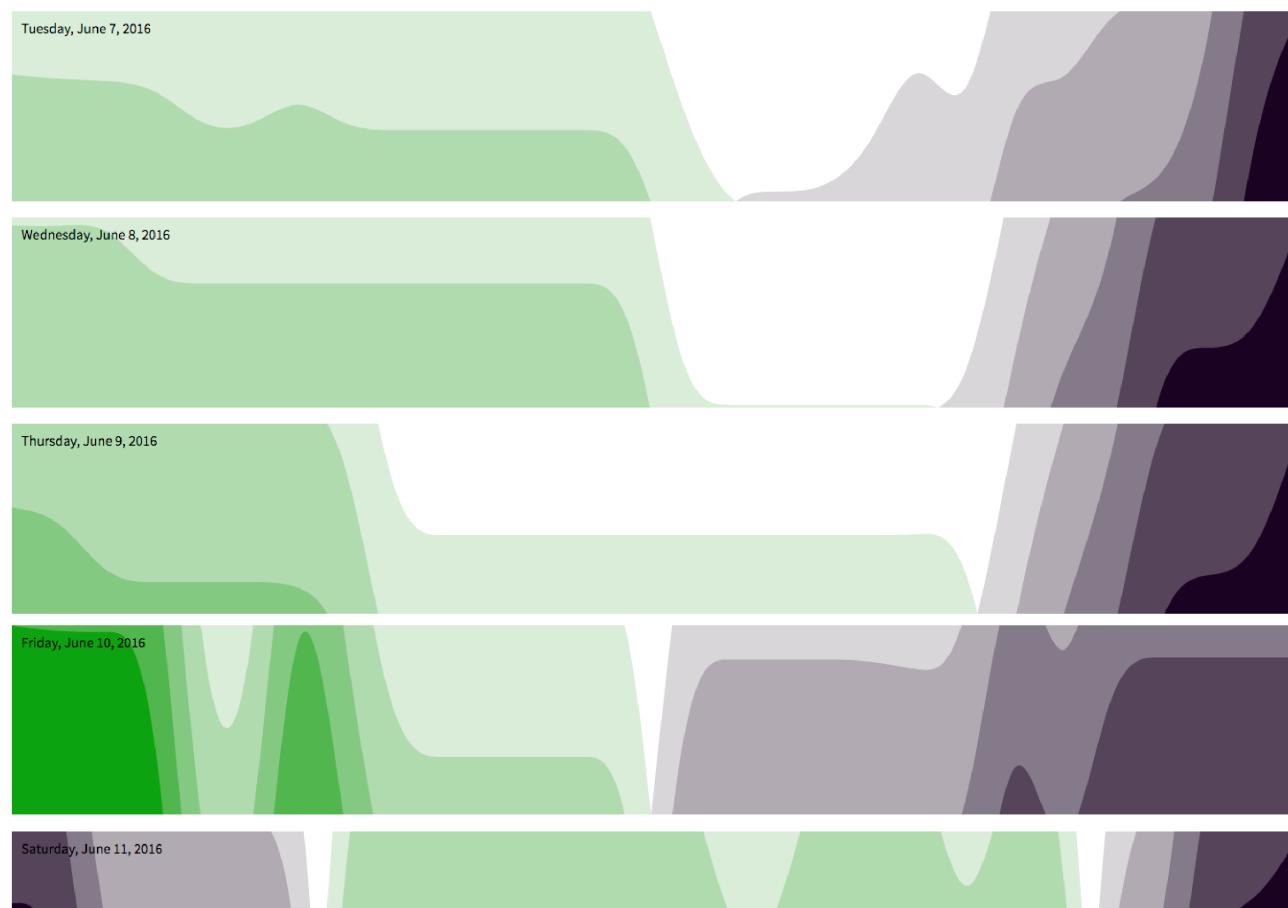


In welcher Woche vor dem Abflug soll ich buchen?

Wieviel kann ich durchschnittlich sparen, wenn ich zum richtigen Zeitpunkt buche?



Preisentwicklung nach Abflugtag



In diesem D3-Prototypen habe ich meine Fragestellungen in einzelnen Grafiken ausprobiert. Die optimalen Wochentage wurden als Donut-Chart, die Preisersparnis als Multiple mit einem Kreis pro Prozent umgesetzt.

### Fazit

Den Horizon-Graph habe ich für die Preisentwicklung ausprobiert, da er eine kompakte Darstellungsform eines Liniendiagramms ist. Tests zeigten, dass dieser nicht leicht zu lesen und deshalb verworfen wurde.

Die Donuts wurden ebenfalls verworfen, da die Vergleichbarkeit von mehr als vier bis fünf Kreissegmenten schwierig ist.

