INSTITUT FÜR STATISTIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Analyse von Flugdaten des Münchner Flughafen

Strukturbruchtests und Monitoring

Bachelorarbeit

Autor:	Giuseppe Casalicchio
Matrikelnummer:	8049501
Gutachter:	Prof. Dr. Torsten Hothorn
Betreuer:	Manuel Eugster
Abgabe:	07. September 2011

Abstract

Die vorliegende Bachelorarbeit befasst sich mit dem Testen auf Strukturbrüche unter Verwendung eines generalisierten M-Fluktuationsprozesses, der zur Erfassung von Schwankungen in einem Modell verwendet werden kann. Falls Strukturbrüche vorhanden sind, wird eine Methode vorgestellt, die das Lokalisieren von Strukturbrüchen ermöglicht. Darüber hinaus wird das Konzept des Monitorings kurz erläutert, welches mit Hilfe eines historischen Datensatzes bei neu hinzukommenden Beobachtungen strukturelle Veränderungen aufdecken kann. Diese Methoden werden anhand von Daten der An- und Abflüge des Flughafen München illustriert.

Inhaltsverzeichnis

1. Einführung

Seit geraumer Zeit war das Fliegen lernen eines der größten Träume der Menschheit. Heutzutage ist aus dem ehemaligen Menschheitstraum eine selbstverständliche und für viele Reisende unverzichtbare Transportmöglichkeit geworden. Die Gründe liegen klar auf der Hand: Wegen der kurzen Reisedauer und der fairen Flugpreise werden Flüge für Urlaubsreisen, Geschäftsreisen und andere Reisegründe immer beliebter.

Dieser Trend lässt sich auch am Franz-Josef-Strauss Flughafen in München beobachten. Der Flughafen liegt 28,5km nordöstlich des Stadtzentrum Münchens und ist nach Frankfurt der zweitgrößte Flughafen in Deutschland. So gab es im Jahr 2001 am Münchner Flughafen 23,6 Millionen Fluggäste im gewerblichen Luftverkehr, 2009 waren es 32,7 Millionen und ein Jahr später 34,7 Millionen [Dr. Reingard Schöttl 2011]. Täglich starten und landen am Franz-Josef-Strauss Flughafen nahezu 1000 Flugzeuge pro Tag (vgl. dazu Tabelle 4.2). Dabei stellt sich die Frage, ob die Anzahl der Flugbewegungen innerhalb eines Zeitintervalls konstant bleibt oder sich (beispielsweise wegen saisonaler Schwankungen) über die Zeit hinweg verändert. Ursachen für (saisonale) Schwankungen beim Flugaufkommen können beispielsweise Wintereinbrüche oder Naturkatastrophen wie Vulkanausbrüche sein, die das Streichen von geplanten Flügen bewirken können. An Sonn- und Feiertagen sowie kurz vor und nach den Ferien könnte man ebenfalls eine Veränderung in der Anzahl an Flugbewegungen vermuten.

In Abbildung 1.1 sieht man einen kurzen Überblick über das tägliche Flugaufkommen am Flughafen München vom 9. November 2010 bis zum 9. Februar 2011. Die Anzahl an Flugbewegungen sinkt bis zur Mitte hin und nimmt dann wieder zu (vgl. Kapitel 4.2.1). Solche Grafiken mit einer zeitlich geordneten Einheit auf der x-Achse und einer beliebigen (reellen) Größe auf der y-Achse stellen Zeitreihen dar (siehe Kapitel 2). Ein wichtiger Aspekt in Zeitreihen ist die Analyse von Strukturbrüchen. In einem Regressionsansatz sind Strukturbrüche vorhanden, wenn sich die geschätzten Regressionsparameter einer Zeitreihe in bestimmten Zeitintervallen unterscheiden.

Ziel dieser Abschlussarbeit ist es zu überprüfen, ob unter Verwendung eines geeigneten Modells solche Änderungen der Regressionsparameter in den vorliegenden Flugdaten existieren, wie oft sie vorkommen und wo sie sich befinden. Zudem soll durch Monitoring (siehe Kapitel 3.5) mit Hilfe eines historischen Datensatzes beobachtet werden, ob in neu hinzukommenden Daten aktuelle Veränderungen in der Struktur einer Zeitreihe vorhanden sind.

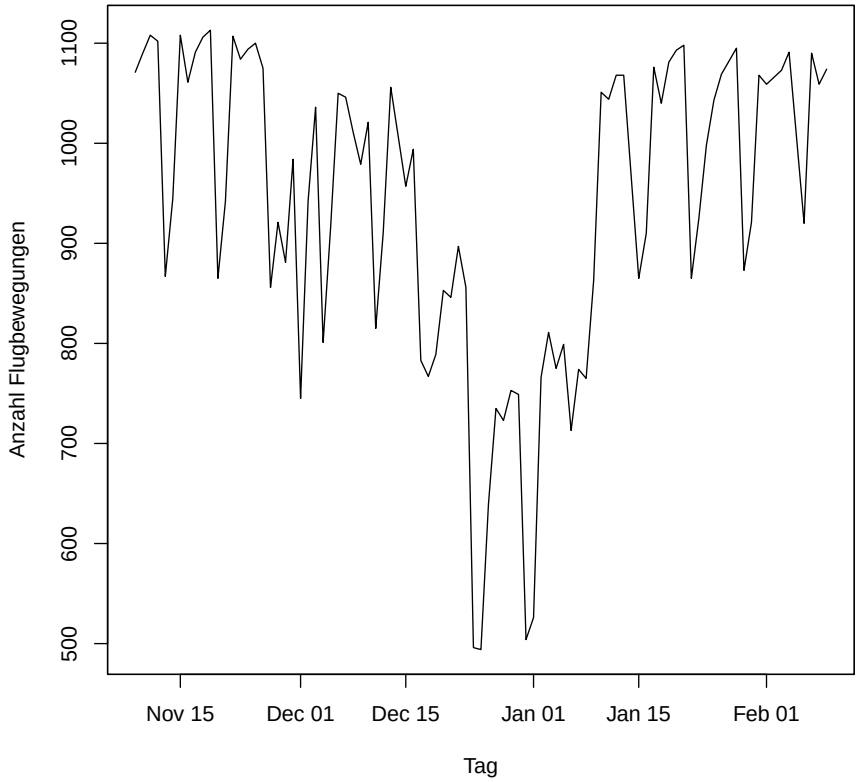

Abbildung 1.1.: Anzahl der Flugbewegungen vom 9. November 2010 bis zum 9. Februar 2011

2. Zeitreihen

Sei $(x_i)_{i \in T}$ eine Folge zeitlich geordneter Realisierungen von Zufallsvariablen X_i eines Wahrscheinlichkeitsraumes $(\Omega, \mathcal{F}, P)$ zu den Zeitpunkten i.

Dann gilt:

1. Die Familie $\{X_i : i \in T\}$ von Zufallsvariablen heißt stochastischer Prozess.

2. Die Folge $(x_i)_{i \in T}$ heißt Zeitreihe.

Demnach ist eine Zeitreihe $(x_i)_{i \in T}$ nach formaler Definition die Realisierung eines stochastischen Prozesses $\{X_i : i \in T\}$, wobei die Indexmenge T im Allgemeinen echte Teilmenge von $\mathbb{R}$ und eine diskrete Menge ist $(T \subset \mathbb{R})$ [Wenzel et al. 2010; Kreiß and Neuhaus 2006].

In einem Datensatz beginnt eine Zeitreihe immer ab einem bestimmten Startzeitpunkt zum Beispiel $i = 1$, zu dem die erste Beobachtung vorliegt. Auch wenn es im Datensatz keine Daten zu den vorangegangenen Zeitpunkten gibt, existieren theoretisch auch Daten zu den vergangenen Zeitpunkten $i-1, i-2, ..., i-n$ $(n \in \mathbb{N})$ einer Zeitreihe. Deshalb ist es sinnvoll für die diskrete Menge T vorauszusetzen, dass sie aus den ganzen Zahlen stammt $(T \in \mathbb{Z})$. Würde man $T \in \mathbb{N}$ fordern, so müsste man für den Fall, dass die Zeitreihe mit älteren Daten erweitert werden kann, den Startwert immer wieder neu festlegen.

Wenn die Zeitpunkte $i \in T$ äquidistant sind, also den gleichen Abstand besitzen, handelt es sich um eine regelmäßige (oder auch äquidistante) Zeitreihe. Andernfalls (wenn z.B. fehlende Werte in der Zeitreihe vorkommen) ist es eine unregelmäßige Zeitreihe. Da regelmäßige Zeitreihen einfacher zu handhaben sind und eine bessere Interpretierbarkeit ermöglichen, werden im Folgenden unregelmäßige Zeitreihen vernachlässigt [Kreiß and Neuhaus 2006].

Auf der x-Achse einer Zeitreihe können Jahre, Monate, Wochen, Tage oder Stunden eingetragen werden. Die y-Achse enthält dabei die zu untersuchende (meist reelle) Größe. In dieser Abschlussarbeit werden als Beobachtungen auf der y-Achse die Anzahl der Flugbewegungen stehen, wobei sich auf der Zeitachse hauptsächlich Tage befinden.

2.1. Das klassische Komponentenmodell

Grundsätzlich kann man eine Zeitreihe in folgende Komponenten aufteilen:

- **Trendkomponente** m_i**:** Stellt den mittleren, langfristigen Verlauf der Zeitreihe dar.
- **Konjunkturkomponente** k_i**:** Eine (nicht unbedingt regelmäßige) Schwankung, die über einen größeren Zeitraum vorhanden sein kann.
- **Saisonkomponente** s_i**:** Stellt eine innerhalb eines Zeitintervalls regelmäßig wiederholende Struktur dar, die beispielsweise jahreszeitlich bedingt sein kann.
- **Restkomponente** r_i**:** Enthält unerklärbare Einflüsse und Störungen, die in der Zeitreihe vorhanden sind.

Man kann die Konjunkturkomponente entweder mit der Trendkomponente zu einer sogenannten **glatten Komponente** oder mit der Saisonkomponente zu einer **zyklischen Komponente** zusammenfassen (vgl. Abbildung 2.1).

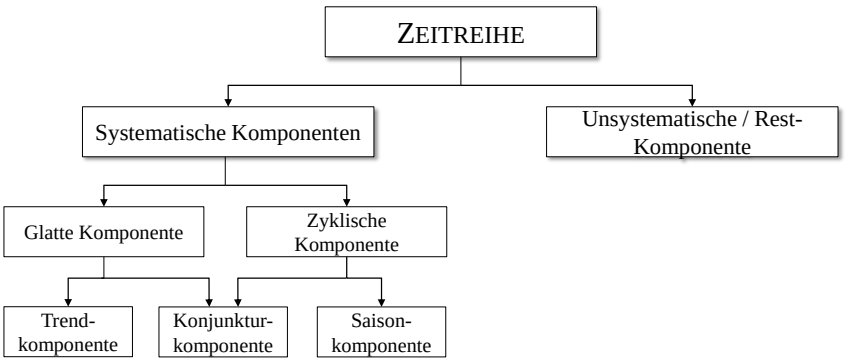

Abbildung 2.1.: Komponentenzerlegung [Wenzel et al. 2010, S. 35]

Da keine langjährige Daten in den vorliegenden Flugdaten vorhanden sind, bleibt die Konjukturkomponente im Folgenden unberücksichtigt.

Eine Zeitreihe kann unter anderem additiv

$$x_i = m_i + s_i + r_i \tag{2.1}$$

oder multiplikativ aufgeteilt werden

$$x_i = m_i \cdot s_i \cdot r_i \tag{2.2}$$

2.2. Zeitreihenzerlegung

In Zeitreihen kommen oft zyklische Schwankungen vor. Die Länge eines Zyklus wird als Periodenlänge $d \in \mathbb{N}$ bezeichnet und hängt vom Datenzyklus ab. Bei wöchentlichen Zyklen ist $d = 7$, bei monatlichen Zyklen $d = 12$ und bei Quartalsdaten $d = 4$ [Kreiß and Neuhaus 2006].

Die Methode des gleitenden Durchschnitts (Moving Average) ist eine einfache Möglichkeit für eine Zeitreihenzerlegung und wird im Folgenden für die Trendschätzung verwendet.

