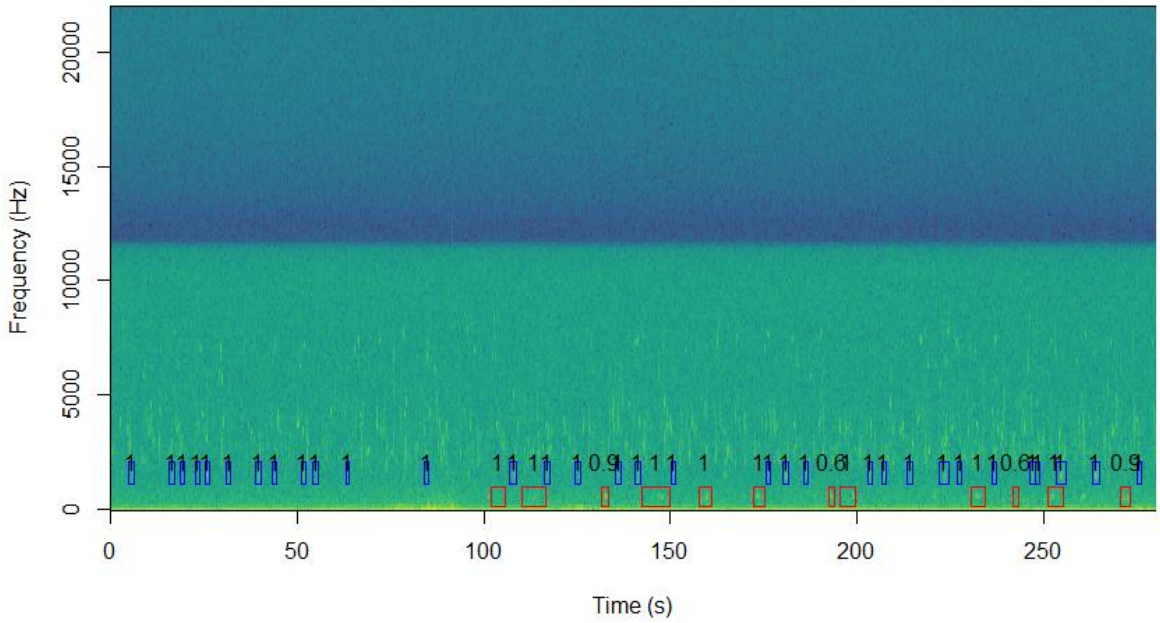
ZÜRCHER HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN
DEPARTEMENT LIFE SCIENCES UND FACILITY MANAGEMENT
INSTITUT UMWELT UND NATÜRLICHE RESSOURCEN

Automatisierte Erkennung der Balzaktivität von Birkhähnen (*Tetrao tetrix*) in R anhand bioakustischer Aufnahmen



Bachelorarbeit FS 2019

von

Burkhalter Felix

Bachelorstudiengang 2016

Studiengang Umweltingenieurwesen, Naturmanagement

Abgabedatum: 08.08.2019

Korrektor:
Dr. Stefan Suter
Forschungsgruppe
Wildtiermanagement
ZHAW Wädenswil

Co-Korrektorin:
Annette Stephani
Forschungsgruppe
Wildtiermanagement
ZHAW Wädenswil

IMPRESSUM

Titelblatt: Spektrogramm mit Suchergebnissen der automatisierten Erkennung. Bild: Burkhalter, 2019

Keywords: Seewave, gibbonR, R, machine learning, Birkhuhn, *Tetrao tetrix*, Balzaktivität, Deep learning, neuronale Netzwerke

Zitiervorschlag: Burkhalter, F. (2019), Automatisierte Erkennung der Balzaktivität von Birkhähnen (*Tetrao tetrix*) in R anhand bioakustischer Aufnahmen, Bachelorarbeit, Bachelorstudiengang Umweltingenieurwesen, ZHAW Wädenswil 2019

Zürcher Hochschule für Angewandte Wissenschaften Wädenswil, Institut Umwelt und Natürliche Ressourcen, Forschungsgruppe Wildtiermanagement WILMA
Grüentalstrasse 18, 8820 Wädenswil

DANKSAGUNG

An dieser Stelle möchte ich mich bei allen externen Personen bedanken, die mich im Laufe der Arbeit mit Antworten und Wissen unterstützt haben. Den (Mit-)Gründern der R-Pakete seewave, monitoR und gibbonR Jérôme Sueur, Josh Hafner und Dena Clink, die Zeit fanden, auf meine Fragen bezüglich ihrer R-Pakete einzugehen und zu beantworten, ist grosser Dank geschuldet.

Einen ganz besonderen Dank möchte ich an meine R-Mentorin Tamara Mainetti richten. Ihre lösungsorientierte Hilfe in R verkürzte mir erheblich den Weg durch die Untiefen der R-Hilfestellungen im Internet.

ZUSAMMENFASSUNG

Die Bestände des Birkhuhns (*Tetrao tetrix*) sind aufgrund diverser Faktoren, wie Lebensraumverlust oder Störung durch den Menschen, in der Schweiz rückläufig. Um die Bestände zu beobachten, werden die Birkhähne während der Balzzeit passiv akustisch überwacht. Diese Form der Überwachung generiert Akustikaufnahmen in grossen Datenmengen. An der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) müssen diese zurzeit manuell an einem Spektrogramm ausgewertet werden, damit festgestellt werden kann, wann ein Birkhahn die beiden Balzgeräusche, Kullern und Zischen, von sich gibt. Diesen finanziell und zeitlich grossen Auswertungsaufwand gilt es mittels Automatisierung zu optimieren. In dieser Arbeit wurden zwei R-Pakete, die sich für eine automatisierte Erkennung eignen, getestet und die Ergebnisse verglichen. Das erste Paket seewave, welches die automatische Erkennung mittels einer Cross-Korrelation zwischen dem Spektrogramm von vier Vorlagen und dem Spektrogramm der Akustikaufnahme ausführt, konnte 25% der gesuchten Balzgeräusche entdecken und markieren. Die Trefferquote konnte auf 77% erhöht werden, nachdem die Spektrogramme der Vorlage und der Akustikaufnahme in eine binäre Form gebracht wurden. Das zweite Paket gibbonR ist spezifisch auf die automatische Erkennung von Geräuschen in Akustikaufnahmen abgestimmt und basiert auf dem Prinzip des deep learning. Die Algorithmen wurden mit je 1000 Akustikaufnahmen des Kullerns und Zischens trainiert. Der Algorithmus, der auf Stützvektormaschinen (SVM) basiert, konnte mit einer Wahrscheinlichkeit von 99.6% Kullern und Zischen richtig zuteilen. Mit diesem Paket konnten 74% der gesuchten Balzgeräusche entdeckt und markiert werden. Während das Potential von seewave ausgeschöpft ist und substanzielle Verbesserungen in der Trefferquote nicht möglich sind, könnte gibbonR durch eine Erhöhung der Trainingsdateien die Trefferquote erhöhen. Zudem ist der Arbeitsablauf bei gibbonR iterativ, wodurch die Trefferquote des Algorithmus zu- und der Zeitaufwand abnimmt. Anders benötigt seewave bei jeder automatischen Auswertung Vorlagen, die aus der Akustikaufnahme stammen, die untersucht wird. Weder wird der Zeitaufwand ab- noch die Trefferquote langfristig zunehmen. Beide Pakete klassifizierten Hintergrundgeräusche falsch. Bei gibbonR könnte dies mit der Erhöhung der Anzahl von Trainingsdaten verbessert werden. Auch sollte gibbonR, bei Verwendung einer Stützvektormaschine (SVM), gezielt mit Hintergrundgeräuschen trainiert werden, um bessere Ergebnisse zu erzielen. Zudem sollte das Potential von gibbonR besser untersucht werden. Für die Anwendung an anderen Tierarten, die passiv akustisch überwacht werden, eignet sich gibbonR durch seine deep learning Ansätze besser als seewave.

ABSTRACT

The numbers of black grouse (Tetrao tetrix) in Switzerland are declining due to various factors such as habitat loss or human disturbance. In order to observe the populations, the black grouse are passively monitored acoustically during the mating season. This form of monitoring generates acoustic recordings in large amounts of data. At the Zurich University of Applied Sciences (ZHAW), these recordings currently require manual evaluation using a spectrogram so that it can be determined when a black grouse makes the two mating noises, Kullern and Zischen. This financially and temporally large evaluation effort has to be optimized by automation. In this work, two R-packages suitable for automated detection were tested and the results compared. The first package seewave, which carries out the automatic recognition by means of a cross correlation between the spectrogram of four originals and the spectrogram of the acoustic recording, was able to detect and mark 25% of the desired mating noises. The hit rate could be increased to 77% after the original and acoustic recording spectrograms had been converted into a binary form. The second package gibbonR is specifically designed for the automatic detection of sound in acoustic recordings and is based on the principle of deep learning. The algorithms were trained with 1000 acoustic recordings each of Kullern and Zischen. The algorithm, which is based on support vector machines (SVM), was able to correctly classify Kullern and Zischen with a probability of 99.6%. With this package, 74% of the searched mating noise could be discovered and marked. While the potential of seewave is exhausted and substantial improvements in the hit rate are not possible, gibbonR could increase the hit rate by increasing the training files. In addition, the workflow in gibbonR is iterative, whereby the hit rate of the algorithm increases and the time required decreases. In contrast, seewave requires templates from the acoustic recording to be examined for each automatic evaluation. Neither the time required nor the hit rate will increase in the long run. Both packages incorrectly classified background noise. With gibbonR this could be improved by increasing the number of training data. GibbonR should also be specifically trained with background noise when using a support vector machine (SVM) in order to achieve better results. In addition, the potential of gibbonR should be better researched. For the application to other animal species that are passively monitored acoustically, gibbonR is better suited than seewave due to its deep learning approaches.

INHALTSVERZEICHNIS

1 EINLEITUNG

Das Birkhuhn (*Tetrao tetrix*) ist ein auf den Voralpen- und Alpenraum beschränktes Raufusshuhn, dessen Bestände und Verbreitung rückläufig sind (Ciach, 2015). Als Gründe hierfür werden die zunehmende Freizeitnutzung im Alpenraum und die Intensivierung der Alpwirtschaft genannt. Gleichzeitig spielen paradoxerweise auch die Nutzungsaufgabe von Alpwirtschaftsbetrieben und die darauffolgende Extensivierung und Vergandung von Lebensräumen eine grosse Rolle. Ausserdem sind kaltnasse Witterung, die den Bruterfolg mindern, Militär, welches in abgelegenen Orten trainiert, Schneeschuhwanderer und Variantenskifahrer, die abseits markierter Routen weit in Birkhuhnhabitate eindringen, weitere Faktoren, die sich negativ auf die Birkhuhnpopulation auswirken. (Spaar, Ayé, & Zbinden, 2012)

In der Schweiz wurde der Bestand der Birkhühner zwischen den Jahren 2013-2016 auf 12’000-16'000 Brutpaare geschätzt („Vogelwarte.ch“, 2019) und befinden sich auf der Roten Liste als potentiell gefährdete (near threatened, NT) Art. Bejagt werden dürfen Birkhühner in sechs Kantonen. (Spaar et al., 2012)

Eine Prognose für die Bestandsentwicklung des Birkhuhns abzugeben ist schwierig, da Wetterfaktoren und Populationsdynamik eine entscheidende Rolle spielen (Spaar et al., 2012). Ein möglichst genaues Monitoring ist somit wichtig, um präzise Prognosen formulieren zu können. Nur so ist es möglich Massnahmen zu treffen, um die Art langfristig schützen und fördern zu können. Die Balzaktivität der Birkhähne eignet sich besonders als Untersuchungsparameter und ist mit relativ geringem Aufwand möglich. Die Balzaktivität der Birkhähne lässt Rückschlüsse über den Bruterfolg der Birkhühner zu, welche wiederum Erkenntnisse über die gesamte Populationsdynamik geben. Über den Einfluss der vielseitigen Störungen auf die Balzaktivität der Birkhähne ist jedoch noch wenig bekannt.

Ein geeignetes Medium zur Untersuchung der Balzaktivität sind Akustikaufnahmen der balztypischen Geräusche. Akustikaufnahmen haben den Vorteil, dass sie wetterunabhängiger als andere Aufnahmemethoden, wie selbstauslösende Fotofallen oder direkte Beobachtungen, sind. So ist es beispielsweise nicht möglich bei Nebel alle balzenden Birkhähne mit Fotofallen erfassen zu können (Heinicke et al., 2015). Direkte Beobachtungen können sich negativ auf die Birkhahnbalz auswirken und diese auch direkt stören. Die Akustikaufnahmen wiederum erfassen über einen vorgegebenen Zeitraum alle Geräusche in einem vordefinierten Frequenzbereich.

Die meisten Akustikaufnahmen sind über eine Stunde lang und je nach Aufnahmemodus der Geräte werden Aufnahmen von sechs Stunden pro Tag generiert und gespeichert (Wildlife Acoustics, 2018). Es fällt ein grosses Datenvolumen von mehreren Gigabyte an, welches bewältigt werden muss. Und hierzu gibt es verschiedene Ansätze. Die Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) in Wädenswil wertet zurzeit das Akustikmaterial manuell aus. Jede Akustikaufnahme wird einzeln in

einem Spektrogramm visuell ausgewertet und Stellen, an denen eine Balzaktivität zu erkennen ist, werden herausgesucht und meist in einer Excel-Datei notiert und gespeichert. Daher ist die manuelle Auswertung mit hohem zeitlichem und auch finanziellem Aufwand verbunden.

Ein anderer Ansatz ist, die Auswertung der Akustikaufnahmen mit einer geeigneten Methode in einem Softwareprogramm durchzuführen und zu automatisieren. Dadurch könnte der zeitliche und finanzielle Aufwand gesenkt werden.

1.1 ZIELE DER ARBEIT

Im Rahmen dieser Arbeit soll geklärt werden, wie sich in Zukunft akustische Aufnahmen automatisch auswerten lassen. Die Entwicklung einer automatisierten Methode zur Erkennung der Birkhahnbalz soll ein Produkt dieser Arbeit sein und um dies zu erreichen, sind folgende Fragestellungen zu beantworten:

- Welche Methoden der automatisierten Erkennung eignen sich für Akustikaufnahmen?
- Wo liegen die Vor- und Nachteile der unterschiedlichen Methoden zur automatischen Erkennung?
- Wie akkurat/zuverlässig ist die automatisierte Auswertung im Vergleich zur manuellen?

Um die Erkennung von akustischer Aktivität vollends zu automatisieren, gilt es eine Methode zu entwickeln, die ausnahmslos alle Balzgeräusche in einer Audioaufnahme erkennt und diese in einer Tabelle mit einer Zeitreferenz zusammenfasst. Diesem Anspruch gerecht zu werden, ist übergeordnetes Ziel dieser Arbeit.

Wenn die automatische Auswertung gute Ergebnisse erzielt, werden noch Akustikaufnahmen ausgewertet und diese mit bereits ausgewerteten Bildern aus Fotofallen verglichen, um zu sehen, was akustisch und was visuell erfasst werden kann.