## Frontend

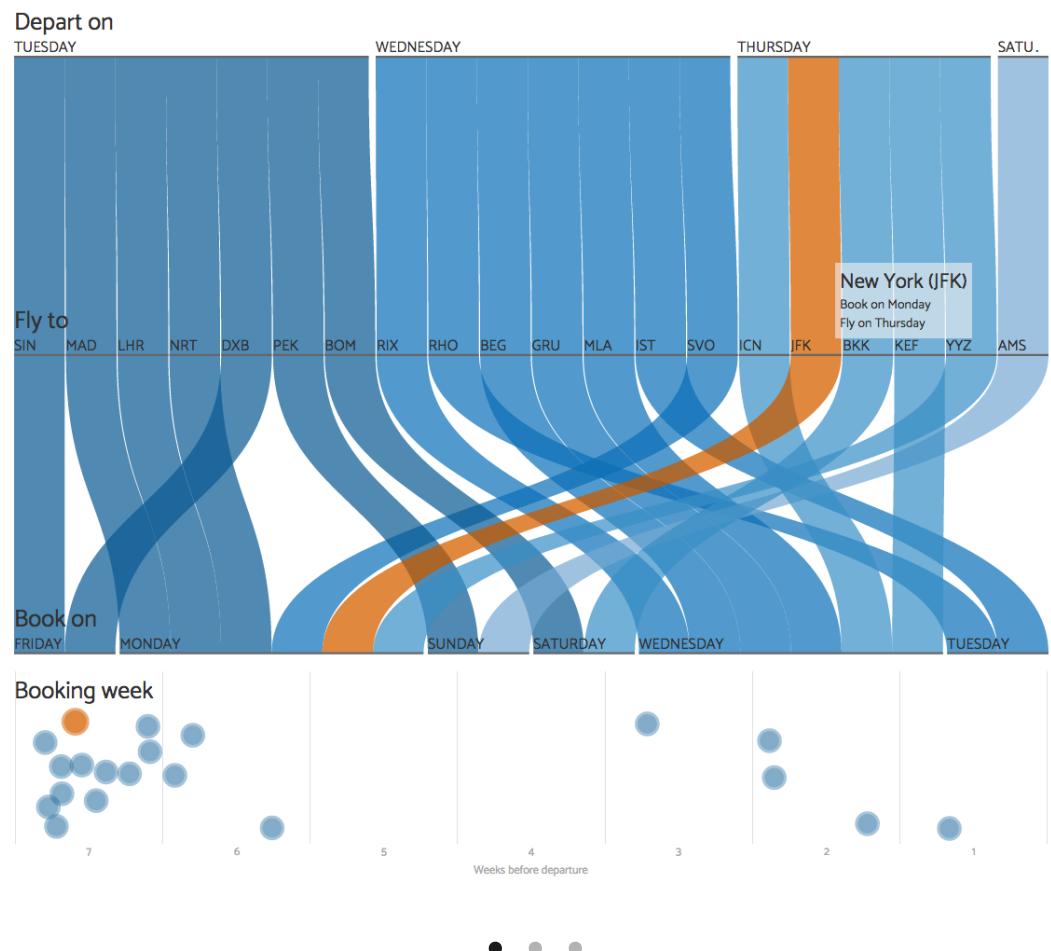
*Im Frontend wird die Datenvisualisierung aufbereitet. Dazu werden die Daten vom Backend via REST-Service abgefragt. Für die Visualisierung werden die Daten mittels D3 berechnet und dargestellt.*

Die Visualisierung besteht aus folgenden Teil-Visualisierungen:

1. Parallel Set kombiniert mit Zeitstrahl.
2. Matrix mit interaktiver Preisentwicklung und -ersparnis.
3. Durchschnittliche Preisersparnis pro Destination.

Folgende Libraries und Beispiele werden im Projekt verwendet:

- **d3-parseset** [\[d3-parsesets\]](#)
  - Library wird bezüglich Anordnung und Beschriftung der Kategorien auf die Bedürfnisse des Projektes individualisiert.
  - Events werden ausgelöst, wenn eine Kurve angewählt wird. Dadurch wird eine Kommunikation mit der Zeitstrahl-Visualisierung möglich.
- **timeseries** [\[timeseries\]](#)
  - Anpassung der Library für die Verwendung. Anstelle von Zeitwerten werden Zahlenwerte (Wochen vor Abflug) angezeigt.
- **Trulia Matrix und Bar Chart** [\[Trulia\]](#)
  - Anpassung des Beispiels, so dass anstelle einer HTML-Tabelle SVG-Code erzeugt wird. Dadurch wird die Visualisierung viel schneller.
- **Vizuly Radial Progress** [\[Vizuly\]](#)
  - Erstellen eines eigenen Themes, das die Farben für das Chart definiert.



Die erste Visualisierung adressiert die folgenden Fragen aus den **Fragestellungen**:

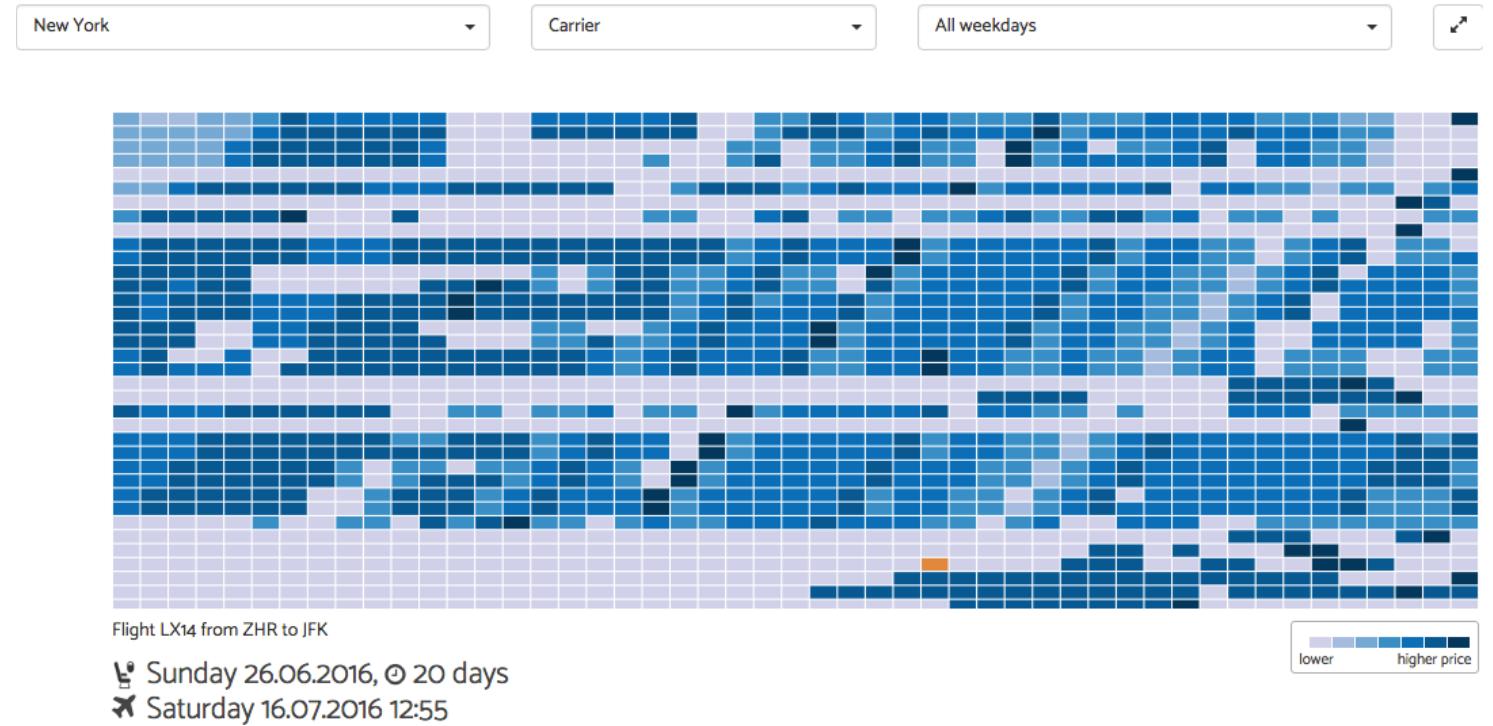
- An welchem Wochentag soll ich abfliegen?**
- An welchem Wochentag soll ich einen Flug buchen?**
- Wie viele Wochen vor dem Abflug soll ich einen Flug buchen?**

Für die Visualisierung habe ich mich für die Visualisierungsform des **Parallel Sets** [d3-parses] entschieden. Die Antworten auf die Fragen a. und b. verlangen beide einen Wochentag. Dies ist eine **kategorische, abgeschlossene Menge von Werten**. Auch die Destination liegt in einem kategorischen Wertebereich.

Ergänzend zum Parallel Set habe ich die Wochen als **Zeitstrahl** [timeseries] visualisiert. Es wäre möglich gewesen, diese auch als weitere Kategorie in das Parallel Set aufzunehmen. Dagegen haben aus meiner Sicht jedoch zwei Gründe gesprochen.

1. Die Anzahl Kategorien wird bei den Wochen mit zunehmender Datenmenge und einer Verlängerung der Zeitreihen ansteigen. Die Anzahl Kategorien sind also nicht zwingend abschliessend.
2. Die **Symmetrie in der Grafik** mit den drei Variablen "Depart on", "Fly to" und "Book on" hat mir sehr gut gefallen. Durch eine weitere Kategorie wäre aus meiner Sicht auch die Verständlichkeit der Grafik reduziert worden.

Durch die **Interaktivität zwischen dem Parallel Set und dem Zeitstrahl** entsteht jedoch eine Verbindung zwischen den beiden Grafiken und diese werden als eine Visualisierung wahrgenommen.