2.2.1. Trendschätzung

Sei $\mathbb{N}_g = \{2, 4, 6, 8, \dots\}$ die Menge der natürlichen geraden Zahlen und $\mathbb{N}_u = \{1, 3, 5, 7, \dots\}$ die Menge der natürlichen ungeraden Zahlen. Für ein festes $q \in \mathbb{N}$ und Gewichte $w_j \in \mathbb{R}$, $j = -q, \dots, q$ wird der symmetrische gleitende Durchschnitt zum Zeitpunkt i aus den q benachbarten Beobachtungen $X_{i-q}, \dots, X_{i+q}$ berechnet, sodass der Trendschätzer durch

$$\hat{m}_i = \sum_{j=-q}^{q} w_j X_{i+j}, \qquad i = q+1, \dots, n-q \tag{2.3}$$

gegeben ist. Dazu werden immer $2q + 1$ Beobachtungen verwendet, sodass die Gewichte typischerweise als $w_j = 1/(2q + 1), j = -q, ..., q$ gewählt werden.

Das feste q ist idealerweise von der Periodenlänge d der Zeitreihe abhängig, sodass man immer den Zyklus einer Periode zusammenfasst:

$$q = \begin{cases} d/2 & d \in \mathbb{N}_g \\ (d-1)/2 & d \in \mathbb{N}_u \end{cases} \tag{2.4}$$

Damit bei konstanten Beobachtungen $x_i = c \ \forall i$ der gleitende Durchschnitt ebenfalls den Wert c annimmt, gilt für die Gewichte $\sum_{j=-q}^{q} w_j = 1$ [Kreiß and Neuhaus 2006, S. 10f].

Falls die Periodenlänge eine ungerade Zahl ist ($d \in \mathbb{N}_g$), geht die erste und letzte Beobachtung X_{i-q} und X_{i+q} mit den Gewichten $w_{-q} = w_q = 0.5/(2q + 1)$ in die Mittelwertsberechnung ein. Dadurch ist der gleitende Durchschnitt symmetrisch um den jeweiligen Beobachtungswert für i [Winkler 2006, Kap. 10.4].

Für $i < q+1$ und $i > n-q$, d.h. für die ersten und letzten q Beobachtungen können, wegen einseitig zu wenig vorhandenen Werten, für die Trendkomponente keine symmetrischen gleitenden Durchschnitte berechnet werden.

2.2.2. Saisonbereinigung

Seien $X_1, \ldots, X_n$ trendbereinigte Daten. Weiterhin wird angenommen, dass m komplette Perioden der Länge d vorliegen, sodass $n = d \cdot m$. Für den r-ten Wert einer Periode lässt sich der saisonale Anteil definieren als

$$\tilde{s}_r = \frac{1}{m} \sum_{j=1}^{m} X_{r+(j-1)\cdot d} \qquad (r = 1, ..., d). \tag{2.5}$$

Die saisonalen Anteile sollen sich aufsummiert aufheben, sodass man für die Schätzung der Saisonkomponente

$$\hat{s}_r = \tilde{s}_r - \frac{1}{d} \sum_{j=1}^{d} \tilde{s}_j \tag{2.6}$$

verwendet. Die geschätzten saisonalen Anteile $\hat{s}_1, ...\hat{s}_d$ sind für alle m Perioden gleich und werden gemäß $\hat{s}_t = \hat{s}_{t-d}$ für $t > d$ periodisch fortgesetzt, sodass $\hat{s}_1, ...\hat{s}_d$ insgesamt m mal vorkommt und für jedes $i \in T$ ein s_i existiert [Kreiß and Neuhaus 2006].

Die Restkomponente kann für eine additive Zerlegung durch Umformung von Gleichung (2.1) nach

$$\hat{r}_i = x_i - \hat{s}_i - \hat{m}_i \tag{2.7}$$

und für eine multiplikative Zerlegung durch Umformung von Gleichung (2.2) nach

$$\hat{r}_i = x_i \cdot \hat{s}_i \cdot \hat{m}_i \tag{2.8}$$

geschätzt werden.

3. Analyse von Strukturbrüchen

3.1. Motivation

In Zeitreihen kommen häufig strukturelle Veränderungen vor, d.h. im Laufe der Zeit kann sich der Einfluss der unabhängigen Variablen auf die abhängige Variable verändern. Die Zeitpunkte, an denen solche strukturelle Veränderungen vorkommen, nennt man Strukturbrüche oder Bruchpunkte.

In einem linearen Regressionsmodell sind Strukturbrüche als Änderung der geschätzten Parameter über die Zeit zu verstehen (vgl. Kapitel 3.3.1). Daher werden in parametrischen Modellen solche strukturelle Veränderungen auch als Parameterinstabilität beschrieben. Falls eventuelle Parameterinstabilitäten in Modellen nicht berücksichtigt werden, führt dies zu verzerrte Inferenzen, ungenauen Prognosen und Parameterschätzungen, die nicht mehr sinnvoll interpretiert werden können [Zeileis and Hornik 2007].

Eine rein visuelle Analyse von Abbildung 1.1 suggeriert, dass es in der Zeitreihe strukturelle Veränderungen gibt, da sich die Anzahl an Flugbewegungen pro Tag nicht durchgehend um einen konstanten Wert bewegt. Im Folgenden wird ein allgemeiner statistischer Test vorgeschlagen, der es ermöglicht diese Vermutung formal zu überprüfen.

3.2. Generalisierte M-Fluktuationstests auf Parameterinstabilität

Die Literatur schlägt eine Vielzahl an Strukturbruchtests vor. Zeileis [2005] unterteilt diese in drei Klassen:

1. Tests basierend auf ML-Scores z.B. der Nyblom-Hansen Test
2. Tests basierend auf F-Statistiken z.B. der supF Test
3. Tests basierend auf OLS Residuen z.B. der OLS-CUSUM Test

Obwohl diese drei Testklassen nahezu unabhängig voneinander entwickelt wurden und unterschiedliche Schätzverfahren (ML-Schätzer, generalisierte Momentenschätzer (GMM) und kleinste Quadrate Schätzer (OLS)) verwenden, stehen sie miteinander mehr in Verbindung als auf dem ersten Blick erwartet und können den generalisierten M-Fluktuationstests

zugeordnet werden. Dabei werden unterschiedliche Funktionen zur Erfassung von Schwankungen verwendet, die auf den selben generalisierten M-Fluktuationsprozess basieren [Zeileis 2005].

In Kapitel 3.3.2 wird als Beispiel gezeigt, dass sich der OLS-CUSUM Prozess aus der dritten Testklasse durch einen generalisierten M-Fluktuationsprozess ausgedrücken lässt. Zeileis [2005] zeigt, dass sich die ersten beiden Testklassen ebenfalls in Abhängigkeit eines generalisierten M-Fluktuationsprozesses hinschreiben lassen.

Generalisierte M-Fluktuationstests können somit als allgemeiner Ansatz zum Testen auf Parameterinstabilitäten verstanden werden. Zur Konstruktion eines solchen Tests sind folgende Schritte nötig [Zeileis and Hornik 2007; Zeileis 2006]:

1. Wahl eines geeigneten Schätzverfahrens bzw. einer Scorefunktion.
2. Verwendung eines empirischen Partialsummenprozesses, der gegen eine brownsche Brücke konvergiert, um die Schwankungen des geschätzten Modells zu erfassen.
3. Schwankung innerhalb des Prozesses wird mit Hilfe einer Funktion (der Teststatistik) gemessen und mit dem kritischen Wert einer Grenzverteilung verglichen.

3.2.1. Generalisierter M-Fluktuationsprozess

In diesem Abschnitt wird ein allgemeiner Fluktuationsprozess definiert, den man zum Erfassen von Parameterinstabilitäten eines Modells benötigt. Dazu betrachtet man n unabhängige Beobachtungen

$$Y_i \sim F(\theta_i) \qquad (i = 1, ..., n), \tag{3.1}$$

die einer Verteilung F mit einem k-dimensionalen Parameter θ_i folgen. Weiterhin wird angenommen, dass es sich bei dem Index $i = 1, ..., n$ um (meist zeitlich) geordnete Beobachtungen handelt.

Man interessiert sich nun dafür, ob die Parameter θ_i über die Zeit i hinweg konstant bleiben. Die Nullhypothese lautet dann

$$H_0 : \theta_i = \theta_0 \qquad (i = 1, ..., n). \tag{3.2}$$

und wird der Alternativhypothese, dass sich mindestens ein Parameter θ_i über die Zeit verändert hat, gegenübergestellt. Wichtig ist hierbei die zeitliche Anordnung, damit strukturelle Veränderungen richtig interpretiert werden können [Zeileis and Hornik 2007].

Um die Hypothese testen zu können, wird mit Hilfe der M-Schätzung mit $\psi(\cdot)$ als Scorefunktion (vgl. dazu Kapitel 3.2.2.1) der Parametervektor θ für alle n Beobachtungen geschätzt.

Für den Erwartungswert des wahren Parameters θ_i gilt $E(\psi(Y_i, \theta_i)) = 0$ [Zeileis 2005, 2006].

Der M-Schätzer $\hat{\theta}$ ist gegeben durch

$$\underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{n} \Psi(Y_i, \theta) = \hat{\theta} \qquad \Leftrightarrow \qquad \sum_{i=1}^{n} \psi(Y_i, \hat{\theta}) = 0. \tag{3.3}$$

Falls man beispielsweise für die Bestimmung von $\hat{\theta}$ die OLS-Methode oder ML-Schätzung verwendet, ist Ψ als Residuenquadratsumme oder Log-Likelihood definiert. Für die kummulierte Summe erster Ordung ψ führen beide Schätzverfahren zu einem Fluktuationsprozess, der Schwankungen erfassen kann [Zeileis 2003; Zeileis et al. 2010].

Unter H_0 ist der Mittelwert der Scores $\psi(Y_i, \hat{\theta})$ gleich Null, jedoch existieren unter der Alternativhypothese systematische Abweichungen, die von Null verschieden sind. Diese Abweichungen können mit Hilfe des kummulativen Summenprozess der Scores

$$W_n(t, \theta) = n^{-\frac{1}{2}} \sum_{i=1}^{\lfloor nt \rfloor} \psi(Y_i, \theta) \tag{3.4}$$

aufgedeckt werden [Zeileis 2006].

Der empirische Fluktuationsprozess $efp(\cdot)$ konvergiert nach Zeileis [2006] gegen eine k-dimensionale brownsche Brücke $W^0(\cdot)$ und ist gegeben durch

$$efp(t) = \hat{J}^{-\frac{1}{2}} W_n(t, \hat{\theta}), \tag{3.5}$$

wobei $\hat{J}$ ein konsistenter Schätzer für die Kovarianzmatrix der Scores ψ ist und zum Beispiel wie folgt aussehen kann [Zeileis and Hornik 2007]:

$$\hat{J} = \frac{1}{n} \sum_{i=1}^{n} \psi(Y_i, \hat{\theta}) \psi(Y_i, \hat{\theta})^\top \tag{3.6}$$

3.2.2. Generalisierte M-Fluktuationstests

3.2.2.1. Wahl des Scores

In generalisierten M-Fluktuationstests wird die M-Schätzung zur Bestimmung der Parameter θ verwendet, sodass durch geeignete Wahl der Scorefunktion ψ auch andere Schätzverfahren hergenommen werden können. Als ψ wählt man meistens die partielle Ableitung einer Zielfunktion Ψ, d.h.

$$\psi(y, \theta) = \frac{\partial \Psi(y, \theta)}{\partial \theta}. \tag{3.7}$$

M-Schätzung deshalb, weil es die ML-Schätzung und OLS-Schätzung impliziert und mit der GMM-Schätzung verwandt ist, sodass der Zusammenhang zwischen den drei Testklassen aus Kapitel 3.2 deutlich wird.

Falls beispielsweise $\Psi(y, \theta) = \Psi_{NLL}(y, \theta) = -log(f(y, \theta))$ die negative Log-Likelihood ist, gilt für Gleichung (3.3)

$$\begin{aligned} \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{n} \Psi(Y_i, \theta) = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{n} -log(f(y, \theta)) \\ \Leftrightarrow \underset{\theta \in \Theta}{\operatorname{argmax}} \sum_{i=1}^{n} log(f(y, \theta)), \end{aligned} \tag{3.8}$$

welches genau der ML-Schätzung für $\hat{\theta}$ entspricht. Analog ergibt sich für $\Psi(y, \theta) = \Psi_{RSS}(y, \theta) = (y_i - x_i^\top \theta)$ die Minimierung der Residuenquadratsumme, was der OLS-Schätzung entspricht.

Die Schätzung von θ basiert also auf eine Score- oder Schätzfunktion ψ beziehungsweise auf eine Momenten- oder Orthogonalitätsbedingung ähnlich zu Gleichung (3.3), für deren partielle Summe auch ein Fluktuationsprozess mit den selben asymptotischen Eigenschaften hergeleitet werden kann [Zeileis and Hornik 2007].

3.2.2.2. Teststatistik

Zur Beschreibung von Parameterinstabilitäten wurde ein empirischer Fluktuationsprozess definiert, der nun zum Aufstellen einer Teststatistik verwendet wird. Dabei betrachtet man nicht direkt den empirischen Fluktuationsprozess $efp(\cdot)$, sondern verwendet eine geeignete Transformation $\lambda(efp(\cdot))$, die auf den Fluktuationsprozess angewendet wird.