2 DAS BIRKHUHN

In der Schweiz ist die Verbreitung des Birkhuhns *(Tetrao tetrix)* auf den Alpenraum beschränkt. Dort besiedelt es Bereiche, die an der oberen Waldgrenze liegen und der Wald zunehmend lichter wird und in einen Zwergstrauchgürtel übergeht. Diese Bedingungen entstehen vor allem da, wo die Waldgrenze durch Holschlag künstlich gesenkt wurde. (Bergmann, Marti, Müller, Vitovic, & Wiesner, 1990)

Birkhähne lassen sich anhand des ausgeprägten Dimorphismus leicht von Birkhennen unterscheiden (Abbildung 1). Die Birkhähne besitzen ein verschiedenfarbiges Prachtkleid, während die Henne farblich schlichter erscheint.

Abbildung 1: Gut erkennbarer Dimorphismus der Birkhühner. Links: Hahn, rechts: Henne.
Quelle: https://www.rspb.org.uk/birds-and-wildlife/wildlife-guides/bird-a-z/black-grouse/

Auch in der Körpermasse sind deutliche Unterschiede vorhanden. Eine adulte Henne wiegt ca. 960 g und ist damit rund 400 g leichter als ein Hahn, der damit etwa die Grösse eines Haushuhns erreicht. (Bergmann et al., 1990) Die Spannweite beträgt zwischen 65 und 80 cm („Vogelwarte.ch“, 2019).

2.1 BALZGERÄUSCHE

Die alljährliche Balz der Birkhähne, die von Ende Januar bis in den Mai dauern kann, dient zur Anwerbung weiblicher Fortpflanzungspartner (Bergmann et al., 1990). Die Balz findet ausschliesslich auf Balzplätzen statt, die immer im offenen Gelände liegen (Potapov & Flint, 1989 in Bergmann et al., 1990). Auf den Balzplätzen spielen feste Landmarken wie Äste, kleine Bäume oder Zäune eine wichtige Rolle für das Einhalten der Reviergrenzen und der räumlichen Orientierung der Hähne. (Bergmann et al., 1990)

Für diese Arbeit wurden die zwei häufigsten Balzgeräusche, das Kullern und Zischen, ausgewählt und näher untersucht. Es existieren weitere, auf die in der vorliegenden Arbeit jedoch nicht eingegangen wird.

2.1.1 DAS KULLERN

Das Kullern spiegelt die dominierende Lautäusserung auf einem Balzplatz wider. In der Literatur wird das Kullern als «anhaltender, tieftonaler, bauchrednerischer» Gesang beschrieben (Bergmann et al., 1990) und ist bei klarem Wetter und bei Windstille meist über 3 km weit wahrzunehmen (Glutz von Blotzheim, 1994). Das für das Kullern typische tiefe Frequenzband von 0.4-0.8 kHz (Abbildung 2) machen es schwer den balzenden Hahn zu lokalisieren, auch wenn dieser wiederholt kullert (Bergmann et al., 1990).

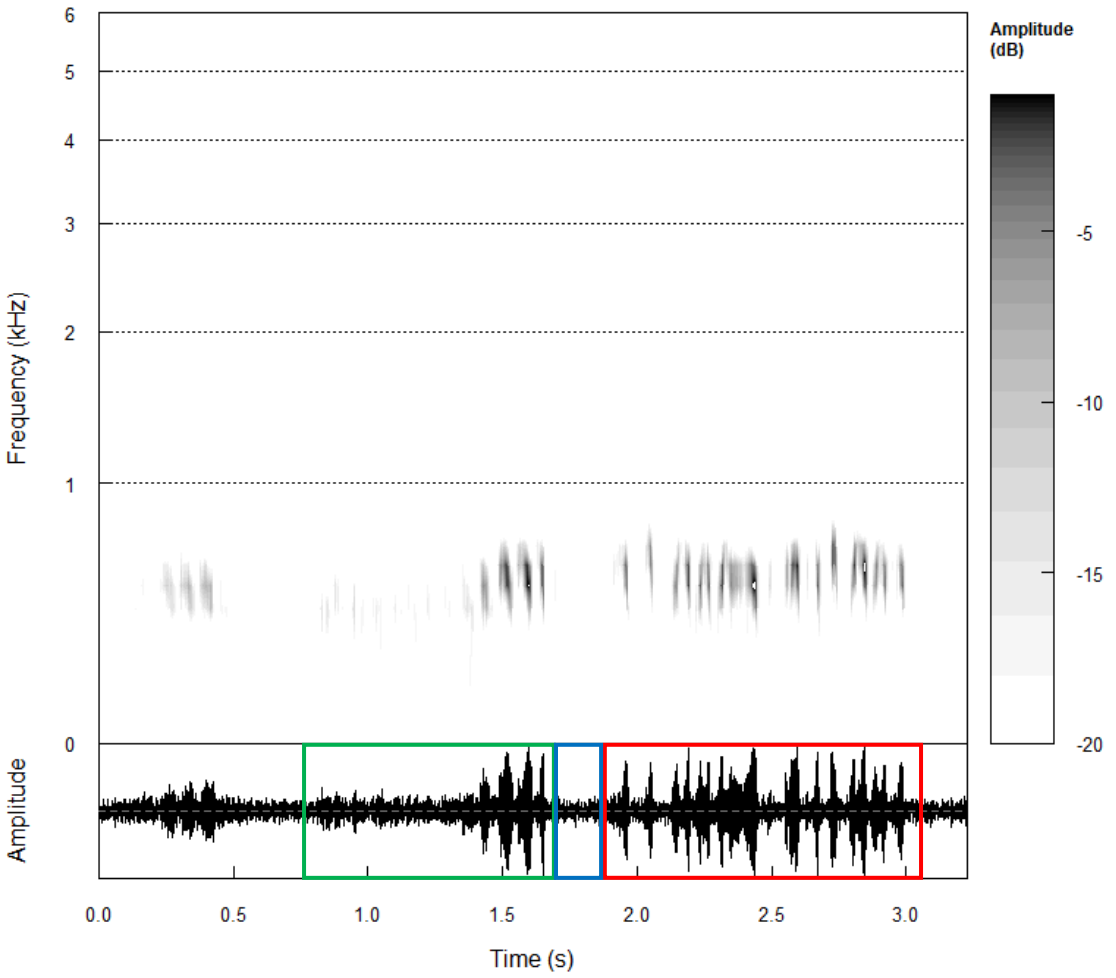


Abbildung 2: Kullern eines Birkhahnes während der Balz. Grün: 1. Teil, blau: Pause, rot: 2. Teil. Quelle: Burkhalter, 2019

Das Kullern besteht aus zwei Teilen, die durch eine 0.15 s kurze Pause getrennt sind. Der erste Teil dauert im Durchschnitt 1 s und gegen Ende des ersten Teils kullert der Birkhahn mit zunehmender Amplitude und Frequenz (Abbildung 2). Der zweite Teil dauert in der Regel etwas über 1.2 s und schliesst das Kullern ab. Dennoch besitzt das Kullern eine grosse Variabilität in seiner Länge. Anhaltendes Kullern der Birkhähne über mehrere Minuten sind keine Seltenheit. Das Kullern kann mit einem Gurgeln verglichen werden (Glutz von Blotzheim, 1994).

2.1.2 DAS ZISCHEN

Anders als das Kullern ist das Zischen gut lokalisierbar und über eine Distanz von etwa 1 km wahrnehmbar. Morgens und abends beginnt bzw. endet die Tagesaktivität der balzenden Hähne oft mit dem Zischen. Gezischt wird von den Hähnen, anders als beim Kullern, auch abseits der Balzplätze. (Bergmann et al., 1990) Bei hoher Balzaktivität zischt der Hahn zwei- bis dreimal pro Minute und dies ist nicht selten an balztypische Flügelschläge geknüpft. Die Länge des Zischens beträgt in der Regel 1 s (Glutz von Blotzheim, 1994) und ist in seiner Struktur homogen.

Das Zischen kann ebenfalls in zwei Teile unterteilt werden (Abbildung 3).

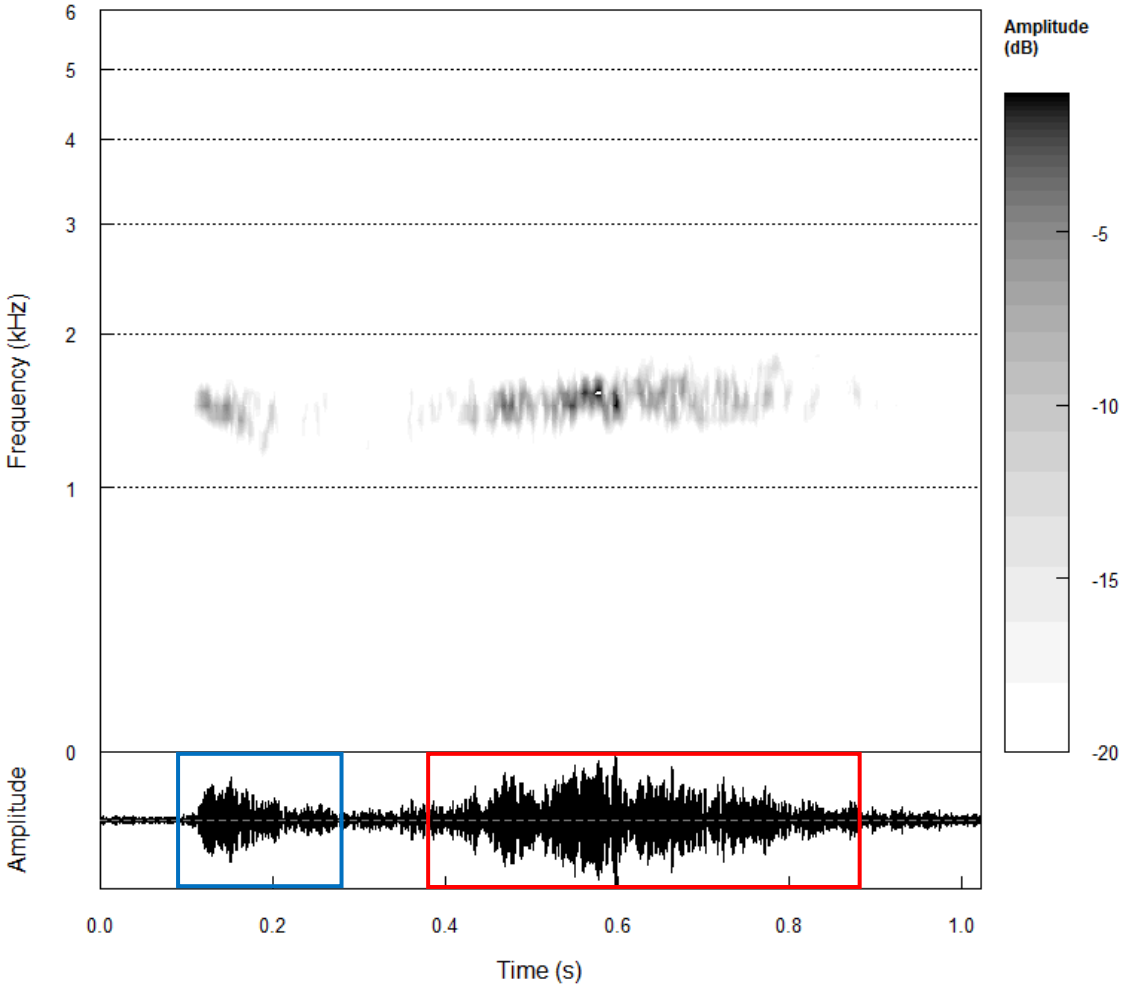


Abbildung 3: Zischen eines Birkhahnes während der Balz. Blau: 1. Teil, rot: 2. Teil. Quelle: Burkhalter, 2019

Der erste Teil wird als «tschu» beschrieben, anschliessend, nach einer 0.15 s kurzen Pause, gefolgt von einem «chäh» (Bergmann et al., 1990). Das Zischen ist in einem Frequenzband von 1100 bis 1900 Hz zu finden, dennoch können teils durch eine Variabilität des Zischens Frequenzbänder von 1100 bis 2100 beansprucht werden.

2.2 MONITORING DER BALZGERÄUSCHE

Bei vielen Tierarten sind heutzutage die visuelle Beobachtung und die Fang- und Wiederfangmethode die gängigsten Monitoringmethoden. Diese eignen sich jedoch unter anderem nicht für Birkhühner, da diese schwierig zu fangen, zu markieren oder zu sichten sind, weshalb ein anderer Ansatz für das Monitoring solcher Tierarten gesucht werden muss. (Marques et al., 2013)

Für das Monitoring der Birkhühner werden zurzeit verschiedene Methoden angewendet. Zum einen besteht die Möglichkeit die Birkhühner direkt zu beobachten. Die Schwierigkeit dabei ist, dass Birkhühner sensibel auf menschliche Störungen reagieren (Spaar et al., 2012) und schnell zur Flucht neigen. Zudem ist diese Methode äusserst teuer aber bestimmungsgenau, da ein Wildhüter oder eine ähnlich ausgebildete Person Birkhühner beobachten sollte. Ob sich die Birkhühner zeigen und tatsächlich beobachtet werden können, entscheidet oft der Zufall, was diese Beobachtungsmethode zeitaufwendig und teuer macht.

Eine weitere Möglichkeit besteht im passiven visuellen Monitoring. Hierzu eignen sich Fotofallen. Diese werden eingesetzt, um Birkhühner zu beobachten. Fotofallen lösen automatisch aus, wenn sich ein Objekt in den Blickwinkel der Kamera bewegt oder sie lösen in einem vorgegebenem Zeitintervall aus. Für ein erfolgreiches Monitoring der Birkhühner können an einschlägig bekannten Balzplätzen Fotofallen aufgestellt werden, um möglichst viele balzende Birkhähne fotografieren zu können. Nachteilig wirkt sich die Wetterabhängigkeit dieser Methode aus. So verunmöglicht Nebel oder Dunkelheit die Produktion qualitativ hochwertiger, auswertbarer Fotos. Dieses Problem besitzt das passive akustische Monitoring (PAM) nicht.

Das PAM wird bereits grossräumig bei Meeressäugern eingesetzt (Munger, Mellinger, Wiggins, Moore, & Hildebrand, 2005), da im Wasser die Sicht limitierender Faktor ist. Für Birkhühner kann diese Methode ebenfalls eingesetzt werden. Dafür werden an bekannten Balzplätzen Aufnahmegeräte platziert, die über einen vorgegebenen Zeitraum und Frequenzbereich alle Geräusche aufnehmen (Wildlife Acoustics, 2018). In den letzten Jahren wurden viele Fortschritte im Bereich der technischen Bioakustik verzeichnet. Während vor 20 Jahren die Batterielebenszeit oder die begrenzten Möglichkeiten der Datenspeicherung limitierende Faktoren für das PAM waren, zählen diese heutzutage durch anhaltende technische Verbesserungen nicht mehr dazu (Clink & Klinck, 2019). Vielmehr ist die manuelle Auswertung der gross anfallenden Datenvolumen limitierend, dies obwohl sich Daten des PAM aufgrund der immer gleichen Datenstruktur für eine automatische Auswertung eignen würden (Marques et al., 2013).