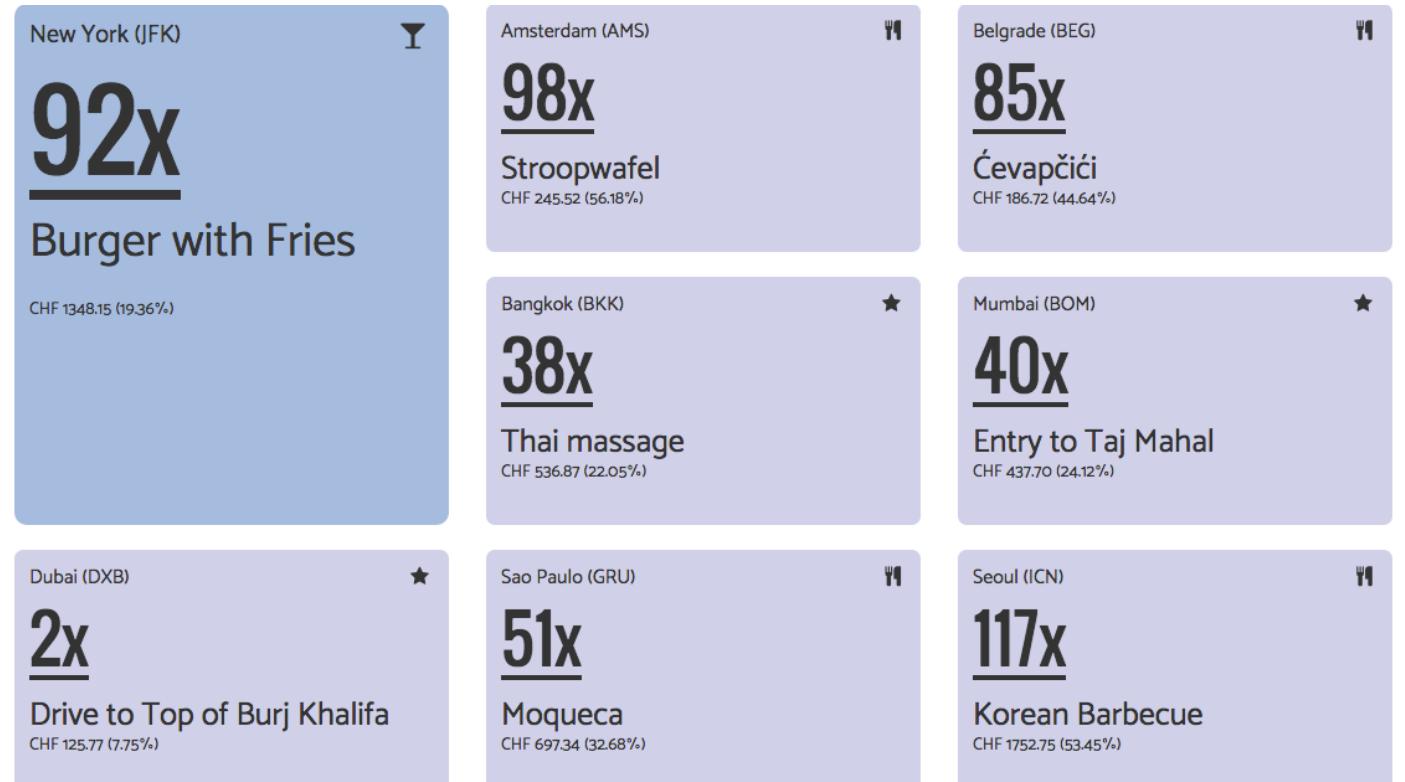
Die zweite Visualisierung erlaubt dem Betrachter eine Vertiefung in die Materie. Sämtliche Datensätze werden in einer Matrix dargestellt. Die Idee und Teile des Codes stammen von Trulia [\[Trulia\]](#).

Durch diese Darstellungsform kann der Benutzer schnell und einfach zwischen den Preisentwicklungen der einzelnen Flüge vergleichen und so Muster erkennen.

Die Schwierigkeit der Visualisierung lag darin, auf beiden Achsen Zeitreihen zu visualisieren. Die **Höhe des Preises** ist durch eine **Farbkodierung** in der einzelnen Zelle abgebildet. Je tiefer der Preis umso heller die Farbe.

Wie in der visuellen Konzeption beschrieben, bietet die Matrix keine exakte Preisinformation. Um den **genauen Preisverlauf** für den Betrachter trotzdem sichtbar zu machen, habe ich unterhalb der Matrix eine Detailansicht durch zwei weitere Grafiken angefügt: Ein **Bar Chart sowie ein Progress-Chart** [\[Vizuly\]](#). Diese zeigen den genauen Preisverlauf des gewählten Fluges sowie die Preisersparnis des aktuell gewählten Flugpreises.

Mittels **interaktiven Filtern** nach Destination, Carrier und Wochentag kann der Betrachter die Visualisierung seinen Bedürfnissen anpassen. Es wird ihm dadurch die Möglichkeit gegeben, in die **Visualisierung einzutauchen und diese zu entdecken**.



• • •

Die letzte Visualisierung ist eine **typografische Darstellung**. Diese präsentiert dem Betrachter noch eine andere, unerwartete Seite auf die Thematik.

Anstelle der Preisentwicklung steht hier die folgende Frage aus den **Fragestellungen** im Zentrum:

**d. Wie viel kann ich durchschnittlich pro Destination sparen, wenn ich zum richtigen Zeitpunkt buche?**

Zur Beantwortung werden nebst der **durchschnittlichen Preisersparnis** in Schweizer Franken, bzw. Prozent auch die Ersparnis in Form eines **typischen Produkts** für jede Destination errechnet. Die Visualisierung nimmt mit den Kacheln die Form der einzelnen Zellen der Matrix auf. Die aktuell gewählte Destination aus der Preis-Visualisierung wird hier ebenfalls hervorgehoben, indem diese als erste Kachel vergrössert dargestellt wird.

# Backend

*Das Backend bereitet die Daten aus der Datenbank auf und stellt diese via REST-Service zur Verfügung. Folgende Schnittstellen stehen zur Auswahl.*

## **1. GET api/carrier**

Liefert alle Carrier von allen Destinationen.

---

## **2. GET api/carrier/{destination}**

Liefert alle Carrier für eine Destination.

---

## **3. GET api/count/{carrier}**

Zählt die Anzahl Flüge für einen Carrier aller Destinationen.

---

## **4. GET api/dayHistogramBook/{destination}**

Liefert das Histogramm der günstigsten Flüge nach Wochentagen für eine Destination.

Beispiel <http://ffv.mochila.ch/api/dayHistogramBook/lhr>

Resultat

{"Thu":546,"Tue":513,"Sat":567,"Wed":626,"Fri":413,"Mon":652,"Sun":561}

---

## **5. GET api/dayHistogramBook/{destination}/{carrier}**

Analog Punkt 4, jedoch zusätzlich nach Carrier eingeschränkt.