Der empirische Fluktuationsprozess ist eine Matrix $(efp_j(i/n))_{i,j}$ mit $i = 1, ..., n$ Beobachtungen über die Zeit und $j = 1, ..., k$ Komponenten des Parametervektors θ. Daher kann man die Funktion λ in λ_{time} (für eine Aggregation über die Zeit) und λ_{comp} (für eine Aggregation über die Komponenten) aufteilen, wodurch zwei mögliche Strategien zur Konstruktion einer Teststatistik entstehen:

1. Zuerst aggregiert man den Prozess über die Zeit, wodurch k unabhängige und univariate Teststatistiken entstehen, die alle einer Komponenten des Prozesses und somit auch des Parametervektors θ zugeordnet werden können.
2. Zuerst aggregiert man den Prozess über die Komponenten, sodass ein Fluktuationsprozess entsteht, der den Zeitpunkt potentieller Strukturbrüche erkennt.

Für λ_{time} können üblicherweise das absolute Maximum oder der Mittelwert gewählt werden. Typische Funktionen λ_{comp} beinhalten die Maximumsnorm L_∞ oder die quadrierte euklidische Norm L_2. Je nachdem, was man zuerst aggregiert, können entweder die Komponenten eines möglichen Strukturbruchs oder die Zeitpunkte identifiziert werden.

Im Folgenden werden Teststatistiken aus der zweiten Strategie mit der Form

$$\lambda_{time}\left(\frac{\lambda_{comp}(efp(t))}{d(t)}\right) \tag{3.9}$$

betrachtet, wobei $d(\cdot)$ eine Gewichtsfunktion für die Form einer Schranke ist und in dieser Abschlussarbeit dafür ein konstanter Wert $d(t) = 1$ verwendet wird.

Die Schranke zur Ablehnung der Nullhypothese hat die allgemeine Form $b(t) = c \cdot d(t)$, wobei mit c das Signifikanzniveau und mit $d(t)$ die Form der Schranke (linear, quadratisch ect.) festgelegt wird [Zeileis 2005; Zeileis and Hornik 2007; Zeileis 2006].

Für die Analysen in Kapitel 4.3 wird der Double Maxium Test und der OLS-CUSUM Test verwendet (siehe Kapitel 3.3.2). Deshalb werden in dieser Abschlussarbeit dise beiden Teststatistiken genauer betrachtet. Weitere mögliche Teststatistiken können in Zeileis and Hornik [2007] nachgeschlagen werden.

Die Double Maximum Teststatistik ist die einzige mit der man sowohl jede Komponente, als auch die Zeitpunkte eines möglichen Strukturbruchs identifizieren kann und lautet für $d(t) = 1$:

$$\max_{i=1,...,n} \max_{j=1,...,k} \left| efp_j(i/n) \right| \tag{3.10}$$

Die Grenzverteilung entspricht dem Maximum einer brownschen Brücke und ist gegeben durch $max_{j=1,...,k} ||W_j^0(t)||_\infty$. Die Schranke für die Teststatistik aus Gleichung (3.10) lautet also $b(t) = c \cdot 1$. Falls ein bestimmter kritischer Wert c der Grenzverteilung mit

einer gegebenen Wahrscheinlichkeit α überschritten wird, ist die Schwankung innerhalb des empirischen Fluktuationsprozesses ungewöhnlich groß, sodass die Nullhypothese auf Signifikanzniveau α verworfen werden kann [Zeileis and Hornik 2007; Zeileis et al. 2002].

3.3. Parameterinstabilität in linearen Regressionsmodellen

In diesem Abschnitt wird der allgemeine Ansatz aus dem vorherigen Kapitel auf das lineare Regressionsmodell angewendet und ein Test vorgestellt, der auf OLS Residuen basiert.

Im Modell aus Gleichung (3.1) wurde angenommen, dass die Dichte $f(y_i, \theta_i)$ die vollständige Verteilung von Y_i beschreibt. Nun können die Beobachtungen, wie in einem Regressionsansatz üblich, auch in $Y_i = (y_i, x_i)^\top$ mit zusätzlichen Regressoren x_i und der abhängigen Variablen y_i aufgeteilt werden. Um für diesen Fall ein Fluktuationstest abzuleiten, wird normalerweise die bedingte Dichte $f(y_i|x_i, \theta_i)$ von y_i unter der Bedingung x_i betrachtet. Damit die Abhängigkeit der Kovariablen deutlich wird, kann die Scorefunktion ψ wie folgt geschrieben werden:

$$\psi(Y_i, \theta_i) = \psi(y_i, x_i, \theta_i) \tag{3.11}$$

Für Gleichung (3.4) folgt damit:

$$W_n(t, \theta) = \frac{1}{\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} \psi(y_i, x_i, \theta) \tag{3.12}$$

Es wird weiterhin angenommen, dass der Erwartungswert null ist. Für die Varianzen gilt:

$$\frac{1}{n} \sum_{i=1}^{n} \mathrm{COV}[\psi(y_i, x_i, \theta_0)] = J_n \quad \xrightarrow{\mathrm{p}} \quad J, \tag{3.13}$$

wobei J der Kovarianzmatrix $\mathrm{COV}[\psi(Y, \theta)]$ ohne den Regressoren x_i entspricht. Diese Konvergenz folgt aus der Annahme, dass die x_i einen schwach abhängigen Prozess ohne einen stochastischen Trend beschreiben [Zeileis and Hornik 2007; Zeileis 2003].

3.3.1. Modellannahmen

Betrachtet wird das Regressionsmodell

$$y_i = {x_i}^T \beta_i + u_i \qquad (i = 1, ..., n) \tag{3.14}$$

mit

y_i:	Response zum Zeitpunkt i
${x_i}^T = (1, x_{i2}, ..., x_{ik})$:	$1 \times k$ Zeilenvektor mit den in das Modell aufgenommenen (unabhängigen) Einflussgrößen zum Zeitpunkt i
β_i:	$k \times 1$ Spaltenvektor der Regressionskoeffizienten
$u_i \sim iid(0, \sigma^2)$:	Vom Zeitpunkt i abhängige Störgröße

Dabei darf der Index i bei den Regressionskoeffizienten β_i nicht weggelassen werden. Andernfalls wäre nicht mehr gewährleistet, dass die Regressionskoeffizienten vom Zeitpunkt i abhängig sein könnten.

Mit Strukturbruchtests soll getestet werden, ob sich die Regressionskoeffizienten β_i über die Zeit hinweg unterscheiden oder konstant bleiben. Die Nullhypothese kann wie folgt formuliert werden:

$$H_0 : \beta_i = \beta_0 \qquad (i = 1, ..., n) \tag{3.15}$$

Wenn das Testergebnis dazu führt, dass H_0 nicht verworfen werden kann, bedeutet dies, dass die Regressionskoeffizienten über die Zeit hinweg immer gleich groß sind und somit kein Strukturbruch vorliegt. Kann H_0 abgelehnt werden, gibt es in der Zeitreihe mindestens einen Strukturbruch [Zeileis et al. 2002].

Der Parametervektor $\theta = (\beta, \sigma^2)^\top$ wird, wie in Kapitel 3.2.2.1 beschrieben, mit Hilfe der ML-Schätzung oder OLS-Schätzung bestimmt. Die Varianz σ^2 wird als lästiger Parameter (nuisance parameter) behandelt, sodass sich für die Scores aus Gleichung (3.11)

$$\psi_\beta(y_i, x_i, \beta) = x_i(y_i - x_i^\top \beta) \tag{3.16}$$

ergibt. Bei Modellen mit Intercept gilt für die erste Komponente der Scores [Zeileis 2005]:

$$(\psi_\beta(y_i, x_i, \hat{\beta}))_1 = (y_i - x_i^\top \hat{\beta}) = \hat{u}_i \tag{3.17}$$

3.3.2. Der OLS-CUSUM Test

Neben den Double Maximum Test aus Gleichung (3.10), der ebenfalls auf das lineare Regressionsmodell anwendbar ist, wird hier noch der OLS-CUSUM Test vorgestellt. Der OLS-CUSUM Prozess basiert auf OLS Residuen und ist gegeben durch

$$W_n(t) = \frac{1}{\hat{\sigma}\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} \hat{u}_i \qquad \forall t \in [0,1]. \tag{3.18}$$

Dabei gilt nach Zeileis et al. [2002] für die geschätzte Varianz des OLS-CUSUM Prozesses

$$\hat{\sigma}^2 = \frac{1}{n-k} \sum_{i=1}^{n} \hat{u}^2. \tag{3.19}$$

Ploberger and Krämer [1992] schlagen folgende Teststatistik für den OLS-CUSUM Test vor

$$\sup_{0 \le t \le 1} |W_n(t)|. \tag{3.20}$$

Man kann zeigen, dass sich diese Teststatistik in Abhängigkeit eines generalisierten M-Fluktuationstest $efp(t)$ hinschreiben lässt, welches die Form aus Gleichung (3.9) hat:

$$\begin{aligned}
\sup_{0 \le t \le 1} \left| \frac{1}{\hat{\sigma}\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} \hat{u}_i \right| &\overset{(3.17)}{=} \sup_{0 \le t \le 1} \left| \frac{1}{\hat{\sigma}\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} (y_i - x_i^\top \hat{\beta}) \right| \\
&\overset{(3.17)}{=} \sup_{0 \le t \le 1} \left| \frac{1}{\hat{\sigma}\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} \left(\psi_\beta(y_i, x_i, \hat{\beta}) \right)_1 \right| \\
&\overset{(3.12)}{=} \sup_{0 \le t \le 1} \left| \frac{1}{\hat{\sigma}\sqrt{n}} \sqrt{n} \left(W_n(t, \hat{\beta}) \right)_1 \right| \\
&\overset{(3.5)}{=} \sup_{0 \le t \le 1} \left| \hat{J}_{1,1}^{-1/2} \left(\hat{J}^{1/2} efp(t) \right)_1 \right|
\end{aligned} \tag{3.21}$$

Dabei entspricht λ_{time} der Supremumsfunktion bezüglich $t \in [0,1]$ und λ_{comp} dem absoluten Wert der ersten Komponente des mit $\hat{J}^{1/2}$ skalierten Prozesses $efp(t)$ [Zeileis 2005].

Da für den OLS-CUSUM Prozess die Schätzung für σ^2 aus Gleichung (3.19) verwendet wird, definiert man die geschätzte Kovarianzmatrix als

$$\hat{J} = \frac{1}{n-k} \sum_{i=1}^{n} \psi(y_i, x_i, \hat{\beta}) \psi(y_i, x_i, \hat{\beta})^\top, \tag{3.22}$$

sodass das erste Diagonalelement $\hat{J}_{1,1}$ der Matrix $\hat{J}$ genau dem $\hat{\sigma}^2$ aus Gleichung (3.19) entspricht.

In Anhang A.1 wird der Beweis aus Formel (3.21) in `R` [R Development Core Team 2010] reproduziert. Dabei sei angemerkt, dass der OLS-CUSUM Prozess $W_n(t)$ im `strucchange` Paket [Zeileis et al. 2002] in der Funktion `efp()` enthalten ist und daher nicht mit dem generalisierten M-Fluktuationsprozess $efp(\cdot)$, der in `strucchange` in der Funktion `gefp()` implementiert ist, verwechselt werden darf.

Beim OLS-CUSUM Test wird die Nullhypothese verworfen, wenn eine bestimmte Schranke der Form $b(t) = c$ überschritten wird. Zeileis et al. [2002] schlägt dafür auch eine alternative Schranke $b_{\text{alternative}}(t) = c \cdot \sqrt{t(1-t)}$ vor, die proportional zur Standardabweichung des entsprechenden Prozesses ist.

3.4. Lokalisieren und Schätzen von Bruchpunkten

Die Klasse der M-Fluktuationstests (dazu gehören z.B. auch der Double Maximum Test und der OLS-CUSUM Test) gibt lediglich Auskunft darüber, ob Strukturbrüche in einem Modell vorhanden sind. In der Praxis interessiert man sich aber auch dafür, wo sich diese Strukturbrüche befinden. Eine Visualisierung der M-Fluktuationstests mit den dazugehörigen Schranken kann bereits Aufschluss darauf geben, an welchen ungefähren Zeitpunkten die Schranke überschritten wird und wo sich die Bruchpunkte befinden (vgl. Kapitel 4.3, insbesondere Abbildung 4.5). Man möchte jedoch eine formale Prozedur um die Anzahl und Lage der Bruchpunkte schätzen zu können. Diese von Bai and Perron [1998] entwickelte Prozedur wird in diesem Abschnitt kurz vorgestellt und ist im `strucchange` Paket in der Funktion `breakpoints()` implementiert [Kleiber and Zeileis 2008].