3 MÖGLICHKEITEN DER AUTOMATISIERTEN ERKENNUNG

Automatisierung – dieses Wort ist allgegenwärtig. In Zeiten der Digitalisierung werden zunehmend manuelle Prozesse automatisiert. Dies betrifft auch die Auswertung von Daten. Als Gründe werden die immer grösseren Datenmengen und die immer leistungsfähigeren Computer genannt (Morfi & Stowell, 2018). So auch in der Bilderkennung. In diesem Bereich ist die Automatisierung weit fortgeschritten und verzeichnet beeindruckende Ergebnisse (Dörfler, Bammer, & Grill, 2017). Aber auch in der Spracherkennung gibt es immer bessere Algorithmen und Methoden (Tóth & Czeba, 2016), um verschiedene Geräusche zuzuordnen und zu klassifizieren. Bereits 1997 wurden Tests zur Entwicklung eines Mehrprogrammhörgeräts mit neuronalen Netzwerken durchgeführt und Klassifizierungsquoten von 96% erreicht (Feldbusch, 1998). Neun Jahre später wurde die Performance der Erkennung von Vogelarten mit Hilfe neuronaler Netzwerke untersucht. Bei dieser Untersuchung konnten 98.7% der Vogelarten richtig zugeordnet werden. (Cai, Ee, Pham, Roe, & Zhang, 2007)

Die für die Untersuchungen benutzten Algorithmen wurden auf den Mel-Frequenz-Cepstrum-Koeffizienten (MFCC) angewendet. Diese bringen ein Frequenzspektrum in eine kompakte Darstellungsform, reduzieren die Datenmenge substanziell ohne hohen Informationsverlust und sind somit auch besser transportierbar. In der Spracherkennung liefern Algorithmen mit neuronalen Netzwerken basierend auf MFCC signifikant bessere Ergebnisse. (Cai et al., 2007) Essenziell für eine hohe Zuordnungs- und Trefferquote sind die Trainingsdatensätze, die verwendet werden. Grundsätzlich gilt, mit je mehr Daten ein neuronales Netzwerk trainiert wird, desto besser ist die Zuordnungs- und Trefferquote bei der Anwendung (Morfi & Stowell, 2018). Zudem scheinen bei Langzeitaufnahmen deep learning Methoden wie neuronale Netzwerke (Kapitel 3.1) besser anwendbar. Dies zeigen Untersuchungen mit fünf verschiedenen Modellen (Li, Dai, Metze, Qu, & Das, 2017).

3.1 DEEP LEARNING UND NEURONALE NETZWERKE

Der Begriff umschreibt die Tiefe mit der eine Methode angewandt auf ein Modell agiert. Die Tiefe repräsentiert die Anzahl Schichten (Layer) die verwendet werden, um die Informationen aus einer Datei herauszufiltern (Abbildung 4). Desto mehr Schichten vorhanden sind, umso mehr Tiefe besitzt der Algorithmus. Dem gegenüber steht das Lernen mit wenigen Schichten, welches shallow learning genannt wird. (Chollet & Allaire, 2018)

Deep learning umfasst fast immer auch den Begriff neuronales Netzwerk. Ein neuronales Netzwerk stapelt Schichten übereinander (Abbildung 4). Die Definition des Begriffs wird teilweise von dem menschlichen Verständnis abgeleitet, wie das menschliche Hirn lernt, also Informationen verarbeitet.

Trotzdem sind neuronale Netzwerke kein Modell des menschlichen Hirns und gleichen nach heutigem Verständnis auch nicht dem des maschinellen neuronalen Netzwerkes. (Chollet & Allaire, 2018)

Vielmehr muss man sich ein tiefes neuronales Netzwerk (deep neural network) als vielstufige Informationsdestillation vorstellen, bei der Informationen sukzessiv durch einen Algorithmus gefiltert und purifiziert werden (Abbildung 4) (Chollet & Allaire, 2018).

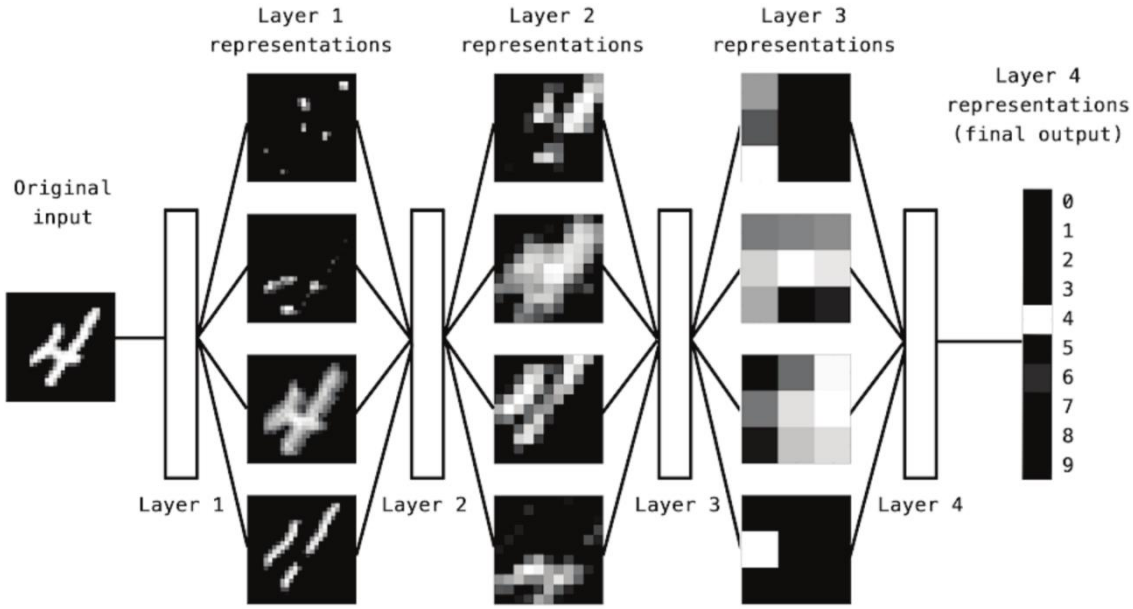


Abbildung 4: Schematische Darstellung des deep learning. Quelle: Chollet & Allaire, 2018

Viele dieser neuronalen Netzwerke sind mittlerweile auf diversen Softwareprogrammen anwendbar und verfügbar. Auch darum übernehmen neuronale Netzwerke immer mehr Bereiche der heutigen Gesellschaft. Vor allem im Bereich der sozialen Netzwerke, in dem gezielt durch Algorithmen Benutzer- und Anwenderdaten verarbeitet werden.

Für die Automatisierung manueller Prozesse ist ein geeignetes Softwareprogramm auszuwählen. Um die unkomplizierte Reproduzierbarkeit einer Arbeit zu gewährleisten, sind frei verfügbare Softwareprogramme von Vorteil.

3.2 STATISTIKPROGRAMM R

Das Statistikprogramm R, welches 1996 in Neuseeland entstand (Ihaka & Gentleman, 1996), ist frei verfügbar und modular aufgebaut, was bedeutet, dass jede Person einen Beitrag zu dem Programm leisten kann. Dies geschieht in Form von R-Paketen, die in R bei Bedarf heruntergeladen, installiert und verwendet werden können. Das macht es für fast alle Bereiche der Datenanalyse einsetzbar. Die Sprache, die R spricht, ist unkompliziert und logisch aufgebaut und dadurch, dass R frei verfügbar und in der Welt der Datenanalyse omnipräsent ist, existieren zahlreiche Foren und Hilfestellung im Internet. (Sueur, 2018) Trotz allen Vorteilen ist R nicht perfekt. Ein grosser Nachteil besteht darin, dass alle R-Objekte auf dem Arbeitsspeicher (RAM) zwischengespeichert werden (Ihaka & Gentleman, 1996). Je nach Grösse der Auswertungsdateien und Komponenten des Computers stösst R schnell an seine Grenzen.

Aufgrund der genannten Vorteile wird in dieser Arbeit ausschliesslich mit R gearbeitet. Die folgenden R-Pakete beinhalten Möglichkeiten die manuelle Auswertung von Akustikaufnahmen zu automatisieren.

3.2.1 SEEWAVE

Dieses R Paket widmet sich stark der Analyse und Synthese von Geräuschen der Bioakustik. So können quantitative Unterschiede zwischen verschiedenen Geräuschen festgestellt werden. Ausserdem sind Analysen über Zeit, Amplitude und Frequenz möglich und Hauptbestandteil dieses Paketes. (Sueur, Aubin, & Simonis, 2008)

3.2.1.1 AUTOMATISIERTE ERKENNUNG

Seewave bietet zusätzlich die Möglichkeit Geräusche automatisch erkennen zu lassen. Hierfür bedient sich das Paket in vielen Funktionen bei anderen R-Paketen, wie tuneR und monitoR. Vor allem monitoR liefert in Bezug auf die automatische Erkennung viele Funktionen, die über seewave abgerufen werden können. In seewave bzw. monitoR werden für die automatische Erkennung zwei «template-matching» Algorithmen zur Verfügung gestellt und angewendet. Zum einen die Spektrogramm Cross-Korrelation (spectrogram cross correlation) und zum anderen die Binäre Punkte Übereinstimmung (binary point matching) (Abbildung 5) (Hafner & Katz, 2018).

Grundsätzlich vergleicht die Spektrogramm Cross-Korrelation einen Referenzausschnitt eines Spektrogramms mit dem Spektrogramm einer Akustikaufnahme und sucht jene Stellen heraus, die mit dem

Referenzausschnitt am besten übereinstimmen. Der Nutzer kann ausserdem eine Schwelle festlegen, ab wieviel Prozent Übereinstimmung, diese als Erkennung markiert werden soll. (Munger et al., 2005).

Die Binäre Punkte Übereinstimmung ist in den Grundzügen ähnlich wie die Spektrogramm Cross-Korrelation. Auch sie vergleicht eine Vorlage mit einer Akustikaufnahme. Doch das Spektrogramm der Vorlage ist binär. Der Nutzer kann in R eine Amplitudenschwelle festlegen, an welcher eine Stelle in einem Spektrogramm als «off» oder als «on» eingeteilt wird. Anschliessend wird das binäre Spektrogramm der Vorlage mit dem Spektrogramm einer Akustikaufnahme verglichen. (Abbildung 5) (Hafner & Katz, 2018)

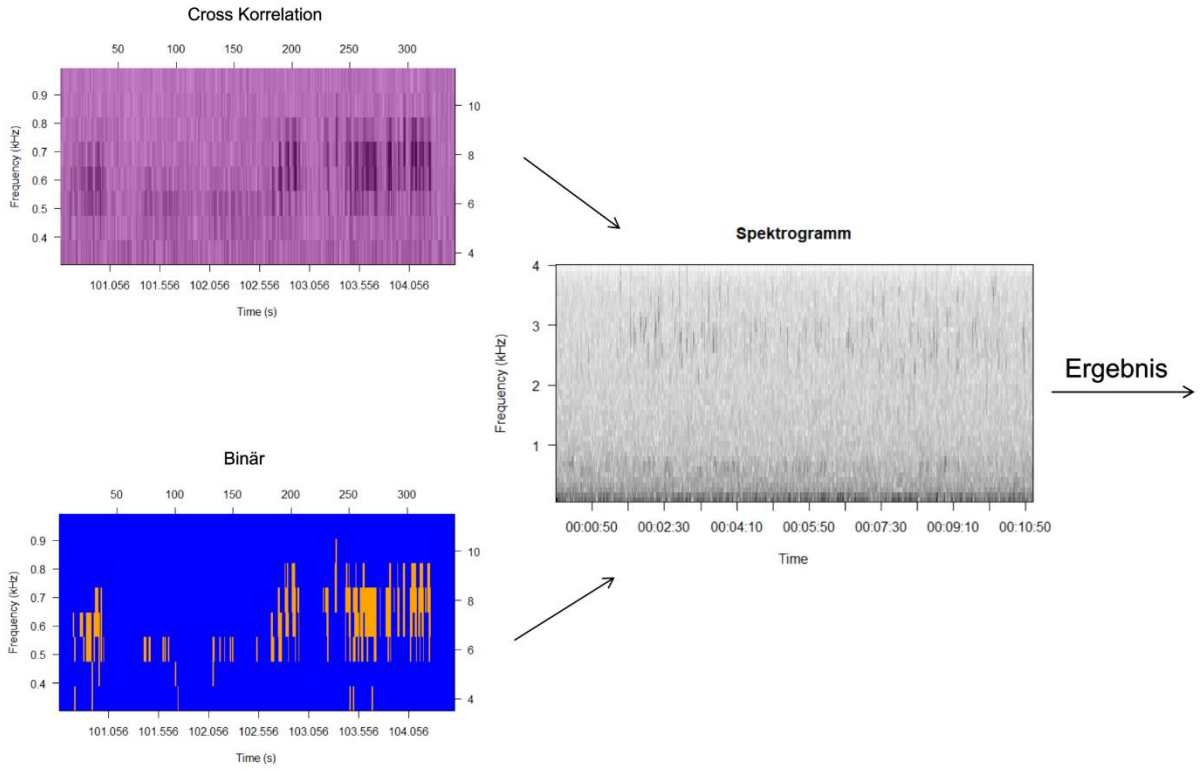


Abbildung 5: Schematische Darstellung der beiden verwendeten Algorithmen im R-Paket seewave bzw. monitoR. Quelle: Burkhalter, 2019

3.2.2 GIBBONR

Dieses R-Paket ist laut Entwicklern (Clink & Klinck, 2019) an Ökologen gerichtet, die an Bioakustikauswertung interessiert sind. Es macht die meist verwendeten Signalverarbeitungstechniken, sowie diverse «machine learning» Algorithmen einfach zugänglich und damit unkompliziert anwendbar. Hauptsächlich soll dieses R-Paket helfen, die Auswertungszeit von Langzeitakustikaufnahmen substanziell zu senken.

3.2.2.1 AUTOMATISIERTE ERKENNUNG

Die Möglichkeiten mit Hilfe des R-Pakets gibbonR Geräusche in Akustikaufnahmen erkennen zu lassen, basieren auf drei verschiedenen Algorithmen. Namentlich Support Vector Machines (SVM), Gaussian Mixture Models (GMM) und Artificial neural networks (NNET). (Heinicke et al., 2015; Keen, Shiu, Wrege, & Rowland, 2017; Mielke & Zuberbühler, 2013) Diese werden unter anderem in der Spracherkennung verwendet, finden aber aufgrund der einfachen Verfügbarkeit immer mehr den Weg in die Auswertung der Bioakustik (Clink & Klinck, 2019). Um eine Übersicht über die Funktionsweise der Algorithmen zu erhalten, werden diese nachfolgend kurz beschrieben.

Support Vector Machines (SVM)

SVM, zu Deutsch «Stützvektormaschine», versucht Datensätze bestmöglich voneinander zu trennen. Dafür werden verschiedene Vektoren in und um die Verteilung der Datensätze gelegt. Der Vektor, der es erreicht, die verschiedenen Datensätze, in dieser Arbeit Kullern und Zischen, voneinander zu trennen und den grössten Abstand zum jeweils anderen Datensatz herstellen kann, bestimmt wie gut die Datensätze voneinander unterschieden werden können. Im Idealfall beträgt dies 100%, welches in Abbildung 6 beide Vektoren betrifft, jedoch Vektor A einen deutlich grösseren Abstand herstellen kann. Deshalb ist dieser geeigneter für beispielsweise eine Anwendung im Bereich der automatischen Erkennung, da diese SVM mit dem Vektor A deutlicher zwischen verschiedenen Datensätzen unterscheiden kann als Vektor B (Berwick, 2003).