---

## **6. GET api/destination**

Liefert alle Destinationen.

---

## **7. GET api/destination/{destination}**

Liefert alle Flugpreise zu einer bestimmten Destination. Dies beinhaltet alle Preise für jeden Flug von 1 bis n Tage vor Abflug (Delta).

Beispiel <http://ffv.mochila.ch/api/destination/kef>

Resultat (gekürzt)

```
[{"carrier": "FI", "number": "569", "destination": "KEF", "origin": "ZHR", "departure": 1473249600000, "prices": [{"request": 1473112800000, "price": 586.18, "delta": 1}, {"request": 1468965600000, "price": 228.16, "delta": 49}], "minPrice": 228.16, "maxPrice": 586.18, "complete": true}, {"carrier": "FI", "number": "569", "destination": "KEF", "origin": "ZHR", "departure": 1471176000000, "prices": [{"request": 1471039200000, "price": 589.46, "delta": 1}, {"request": 1466892000000, "price": 447.61, "delta": 49}], "minPrice": 445.0, "maxPrice": 607.16, "complete": true}, ...]
```

---

## **8. GET api/destination/{destination}/{carrier}**

Analog Punkt 7, jedoch nach Carrier eingeschränkt.

---

## **9. GET api/flights/{destination}**

Analog Punkt 7, jedoch werden die Daten in der Ausgabe nicht verschachtelt ausgegeben.

Beispiel <http://ffv.mochila.ch/api/destination/kef>

Resultat (gekürzt)

```
[{"destination": "KEF", "origin": "ZHR", "carrier": "FI", "flightNumber": "FI569", "departureDate": "2016-09-07", "departureTime": "14:00:00", "dts": 1473249600000, "departureWeekday": "Mi", "requestDate": "2016-09-06", "deltaTime": 1, "price": 586.18, "bin": 6}, {"destination": "KEF", "origin": "ZHR", "carrier": "FI", "flightNumber": "FI569", "departureDate": "2016-09-07", "departureTime": "14:00:00", "dts": 1473249600000, "departureWeekday": "Mi", "requestDate": "2016-09-05", "deltaTime": 2, "price": 583.97, "bin": 5}, ...]
```

---

---

**10. GET api/flights/{destination}/{carrier}**

Analog Punkt 9 mit zusätzlicher Einschränkung nach Carrier.

---

**11. GET api/min/{destination}**

Liefert den günstigsten Preis für eine Destination über alle Flüge. Als Resultat wird der Tag vor Abflug sowie dessen Wahrscheinlichkeit zurückgegeben.

Beispiel <http://ffv.mochila.ch/api/min/kef>

Resultat {"value": "14", "propability": 0.181818181818182}

---

**12. GET api/min/{destination}/{carrier}**

Analog Punkt 11, jedoch weitere Einschränkung nach Carrier.

---

**13. GET api/minWeekdayBook/{destination}**

Liefert den günstigsten Wochentag zum Buchen für die angegebene Destination. Als Ergebnis wird der Wochentag mit der zugehörigen Wahrscheinlichkeit geliefert.

Beispiel <http://ffv.mochila.ch/api/minWeekdayBook/kef>

Resultat {"value": "Wed", "propability": 0.1661807580174927}

---

**14. GET api/minWeekdayBook/{destination}/{carrier}**

Analog Punkt 13 mit zusätzlicher Einschränkung nach Carrier.

---

**15. GET api/minWeekdayFlight/{destination}**

Liefert den günstigsten Wochentag zum Abfliegen für die angegebenen Destination. Als Ergebnis wird der Wochentag mit der zugehörigen Wahrscheinlichkeit geliefert.

Beispiel <http://ffv.mochila.ch/api/minWeekdayFlight/kef>

Resultat {"value": "Thu", "propability": 0.2727272727272727}

---

**16. GET api/minWeekdayFlight/{destination}/{carrier}**

Analog Punkt 15 mit zusätzlicher Einschränkung nach Carrier.

---

## **17. GET api/minhist/{destination}**

Liefert ein Histogramm, das über alle Flüge einer Destination die günstigsten Flüge pro Delta-Tag (1 bis n Tage vor Abflug) aufzeigt.

Beispiel <http://ffv.mochila.ch/api/minhist/kef>

Resultat

[1,0,0,1,0,0,5,0,0,0,0,1,1,12,2,1,3,0,2,0,2,1,4,2,2,2,0,2,2,1,2,1,0,0,2,1,1,0,1,1,3,7,0,0,0,0,0,0]

---

## **18. GET api/minhist/{destination}/{carrier}**

Analog Punkt 17 mit zusätzlicher Einschränkung nach Carrier.

---

## **19. GET api/savings/{destination}**

Liefert die durchschnittliche Preisersparnis für die gewählte Destination. Dazu wird für jeden Flug die Differenz zwischen minimalem und maximalem Flugpreis berechnet. Am Ende wird von allen Differenzen der Mittelwert berechnet.

Das Resultat wird als absolute Preisdifferenz sowie in Prozent zurückgeliefert.

Beispiel <http://ffv.mochila.ch/api/savings/kef>

Resultat {"absolut":324.95257575757574,"relative":0.40469160161980877}

---

## **20. GET api/savings/{destination}/{carrier}**

Analog Punkt 20 mit zusätzlicher Einschränkung nach Carrier.