3.4.1. Segmentiertes Regressionsmodell

Wenn die Anzahl an Bruchpunkten m ist, gibt es $m + 1$ segmentierte Modelle, deren Regressionskoeffizienten konstant sind. Das Modell aus Gleichung (3.14) lässt sich dann umschreiben als:

$$y_i = {x_i}^T \beta^{(j)} + u_i \qquad (i = i_{j-1} + 1, ..., i_j, \quad j = 1, ..., m + 1) \tag{3.23}$$

Dabei entspricht j dem Segmetindex und $\beta^{(j)}$ den Regressionskoeffizienten des j-ten Segments. Die Indizes $\{i_1, ..., i_m\}$ werden als Menge aller Bruchpunkte an denen gesplittet wird bezeichnet, wobei konventionell $i_0 = 0$ und $i_{m+1} = n$ gilt [Zeileis et al. 2003; Kleiber and Zeileis 2008].

3.4.2. Schätzen von Bruchpunkten

Ziel ist es ein segmentiertes Modell mit m Bruchpunkten und $m+1$ segmentspezifischen Regressionskoeffizienten zu finden für das die Residuenquadratsumme minimal wird [Kleiber and Zeileis 2008].

Die segmentierte Zielfunktion basierend auf Ψ lässt sich hinschreiben als

$$\begin{aligned} RSS(i_1, ..., i_m) &= \sum_{j=1}^{m+1} rss(i_{j-1}+1, i_j) \\ rss(i_{j-1}+1, i_j) &= \sum_{i=i_{j-1}+1}^{i_j} \Psi(y_i, x_i, \hat{\beta}^{(j)}), \end{aligned} \tag{3.24}$$

wobei $rss(i_{j-1}+1, i_j)$ der minimalen Residuenquadratsumme im j-ten Segment entspricht (mit $\Psi = \Psi_{RSS}$ aus Kapitel 3.2.2.1).

Gesucht sind nun die Schätzer $(\hat{i}_1, ...\hat{i}_m)$, welche die segmentierte Zielfunktion minimieren. Diese erhält man durch Lösung der Gleichung

$$(\hat{i}_1, ...\hat{i}_m) = \underset{(i_1,...,i_m)}{\operatorname{argmin}} RSS(i_1, ..., i_m), \tag{3.25}$$

wobei die Größe der Segmente $i_j - i_{j-1} \geq n_h \geq k$ entweder direkt gewählt oder von einem h abgeleitet werden, sodass $n_h = \lfloor nh \rfloor$ die minimale Anzahl an Beobachtungen pro Segment ist. Dabei ist $\mathcal{I}_{(m,n)} = \{\hat{i}_1, ...\hat{i}_m\}$ die optimale Menge der Bruchpunkte aus Gleichung (3.25) und wird m-Partition genannt.

Die direkte Lösung von Gleichung (3.25) durch erschöpfende Suche über alle möglichen Partitionen ist rechnerisch sehr aufwendig, da die Zeitkomplexität von der Ordnung $O(n^m)$ ist. Bai and Perron [1998] verwenden das Optimalitätsprinzip von Bellman, sodass die Lösung der folgenden Rekursionsformel mit einer Zeitkomplexität von $O(n^2)$ genügt:

$$RSS(\mathcal{I}_{(m,n)}) = \min_{mn_h \leq i \leq n-n_h} [RSS(\mathcal{I}_{(m-1,i)}) + rss(i+1, n)]. \tag{3.26}$$

Dieser Algorithmus berechnet eine Dreiecksmatrix $rss(i, j)$ für alle $j - i \geq \lfloor nh \rfloor$ und $i = 1, ..., n - \lfloor nh \rfloor + 1$. Basierend auf dieser Matrix kann die Minimierung von Gleichung (3.25) durch Anwendung von Gleichung (3.26) für eine beliebige Anzahl an Bruchpunkten m gelöst werden.

Für $m = 0, 1, ...$ mögliche Bruchpunkte liegen $m+1$ segmentierte Modelle vor. Die Segmentierung ist also abhängig von der Anzahl an Bruchpunkten m. Man interessiert sich für

welches m eine optimale Segmentierung vorhanden ist. Dazu wählt man das segmentierte Modell, welches das BIC (Bayesianisches Informationskriterium) miniminiert [Zeileis et al. 2010, 2003; Zeileis 2003].

3.5. Monitoring

Monitoring beschäftigt sich mit der Frage, ob bei neu hinzukommenden Daten ein Strukturbruch vorliegt. Dazu werden die Parameter zunächst sequentiell aus allen verfügbaren (historische und hinzugekommene) Daten geschätzt und nur mit der Schätzung aus den historischen Daten verglichen. Die Nullhypothese, dass in den hinzugekommenen Daten aus dem Monitoring Zeitraum kein Strukturbruch vorhanden ist, wird abgelehnt, wenn beide Schätzungen zu stark voneinander abweichen.

In linearen Regressionsmodellen lautet die Nullhypothese

$$H_0 : \beta_i = \beta_0 \qquad (i > n) \tag{3.27}$$

und wird der Alternativhypothese, dass es einen Punkt in der Zukunft gibt, an dem sich der Vektor β_i ändert, gegenübergestellt. Das dazugehörige Modell aus Gleichung (3.14) kann für die hinzugekommenen Beobachtungen $i > n$ erweitert werden:

$$y_i = {x_i}^T \beta_i + u_i \qquad (i = 1, \dots, n, n+1, \dots). \tag{3.28}$$

Das Monitoring beginnt ab dem Zeitpunkt n, sodass die Daten zu den Zeitpunkten $1, \dots, n$ als historischer Datensatz bezeichnet werden können und die neu hinzugekommenen Daten ab dem Zeitpunkt $n + 1$, dem Monitoring Zeitraum, vorliegen.

Diese Abschlussarbeit beschränkt sich auf den OLS-CUSUM Prozess, der auch für das Monitoring verwendet werden kann. Weitere mögliche Prozesse für das Monitoring können in Hornik et al. [2005] nachgeschlagen werden. Die Regressionskoeffizienten werden nur einmal für den historischen Datensatz geschätzt und auf Basis dieser Schätzungen die Residuen der Beobachtungen in den neu hinzugekommenen Daten berechnet. Falls es einen Strukturbruch in den hinzugekommenen Daten gibt, weichen die Residuen systematisch von ihrem Mittelwert von Null ab [Zeileis et al. 2002; Zeileis 2005].

Die OLS Residuen des historischen Datensatzes (bis zur n-ten Beobachtung) sind gegeben durch

$$\hat{u}_i^{(n)} = y_i - x_i^\top \hat{\beta}^{(n)}, \tag{3.29}$$

sodass der OLS-CUSUM Prozess wie in Gleichung (3.18) definiert ist und die Teststatistik wie folgt geschrieben werden kann [Hornik et al. 2005]:

$$\sup_{t \geq 0} \left| \frac{1}{\hat{\sigma}\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} \hat{u}_i^{(n)} \right|. \tag{3.30}$$

Der einzige Unterschied ist, dass t nicht mehr nur bis 1 beschränkt ist, sondern auch für die hinzugekommenen Daten den Wert $T > 1$ annehmen kann, sodass $t \geq 0$ gilt. Nach Hornik et al. [2005] wird die Nullhypothese abgelehnt, wenn der Prozess im Monitoring Zeitraum $1 < t < T$ eine bestimmte Schranke

$$b_{\text{monitor}}(t) = \sqrt{t(t-1)\left[c^2 + \log \frac{t}{t-1}\right]} \tag{3.31}$$

überschreitet.

4. Anwendung

4.1. Datensatz

Die Daten für die statistischen Analysen in dieser Abschlussarbeit wurden im Zeitraum vom 09. November 2010 bis zum 09. Februar 2011 im Rahmen der Veranstaltung *Fortgeschrittene Programmierung mit R* im Wintersemester 2010/2011 erstellt und sind im `MUCflights` Paket [students of the 'Advanced R Programming Course' Basil Abou El-Komboz et al. 2011] frei verfügbar. In `MUCflights` ist eine Funktion `flights()` implementiert, mit der Fluginformationen für den oben genannten Zeitraum ausgegeben werden können:

```
> dataset <- flights(from = "2010-11-09", to = "2011-02-09")
> info <- dataset[, c("lsk", "stt", "lvg", "lde")]
```

Der Datensatz `dataset` ist von der Klasse `flights` und hat 18 Variablen, die Auskunft über die einzelnen Flüge am Franz-Josef-Strauss Flughafen geben. Für die Auswertungen in dieser Abschlussarbeit wird der Datensatz auf folgende 4 Variablen reduziert:

- `lsk`: Ausprägungen `S` (für gestartete Flüge) und `L` (für gelandete Flüge)
- `stt`: geplante Startzeit (für gestartete Flüge) bzw. Landezeit (für gelandete Flüge)
- `lvg`: Name der Fluggesellschaft
- `lde`: Zielland (für gestartete Flüge) bzw. Herkunftsland (für gelandete Flüge)

Tabelle 4.1 veranschaulicht die Form des reduzierten Datensatzes `info`. Ziel ist es diesen Datensatz in eine Zeitreihe umzuwandeln, in der die Anzahl der Flugbewegungen pro Tag ablesbar ist. Dazu wird die Variable `stt` ohne die Uhrzeit verwendet, nach den unterschiedlichen Tagen gruppiert und gezählt wie oft ein Tag vorkommt.

	lsk	stt	lvg	lde
1	L	2010-11-09 05:10:00	Singapore Airlines	Singapur
2	L	2010-11-09 05:15:00	Lufthansa	China
3	L	2010-11-09 05:40:00	Lufthansa	Saudi-Arabien

Tabelle 4.1.: Form des reduzierten Datensatzes

Dies wird mit der Funktion `dailyflights()` (siehe Listing B.1 in Anhang B.2) realisiert, sodass eine Zeitreihe `zootab` der Klasse `zooflights` mit dem täglichen Flugaufkommen ausgegeben wird:

```
> zootab <- dailyflights(info)
> tail(zootab)

2011-02-03 2011-02-04 2011-02-06 2011-02-07 2011-02-08 2011-02-09
      1073       1091        920       1090       1059       1074
```

Ein Objekt der Klasse `zooflights` erbt von der Klasse `zoo` [Zeileis and Grothendieck 2005] und stellt somit ein Zeitreihenobjekt eines Flugdatensatzes der Klasse `flights` dar. Das `zoo` Paket bietet die Möglichkeit sowohl mit regelmäßigen (äquidistanten) als auch unregelmäßigen Zeitreihen umzugehen und kann im Gegensatz zu Objekten der Klasse `ts` den Zeitindex direkt als Datumsformat `Date` anzeigen. Deshalb werden in dieser Abschlussarbeit für die Darstellung einer Zeitreihe Objekte der Klasse `zoo` verwendet. Das Zeitreihenobjekt `zootab` beinhaltet die zeitlich geordneten Beobachtungen zu allen Tagen, an denen Fluginformationen vorhandenen sind. Beispielsweise wurde im obigen Output der Zeitindex `2011-02-05` übersprungen, da im Datensatz `dataset` keine Informationen zu diesem Datum vorliegen.