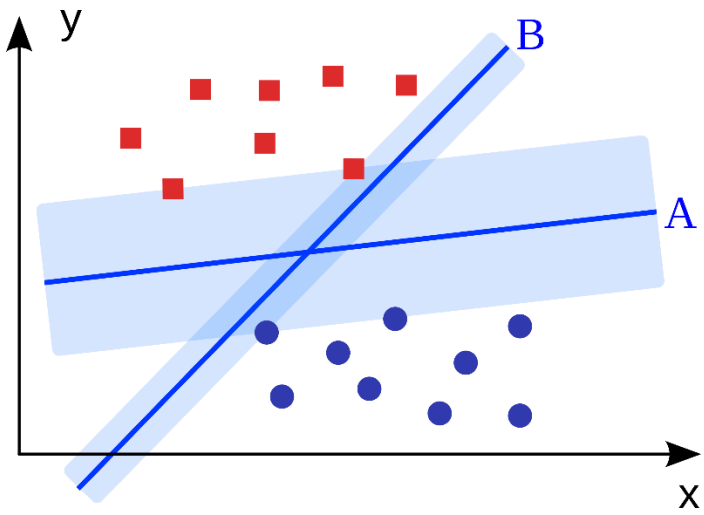


Abbildung 6: Prinzip der SVM. Vektor A besser als Vektor B. Quelle: Von Ennepetaler86 - Eigenes Werk, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=11523156

Gaussian Mixture Models (GMM)

Die Gausssche Mischverteilung (GMM) ist ein häufig vorkommender Spezialfall der Mischverteilung. Als Beispiel für eine Mischverteilung kann das Gewicht von Birkhuhn und Birkhahn genommen werden. Werden beide Gewichte zusammengefasst, parametrisiert und in einer Dichtefunktion dargestellt, würde eine Grafik zwei Spitzen aufweisen (Abbildung 7). Da Birkhuhn und Birkhahn aber ein verschiedenes Gewicht aufweisen, kann die Mischverteilung in zwei Normalverteilungen aufgeteilt werden. (Reynolds, 2015) Dieses Prinzip macht sich der Algorithmus zu Nutze, um zwischen Birkhuhn und Birkhahn anhand des Gewichts zu unterscheiden.

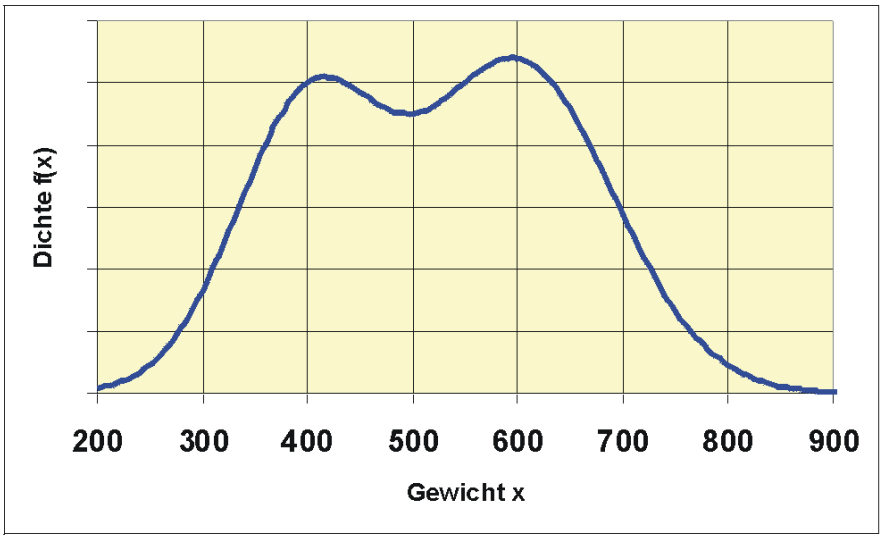


Abbildung 7: Klassische Mischverteilung. Gut zu erkennen sind die zwei Spitzen, was darauf schliessen lässt, dass zwei verschiedene Grössen zusammengefasst wurden. Quelle: Von Philipendula - selbst erstellt von Philipendula, CC BY-SA 3.0, https://de.wikipedia.org/w

Anders als das Gewicht der Birkhühner, sind die Balzgeräusche der Birkhähne unterschiedlich und nicht aufgrund eines Parameters unterscheidbar. Mit der Gaussschen Mischverteilung ist es möglich, mit Hilfe von Trainingsdaten und einem Expectation-Maximization (EM) Algorithmus, der iterativ angewendet wird, die verschiedenen Datensätze in Gruppen zuzuordnen. In dieser Arbeit wären dies erneut die Balzgeräusche der Birkhähne, die anhand dieser Methode in Kullern und Zischen eingeteilt werden können. (Reynolds, 2015)

Artificial neural networks (NNET)

Das Prinzip der artificial neural networks (auf Deutsch «künstliche neuronale Netzwerke») basiert auf dem in Kapitel 3.1 beschriebenen neuronalen Netzwerken. Nach der Konstruktion eines künstlichen neuronalen Netzwerks kann dieses mit klassifizierten Daten gespeist werden. So können dem Algorithmus verschiedene «Begriffe» beigebracht werden, die anschliessend bei einer Anwendung abgerufen werden können. (Ching, Zhu, & Garmire, 2018)

4 MATERIAL UND METHODE

4.1 AUSGANGSLAGE

Die beschriebenen R-Pakete verwenden vielversprechende Ansätze für die automatische Erkennung von Geräuschen in Akustikaufnahmen. Es werden die zwei beschriebenen R-Pakete getestet, um herauszufinden, welches sich für die Praxis besser eignet. Unter anderem vergleiche ich den Zeitaufwand, der pro Methode anfällt, um eine Akustikaufnahme auszuwerten. Dazu führe ich verschiedene Tests durch.

Ausserdem soll der gesamte Arbeitsaufwand pro Methode untersucht und verglichen werden. Dieser beinhaltet Einarbeitung in die R-Pakete und das Erstellen der R-Skripts. So kann für zukünftige Anwender ein grober Zeitaufwand pro Methode angegeben werden.

4.2 VERWENDETE MATERIALIEN

Für diese Arbeit verwende ich R-Studio (Version 3.5.1) und die R-Pakete seewave, tuneR, signal, audio, monitoR, warbleR, ggplot2, dtw, caTools, phonTools, tidyverse, remote, gibbonR, viridis, mixtools, data.table, cowplot und lubridate. Einige dieser R-Pakete überschneiden sich in gewissen Funktionen.

In R verwende ich selbst verfasste Skripte (Anhang A) basierend auf den Paketen seewave und gibbonR.

Basierend auf dem R-Paket seewave:

- «DarstellungEinzelnerErgebnisse.R» für die genaue Darstellung einzelner Silben oder Geräuschen
- «AutomaticDetection.R» für die automatische Erkennung mit der Spektrogramm Cross Korrelation. Fortan in der Arbeit «CC Seewave» genannt.
- «BinAutoDec.R» für die automatische Erkennung mit der Binäre Punkteübereinstimmung. Fortan in der Arbeit «Bin Seewave» genannt.

Basierend auf dem R-Paket gibbonR

- «AutomatischeErkennung.R» für die automatische Erkennung mit verschiedenen Algorithmen. Fortan in der Arbeit «gibbonR» genannt.

Für die exakte Begutachtung von Spektrogrammen setze ich das Programm Sonic Visualiser (Version 3.2.1) der Queen Mary, University of London ein. Dieses Programm erlaubt eine äusserst genaue Betrachtung, Analyse und Bearbeitung des Spektrogramms.

Da ich in dieser Arbeit mit grossen Audiodateien (>1 GB) arbeiten und in R bearbeiten muss, verwende ich einen Computer mit einer Arbeitsspeicherkapazität (RAM) von 32 GB.

4.2.1 DARSTELLUNG

Um die Ergebnisse in einer geeigneten Form darstellen zu können, habe ich das R-Skript «DarstellungErgebnisse.R» verfasst. Es lehnt sich in der Darstellungsform von den bisherigen Auswertungen der Forschungsgruppe Wildtiermanagement der ZHAW Wädenswil an und teils wurden ihre R-Codes Grundlage verwendet.

4.2.2 VERGLEICH FOTOFALLEN UND AKUSTIKAUFNAHMEN

Um die beiden Methoden vergleichen zu können, verwende ich die ausgewerteten Daten der Fotofallen (Plotwatcher) als Referenz. Danach untersuche ich Akustikaufnahmen (Songmeter) des Jahres 2016, die in der Hochalp im Kanton Appenzell Ausserhoden im Einsatz waren. Ausgesucht habe ich den Songmeter 2 (SM2), dieser steht in unmittelbarer Nähe zum Plotwatcher 3 (PL3) (Abbildung 8) und eignet sich deshalb für einen Vergleich zwischen akustischer und visueller Aufnahmemethode. Von Songmeter 2 untersuche ich die Tage vom 11.05. bis zum 22.05.2016, da in diesem Zeitbereich abwechselnd viel bzw. wenig Balzaktivität mit dem PL3 erfasst wurde. Zur abschliessenden Darstellung der Ergebnisse verwende ich auch hierfür das R-Skript «DarstellungErgebnisse.R».

Abbildung 8: Positionen des PL3 und SM2 und ihrer geographischen Nähe zueinander. Quelle: https://earth.google.com/web/@47.27719356,9.25809557,1278.9163471a,632.33126573d,35y,0h,0t,0r

4.3 METHODE

Als Ausganglage dienen mir drei Testdateien (Tabelle 1), die ich manuell mit dem Programm Sonic Visualiser auswerte und alle vorkommenden Balzgeräusche in einer CSV-Datei notiere. Dies bildet die Grundwahrheit der anschliessenden Auswertung. Die drei anzuwendenden R-Skripts werden mit dieser Grundwahrheit verglichen.

Tabelle 1: Übersicht über die untersuchten Testdateien. Quelle: Burkhalter, 2019.

	Testkurz	Testdatei	Testlang
Länge [s]	280	660	7200

Vorab habe ich einige Grundsätze zu den Balzgeräuschen definiert. Zudem beantworten diese die Frage, ab wann in dieser Arbeit ein Balzgeräusch als ein Balzgeräusch angesehen wird. Hier habe ich bewusst grössere Frequenzbänder gewählt, um möglichst viele Balzgeräusche zu entdecken.

- Kullern: Frequenzbereich: 0.2-1 kHz, Dauer: mind. 3 s, max. offen
- Zischen: Frequenzbereich: 1.2-2 kHz, Dauer: mind. 1 s, max. 3 s

In den R-Skripts verwende ich diese definierten Grundsätze in den Argumenten der R-Funktionen.

4.3.1 ANWENDUNG MIT SEEWAVE

Der Musterarbeitsablauf für die automatische Erkennung von Geräuschen mit dem R-Paket seewave bzw. monitoR kann wie folgt beschrieben werden (Sueur, 2018). Die detaillierten R-Codes befinden sich im Anhang A:

1. Auswahl von mehreren Vorlagen (templates) aus einem projizierten Spektrogramm, die das Geräusch enthalten, welches gesucht wird. Diese Vorlagen sollten eine gute Übersicht über die Frequenz- und Zeitvariabilität und Hintergrundgeräusche der Sound of Interest (SOI) geben. Die Auswahl der Vorlagen sollten aus der Akustikaufnahme stammen, die untersucht werden soll.
2. Setzen einer Schwelle, ab welcher eine Übereinstimmung als Übereinstimmung gelten soll.
3. Anwendung der Vorlagen auf eine Akustikaufnahme, die untersucht werden soll.
4. Ausgabe der Punkte mit hohen Übereinstimmungen.
5. Nachbearbeitung der Ausgabe, um das Gesamtergebnis zu verbessern. Unter anderem soll die «false positive» bzw. «false negative rate» minimiert werden, also Punkte die fälschlicherweise als falsch bzw. richtig eingeteilt wurden.
6. Wenn erwünscht: Visuelle Darstellung der Ergebnisse.

Aufgrund dieses Musterablaufs habe ich mit dem R-Paket seewave zwei R-Skripte verfasst, die speziell an die Balzgeräusche der Birkhähne angepasst sind. Es werden vier Vorlagen (Sueur, 2018) der beiden gesuchten Balzgeräusche von unterschiedlichen Akustikaufnahmen erstellt, an denen sich der Suchalgorithmus orientiert. Nun kann ich mit Hilfe der R-Skripts versuchen, alle vorkommenden Balzgeräusche zu finden. Die durchsuchten Akustikaufnahmen sind diejenigen, aus denen die vier Vorlagen stammen und entspricht damit den Empfehlungen der Entwickler der R-Pakete (Sueur, Hafner, 2019, persönliche Mitteilung). Jedes der beiden R-Skripte wird für die beiden Balzgeräusche einzeln angewendet und die alle Ergebnisse zusammen in einer CSV-Datei gespeichert. Dies wird für alle drei Testdateien (Tabelle 1) durchgeführt. Somit wende ich das R-Skript von seewave zwölfmal an und speichere die Ergebnisse zur weiteren Auswertung in sechs CSV-Dateien.

4.3.1.1 AUSGABE DER ERGEBNISSE

Seewave erstellt nach einer automatischen Auswertung eine Liste in R, die in einen dataframe umgewandelt wird, um die Ergebnisse anschliessend als CSV-Datei zu exportieren (Abbildung 9).

	A	B	C	D	E
1	template	date.time	time	score	detection
2	t3	13.05.2015 06:26	84.5090249	0.18066372	TRUE
3	t3	13.05.2015 06:26	90.4068934	0.29180932	TRUE
4	t4	13.05.2015 06:26	90.4185034	0.33657354	TRUE
5	t1	13.05.2015 06:26	90.4301134	0.39702185	TRUE
6	t2	13.05.2015 06:26	90.4649433	0.34147472	TRUE

Abbildung 9: Ausschnitt aus einer CSV-Datei mit Ergebnissen aus einer Auswertung mit dem R-Paket "seewave". Spalte "template" = Vorlage, die einen Treffer erzielen konnte; Spalte "date.time" = genaue Zeitangabe wann der Treffer gefunden wurde; Spalte "time" = in der wievielten Sekunde der untersuchten Aufnahme der Treffer gefunden wurde; Spalte «score» = Übereinstimmung in Prozent (1=100%) mit der Vorlage; Spalte «detection» = ob der Treffer den Schwellenwert überschritten hat und als Treffer gezählt wurde. Quelle: Burkhalter, 2019

Es werden fünf Spalten ausgegeben. Spalte «template» beschreibt welche Vorlage einen Treffer mit einer Stelle in der untersuchten Datei erzielen konnte. Spalte «date.time» gibt mit einem Datum an, wann der Treffer gefunden wurde. Hierbei ist es erforderlich, dass in R die korrekte Zeitzone und Anfangszeit angegeben wurde. Spalte «time» stellt dar, in der wievielten Sekunde der Treffer in der untersuchten Aufnahme gemacht werden konnte. In der Spalte «score» ist die Übereinstimmung in Prozent (1 = 100%) angegeben. Spalte «detection» beschreibt, ob der Treffer den vorgegebenen Schwellenwert überschritten hat und als Treffer gezählt wurde. Es besteht keine Möglichkeit die Treffer als Audiodatei auszugeben und akustisch zu überprüfen.

4.3.2 ANWENDUNG MIT GIBBONR

Der Ablauf einer automatischen Erkennung wird wie folgt beschrieben (Clink & Klinck, 2019) (Abbildung 10), die dazugehörigen R-Codes befinden sich im Anhang A:

1. Referenzdaten klassifizieren. Um dem Algorithmus eine Grundwahrheit zur Verfügung zu stellen, müssen Geräusche, die gesucht werden, klassifiziert werden. Die gleichen Geräusche erhalten die gleichen Namen.
2. Training des machine learning Algorithmus.
3. Detektor über eine Akustikaufnahme laufen lassen.
4. Erfolgskontrolle durch den Nutzer.
5. Wiederholen (falls notwendig).