---



# Form und Methode

*In den Modulen “Storytelling” und “Visuelle Identität/Detailgestaltung” wurde die Umgebung der Visualisierung betrachtet. Das Storytelling thematisierte das Einbetten der Visualisierung in eine Geschichte. In der Detailgestaltung wurden die gestalterischen Aspekte der Visualisierungen beleuchtet und erlernt.*

Das Storytelling ist ein wichtiger Aspekt einer Visualisierung. Sinn und Zweck ist es die Aufmerksamkeit des Betrachters auf die Geschichte zu lenken. Bei Webseiten entscheidet ein Benutzer innerhalb von 10-20 Sekunden, ob er auf einer Webseite verbleibt oder abspringt [\[Jakob Nielsen\]](#).

Die Story muss daher den Benutzer von Beginn weg in den Bann ziehen. Oder wie Nielsen es formuliert: “To gain several minutes of user attention, you must clearly communicate your value proposition within 10 seconds.”

### **Elemente des Storytellings**

Gemäss Walther von La Roche [\[Wikipedia\]](#) gibt es verschiedene Elemente, welche die Aufmerksamkeit von Stories erzeugen:

- |                                     |                                   |
|-------------------------------------|-----------------------------------|
| - Nähe (geografisch oder psychisch) | - Konflikt, Macht, Kampf          |
| - Sex, Erotik                       | - Kuriosität, Humor, Überraschung |
| - Fortschritt, Erfindung, Neuigkeit | - Kinder, Tiere, Gefühl           |
| - Spannung, Dramatik, Folgenschwere | - Übersinnliches                  |

Bestandteile einer guten Story sind: Bild, Headline, Untertitel und Lead. Bei Infografiken ist aber auch die erste Grafik entscheidend, die der Betrachter sieht.

Beim Verfassen einer Geschichte ist es wichtig, die Storykurve zu beachten. Mit einem Höhepunkt beginnen, danach die Details bzw. Pflichtinfos preisgeben und zum Ende nochmals die Spannung steigern [MAZ Blog]. Zudem muss beachtet werden, dass die Geschichte von konkreten zu abstrakten Inhalten aufgebaut wird. Beginnt die Geschichte zu abstrakt, dann wendet sich der Leser von ihr ab.



#### Formate im DDJ (Data Driven Journalism)

Es gibt verschiedenste Formate im DDJ. Im folgenden habe ich die aus meiner Sicht wichtigsten Formate aufgelistet.

##### 1. Explorable

- Datensatz erfahrbar und zugänglich machen.
- Geschichte beim Betrachter im Kopf entstehen lassen.

##### 2. Explainer

- Erklärstück mit interaktiven Elementen.
- Wenn möglich gleiche "Währung" in allen Grafiken verwenden, z.B. Anzahl Unfälle.

##### 3. Talkie

- Visualisierung als Film/Erzählung gestalten.

- Mit Audioelementen den Benutzer leiten.
- Wichtig: Erzähltempo berücksichtigen.
- Vorsicht: Viele Benutzer mögen es nicht, wenn automatisch ein Ton abgespielt wird (z.B. bei Zugriff via Mobile im öffentlichen Raum).

#### **4. Storymap/Map**

- Geschichten entlang von Karten erzählen.
- Geographische Pattern darstellen.

#### **5. Quiz/Game und Try Yourself**

- Gamification zieht anderes Zielpublikum an.
- Spassfaktor beachten.
- Grossen Lerneffekt erzielen.

#### **6. Animierte GIFs**

- Eignen sich für Social Media-Kanäle.
- Können als Teaser für die eigentliche Visualisierung genutzt werden.

#### **7. Photo-Graph**

- Visualisierung mit Bildern, dadurch nahe an der Realität.

#### **Visuelle Identität/Detailgestaltung**

Die Detailgestaltung setzt sich damit auseinander, die Visualisierung für den Betrachter ansprechend zu gestalten. Dabei werden nochmals die Farben überarbeitet, ein Stil für die Visualisierung und das Raster definiert sowie eine Typografie ausgewählt.

Bei der Detailgestaltung werden nochmals die Visu-Prinzipien [[Coarse Grained Blog](#)] von Edward Tufte [[Wikipedia](#)] berücksichtigt.

## **Form und Methode**

# Umsetzung

*Die Fülle von gesammelten Daten wird dem Betrachter in Form eines "Explorable" zugänglich gemacht. Aus diesem Grund wird die Visualisierung als interaktive Webseite realisiert. Um dem Betrachter trotzdem einen Teil einer Geschichte zu erzählen werden die Grafiken um Texte ergänzt. Der Rest der Geschichte entsteht im Kopf des Betrachters.*

Im Laufe des Projektes hat sich die ursprünglich gewählte Form einer interaktiven Webseite weiter bestätigt. Um einen möglichst grossen Benutzerkreis anzusprechen, habe ich die **Webseite für verschiedene Endgeräte gestaltet (Responsive Design)**. Aufgrund der Fülle von Daten werden die Visualisierungen auf mobilen Endgeräten teilweise etwas klein. Jedoch kann für eine detaillierte Betrachtung die Desktop-Webseite konsultiert werden.