Die fehlenden Daten liegen zu folgenden Tagen vor:

```
> zootabNA <- dailyflights(info, option = "na.include")
> zootabNA[is.na(zootabNA)]

2010-12-14 2011-01-14 2011-02-05
        NA         NA         NA
```

Während die Zeitreihe `zootab` an fehlenden Tagen den Zeitindex überspringt, nimmt die Zeitreihe `zootabNA` stattdessen den Wert `NA` an, wodurch die fehlenden Tage ermittelt werden können. Da für die Zeitreihenzerlegung eine äquidistante (bzw. regelmäßige) Zeitreihe (in der keine fehlenden Werte vorkommen) benötigt wird, werden die Daten zu den 3 fehlenden Tagen zum Beispiel durch lineare Interpolation berechnet und erhalten somit eine äquidistante Zeitreihe `zootabAP`:

```
> zootabAP <- dailyflights(info, option = "na.approx")
> zootabAP[is.na(zootabNA)]

2010-12-14 2011-01-14 2011-02-05
    1006.5      966.5     1005.5

> is.regular(zootabAP, strict = TRUE)

[1] TRUE
```

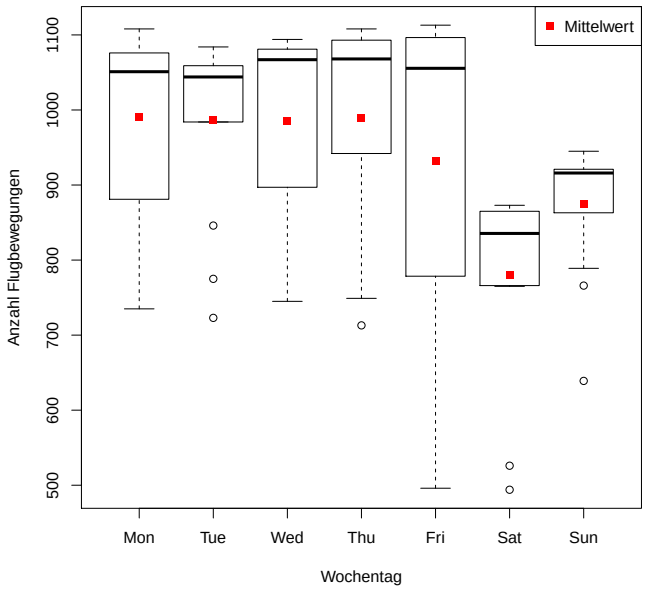

Abbildung 4.1.: Anzahl der Flugbewegungen pro Wochentag

4.2. Explorative Datenanalyse

4.2.1. Deskriptive Analyse

Für die deskriptive Analyse werden die Originaldaten der Zeitreihe `zootab` ohne interpolierte Werte verwendet. Die Verteilung der Anzahl an Flugbewegungen pro Tag ist nahezu symmetrisch, da sich Mittelwert und Median nur geringfügig unterscheiden (vgl. Tabelle 4.2).

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
494	847.8	968	936.6	1070.5	1113

Tabelle 4.2.: Summary von `zootab`

Abbildung 1.1 aus Kapitel 1 zeigt den Verlauf der Zeitreihe `zootab`. Die regelmäßig und in gleichen Abständen vorkommenden spitzen Ecken suggerieren einen immer wiederkehrenden Rhythmus in den Beobachtungen. Dieser Rhythmus scheint nach Abbildung 4.1 wöchentlich zu sein, da am Wochenende die durchschnittliche Anzahl an Flugbewegungen niedriger ist als unter der Woche.

Die ersten beiden Boxplots aus Abbildung 4.2 veranschaulichen die Anzahl an Flugbewegungen getrennt nach der Variablen `lsk` (gestarteten und gelandeten Flügen). In den

mittleren zwei Boxplots wird die Variable `lvg` nach Flügen mit der Airline Lufthansa und Flügen mit allen anderen Airlines getrennt. Die letzten beiden Boxplots gruppieren die Variable `lde` nach Flugbewegungen vom/ins Ausland und Flugbewegungen innerhalb Deutschland. Aus den Boxplots lassen sich folgende drei Aussagen ableiten:

1. Die Anzahlen zwischen gestarteten und gelandeten Flügen ähneln sich.
2. Täglich gibt es mehr Flüge mit der Airline Lufthansa als mit anderen Airlines.
3. Die Anzahl an Flugbewegungen aus dem Ausland (für gelandete Flüge) bzw. in das Ausland (für gestartete Flüge) ist höher als die Anzahl an Flugbewegungen innerhalb Deutschland.

Darauf aufbauend kann man in Abbildung 4.3 sehen, wie sich das Verhältnis (Odds) der Anzahl an Flugbewegungen zwischen

1. gestarteten und gelandeten Flügen
2. Flüge der Airline Lufthansa und den restlichen Airlines
3. nicht innerdeutschen und innerdeutschen Flügen

über die Zeit verhält. Durchschnittlich gibt es 1.8 mal mehr Flüge mit der Airline Lufthansa im Vergleich zu anderen Airlines und 2.7 mal mehr Flüge ins/aus dem Ausland als innerdeutsche Flüge. Das Verhältnis der Anzahl zwischen gestarteten und gelandeten Flügen bleibt fast durchgehend konstant.

Ab Mitte Dezember bis Mitte Januar (während der Weihnachtsferien) ist das Verhältnis der Anzahl an Flugbewegungen zwischen der Airline Lufthansa und den restlichen Airlines (grüne Linie) niedriger als sonst. Daher sind in diesem Zeitraum verhältnismäßig weniger Flüge mit der Airline Lufthansa vorhanden. Gleichzeitig sind verhältnismäßig weniger innerdeutsche Flüge als Flüge ins/aus dem Ausland (blaue Linie) in den Flugdaten vorhanden, welches im Zusammenhang mit dem Wintereinbruch und den dadurch gestrichenen Flügen – insbesondere innerhalb Deutschlands – stehen könnte dpa [2010b].

4.2.2. Zeitreihenzerlegung

Wegen des vermuteten wöchentlichen Rhythmus aus Abbildung 4.1 wird als Periodenlänge $d = 7$ vorausgesetzt. Um die Zeitreihe zu zerlegen, benötigt man m **vollständige** Perioden der Länge $d = 7$ einer äquidistanten Zeitreihe. Die Zeitreihe `zootab` enthält fehlende Daten und ist somit nicht äquidistant, daher wird statt der Zeitreihe `zootab` zunächst von der (nicht periodischen) äquidistanten Zeitreihe `zootabAP` ausgegangen.

Um die Zeitreihe in ihre Trend-, Saison- und Restkomponente zu zerlegen ist eine periodische Zeitreihe mit mehr als zwei Perioden nötig, sodass `zootabAP` mit Hilfe der Funktion

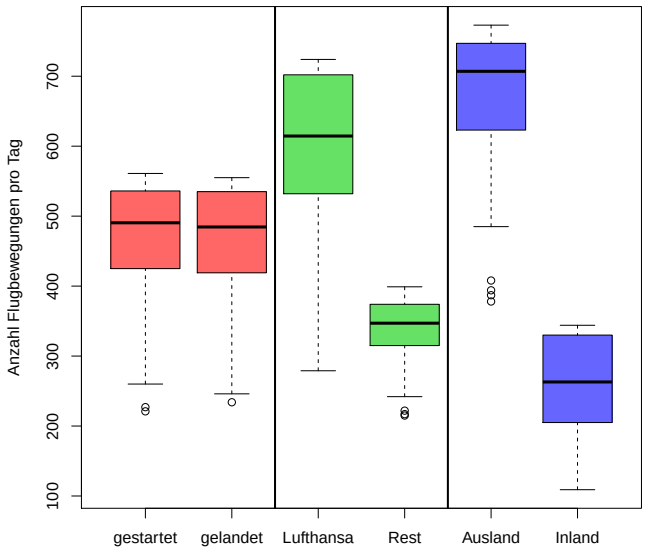

Abbildung 4.2.: Flugbewegungen getrennt nach: gestartet und gelandet (links), Lufthansa und Rest (mitte), Auslandsflüge und Inlandsflüge (rechts)

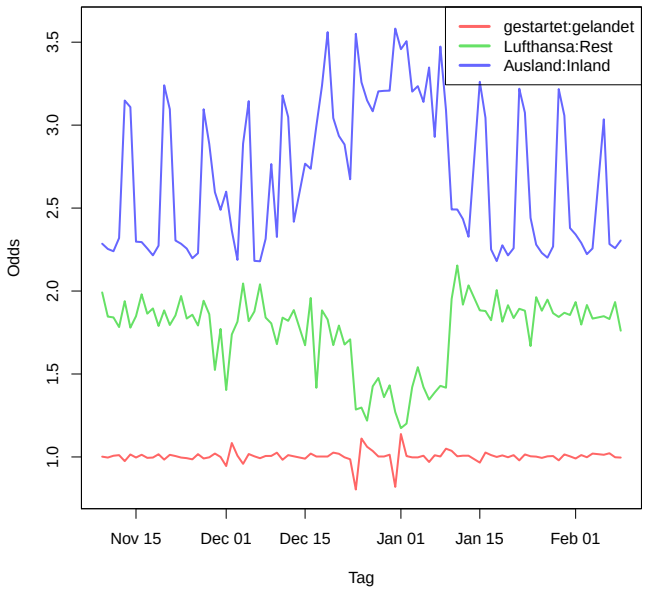

Abbildung 4.3.: Odds gestartet:gelandet (rot), Lufthansa:Rest (grün) und Ausland:Inland (blau)

`ts()` unter Angabe der Periodenlänge $d = 7$ in eine periodische Zeitreihe `periodic` transformiert wird:

```
> d <- 7
> m <- length(zootabAP)%/%d
> periodic <- ts(coredata(zootabAP), start=c(1,1), end=c(m,d), frequency=d)
> periodic

Time Series:
Start = c(1, 1)
End = c(13, 7)
Frequency = 7
 [1] 1071.0 1090.0 1108.0 1102.0  867.0  945.0 1108.0
 [8] 1061.0 1091.0 1106.0 1113.0  865.0  942.0 1107.0
[15] 1084.0 1094.0 1100.0 1075.0  856.0  921.0  881.0
[22]  984.0  745.0  942.0 1036.0  801.0  916.0 1050.0
[29] 1046.0 1011.0  979.0 1021.0  815.0  911.0 1056.0
[36] 1006.5  957.0  994.0  783.0  767.0  789.0  853.0
[43]  846.0  897.0  856.0  496.0  494.0  639.0  735.0
[50]  723.0  753.0  749.0  504.0  526.0  766.0  811.0
[57]  775.0  799.0  713.0  774.0  765.0  863.0 1051.0
[64] 1044.0 1068.0 1068.0  966.5  865.0  910.0 1076.0
[71] 1040.0 1081.0 1093.0 1098.0  865.0  925.0  998.0
[78] 1043.0 1069.0 1082.0 1095.0  873.0  921.0 1068.0
[85] 1059.0 1066.0 1073.0 1091.0 1005.5  920.0 1090.0
```

Jede Zeile von `periodic` entspricht einer Periode. Die einzelnen Werte entsprechen der Anzahl an Flugbewegungen pro Tag (vom 9. November 2010 bis zum 7. Februar 2011). Die Daten vom 8. und 9. Februar 2011 wurden in `periodic` weggelassen, damit insgesamt $m = 13$ vollständige Perioden vorliegen.

Da die Schätzung der Saisonkomponente in der Funktion `decompose()` aus dem `stats` Paket [R Development Core Team 2010] für ungerade Periodenlängen fehlerhaft ist (für den Beweis siehe Anhang A.2), wird basierend auf `decompose()` zur Zerlegung der Zeitreihe eine selbst korrigierte Funktion `decomp()` (siehe Listing B.2 Anhang B.2) verwendet.

In Abbildung 4.4 wurde die Zeitreihe additiv mit Hilfe der Funktion `decomp()` in Trendkomponente, Saisonkomponente und Restkomponente zerlegt. Auf der x-Achse sind die Perioden (in diesem Beispiel Wochen) aufgetragen und auf der y-Achse die Anzahl an Flugbewegungen. Zur Orientierung wird der Abbildung jeden Samstag eine senkrechte gestrichelte Linie hinzugefügt.

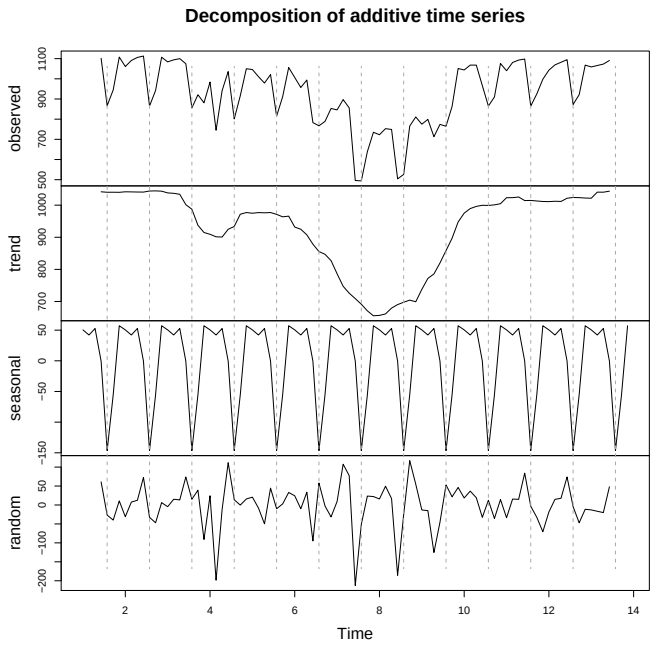

Abbildung 4.4.: Komponentenzerlegung

Die erste Grafik von Oben entspricht dem Verlauf der periodischen Zeitreihe `periodic`, die zweite stellt die mit dem gleitenden Durchschnitt geschätzte Trendkomponente dar (vgl. Kapitel 2.2). Der Trend verläuft anfänglich konstant, fällt dann in der dritten Woche, steigt kurzzeitig leicht an, fällt nochmals in der sechsten Woche und nimmt dann ab der achten Woche wieder zu, bis der anfängliche konstante Wert erreicht wird. Die unteren beiden Grafiken entsprechen zum einen der Saisonkomponente und der Restkomponente. Den kleinsten Wert nimmt die Saisonkomponente immer Samstags an, was den Verlauf des Mittelwerts aus Abbildung 4.1 bestätigt.