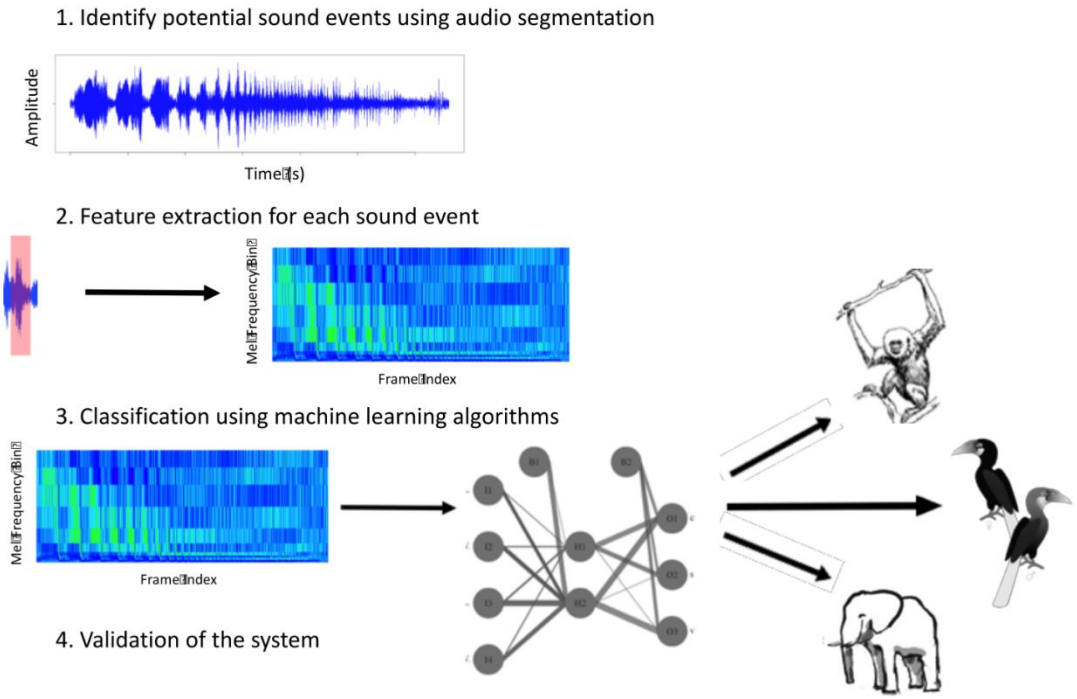


Abbildung 10: Schematischer Arbeitsweise des R-Paketes gibbonR. Quelle: Clink & Klinck, 2019

Hier habe ich ebenfalls ein auf den Birkhahn massgeschneidertes R-Skript (Anhang A) verfasst, das es mir ermöglicht Balzgeräusche automatisch erkennen zu lassen. Da ein machine learning Algorithmus einen Trainingsdatensatz mit vom Nutzer klassifizierten Daten benötigt, habe ich vorgängig 1000-mal ein Kullern und 1000-mal ein Zischen von verschiedenen Akustikaufnahmen extrahiert. Dafür habe ich unterschiedliche Songmeteraufnahmen von der Hochalp (AR) 2016 und vom Kanton Luzern 2018 mit gibbonR geladen. Anschliessend lässt sich ein Frequenzbereich vorgeben, welcher abgesucht wird und

potenzielle gesuchte Geräusche ermittelt und als WAV-Datei in einem Ordner speichert. Die verwendeten Frequenzbereiche sind vom Kullern bzw. Zischen übernommen (Kapitel 2.2.1 & 2.2.2). Danach habe ich die potenziellen Geräusche von Hand in Kullern bzw. Zischen klassifiziert und als WAV-Datei in einem Ordner gespeichert. Die mit dem R-Paket gibbonR klassifizierten Trainingsdateien stammen aus Akustikaufnahmen, die anschliessend nicht als Testdateien (Testkurz, Testdatei, Testlang) verwendet werden.

GibbonR besitzt die Funktion, aus einer Liste mit WAV-Dateien die Mel-Frequenz-Cepstrum-Koeffizienten (MFCC) zu berechnen. Diese Koeffizienten verwenden die machine learning Algorithmen als Ersatz für ein Spektrogramm. Der ganze Ablauf wird pro Balzgeräusch einmal durchgeführt, sodass ich am Ende zwei CSV-Dateien (Abbildung 11) mit den MFCC der Balzgeräusche Kullern und Zischen erhalte. Diese Form erleichtert eine Weitergabe an weitere Personen.

	A	B	C	D	E	F	G
1	class	1	2	3	4	5	6
2	Kullern	-0,234802616	-10,18091638	-3,716773388	6,209345909	5,576613364	-5,232216581
3	Kullern	-1,161968980	-1,669917603	2,285782033	0,190287846	0,524676899	-0,819791169
4	Kullern	0,306830488	-1,949322894	-2,186525458	2,384336378	4,108017270	0,395017020
5	Kullern	-0,831847333	-4,951642005	0,426161167	4,463138722	-1,306692647	-5,762183743
6	Kullern	-1,047189091	-1,580465945	-0,825570128	2,506324464	-0,097134468	-2,354418587
7	Kullern	-2,069451022	-10,50075419	3,046171788	11,63116159	-6,819554064	-6,611159850

Abbildung 11: Ausschnitt aus einer CSV-Datei mit enthaltenen MFCC für das Balzgeräusch Kullern. Angezeigt werden nur die ersten 6 von 178 Spalten. Quelle: Burkhalter, 2019.

Für gibbonR stellte sich nach diversen Tests mit den Algorithmen heraus, dass die Support Vector Machines (SVM) die besten Ergebnisse liefert. Der dazugehörige biplot (Abbildung 12) vermittelt auf den ersten Blick den Eindruck, als wären die beiden Balzgeräusche stark überlappend und somit für eine automatische Erkennung ungeeignet.

Abbildung 12: biplot des Trainingsdatensatzes, der 1000 Kullern und 100 Zischen enthält und wie stark sich diese überlagern. Quelle: Burkhalter, 2019

Die statistische Überprüfung der Genauigkeit des Algorithmus zeigte jedoch, dass dieser mit einer Wahrscheinlichkeit von 99.56% Zischen und Kullern richtig voneinander unterscheiden kann, während Gaussian Mixture Models (GMM) und Artificial Neural Networks (NNET) Genauigkeiten von 94.8% bzw. 90.6% besitzen. Somit wird der Algorithmus Support Vector Machines (SVM) für die Auswertung angewendet.

4.3.2.1 AUSGABE DER ERGEBNISSE

GibbonR führt die Ergebnisse einer automatischen Auswertung in Form eines dataframes in R auf, welcher in einer gewünschten Form exportiert werden kann. Geeignet sind CSV-Dateien.

	A	B	C	D	E
1	timing.df.detect.num	timing.df.signal	timing.df.start.time	timing.df.end.time	timing.df.signal.probability
2	21	Kullern	4159.628325	4166.796346	0.975215708
3	34	Kullern	4257.420615	4266.316641	0.997456658
4	48	Kullern	4271.244656	4277.068673	0.900202875
5	72	Kullern	4401.741042	4410.189067	0.915159386
6	89	Kullern	4588.557596	4593.997612	0.916941294

Abbildung 13: Ausschnitt aus einer CSV-Datei mit Ergebnissen aus einer Auswertung mit dem R-Paket gibbonR. Spalte "timing.df.detect.num" = Nummer des gefundenen Treffers in aufsteigender Reihenfolge; Spalte «timing.df.signal» = welches Geräusch gefunden wurde; Spalte "timing.df.start.time" = in der wievielten Sekunde der untersuchten Aufnahme das Geräusch beginnt aufzutreten; Spalte "timing.df.end.time" = in der wievielten Sekunde der untersuchten Aufnahme das Geräusch aufhört aufzutreten; Spalte «timing.df.signal.probability» = Wahrscheinlichkeit, dass das dieser Treffer das gesuchte Geräusch ist, in Prozent (1=100%). Quelle: Burkhalter, 2019

Es werden fünf Spalten ausgegeben. Spalte "timing.df.detect.num" gibt die Nummer des gefundenen Treffers in aufsteigender Reihenfolge an. Die Spalte «timing.df.signal» bezeichnet das Geräusch welches gefunden wurde. Spalte "timing.df.start.time" gibt an, in der wievielten Sekunde der untersuchten Aufnahme das Geräusch beginnt. Spalte "timing.df.end.time" gibt an, in der wievielten Sekunde der untersuchten Aufnahme das Geräusch aufhört. Die Spalte «timing.df.signal.probability» beschreibt die Wahrscheinlichkeit, dass der gefundene Treffer das gesuchte Geräusch ist, in Prozent (1=100%). Ein Schwellenwert muss in R vordefiniert werden und nur wenn dieser erreicht oder überschritten ist, wird der Treffer in dem dataframe aufgelistet. Für diese Treffer ist es möglich einen Speicherort auszuwählen, an dem die gefundenen Geräusche als WAV-Datei gespeichert werden. Anschliessend ist es mit geringem Aufwand möglich festzustellen, ob die Ergebnisse die gewünschten Geräusche enthalten. Die exportierten WAV-Dateien können dann beispielsweise iterativ als Vorlage verwendet werden.

4.3.3 VERGLEICH FOTOFALLEN UND AKUSTIKAUFNAHMEN

Für das Untersuchen der Akustikaufnahmen verwende ich die Methode von gibbonR mit dem R-Skript «AutomatischeErkennung.R». Dies aufgrund der Anwendbarkeit der Methode auf verschiedene Akustikaufnahmen, sowie der geringen Rechenzeit und Ausgabe der Ergebnisse. Diese werden wie in Kapitel 4.3.2 beschriebenen Weise gespeichert.

5 ERGEBNISSE

Der Zeitaufwand für die Einarbeitung in das R-Paket seewave sowie das Verfassen der R-Skripts beanspruchte einmalig ca. 170 Arbeitsstunden. Für das R-Paket gibbonR und dessen R-Skripte wurden einmalig ca. 120 Arbeitsstunden aufgewendet. Der hohe Aufwand ist vor allem der Einarbeitung in die R-Pakete und den dazugehörigen zugehörigen Algorithmen geschuldet.

Die manuelle Auswertung der drei Testdateien ist in Tabelle 2 zusammengefasst. Am meisten Ereignisse besitzt die längste Datei «Testlang» mit gesamthaft 543 Ereignissen, gefolgt von der «Testkurz» mit 51 und der «Testdatei» mit 48 Ereignissen.

Tabelle 2: Ergebnisse der manuellen Auswertung der drei Testdateien. Quelle: Burkhalter, 2019.

	Testkurz	Testdatei	Testlang
Länge [s]	280	660	7200
Kullern [Anzahl]	14	30	195
Zischen [Anzahl]	37	18	348

5.1 ZEITAUFWAND

Die Zeit für die Auswertung der drei Testdateien unterschied sich, relativ zur Gesamtlänge der Dateien gesehen, nicht. Wenig Zeit beanspruchte die automatische Auswertung im Gegensatz zur Manuellen. GibbonR benötigte für «Testlang» mit 13 min, ca. 14-mal weniger Zeit als die manuelle Auswertung (Tabelle 3). Für «Testkurz» bzw. «Testdatei» beanspruchte gibbonR 7-mal bzw. 8-mal weniger Zeit als die manuelle Auswertung. Die Unterschiede zwischen den beiden R-Pakete sind kleiner. GibbonR ist hier ca. 2-mal schneller als CC und Bin Seewave (Tabelle 3).

Tabelle 3: Zeitaufwand in Minuten für die Auswertung der Testdateien. Quelle: Burkhalter, 2019.

Zeitaufwand [min]	Testkurz	Testdatei	Testlang
manuell	7	12	180
gibbonR	1	1.5	13
CC Seewave	3	5	29
Bin Seewave	2	4	26

5.2 PERFORMANCE DER R-PAKETE

Der Übersichtsplot der Testdatei «Testkurz» visualisiert, dass das Skript mit der Cross Korrelation (CC Seewave) am wenigsten Ereignisse erkannte (Abbildung 14). 5 von 14 Kullern bzw. 11 von 37 Zischen wurden von CC Seewave entdeckt (Tabelle 4). GibbonR und Bin Seewave gleichen auf den ersten Blick den Ergebnissen der manuellen Auswertung (Abbildung 14).

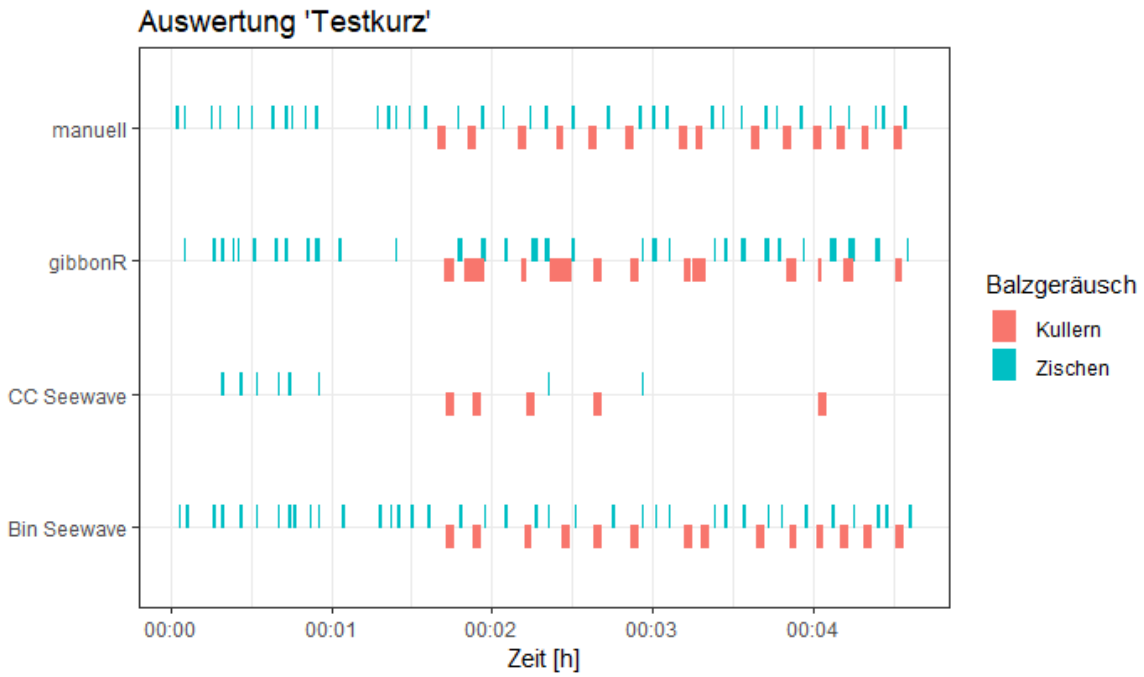


Abbildung 14: Auswertung der Testdatei "Testkurz". Aufgeteilt in die beiden Balzgeräusche Kullern und Zischen. Darstellung in einem ggplot. Quelle: Burkhalter, 2019

Dieser Eindruck wird bei Betrachtung der Anzahl erkannten Ereignisse bestätigt (Tabelle 4). Bin Seewave erkannte alle gesuchten Geräusche der Datei, 37 von 37 Zischen und 14 von 14 Kullern. Ein Zischen wurde kurz nach der ersten Minute zusätzlich als Balzgeräusch markiert (Abbildung 14). Was einer Fehlerquote von 3% entspricht.