Aus mehreren Gründen habe ich mich im Verlaufe des Projektes für **mehr als eine Grafik** entschieden:

- Auf einem Bildschirm, bzw. Mobile Device, steht nur beschränkter Platz zur Verfügung. Um die **Verständlichkeit zu gewähren**, muss die Grafik unterteilt werden.
- Eine einzelne Grafik soll - zumindest auf dem Desktop - auf einem Bildschirm mit heutiger Auflösung ohne Scrollen Platz finden.

- Die unterschiedlichen Fragestellungen besitzen **unterschiedliche Betrachtungslevels**. Diese in einer gemeinsamen Grafik zu visualisieren wäre nicht sinnvoll möglich.
- Aus Sicht des Storytellings macht es durchaus Sinn, eine Visualisierung in mehrere Grafiken zu unterteilen. Dies erlaubt es, eine Geschichte glaubwürdig anhand der Storykurve aufzubauen.

### Schriften

Im Rahmen der Detailgestaltung habe ich mich auf folgende zwei Schriftarten von Google Fonts [[Google Fonts](#)] festgelegt.

**Oswald Regular** Verwendung in Headline, Lead und Zwischentiteln.

Catamaran Regular Verwendung in Texten und allen Visualisierungen.

### Farben

Bei den Farben habe ich mich an der Farbkodierung von Cynthia Brewer [[Colorbrewer](#)] orientiert. Ich habe mich für das Blau-Violette Schema entschieden, da Blau die Farbe des Himmels ist und ich dies mit Fliegen assoziere.

Um eine einheitliche Farbgebung innerhalb der gesamten Webseite zu erreichen habe ich bei allen Visualisierungen dieselben Farbtöne eingesetzt. Um die Selektionen des Betrachters hervorzuheben, wurde die Farbe Orange gewählt.



## Piktogramme

Folgende Piktogramme von Font Awesome [Font Awesome] und Icons8 [Icons8] werden auf der Webseite als **erklärende Gestaltungselemente** eingesetzt.

- ⌚ ✈️ ⏰ Buchungstag, Abflugtag und Delta-Tage
- 💵 💵 💸 Maximaler, minimaler Preis und Preisersparnis
- ⭐ 🍽️ 🍷 Sehenswürdigkeit, Essen und Trinken

## Kommunikationsmittel Präsentation/Ausstellung

Nebst der Webseite wurden für die Ausstellung ein Banner sowie Visitenkarten erstellt. Das Banner vermittelt einen visuellen Eindruck über die Fülle von Daten und fungiert als “Eyecatcher”. Die Visitenkarten dienen als “Brücke” zwischen der physischen und der digitalen Welt. Sie ermöglichen den Besuchern der Ausstellung einen Teil von Flight Fare Visualization mit nach Hause zu nehmen.



Anhang

# Anhang

## Lessons Learned

Folgende Erfahrungen habe ich im Verlaufe des Autorenprojektes gemacht.

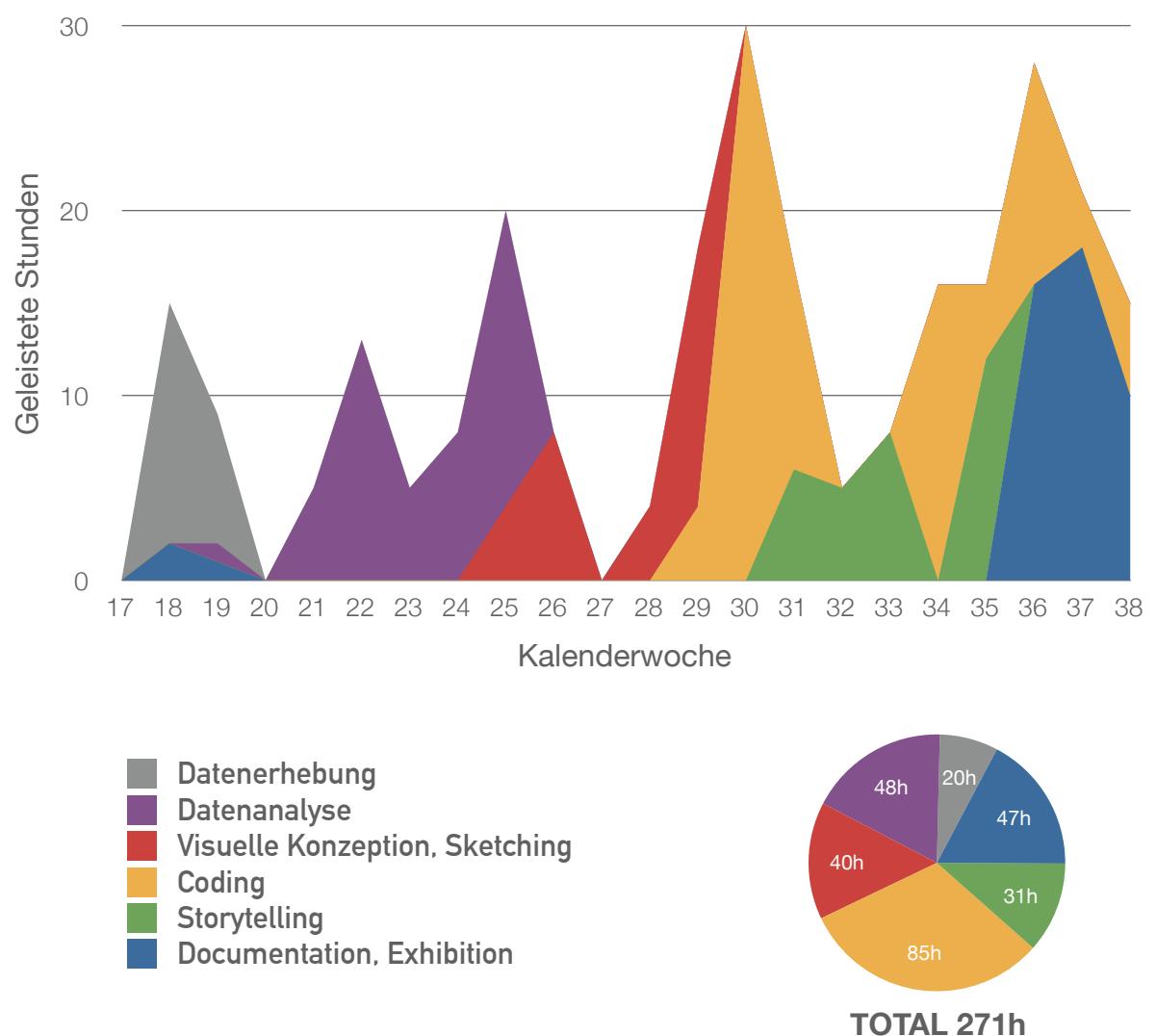
1. Erst dann an die Visualisierung herangehen, wenn eine **klare Vorstellung** davon vorhanden ist.
2. In **kleinen Iterationen** (Rapid Prototyping) einzelne Visualisierungen mit den echten Daten prüfen.
3. **Interaktion benötigt** in der Umsetzung **viel Aufwand**, weshalb sie nur gezielt eingesetzt werden soll (vgl. Punkt 5). Außerdem muss diese im Optimalfall noch auf das Endgerät (Desktop, Mobile) angepasst werden, da gewisse Interaktionen nur für bestimmte Endgeräte sinnvoll sind (z.B. MouseOver).
4. Eine **klare Priorisierung der Features** vornehmen. Wichtiges von “Nice-To-Have” trennen.
5. **Kosten und Nutzen** stets abwägen.