4.3. Analyse von Strukturbrüchen

Der Einfachheit wird im Folgenden von der äquidistanten Zeitreihe `zootabAP` mit den interpolierten Werten und einem Modell `y` $\sim$ `1` ohne Einflussgrößen ausgegangen. Bei Anwendung der Funktion `efp()` auf ein Objekt der Klasse `zoo` oder auf eine unregelmäßige Zeitreihe mit fehlenden Werten (wie zum Beispiel `zootab`) geht die Datumsinformation, die in der Spalte `$process` eines `efp` Objektes abgespeichert ist, verloren. Deshalb werden für die Strukturbruchanalyse in dieser Abschlussarbeit Objekte der Klasse `ts` verwendet. Mit der Funktion `tsflights()` (vgl. Listing B.3 in Anhang B.2) wird eine neue Klasse `tsflights` eingeführt, die auf ein Objekt der Klasse `zooflights` angewendet werden kann. Damit kennzeichnet man, dass es sich um eine Zeitreihe bestehend aus Flugdaten der Klasse `ts` und nicht mehr um Flugdaten der Klasse `zoo` handelt.

Es werden der OLS-CUSUM Prozess `ocus` und der generalisierte M-Fluktuationsprozess `gefp` definiert:

```
> y <- tsflights(zootabAP)
> ocus <- efp(y ~ 1, type="OLS-CUSUM")
> gefp <- gefp(y ~ 1, fit=lm)
```

Mit dem Argument `functional="max"` auf dem OLS-CUSUM Prozess `ocus`, wird für den Strukturbruchtest die Teststatistik aus Gleichung (3.20) hergenommen. Es ergibt sich folgender `R` Output:

```
> sctest(ocus, functional="max")

        OLS-based CUSUM test

data:  ocus
S0 = 1.7705, p-value = 0.003786
```

Durch die Wahl des Funktionals `maxBB` auf den generalisierten M-Fluktuationsprozess, wird die Double Maximum Teststatistik aus Gleichung (3.10) für den Strukturbruchtest verwendet:

```
> sctest(gefp, functional=maxBB)

        M-fluctuation test

data:  gefp
f(efp) = 1.7801, p-value = 0.003537
```

Bei beiden Tests ist der p-Wert kleiner als $\alpha = 0.05$, wodurch die Nullhypothese, dass kein Strukturbruch vorhanden ist abgelehnt werden kann. Abbildung 4.5 visualisiert das Testergebnis. Ein kritischer Wert $b(t) = c$ (rote Linie) wird bei beiden Tests mehrmals überschritten. Verwendet man beim OLS-CUSUM Test die alternative Schranke $b_{\text{alternative}}(t) = c \cdot \sqrt{t(1-t)}$, wird auch diese mehrmals überschritten (vgl. Abbildung 4.6). Dies kann ein Indiz dafür sein, dass in der vorliegenden Zeitreihe mehrere Bruchpunkte vorhanden sein könnten.

Die genaue Anzahl der Bruchpunkte ist noch unbekannt und kann mit der Funktion `breakpoints()` aus dem `strucchange` Paket [Zeileis et al. 2002] unter Anwendung der in Kapitel 3.4 vorgestellten Methode geschätzt werden:

```
> (breakp <- breakpoints(y~1))

         Optimal 4-segment partition:

Call:
breakpoints.formula(formula = y ~ 1)

Breakpoints at observation number:
18 45 61

Corresponding to breakdates:
14939 14966 14982
```

Der `R` Output von `breakp` besagt, dass das beste Modell (mit minimalen BIC) aus 4 Segmenten mit 3 Bruchpunkten besteht. Die Werte in `Corresponding to breakdates` entsprechen standardmäßig der Anzahl vergangener Tage vom 1. Januar 1970, sodass sich durch Anwendung der Funktion `as.Date()` auf die jeweiligen Werte das entsprechende Datum der Bruchpunkte herleiten lässt.

Das dazugehörige Datum der Bruchpunkte ist an den Tagen 2010-11-26, 2010-12-23, 2011-01-08 und wird in Abbildung 4.7 visualisiert. Die minimale Anzahl an Beobachtungen pro Segment beträgt standardmäßig maximal 15% von der Gesamtanzahl an Beobachtungen. Die Grafik suggeriert, dass sich im ersten und letzten Segment die Anzahl an Flugbewegungen um einen mittleren konstanten Wert streut, der über 1000 liegt. Im zweiten und dritten Segment ist dieser mittlere Wert deutlich geringer.

Es können auch andere optimale Segmentierungen berechnet werden. Diese Modelle haben dann aber nicht mehr den minimalen BIC. Abbildung 4.8 zeigt sowohl RSS als auch BIC für eine unterschiedliche Anzahl an Bruchpunkten. Obwohl die Residuenquadratsumme für $m = 5$ Bruchpunkte minimal ist, ist nach dem BIC das Modell mit $m = 3$ Bruchpunkten das beste.

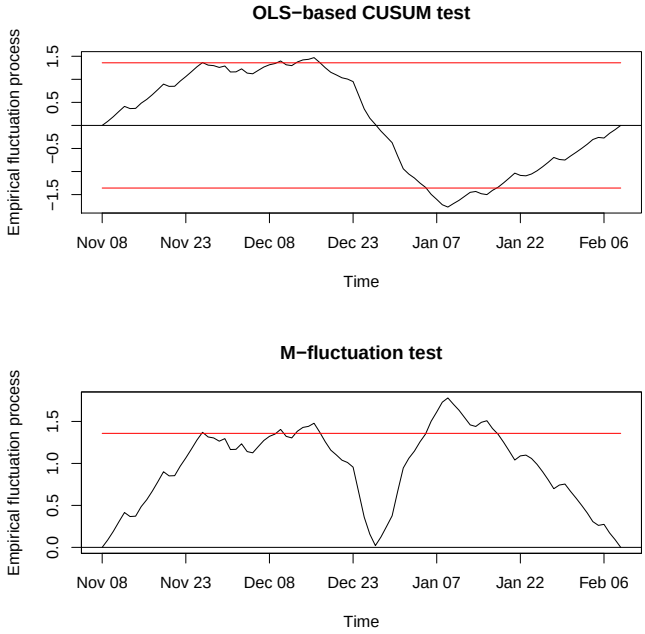

Abbildung 4.5.: OLS-CUSUM Prozess und generalisierter M-Fluktuationsprozess

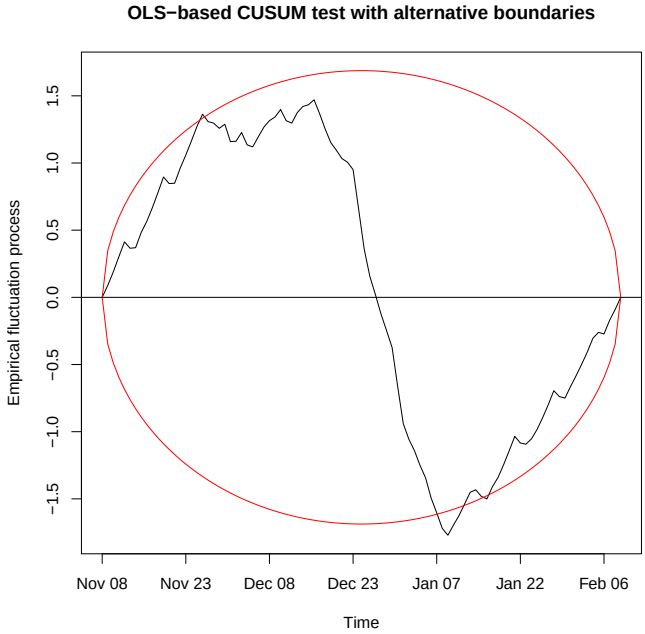

Abbildung 4.6.: OLS-CUSUM Prozess mit alternativer Schranke $b_{\text{alternative}}(t)$

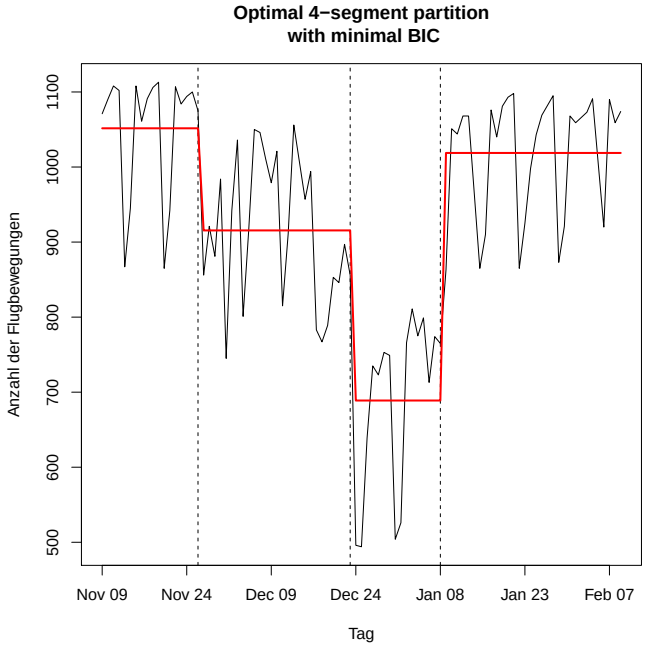

Abbildung 4.7.: optimales segmentiertes Modell mit 3 Bruchpunkten

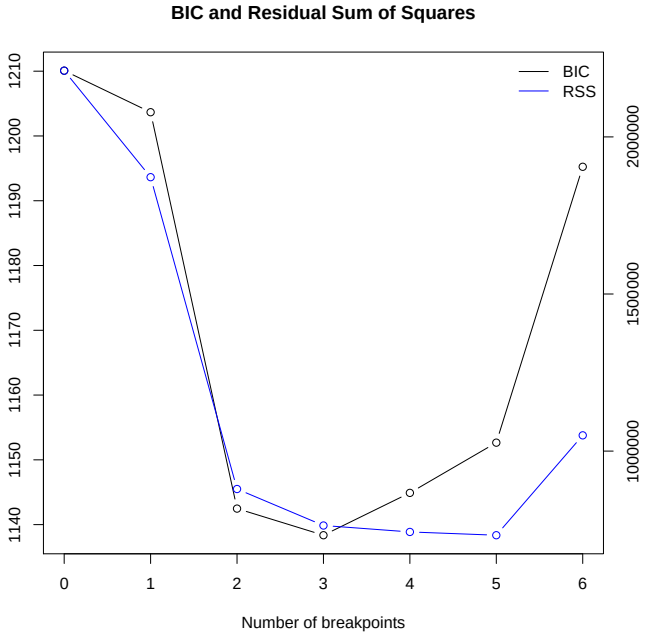

Abbildung 4.8.: BIC und RSS in Abhängigkeit von der Anzahl an Bruchpunkten

4.4. Monitoring

Es werden die ersten drei Wochen als historischer Datensatz `historic` verwendet. Mit der Funktion `mefp()` wird für diesen Zeitraum ein stabiles Regressionsmodell gefittet. Da das Monitoring ein sequentielles Verfahren ist, muss das Signifikanzniveau `alpha` bereits im Vorfeld definiert werden. Der folgende Output enthält Informationen über die Größe des historischen Datensatzes und über die dazugehörigen geschätzten Parameter [Zeileis 2003].

```
> historic <- window(y, start=min(index(y)), end= min(index(y)) + 20)
> (ocus.mefp <- mefp(historic~1, type="OLS-CUSUM", alpha=0.05))

Monitoring with OLS-based CUSUM test

Initial call:
  mefp.formula(formula = historic ~ 1, type = "OLS-CUSUM", alpha = 0.05)

Last call:
  mefp.formula(formula = historic ~ 1, type = "OLS-CUSUM", alpha = 0.05)

Significance level   :  0.05
Critical value       :  2.795483
History size         :  21
Last point evaluated :  21

Parameter estimate on history :
(Intercept)
   1027.952
```

Der kritische Wert beträgt $c = 2.795483$, sodass mit der Formel aus Gleichung (3.31) die Schranke zur Ablehnung der Nullhypothese berechnet werden kann.