Tabelle 4: Anzahl gefundener Ereignisse der Testdatei "Testkurz". Quelle: Burkhalter, 2019.

«Testkurz»	manuell	gibbonR	CC Seewave	Bin Seewave
Kullern [Anzahl]	14	12	5	14
Zischen [Anzahl]	37	29	11	37
Gefundene Geräusche [%]	Kullern 100 Zischen 100	Kullern 85.7 Zischen 78.4	Kullern 35.7 Zischen 29.7	Kullern 100 Zischen 100

Der ggplot (Abbildung 15) der Auswertung für die Testdatei «Testdatei» gleicht jenem der Testdatei «Testkurz». CC Seewave fand lediglich 7 von 30 Kullern. Bin Seewave und gibbonR dagegen lieferten mehr Ergebnisse.

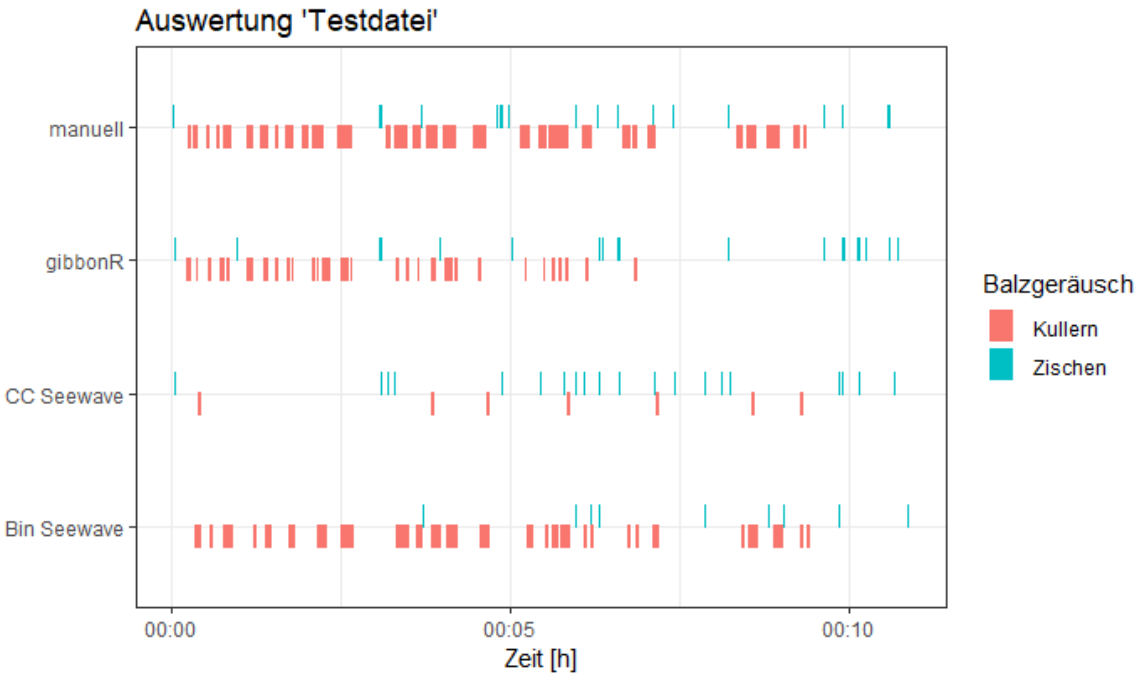


Abbildung 15: Auswertung der Testdatei "Testdatei". Aufgeteilt in die beiden Balzgeräusche Kullern und Zischen. Darstellung in einem ggplot. Quelle: Burkhalter, 2019

Bin Seewave fand 28 der 30 gesuchten Kullern und 4 von 18 Zischen. 9 Zischen wurden mit Bin Seewave gefunden, 5 davon wurden falsch markiert, was einer Fehlerquote von 55% entspricht. GibbonR markiert 25 von 28 Kullern und 9 von 18 Zischen (Tabelle 5). Gesamthaft entdeckte gibbonR 20 Zischen, wobei 11 davon fälschlicherweise als Zischen markiert wurden, was ebenfalls einer Fehlerquote von 55% entspricht.

Tabelle 5: Anzahl gefundener Ereignisse der Testdatei "Testdatei". Quelle: Burkhalter, 2019.

«Testdatei»	manuell	gibbonR	CC Seewave	Bin Seewave
Kullern [Anzahl]	30	24	7	28
Zischen [Anzahl]	18	9	12	4
Gefundene Geräusche [%]	Kullern 100 Zischen 100	Kullern 80 Zischen 44.4	Kullern 23.3 Zischen 61.1	Kullern 93.3 Zischen 22.2

Ein erster Blick auf den ggplot der Testdatei «Testlang» verrät, dass CC Seewave erneut nicht alle Ereignisse gefunden und markiert hat (Abbildung 16). GibbonR und Bin Seewave markieren fälschlicherweise Balzgeräusche zu Beginn der Testdatei «Testlang».

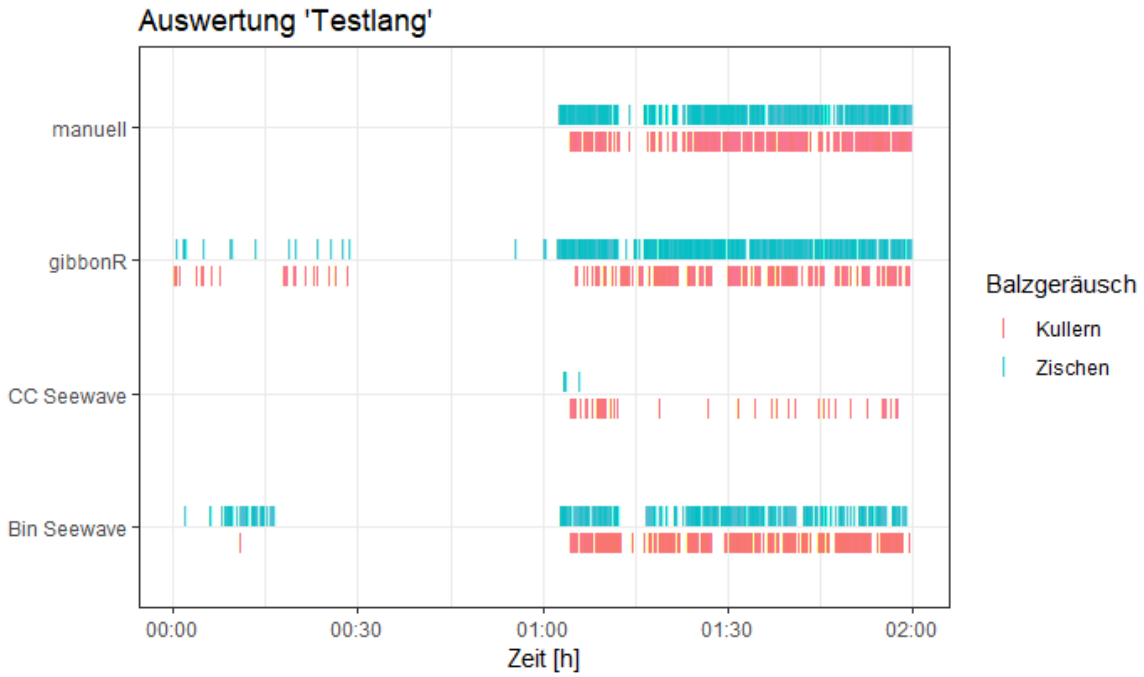


Abbildung 16: Auswertung der Testdatei "Testlang". Aufgeteilt in die beiden Balzgeräusche Kullern und Zischen. Darstellung in einem ggplot. Quelle: Burkhalter, 2019

Die Auswertung der Testdatei «Testlang» ist etwas komplexer. Die Anzahl gefundener Geräusche lässt darauf schliessen, dass alle Methoden alle Balzgeräusche entdecken konnten (Tabelle 6). Jedoch werden hier die falschen Detektionen (false positives) vernachlässigt. Zudem zeigt sich ein weiteres Problem. Bei der manuellen Auswertung wird die Länge der Balzgeräusche als ein Ereignis zusammengefasst. Das bedeutet, dass ein Kullern, welches 9 s anhält, in der manuellen Auswertung ein Ereignis bezeichnet. Dieses 9 s lange Kullern besteht aber aus 3 aufeinanderfolgenden Kullern, welche theoretisch mit den Methoden «gibbonR», «CC Seewave» und «Bin Seewave» als 3 Ereignisse bezeichnet werden können.

Tabelle 6: Anzahl gefundener Ereignisse in der Testdatei "Testlang" mit nicht relativierten Zahlen. Quelle: Burkhalter, 2019.

«Testlang»	manuell	gibbonR	CC Seewave	Bin Seewave
Kullern [Anzahl]	195	274	214	412
Zischen [Anzahl]	348	388	348	361

Um die Zahlen zu relativieren und in einen Kontext zu stellen, wurde gezählt, wie oft ein manuell gefundenes Ereignis sich mit einem automatischen überschneidet. So wird verhindert, dass einige automatisch gefundene Ereignisse mehrfach gezählt werden.

Die Anzahl überschneidender Ereignisse, also korrekter Treffer (true positives), wurde anschliessend mit der Gesamtzahl gefundener Ereignisse aus Tabelle 7 verrechnet, um die Trefferquote zu ermitteln.

Tabelle 7: Anzahl gefundener Ereignisse der Testdatei "Testlang". Quelle: Burkhalter, 2019.

«Testlang»	manuell	gibbonR	CC Seewave	Bin Seewave
Kullern [Anzahl]	195	189	57	366
Zischen [Anzahl]	348	332	5	210
Gefundene Geräusche [%]	Kullern 100 Zischen 100	Kullern 68.7 Zischen 85.5	Kullern 26.6 Zischen 1.4	Kullern 88.8 Zischen 58.1

5.2.1 ZUSAMMENFASSUNG DER ERGEBNISSE

Abschliessend können die Ergebnisse wie folgt zusammengefasst werden: Die Prozentzahlen sind die Trefferquoten der Methoden gemittelt über die drei Testdateien. «Bin Seewave» und «gibbonR» finden mit Trefferquoten von 60-94% die meisten gesuchten Balzgeräusche. (Tabelle 8) Deutlich weniger mit 19-30% findet «CC Seewave».

Tabelle 8: Gesamtprozentzahl der gefundenen Geräusche, kumuliert über alle Testdateien. Quelle: Burkhalter, 2019.

	manuell	gibbonR	CC Seewave	Bin Seewave
Kullern [%]	100	78.1	19.6	94
Zischen [%]	100	69.4	30.7	60.1
Aufwand [min]	199	15.5	37	32

5.3 VERGLEICH FOTOFALLEN UND AKUSTIKAUFNAHMEN

Die Ergebnisse zeigen deutlich, dass der Plotwatcher weniger Aktivität feststellen konnte, während die automatische Auswertung der Akustikaufnahmen mehr Balzgeräusche ermittelte (Abbildung 17). «Plotwatcher manuell» repräsentiert fotografierte Birkhühner. Bei «Songmeter automatisch» werden Kullern und Zischen aus Darstellungsgründen als ein Geräusch zusammengefasst.

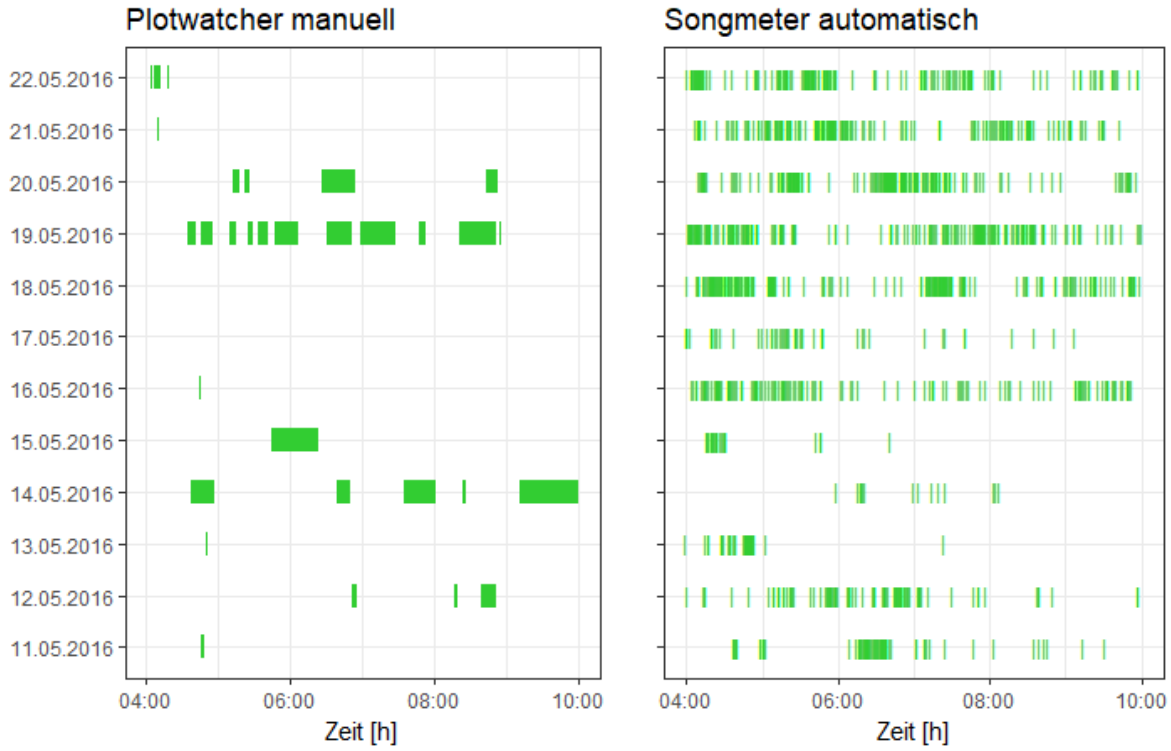


Abbildung 17: Vergleich zwischen Auswertung visueller (links) und akustischer (rechts) Aufnahmen. Grün = Balzgeräusch/Birkhuhn. Quelle: Burkhalter, 2019

«Songmeter automatisch» stellt nicht zwingend alle Balzgeräusche dar. Zudem ist es möglich, dass einige Geräusche fälschlicherweise als Balzgeräusche klassifiziert wurden. Dies aufgrund der ermittelten Trefferquoten, die in Kapitel 5.2.1 beschrieben wurden.

6 DISKUSSION

6.1 VERGLEICH SEEWAVE UND GIBBONR

Seewave

Zwischen dem R-Skript CC Seewave und Bin Seewave sind statistisch grosse Unterschiede in der Trefferquote feststellbar, obwohl sie auf dem gleichen R-Paket basieren. Die Gründe für diese Unterschiede liegen womöglich in der Methode, die verwendet wurde. CC Seewave basiert streng auf dem R-Paket und Buch von Sueur (2018). Im Buch ist die Methode auf andere Tiere ausgerichtet und dies macht einen Transfer zu anderen Tierarten, wie dem Birkhuhn, grundsätzlich schwierig. Zudem ist die relativ lange Nachbearbeitungszeit fehleranfällig. Nach dem Heraussuchen potenzieller Balzgeräusche aus einer Akustikaufnahme, muss der geeignete Schwellenwert errechnet werden, also die Schwelle, ab wieviel prozentiger Übereinstimmung mit der Referenz, ein gefundenes Geräusch als solches gekennzeichnet wird. Doch diese Berechnung, die vom Paket seewave vorgegeben wird, führt offensichtlich zu einem zu hohen Schwellenwert (Abbildung 18).