# Anhang

## Administratives

### Arbeitsnachweis

Die aufgeführten Stunden sind die Aufwände, die ausserhalb der Unterrichtsmodule an der HKB für das Autorenprojekt investiert wurden.



# Anhang

## Links

Die Linkssammlung enthält Webseiten, die mir während dem CAS weitergeholfen haben, oder die ich noch im Detail betrachten möchte.

### **Datenerhebung/-analyse**

RStudio Cheat Sheets

<https://www.rstudio.com/resources/cheatsheets/>

Flowing Data

<http://flowingdata.com/>

### **Coding D3**

Learn JS Data - Data manipulation, munging and processing in JavaScript  
<http://learnjsdata.com/index.html>

Data Scrolling for Explainers - A tutorial  
<http://vallandingham.me/scroller.html>

Textures.js - SVG patterns for Data Visualization  
<http://riccardoscalco.github.io/textures>

d3.tip: Tooltips for d3.js visualizations  
<https://github.com/Caged/d3-tip>

d3 SVG Legend  
<http://d3-legend.susielu.com>

SVG Crowbar - Export SVGs from D3.js  
<http://nytimes.github.io/svg-crowbar>

### **Coding Alternativen**

Library for computational and generative design for InDesign  
<http://basiljs.ch>

Library for vector graphics scripting (also for Illustrator scripting)  
<http://paperjs.org>

Javascript 3D library (auch für D3)  
<http://threejs.org>

Javascript library based on Processing  
<https://p5js.org>

dc.js - Dimensional Charting Javascript Library  
<https://dc-js.github.io/dc.js>

### **Visuelle Konzeption, Inspiration**

Data Visualization 101: How to Design Charts and Graphs  
[https://cdn2.hubspot.net/hub/53/file-863940581-pdf/Data\\_Visualization\\_101\\_How\\_to\\_Design\\_Charts\\_and\\_Graphs.pdf](https://cdn2.hubspot.net/hub/53/file-863940581-pdf/Data_Visualization_101_How_to_Design_Charts_and_Graphs.pdf)

The Wall Street Journal Guide to Information Graphics, Dona M. Wong  
<http://donawong.com/>

### **Data-Driven Journalism (DDJ)**

538 Datablog  
<http://fivethirtyeight.com>

Data Journalism Awards (Global Editor Networks)  
<http://www.globaleditorsnetwork.org/programmes/data-journalism-awards>

KnightLab (tools for mediamakers like TimelineJS  
<https://knightlab.northwestern.edu/projects/>

Archie Tse's Rules  
<http://www.tapestryconference.com/blog/2016/lessons-malofiej-what-do-and-what-not-do-ever-again>

### **Dokumentation, Exhibition**

iBooks Author Templates  
<https://itunes.apple.com/us/app/templates-for-ibooks-author/id527161787?mt=12>  
<https://www.ibooksauthortemplates.com>

reveal.js - The HTML Presentation Framework  
<http://lab.hakim.se/reveal-js/#/>

# Copyright

© 2016 Ruth Ziegler, CAS Data Visualization, HKB Bern

Ruth Ziegler  
Steinhofstrasse 44  
CH-6005 Luzern  
[ziegler.ruth@gmail.com](mailto:ziegler.ruth@gmail.com)  
<http://www.mochila.ch>