Falls beispielsweise für die nächsten zwei Wochen neue Daten hinzugekommen sind, wird `historic` um diese zwei Wochen erweitert. Mit Hilfe der Funktion `monitor()` auf ein Objekt der Klasse `mefp` wird das Objekt `ocus.mefp` automatisch um die neu hinzugekommenen Beobachtungen aktualisiert und ein sequentieller Test auf eine strukturelle Änderung für jede neue Beobachtung ausgeführt:

```
> historic <- window(y, start=min(index(y)), end= min(index(y)) + 34)
> ocus.mefp <- monitor(ocus.mefp)
```

Es wurde keine Meldung für ein Strukturbruch ausgegeben. In Abbildung 4.9 wird die Schranke (rote Linie) nicht überschritten. Dies bedeutet, dass in den hinzugekommenen zwei Wochen kein Strukturbruch vorhanden ist. Die Abgrenzung zwischen historischem Datensatz und neu hinzugekommenen Daten ist als gestrichelte Linie gekennzeichnet.

Nun wird `historic` um alle vorliegenden Beobachtungen bis einschließlich dem 9. Februar 2011 erweitert und das Monitoring erneut ausgeführt:

```
> historic <- window(y, start=min(index(y)), end= max(index(y)))
> ocus.mefp <- monitor(ocus.mefp)

Break detected at observation # 41
```

Diesmal wurde ein Strukturbruch bei Beobachtung 41 gefunden. Abbildung 4.10 zeigt den Monitoringprozess mit deren Schranke (rote Linie). Die Beobachtung, die erstmals die rote Linie überschreitet, wird als Bruchpunkt bezeichnet und ist im Objekt `ocus.mefp` abgespeichert. Der Output sieht folgendermaßen aus:

```
> ocus.mefp

Monitoring with OLS-based CUSUM test

Initial call:
  mefp.formula(formula = historic ~ 1, type = "OLS-CUSUM", alpha = 0.05)

Last call:
  monitor(obj = ocus.mefp)

Significance level   :  0.05
Critical value       :  2.795483
History size         :  21
Last point evaluated :  93
Structural break at  :  41

Parameter estimate on history :
(Intercept)
   1027.952
```

Hinzugekommen ist im Output eine Zeile `Structural break at`, in der sich die Nummer der Beobachtung befindet, an dem erstmals die Schranke mit dem kritischen Wert überschritten wird. Dies bedeutet, dass ab dieser Beobachtung der geschätzte Parameter des historischen Datensatzes (mit dem Wert 1027.952) auf einem Signifikanzniveau von 5% keine gute Anpassung für das Modell ohne Einflussgrößen mehr ist.

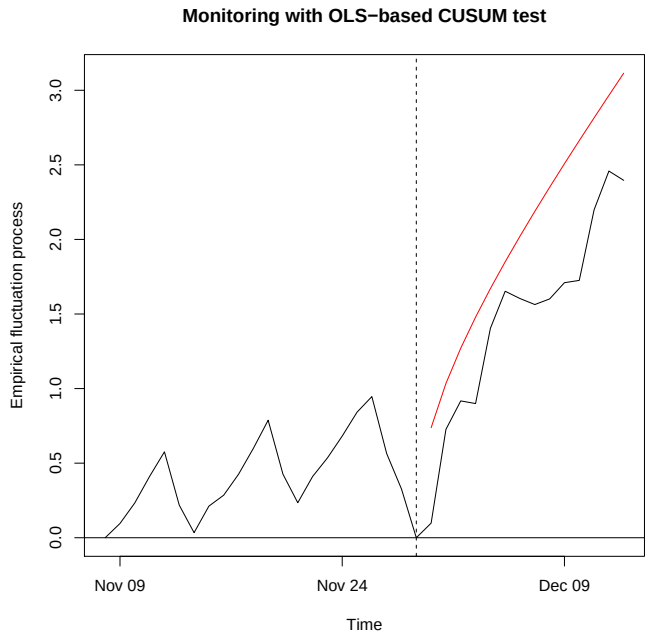

Abbildung 4.9.: Monitoring für folgende 14 Tage bezüglich des historischen Datensatzes

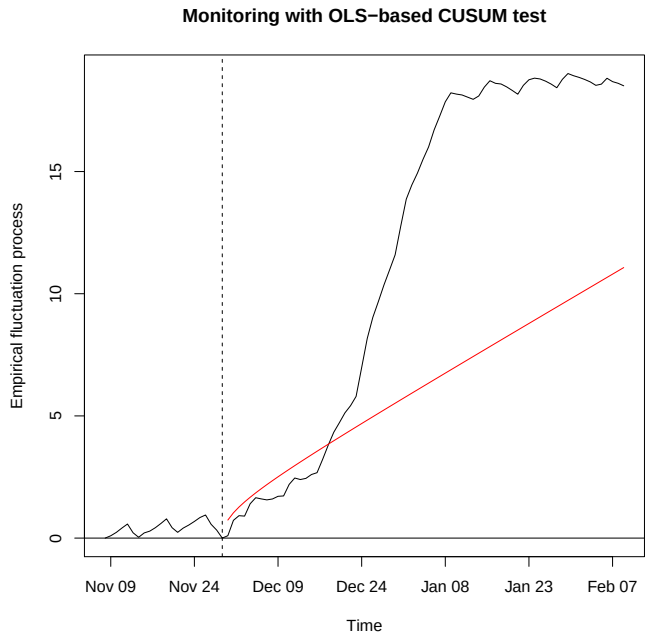

Abbildung 4.10.: Monitoring aller vorhandenen Daten bezüglich des historischen Datensatzes

5. Zusammenfassung und Ausblick

In dieser Abschlussarbeit wurde ein reines Interceptmodell für die Auswertungen der Anzahl an Flugbewegungen im Zeitraum vom 9. November 2010 bis zum 9. Februar 2011 verwendet. Dabei wurde ein Modell mit drei Strukturbrüchen gefunden, für das das BIC minimal wird. Der erste Strukturbruch fällt in den Zeitraum, an dem zahlreiche Pressemeldungen den Wintereinbruch angekündigt hatten, was das Ausfallen etlicher Flüge bewirkte [dpa 2010a]. Der zweite Strukturbruch kurz vor Weihnachten (am 23. Dezember) könnte dadurch erklärt werden, dass „neue starke Schneefälle in vielen Teilen Europas zu Hunderten von Flugausfällen“ geführt hatten [FAZ.NET 2010]. Hinzu kommt noch, dass die folgenden 3 Tagen nach dem zweiten Strukturbruch (Heiligabend, 1. Weihnachtstag und 2. Weihnachtstag) Feiertage sind und über das Wochenende verteilt sind. An diesen finden, wie in Abbildung 4.1 zu sehen, weniger Flugbewegungen statt. Der letzte Strukturbruch, an dem die Anzahl an Flugbewegungen wieder anstieg, wurde am letzten Wochenende der Weihnachtsferien lokalisiert.

Für die Strukturbruchanalyse wären auch kompliziertere Modelle, die neben dem Intercept auch weitere Einflussgrößen enthalten, denkbar. Dabei kann wieder der generalisierte M-Fluktuationsprozess mit Hilfe einer Teststatistik (z.B. der Double Maximum Teststatistik) als exploratives Werkzeug zum Erkennen von eventuell vorhandenen Strukturbrüchen verwendet werden. Falls der Test die Nullhypothese, dass kein Strukturbruch vorhanden ist, ablehnt, können die Strukturbrüche mit der in Kapitel 3.4 beschrieben Methode geschätzt werden.

Zur Zeit werden seit dem 24. Mai 2011 unter der Aufsicht von Manuel Eugster weitere Flugdaten gesammelt. Die Auswertungen in dieser Abschlussarbeit und die dabei verwendeten selbstgeschriebenen Funktionen können auf diese neuen Flugdaten ebenfalls angewendet werden, sodass diese Abschlussarbeit auch als Demonstration zur Analyse von Strukturbrüchen eines Flugdatensatzes gesehen werden kann. Dazu wird unter Absprache mit den Autoren von `MUCflights` das Paket um weitere hilfreiche Funktionen zur Strukturbruchanalyse erweitert. Darüber hinaus wurde eine Funktion geschrieben, die es einem Benutzer beim täglichen Ausführen der Funktion ermöglicht eigene Flugdaten zu sammeln (siehe Listing B.4 in Anhang B.2).

Literaturverzeichnis

Bai, J. and Perron, P. (1998). Estimating and testing linear models with multiple structural changes. *Econometrica*, pages 47–78. 3.4, 3.4.2

dpa (2010a). Schneechaos auf Autobahnen und Flughäfen. *Frankfurter Allgemeine*. Available from: http://www.faz.net/artikel/S30176/wetter-schneechaos-auf-autobahnen-und-flughaefen-30319568.html [Online ; accessed 26. August 2011]. 5

dpa, afp, rtr (2010b). Ramsauer für Nachtflüge vor Weihnachten. *Frankfurter Rundschau*. Available from: http://www.fr-online.de/rhein-main/ramsauer-fuer-nachtfluege-vor-weihnachten/-/1472796/5028998/-/index.html [Online ; accessed 26. August 2011]. 4.2.1

Dr. Reingard Schöttl, Helene Hergt, Judith Hofstetter (2011). Zahlen und Fakten. Available from: www.munich-airport.de/media/download/general/publikationen/de/zahlen_und_fakten.pdf. 1

FAZ.NET (2010). Schneemassen legen Verkehr in Europa lahm. *Frankfurter Allgemeine*. Available from: http://www.faz.net/artikel/S30176/winterchaos-schneemassen-legen-verkehr-in-europa-lahm-30322251.html [Online ; accessed 26. August 2011]. 5

Kurt Hornik and Friedrich Leisch and Christian Kleiber and Achim Zeileis (2005). Monitoring structural change in dynamic econometric models. *Journal of Applied Econometrics*, 20(1):99–121. Available from: http://ideas.repec.org/a/jae/japmet/v20y2005i1p99-121.html. 3.5, 3.5, 3.5

Kleiber, C. and Zeileis, A. (2008). *Applied econometrics with R*. Springer Verlag. 3.4, 3.4.1, 3.4.2

Kreiß, J.P. and Neuhaus, G. (2006). *Einführung in die Zeitreihenanalyse*. Springer Verlag. 2, 2.2, 2.2.1, 2.2.2

Ploberger, W. and Krämer, W. (1992). The CUSUM test with OLS residuals. *Econometrica: Journal of the Econometric Society*, pages 271–285. 3.3.2

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. Available from: http://www.R-project.org. 3.3.2, 4.2.2

The students of the 'Advanced R Programming Course' Basil Abou El-Komboz and Andreas Bender and Abdelilah El Hadad and Laura Goeres and Roman Hornung and Max Hughes-Brandl and Christian Lindenlaub and Christina Riedel and Ariane Straub and Florian Wickler under the supervision of Manuel Eugster and Torsten Hothorn (2011). *MUCflights: Munich Franz-Josef-Strauss Airport Pattern Analysis*. R package version 0.0-3. Available from: http://CRAN.R-project.org/package=MUCflights. 4.1

Wenzel, A. and Hofmann, B. and Schufft, W. and Rückert, M.S.N. (2010). Komponentenzerlegung des Regelleistungsbedarfs mit Methoden der Zeitreihenanalyse. 2, 2.1

Winkler, P. (2006). Empirische Wirtschaftsforschung und Ökonometrie, 2. *Auflage, Heidelberg*. 2.2.1

Zeileis, A. (2003). Testing for Structural Change - Theory, Implementation and Applications. 3.2.1, 3.3, 3.4.2, 4.4

Zeileis, A. (2005). A unified approach to structural change tests based on ML scores, F statistics, and OLS residuals. *Econometric Reviews*, 24(4):445–466. 3.2, 3.2, 3.2.1, 3.2.2.2, 3.3.1, 3.3.2, 3.5, A.1

Zeileis, A. (2006). Implementing a class of structural change tests: An econometric computing approach. *Computational Statistics & Data Analysis*, 50(11):2987–3008. 3.2, 3.2.1, 3.2.1, 3.2.2.2

Achim Zeileis and Gabor Grothendieck (2005). zoo: S3 Infrastructure for Regular and Irregular Time Series. *Journal of Statistical Software*, 14(6):1–27. Available from: http://www.jstatsoft.org/v14/i06/. 4.1

Zeileis, A. and Hornik, K. (2007). Generalized M-fluctuation tests for parameter instability. *Statistica Neerlandica*, 61(4):488–508. 3.1, 3.2, 3.2.1, 3.2.1, 3.2.2.1, 3.2.2.2, 3.2.2.2, 3.3

Zeileis, A. and Kleiber, C. and Krämer, W. and Hornik, K. (2003). Testing and dating of structural changes in practice. *Computational Statistics & Data Analysis*, 44(1-2):109–123. 3.4.1, 3.4.2

Zeileis, A. and Leisch, F. and Hornik, K. and Kleiber, C. (2002). strucchange: An R Package for Testing for Structural Change in Linear Regression Models. *Journal of Statistical Software*, 7(2):1–38. Available from: http://www.jstatsoft.org/v07/i02/. 3.2.2.2, 3.3.1, 3.3.2, 3.3.2, 3.5, 4.3