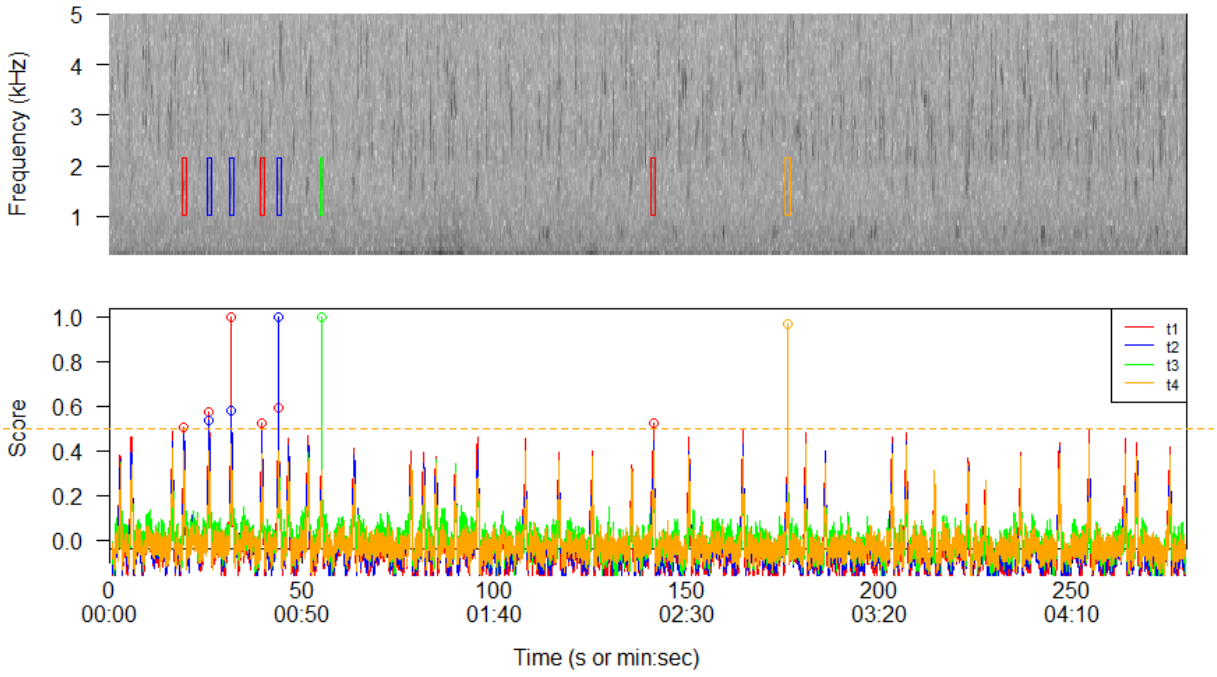


Abbildung 18: Spektrogramm der ausgewerteten Testdatei "Testkurz" mit dem R-Skript CC Seewave. Gut zu sehen, sind viele Spitzen, die knapp nicht als Balzgeräusch bezeichnet wurden. Grün, blau, rot, orange = templates. Quelle: Burkhalter, 2019

Es ist gut ersichtlich, dass viele potenzielle Balzgeräusche, in Abbildung 18 als Spitzen dargestellt, knapp unter dem nach Anleitung errechneten Threshold liegen. Vier gefundene Geräusche haben jedoch sehr hohe Überstimmungen, welche aber lediglich die Geräusche, die als Vorlage (templates) verwendet wurden, repräsentieren. Grundsätzlich sagen diese vier hohen Übereinstimmungen nur aus, dass das gesuchte Geräusch sich stark untereinander unterscheidet. Theoretisch wäre es möglich,

den Threshold manuell nach unten zu korrigieren, was allerdings ein weiteres Problem nach sich zieht. Gemeint ist die Minimierung der «false positive» und «false negative events». Also Ereignisse die fälschlicherweise als Übereinstimmung oder keine Übereinstimmung markiert wurden. Mit Hilfe von seewave lässt sich der Threshold berechnen, an dem die «false positive» und «false negative events» bei 0% liegen. Wird nun der Schwellenwert nach unten korrigiert, lassen sich möglichweise mehr Balzgeräusche entdecken, doch die Rate von «false positive» und «false negative» würde damit erhöht und die Auswertung verliert ihre statistische Aussagekraft.

Geringen Handlungsbedarf besitzt Bin Seewave, welches gute Ergebnisse erzielen konnte. Auch hier wurde strikt nach dem Paket seewave und nach Sueur, 2018 gearbeitet. Der errechnete Threshold eignete sich hier deutlich besser, was in den Ergebnissen (Kapitel 5) ersichtlich ist.

Grundsätzlich sind die Ergebnisse von Bin Seewave zufriedenstellend. CC Seewave, welches vom R-Paket seewave als primäre und besser funktionierende Funktion dargestellt wird, liefert unbefriedigende Ergebnisse. Auch ist bei seewave die Anwendbarkeit schwierig und langwierig. Die Nachbearbeitung ist fehleranfällig und besitzt wenig Spielraum für Änderungen, aber auch die Vorarbeit ist aufwendig. Da die Vorlagen aus der Akustikaufnahme stammen sollte, die anschliessend untersucht wird, müssen vorab in einem Spektrogramm Stellen gesucht werden, die als Vorlage verwendet werden können. R ist dazu ungeeignet, da es sehr viel Zeit beansprucht, um ein Spektrogramm darzustellen (mindestens 30 min Rechenzeit für 10 min Spektrogramme). Zudem ist das paketeigene Werkzeug von seewave zur Markierung von Vorlagen in einem Spektrogramm sehr grob und ungeeignet für Geräusche mit schmalem Frequenzband. Das wiederum bedeutet, dass man auf Alternativprogramme zurückgreifen muss. Sonic Visualiser ist ein geeignetes Programm für die Betrachtung von Spektrogrammen. Damit können Stellen, die sich als Vorlage eignen, gefunden und mit einer Zeitreferenz notiert werden. Danach müssen diese Werte in R übertragen werden, welches wiederum auf die Akustikaufnahme zugreift und die Stellen heraussucht und als Vorlage markiert. Dieser Aufwand muss für jede Aufnahme einzeln wiederholt werden, um die maximale Leistung bei der automatischen Erkennung zu erreichen (Hafner, persönliche Mitteilung), was dazu führt, dass der manuelle Aufwand langfristig kaum abnimmt. Auch schliesst es eine Erhöhung der Anzahl der Vorlagen teilweise aus, da der Arbeitsaufwand pro Akustikaufnahme erheblich gesteigert werden müsste. Versuche im Rahmen dieser Arbeit mit zehn Vorlagen führten zu keinerlei besseren Ergebnissen, nur zu deutlich längeren Rechenzeiten.

Ausserdem leidet die Transportierbarkeit der Methode. Dadurch, dass R immer eine Akustikaufnahme als Referenz benötigt, um Vorlagen einzulesen, muss bei einer Weitergabe an andere Personen auch immer die zugehörige Akustikaufnahme mitgegeben werden. Dies ganz unabhängig davon, ob Vorlagen von einen Akustikaufnahmen auf andere anwendbar wären.

Trotzdem sollte diese Methode weiterverwendet werden. Für kürzere Aufnahmen, die homogene Geräusche enthalten, eignet sich diese Methode. Bei einer 30-sekündigen Aufnahme eines Glanzschenkel-Baumsteigers (*Allobates femoralis*), ein Frosch, der in Südamerika vorkommt, konnten mit dieser Methode 27 von 28 Geräuschen mit Übereinstimmungen von über 65% gefunden werden (Sueur, 2018).

GibbonR

Mit den 1000 Zischen und 1000 Kullern, die als Trainingsdaten verwendet wurden, sind zufriedenstellende Ergebnisse erzielt worden. Die grosse Frage, die sich stellt, ist, wie viele dieser Trainingsdaten benötigt werden, um 100% aller gesuchten Geräusche mit dem Detektor zu finden. Auch die Entwickler des R-Paketes können dazu keine Antwort liefern (Clink & Klinck, 2019). Die Zahl der Trainingsdaten zu erhöhen und zu untersuchen, welche Ergebnisse damit erreicht werden können, sollte Gegenstand weiterer Untersuchungen sein.

GibbonR ist einfach anwendbar und benötigt keine fehleranfällige Vor- und Nacharbeit. Dennoch ist ein einmaliger hoher Zeitaufwand nötig, um Trainingsdaten zu klassifizieren. Da die in einer CSV-Datei als MFCC gespeicherten Trainingsdaten auch anwendbar auf andere Akustikdateien sind und nicht zwingend aus derselben Aufnahme stammen müssen, ist über einen längeren Zeitraum gesehen der Zeitaufwand für eine Auswertung gering. Durch die Umwandlung in MFCC senkt sich das Datenvolumen erheblich. Die 1000 Kullern benötigen in der Form der MFCC ca. 6 MB und erfordern, anders als bei seewave, auch keine Audiodateien als Referenz, was wiederum die Transportierbarkeit erhöht und die Weitergabe an andere Personen erleichtert.

Die Ausgabe der Ergebnisse einer Auswertung als WAV-Dateien vereinfacht das Erweitern der Trainingsdaten. Die WAV-Dateien können wieder in R eingelesen werden. Das Ganze muss als iterativer Prozess angesehen werden, der die Trainingsdaten ständig verbessert und erweitert. Durch eine höhere Anzahl an Trainingsdaten und der damit verbundenen hohen Variabilität der Trainingsdaten ist es möglich, variables Kullern zu entdecken. Das macht dieses R-Paket auch anderweitig anwendbar. Es könnten diverse Geräusche von Tierarten in R eingelesen werden und die MFCC berechnet werden. Das bedeutet es würde eine CSV-Datei mit MFCC existieren. Basierend auf dieser einen CSV-Datei könnten dann verschiedene Akustikaufnahmen nach verschiedenen Geräuschen abgesucht werden.

Während den Versuchen trat ein Fehler wiederholt auf. Oft wurden Wind und Kuhglocken fälschlicherweise als Kullern oder Zischen markiert. Und das mit Wahrscheinlichkeiten von teils über 99%. Bei der Hörprobe und bei Betrachtung des Spektrogramms konnte festgestellt werden, dass der Wind grosse Frequenzbänder «blockiert» und für den Hörer sehr dem Kullern der Birkhähne gleicht. Die Kuhglocken

decken unglücklicherweise oft ein ähnliches Frequenzband ab wie das Kullern. Diese Fehlerkennungen könnten durch die Erhöhung der Trainingsdaten vermindert werden. Zudem sollten noch Trainingsdatensätzen erstellt werden, der Hintergrundgeräusche enthalten. Wird der Algorithmus mit diesen Trainingsdatensätzen gespeist, wäre eine Unterscheidung zwischen Balzgeräusch und Hintergrundgeräusch besser möglich.

GibbonR befindet sich zurzeit in der Anfangsphase. Das Paket wurde erst Ende Mai 2019 auf GitHub zur Verfügung gestellt und ist daher noch nicht ausgereift. Erschwerend kommt hinzu, dass aufgrund des jungen R-Paketes, noch sehr wenig Hilfestellungen auf Foren im Internet vorhanden sind. Laut den Entwicklern ist das Paket auf Erfahrungen und Rückmeldungen der Anwender angewiesen, um gibbonR noch weiter zu verbessern. Das Paket wurde auf den Müller-Gibbon (*Hylobates muelleri*) massgeschneidert, einen auf Borneo endemisch vorkommenden Primaten (Clink & Klinck, 2019). Um dieses Paket breiter anwendbar zu machen, ist es wichtig zu untersuchen, welche Ergebnisse gibbonR auf andere Tierarten erzielt.

Zusammenfassung

Seewave ist ein hervorragendes Paket zur Bearbeitung und Untersuchung von Bioakustik in R. Die automatische Erkennung umfasst nur einen kleinen Teil, des sonst äusserst ausgereiften Paketes. Die lange und fehleranfällige Nachbearbeitungszeit, die schlechte Transportierbarkeit und die teils schlechten Ergebnisse mit CC Seewave (Tabelle 9), machen seewave für Anwendung der automatisierten Erkennung der Balzaktivität der Birkhähne nicht brauchbar.

Tabelle 9: Zusammenfassung und Vergleich der beiden R-Pakete. Rot = schlecht, orange = gut, grün = sehr gut. Quelle: Burkhalter, 2019.

Kriterien	gibbonR	CC Seewave	Bin Seewave
Trefferquoten bei der Auswertung akustischer Aufnahmen	gut	schlecht	gut
Transportierbarkeit	sehr gut	schlecht	schlecht
Benutzerfreundlichkeit	sehr gut	schlecht	schlecht
Zeitaufwand	sehr gut	gut	gut
Potential ausgeschöpft?	nein	ja	ja

GibbonR ist benutzerfreundlich und logisch aufgebaut. Es bedarf keiner Nachbearbeitung, Trainingsdaten können unkompliziert via CSV-Datei weitergegeben werden und auch der Zeitaufwand ist sehr gering. Die Trefferquoten sind bei Bin Seewave und gibbonR momentan ähnlich. Doch hier darf nicht ausser Acht gelassen werden, dass seewave, anders als gibbonR, sein Potential im Bezug auf die automatische Erkennung bereits ausgeschöpft hat (Tabelle 9). GibbonR liefert bereits im unausgereiften

Zustand ähnlich gute Ergebnisse wie Bin Seewave, weshalb zukünftig der Fokus klar auf das Paket gibbonR gelegt werden sollte.

6.2 VERGLEICH MANUELLE UND AUTOMATISCHE AUSWERTUNG

Die automatische Auswertung liefert bereits jetzt zufriedenstellende Ergebnisse. Einzig die Trefferquoten müssen verbessert werden, um wirklich alle vorkommenden Geräusche in einer Akustikaufnahme erkennen zu können. Mit der manuellen Auswertung ist dies bereits jetzt möglich. Diese hat aber zwei essenzielle Nachteile. Zum ersten der Zeitaufwand, welches deutlich höher ist als jener der automatischen Auswertung (Tabelle 10). Zum anderen ist die manuelle Auswertung der Subjektivität der auswertenden Person unterlegen. So steigt die Fehlerquote mit der Zahl der Personen, die an der Auswertung beteiligt sind. Dieser Faktor ist bei der automatischen Methode gänzlich verschwunden und die Ergebnisse sind ohne grossen Aufwand reproduzierbar.

Tabelle 10: Vergleich des Zeitaufwandes der verschiedenen Methoden. Rot = schlecht, orange = gut, grün = sehr gut. Quelle: Burkhalter, 2019.

Zeitaufwand [min]	Testkurz (280s)	Testdatei (660s)	Testlang (7200s)	Spektrogramm
manuell	7	12	180	Nein
gibbonR	1	1.5	13	Ja
CC Seewave	3	5	29	Ja
Bin Seewave	2	4	26	Ja

Die automatische Auswertung bietet zudem die Möglichkeit die Ergebnisse in einem Spektrogramm darstellen zu lassen. Dies ist besonders bei einer Auswertung einer langen Akustikdatei hilfreich. So kann bereits eine kleine Übersicht über die Ergebnisse gewonnen werden, ganze ohne Umweg über CSV-Dateien und erneuter Bearbeitung in R, wie dies bei der manuellen Auswertung nötig wäre.

