Treffen mit Sebastian am Dienstag, 18.6. an der Uni Augsburg

# Ziel

* Besprechen der bisherigen Ergebnisse

🡺 Sicherstellen, dass ich es korrekt verstanden habe

* Abstecken der Ziele / Gliederung für die Masterarbeit  
  🡺 kann mit Schreiben beginnen

# Dokumente

* LaTeX
* 0618 - 1 Exploring exponential smoothing

# Erreicht

* Alle Codes skriptbasiert   
  🡺 Ergebnisse werden geloggt (zB Zeit zum Laufen des Skripts wird gespeichert)

🡺 reproduzierbarer Code

* Start des Schreibens der Masterarbeit in LaTeX und auf Englisch
* Algorithmen getext in recht mathematischer Form
* Value expected single leg flight example
* API single leg flight
* Vergleich verschiedener Strategien

# Ideen

* Kapazitätsbedarf von einem Produkt für eine Ressource in {0, 1, …, N} statt nur in {0, 1}
* Mathematischer Beweis zu exponential smoothing = Mittelwert (bisher schon empirisch)
* Vergleich verschiedener Strategien  
  1. Jede mit Baseline (zB DP) vergleichen  
  2. Jede mit jeder vergleichen (paarweise) und dann in Matrixschreibweise darstellen (statistische Bedenken, ob das erlaubt ist; weil Abläufe bzgl Kapazitätsentwicklung stattfinden werden)  
  3. Alle miteinander vergleichen: Für jedes Simulationssetting (K-Index) ermitteln, welche Strategie

# Große Punkte

* Rote Kästen im LaTeX
* API linear exponential smoothing  
  Dokument: 0618 - 1 Exploring exponential smoothing (hier ohne epsilon-greedy)  
  Dokument 0618 - 2 E\_API\_lin\_single\_leg with epsilon greedy  
  exponential smoothing wie in Koch, 3.3. Update of parameters erwähnt, ist im Wesentlichen eine Durchschnittsbildung
* T-Test für Vergleich verschiedener Policies  
  Code: T\_compare\_strategies\_single\_leg  
  matched pairs t-test mit Null-Hypothese API - DP < 0 (erwarte, dass DP besser ist)

# Kleine Punkte

* Sprache: A customer arrives. “He” oder “She” oder “It” buys a product.
* Arrival probability wird bei Bront et al über einen Poisson Prozess modelliert, was zu mehr als einem Customer pro Zeitschritt führen könnte. Im Erwartungswert lambda pro Zeitschritt. Ich habe implementiert mit maximal 1 customer pro Zeitschritt (Summe der Ankunftswkeiten < 1), relevant bei simulate\_sales(offer\_set).
* Algorithmus determine\_offer\_tuple (von Bront et al) anpassen, dass nur mögliche Produkte in s\_prime berücksichtigt werden (Kapazität); passt schon, da cap = 0 => pi = \infty