Zeileis, A. and Shah, A. and Patnaik, I. (2010). Testing, monitoring, and dating structural changes in exchange rate regimes. *Computational Statistics & Data Analysis*, 54(6):1696–1706. 3.2.1, 3.4.2

A. Beweise in R

A.1. OLS-CUSUM Prozess

Der auf OLS-Residuen basierte OLS-CUSUM Prozess aus Gleichung (3.18) wird mit der Funktion `efp()` generiert und als Objekt `ocus` abgespeichert:

```
> y <- tsflights(zootabAP)
> ocus <- efp(y ~ 1, type = "OLS-CUSUM")
```

Mit der Funktion `gefp()` wird der generalisierte M-Fluktuationsprozess berechnet. Für die Kovarianzmatrix wird die adjustierte Kovarianzmatrix aus Gleichung (3.22) verwendet. Diese kann mit der Funktion `olsvar()` berechnet werden und wird der Funktion `gefp()` mit dem Argument `vcov` übergeben:

```
> olsvar <- function(obj, ...) sandwich(obj, adjust = TRUE, ...)
> gefp <- gefp(y ~ 1, fit = lm, vcov = olsvar)
```

Der generalisierte M-Fluktuationsprozess wird als `gefp_t` und die Wurzel der Kovarianzmatrix $\hat{J}$ als `J12` abgespeichert:

```
> gefp_t <- gefp$process
> J12 <- gefp$J12
```

Die Standardabweichung $\hat{\sigma}$ aus der vorletzten Zeile von Formel (3.21) entspricht nach Zeileis [2005] dem ersten Diagonalelement der Kovarianzmatrix $\hat{J}$, welches im Folgenden als `sigma` bezeichnet wird:

```
> J <- crossprod(J12)
> sigma <- (diag(J)[1])^(1/2)
```

Der manuell berechnete OLS-CUSUM Prozess in Abhängigkeit vom generalisierten M-Fluktuationsprozess `gefp_t` wird als `ocusgen` abgespeichert und kann durch die letzte Zeile aus Formel (3.21) ermittelt werden:

```
> ocusgen <- 1/sigma * (t(J12 %*% t(gefp_t))[, 1])
```

Abschließend wird der OLS-CUSUM Prozess `ocus` mit dem manuell berechneten OLS-CUSUM Prozess `ocusgen`, der auf einen generalisierten M-Fluktuationsprozess basiert, verglichen. Mit Hilfe der Funktion `all.equal()` wird gezeigt, dass beide Prozesse identisch sind:

```
> all.equal(as.vector(ocusgen), as.vector(ocus$process))

[1] TRUE
```

A.2. Fehler in `decompose()`

Im Folgenden wird von einer ungeraden Periodenlänge z.B. $d = 7$ (für Wochentage Montag bis Sonntag) und von $m = 3$ Perioden (für erste bis dritte Woche) ausgegangen. Die Trendkomponente hat für die ersten und letzten $q = (d-1)/2 = 3$ Beobachtungen den Wert `NA` (vgl. Kapitel 2.2.1).

Zunächst wird eine trendbereinigte Zeitreihe mit der Periodenlänge $d = 7$ konstruiert. Danach kann gezeigt werden, dass die Funktion `decompose()` die falschen trendbereinigten Werte für die Berechnung des saisonalen Anteils $\tilde{s}$ aus Gleichung (2.5) verwendet:

```
> d <- 7
> periods <- 3
> q <- (d-1)/2
> x <- rep(seq(d), periods)
> x <- x + rep(seq(periods) * 10, each = d)
> x <- ts(x, start(1, 1), frequency = d)
> trend <- c(rep(NA, q), rep(0, length(x) - 2*q), rep(NA, q))
> trend <- ts(trend, start(1, 1), frequency = d)
> (season <- x - trend)

Time Series:
Start = c(1, 1)
End = c(3, 7)
Frequency = 7
 [1] NA NA NA 14 15 16 17 21 22 23 24 25 26 27 31 32 33 34 NA NA NA
```

Die trendbereinigte Zeitreihe `season` wurde so konzipiert, dass die erste Ziffer für die i-te Woche steht (mit $i = 1, 2, 3$) und die zweite Ziffer für den j-ten Tag einer Woche (mit $j = 1, ..., 7$). Dabei haben die ersten und letzten $q = 3$ Beobachtungen den Wert `NA`.

Der folgende `na.omit()` Aufruf und der verwendete `index` verursachen den Fehler in der Funktion `decompose()`:

```
> season <- na.omit(c(as.numeric(window(season, start(x) + c(1, 0), end(x))),
+           as.numeric(window(season, start(x), start(x) + c(0, d)))))
> index <- c(0, cumsum(rep(d, periods - 2)))
```

Die folgende Matrix zeigt in jeder Spalte an, welche trendbereinigten Werte für die Mittelwertsberechnung des saisonalen Anteils $\tilde{s}$ des j-ten Tages herangezogen werden:

```
> figure <- list()
> figure <- sapply(1L:d, function(i) figure[[i]] <- season[index + i])
> figure
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   21   22   23   24   25   26   27
[2,]   31   32   33   34   14   15   16
```

Man erkennt, dass beispielsweise für den 5. Wochentag (`figure[,5]`) die Werte 25 und 14, also der 4. Tag der 1. Woche und der 5. Tag der 2. Woche, zur Mittelwertsberechnung verwendet werden, richtig wären aber die Werte 25 und 15.

Der Fehler kann behoben werden, indem man folgende Zeilen aus der Funktion `decompose()` entfernt:

```
> season <- na.omit(c(as.numeric(window(season, start(x) + c(1, 0), end(x))),
+           as.numeric(window(season, start(x), start(x) + c(0, f)))))
> index <- c(0, cumsum(rep(f, periods - 2)))
> for (i in 1L:f) figure[i] <- mean(season[index + i])
```

Und mit folgenden Zeilen ersetzt (vgl. Listing B.2 in Anhang B.2):

```
> index <- seq(1, length(x), by=f) - 1
> for (i in 1L:f) figure[i] <- mean(season[index + i], na.rm=TRUE)
```

B. R-Code

B.1. Technische Details

Der folgende Output beschreibt die verwendete Version von R mit weiteren hilfreichen Informationen und gibt eine Übersicht über die verwendeten Pakete mit ihrer Versionsnummer.

```
> sessionInfo()

R version 2.12.2 (2011-02-25)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=German_Germany.1252     LC_CTYPE=German_Germany.1252
[3] LC_MONETARY=German_Germany.1252    LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] tseries_0.10-25   quadprog_1.5-4    gtools_2.6.2      xtable_1.5-6
 [5] strucchange_1.4-4 sandwich_2.2-6    zoo_1.6-5         MUCflights_0.0-3
 [9] NightDay_1.0.1    maps_2.1-5        RSQLite_0.9-4     DBI_0.2-5
[13] sp_0.9-79         geosphere_1.2-19  XML_3.2-0.2

loaded via a namespace (and not attached):
[1] grid_2.12.2     lattice_0.19-17
```

B.2. Erwähnte Funktionen

Im Folgenden werden die im Text dieser Bachelorarbeit vorkommenden selbstgeschriebenen Funktionen aufgelistet.

Listing B.1: Funktionen zum transformieren eines `flights` Datensatzes in eine Zeitreihe

```
dailyflights <- function(input, subset, ...){
  if(!inherits(input, "flights")) stop("kein 'flights' Objekt übergeben")
  if(!missing(subset)) input <- subset(input, subset)
  output <- countdaily(input$stt, ...)
  class(output) <- c("zooflights", class(output))
  output
}

countdaily <- function(input, option=c("na.remove", "na.approx", "na.include")){
  require(zoo)
  option <- match.arg(option)
  days <- as.Date(input)
  tab <- tabulate(as.numeric(days-(min(days)-1)))
  tab[which(tab==0)] <- NA
  output <- zoo(as.numeric(tab), seq(min(days), max(days), "days"))
  return(switch(option, "na.approx"=na.approx(output),
    "na.remove"=na.omit(output), "na.include"=output))
}
```

Listing B.2: Fehlerfreie Funktion zur Zeitreihenzerlegung

```
decomp <- function(x, type=c("additive", "multiplicative"), filter=NULL) {
  type <- match.arg(type)
  l <- length(x)
  f <- frequency(x)
  if (f <= 1 || length(na.omit(x)) < 2 * f)
      stop("time series has no or less than 2 periods")
  if (is.null(filter))
      filter <- if (!f%%2)
          c(0.5, rep(1, f - 1), 0.5)/f
      else rep(1, f)/f
  trend <- filter(x, filter)
  season <- if (type == "additive")
      x - trend
  else x/trend
  periods <- l%/%f
  index <- seq(1, length(x), by=f)-1
  figure <- numeric(f)
  for (i in 1L:f) figure[i] <- mean(season[index + i], na.rm=TRUE)
```

```
19   figure <- if (type == "additive")
20       figure - mean(figure)
21   else figure/mean(figure)
22   seasonal <- ts(rep(figure, periods + 1)[1:l], start = start(x),
23       frequency = f)
24   structure(list(x = x, seasonal = seasonal, trend = trend,
25       random = if (type == "additive") x - seasonal - trend
26                else x/seasonal/trend,
27       figure = figure, type = type), class = "decomposed.ts")
28 }
```

Listing B.3: Funktion zum transformieren eines Objekts `zooflights` in `tsflights`

```
1  tsflights <- function(zooflights, na.remove=FALSE){
2    require(tseries)
3    if(!inherits(zooflights, "zooflights"))
4      stop("Objekt hat nicht die Klasse 'zooflights'")
5    output <- as.ts(zooflights)
6    class(output) <- c("tsflights", class(output))
7    if(na.remove) na.remove(output) else output
8  }
```

Listing B.4: Funktionen zur selbstständigen Sammlung von Flugdaten

```
1  update.flights <- function(file, ...){
2    if(file.exists(file)){
3      daten <- read.flights(file)
4      neu <- getFlights(...)
5      datum <- as.Date(neu$stt)
6      if(!any(as.Date(daten$stt) %in% datum)){
7        updated <- rbind(daten, neu)
8        write.flights(updated, file)
9      } else stop(paste("Neue Daten enthalten gleiches Datum wie in '", file, "'",
           sep=""))
10   } else stop(paste("Datei '", file, "' nicht vorhanden!", sep=""))
11 }
12
13 write.flights <- function(obj, file="flugdaten.csv", ...){
14   if(!inherits(obj, "flights")) stop("Kein 'flights' Objekt übergeben!")
15     obj$stt <- as.character(obj$stt)
16     obj$ett <- as.character(obj$ett)
17     write.table(obj, file = file, row.names = FALSE, ...)
18 }
19
20 read.flights <- function(file="flugdaten.csv", ...){
21   ret <- read.table(file, header=TRUE, colClasses="character", ...)
```

```
22   ret$stt <- as.POSIXct(strftime(ret$stt, format = "%Y-%m-%d %H:%M:%S.0"))
23   ret$ett <- as.POSIXct(strftime(ret$ett, format = "%Y-%m-%d %H:%M:%S.0"))
24   class(ret) <- c("flights", class(ret))
25   ret
26 }
```

C. Inhalt der CD-ROM

Der Inhalt der CD-ROM umfasst folgende Dateien und Ordner:

- Im Ordner `Grafiken\` befinden sich alle in dieser Bachelorarbeit vorkommenden Grafiken im PDF-Format.
- Im Ordner `R-Code\` befinden sich folgende Dateien:
  - `BA_Casalicchio.RData`: Beinhaltet alle `R` Objekte für die Erstellung der Outputs und Grafiken in dieser Bachelorarbeit.
  - `flugdaten.csv`: Beinhaltet einen mit der Funktion `update.flights()` selbst gesammelten Datensatz mit Fluginformationen vom 16. Juli 2011 bis zum 29. August 2011.
  - `functions.R`: Beinhaltet alle selbstgeschriebenen Funktionen (mit Kommentaren), die sowohl für die Auswertungen in Kapitel 4 benötigt wurden, als auch zur eventuellen Erweiterung des `MUCflights` Paketes gedacht sind.
  - `R-Code.R`: Enthält einen kommentierten R-Code, mit dem alle Grafiken und Outputs dieser Bachelorarbeit reproduziert werden können.
  - `updateflightsDEMO.R`: Enthält einen kommentierten R-Code, der die Verwendung der Funktion `update.flights()` erklärt.
- Die Bachelorarbeit in elektronischer Form `BA_Casalicchio.pdf`.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30. August 2011

. . . . . .

(Unterschrift des Kandidaten)