Aufgrund dieser Ergebnisse ist die automatische Methode klar der automatischen vorzuziehen.

6.3 VERGLEICH FOTOFALLEN UND AKUSTIKAUFNAHMEN

Interessanterweise sind die Ergebnisse sehr unterschiedlich (Abbildung 17). Natürlich darf nicht vergessen werden, dass der Detektor mit dem Paket gibbonR aus den in Kapitel 6.1 genannten Gründen, noch nicht alle gesuchten Geräusche findet oder anders formuliert noch zu viele falsche Geräusche als Ereignisse markiert. Die Struktur, die bei dem Plot von «Songmeter automatisch» erkennbar ist, lässt diesen Schluss zu. Immer wieder sind einzelne kurze Balzaktivitäten dargestellt, was dem Balzverhalten

der Birkhühner widerspricht. Trotzdem lässt sich durch eine Anhäufung der Balzaktivität erahnen, wann gebalzt wurde. Interessant ist der 18.05.2016. An diesem Tag konnten bei «Songmeter automatisch» Balzaktivität der Birkhähne über den ganzen Morgen festgestellt werden (Abbildung 17), während bei «Plotwatcher manuell» kein Birkhahn fotografiert werden konnte. Ein gegenteiliger Fall wird hier am 14.05.2016 offensichtlich. Während die Plotwatcher die Anwesenheit eines Birkhahns über einen längeren Zeitraum feststellen konnten, weisen die Songmeterdaten auf eine geringe Aktivität hin. Dieses Schema setzt sich gewissermassen über alle untersuchten Tage fort (Abbildung 17). Der Hauptgrund dafür liegt womöglich in der Erfassungsmethode. Da ein Songmeter einen grösseren Bereich abdeckt, kann nicht ausgeschlossen werden, dass andere, auf den Fotofallen nicht sichtbare Birkhähne, durch den akustischen Detektor erfasst wurden. Auch zeigte eine zufällige Hörprobe der gefundenen Geräusche, dass auch sehr leise und nur schlecht wahrnehmbare Balzgeräusche mit dem Detektor erfasst wurden. Dies deutet darauf hin, dass auch weit entfernte Birkhähne erfasst wurden. Um genau die Birkhähne akustisch zu erfassen, die auch visuell erfasst wurden, würde sich eine Ortung der Birkhähne mittels Triangulation anbieten. So könnten Birkhähne, die sich nicht im Bereich der Fotofallen aufhalten, ausgeschlossen werden.

Ein ähnliches Werkzeug enthält gibbonR bereits jetzt. Es besteht die Möglichkeit einen call-density-plot zu erstellen. Also eine Grafik, die anzeigt, bei welchen Aufnahmegeräten die meisten Balzgeräusche aufgenommen wurden (Abbildung 19). So könnten beliebte Balzplätze der Birkhähne besser lokalisiert und das Monitoring geografisch genauer abgestimmt werden.

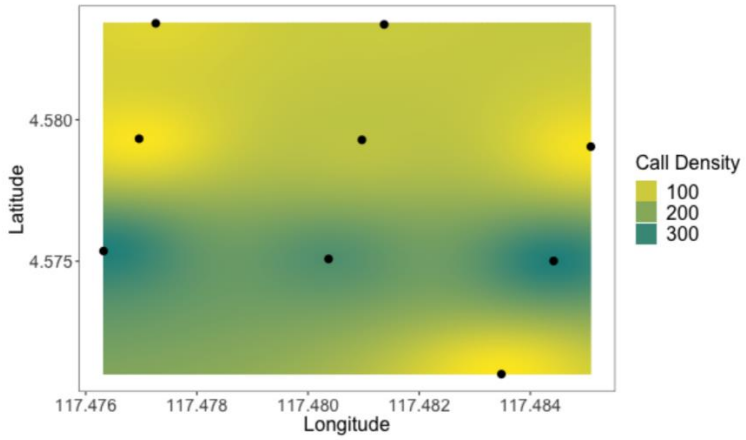


Abbildung 19: Call-densitiy-plot von der Untersuchung des Müller-Gibbon. Quelle: Clink & Klinck, 2019.

Somit ist es wichtig, das Monitoring der Birkhähne sowohl visuell als auch akustisch durchzuführen. Denn ein balzender Birkhahn ist nicht gleich ein sichtbarer Birkhahn. Und ein sichtbarer Birkhahn ist nicht gleich ein balzender Birkhahn. Das bedeutet, die visuelle und akustische Methode sollten komplementär eingesetzt werden.

6.4 SCHLUSSFOLGERUNG UND AUSBLICK

Mit dieser Arbeit konnte gezeigt werden, dass die automatische Auswertung von Akustikaufnahmen basierend auf neuronalen Netzwerken, bereits möglich ist. Auch sind die benötigten Algorithmen frei verfügbar und von jedermann einsetzbar. Was fehlt, ist der Feinschliff und die exakte Abstimmung auf die Birkhühner. Dies sollte in weiteren Untersuchungen unternommen werden.

Die beiden untersuchten R-Pakete, gibbonR und seewave, sind sehr unterschiedlich in deren Ansätze der automatischen Erkennung und lieferten deshalb auch unterschiedliche Ergebnisse. Für kurze Akustikaufnahmen, bei denen der Aufwand einen Algorithmus mit klassifizierten Daten zu trainieren zu hoch ist, eignet sich das Paket seewave besser, da der Zeitaufwand geringer ist. Bei grossen Datenmengen, wie beim passivem akustischen Monitoring der Birkhühner, eignet sich gibbonR deutlich besser, da der Zeitaufwand einmalig hoch ist, jedoch über einen längeren Zeitraum gesehen sehr gering ist, um eine lange Akustikaufnahme auszuwerten.

Bei gibbonR ist zum jetzigen Zeitpunkt nicht klar, wie sensitiv der Detektor Balzaktivität erfasst. Es ist gut möglich, dass einige der «false negative events» in Wahrheit sehr schwache Signale von Balzgeräuschen sind, die bei der manuellen Auswertung im Spektrogramm nicht sichtbar waren. Um genau zu untersuchen, was der Detektor von gibbonR erfasst und was nicht, könnte beispielsweise mit seewave weiter untersucht werden. «False negative events» könnten so genauer analysiert werden.

Eine Frage, die sich stellt, ist in wie weit Birkhühner erfasst werden müssen. Das akustische Monitoring der Birkhühner befasst sich zurzeit ausschliesslich mit den Birkhähnen. Birkhennen sind somit momentan nur visuell zu beobachten. In der Literatur ist beschrieben, dass auch die Birkhennen während der Balz Rufe erzeugen und verglichen zur Auerhenne auch deutlich ruffreudiger ist (Bergmann et al., 1990) und sich daher auch für ein akustisches Monitoring eignen würde. Um eine geschlechterumfassende Beobachtung der Birkhühner umzusetzen, sollte der Fokus auch die Birkhenne und ihre Rufe umfassen.

Die beiden beschriebenen R-Pakete bieten spannende und hilfreiche Möglichkeiten Akustikaufnahmen zu analysieren. Vor allem gibbonR bietet in der automatischen Erkennung neue Möglichkeiten. Und dies mit einem frei verfügbaren und modular aufgebauten Programm, welches sich in Form von R-Paketen ständig weiterentwickelt.

7 LITERATURVERZEICHNIS

Bergmann, H.-H., Marti, C., Müller, F., Vitovic, O., & Wiesner, J. (1990). *Die Birkhühner: Tetrao tetrix und T. mlokosiewiczi* (1. Aufl; S. Klaus, Hrsg.). Wittenburg Lutherstadt: A. Ziemsen.

Berwick, R. (2003). *An Idiot's guide to Support vector machines (SVMs)*.

Cai, J., Ee, D., Pham, B., Roe, P., & Zhang, J. (2007). Sensor network for the monitoring of ecosystem: Bird species recognition. *2007 3rd international conference on intelligent sensors, sensor networks and information*, 293–298. IEEE.

Ching, T., Zhu, X., & Garmire, L. X. (2018). Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data. *PLoS computational biology*, *14*(4), e1006076.

Chollet, F., & Allaire, J. J. (2018). *Deep learning with R*. Shelter Island, NY: Manning Publications Co.

Ciach, M. (2015). Rapid decline of an isolated population of the black grouse Tetrao tetrix: The crisis at the southern limit of the range. *European journal of wildlife research*, *61*(4), 623–627.

Clink, D., & Klinck, H. (2019). *GibbonR: An R package for classification, detection and visualization of acoustic signals using machine learning*.

Dörfler, M., Bammer, R., & Grill, T. (2017). Inside the spectrogram: Convolutional Neural Networks in audio processing. *2017 International Conference on Sampling Theory and Applications (SampTA)*, 152–155. IEEE.

Feldbusch, F. (1998). Geräuscherkennung mittels Neuronaler Netze. *Zeitschrift für Audiologie*, *1*(1998), 30–36.

Glutz von Blotzheim, U. N. (1994). *Handbuch der Vögel Mitteleuropas* (2. Aufl., Bd. 5). Wiesbaden: AULA.

Hafner, S. D., & Katz, J. (2018). *A short introduction to acoustic template matching with monitoR*. 20.

Heinicke, S., Kalan, A. K., Wagner, O. J. J., Mundry, R., Lukashevich, H., & Kühl, H. S. (2015). Assessing the performance of a semi-automated acoustic monitoring system for primates. *Methods in Ecology and Evolution*, *6*(7), 753–763. https://doi.org/10.1111/2041-210X.12384

Ihaka, R., & Gentleman, R. (1996). *R: a language for data analysis and graphics J Comput Graph Stat 5: 299–314*.

Keen, S. C., Shiu, Y., Wrege, P. H., & Rowland, E. D. (2017). Automated detection of low-frequency rumbles of forest elephants: A critical tool for their conservation. *The Journal of the Acoustical Society of America*, *141*(4), 2715–2726.

Li, J., Dai, W., Metze, F., Qu, S., & Das, S. (2017). A comparison of deep learning methods for environmental sound detection. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 126–130. IEEE.

Marques, T. A., Thomas, L., Martin, S. W., Mellinger, D. K., Ward, J. A., Moretti, D. J., … Tyack, P. L. (2013). Estimating animal population density using passive acoustics. *Biological Reviews*, *88*(2), 287–309.

Mielke, A., & Zuberbühler, K. (2013). A method for automated individual, species and call type recognition in free-ranging animals. *Animal behaviour*, *86*(2), 475–482.

Morfi, V., & Stowell, D. (2018). Deep learning for audio event detection and tagging on low-resource datasets. *Applied Sciences*, *8*(8), 1397.

Munger, L., Mellinger, D., Wiggins, S., Moore, S., & Hildebrand, J. (2005). Performance Of Spectrogram Cross-Correlation In Detecting Right Whale Calls In Long-Term Recordings From The Bering Sea. *Canadian Acoustics / Acoustique canadienne*, *33*(2).

Potapov, R. L., & Flint, V. E. (1989). Handbuch der Vögel der Sowjetunion. *Galliformes, Gruiformes. Ziemsen, Wittenberg Lutherstadt*.

Reynolds, D. (2015). Gaussian mixture models. *Encyclopedia of biometrics*, 827–832.

Spaar, R., Ayé, R., & Zbinden, N. (2012). *Elemente für Artenförderungsprogramme Vögel Schweiz – Update 2011. Koordinationsstelle des Rahmenprogramms «Artenförderung Vögel Schweiz»* (S. 89). Zürich und Sempach: Schweizer Vogelschutz SVS/BirdLife Schweiz und Schweizerische Vogelwarte.

Sueur, J. (2018). *Sound analysis and synthesis with R* (1st edition). New York, NY: Springer Berlin Heidelberg.

Sueur, J., Aubin, T., & Simonis, C. (2008). Seewave, a free modular tool for sound analysis and synthesis. *Bioacoustics*, *18*(2), 213–226.

Tóth, B. P., & Czeba, B. (2016). Convolutional Neural Networks for Large-Scale Bird Song Classification in Noisy Environment. *CLEF (Working Notes)*, 560–568.

Vogelwarte.ch. (2019). Abgerufen 28. Mai 2019, von Birkhuhn website: https://www.vogelwarte.ch/de/voegel/voegel-der-schweiz/birkhuhn

Wildlife Acoustics. (2018). *Song Meter SM3 Bioacoustics Recorder User Guide*. Abgerufen von https://www.wildlifeacoustics.com/images/documentation/SM4-USER-GUIDE-DE.pdf

VERZEICHNIS DER ABBILDUNGEN

VERZEICHNIS DER TABELLEN

ANHANGSVERZEICHNIS

Anhang A:

DarstellungEinzelnerErgebnisse.R

```
##### Darstellung einzelner Geräusche #####

library("seewave")
library("tuneR")
library("signal")
library("audio")
library("monitoR")
library("warbleR")
library("ggplot2")
library("dtw")
library("caTools")
library("phonTools")
library("tidyverse")
library("gibbonR")
library("viridis")
library("mixtools")
library("lubridate")



#Wave mit Referenzsound laden
kulzis <- readWave("sample/Form_2015-05-13_060600_CH.wav")

#Zuerst das kullern extrahieren
zischen <- extractWave(kulzis, from=31.0451, to=32.0668, xunit="time")
zischen1 <- fir(zischen, from = 1100, to = 1900, output = "Wave")
oscillo(zischen1)
```

```
#Farbenreihe für den plot
col <- c(rgb(1,0,0,0.5), rgb(0,0,1,0.5))

#Plot von Zischen
spectro(zischen1, ovlp=75, collevels=c(-exp(seq(log(20),  0, length=30))),
      palette=reverse.gray.colors.2, flog=TRUE,
      osc=TRUE, scalefontlab = 2, flim = c(0,6))

#gleiche Prozedur für Kullern
kullern <- extractWave(kulzis, from=111.363, to=114.59, xunit="time")
kullern2 <- fir(kullern, from = 200, to = 900, output = "Wave")

#Plot Kullern
spectro(kullern2, ovlp=75, collevels=c(-exp(seq(log(20),  0, length=30))),
      palette=reverse.gray.colors.2, flog=TRUE,
      osc=TRUE, scalefontlab = 2, flim = c(0,6))


silbe2 <- kullern2

#Die einzelne Silbe genau geplottet
col <- c(rgb(1,0,0,0.5), rgb(0,0,1,0.5))

spectro(silbe2, ovlp=75, collevels=c(-exp(seq(log(20),  0, length=30))),
      palette=reverse.gray.colors.2, flog=TRUE,
      osc=TRUE, scalefontlab = 2, flim = c(0,6))

#Wenn noch dominante und fundamentale Frequenz dazu kommen sollen
df <- dfreq(silbe2, ovlp=50, threshold=25, plot=FALSE)
```

```
ff <- fund(silbe2, fmax=1000, threshold=25, plot=FALSE)

points(ff, pch=15, col=col[1])

points(df, pch=19, col=col[2])

legend("topleft", legend=c("dominante Frequenz", "fundamentale Frequenz"), pch=c(15,19),
col=col, bty="n")



#Amplitudenfilter mit verschiedenen thresholds

res1 <- afilter(silbe2, threshold=1, output="Wave")

res2 <- afilter(silbe2, threshold=5, output="Wave")

res3 <- afilter(silbe2, threshold=10, output="Wave")



#Wie sehen die Elemente nach dem Filter aus?

#spectro(res3, flog=TRUE, ovlp=75)


#Anhoeren, zeigt aber, dass um so hoeher der threshold, desto unangenehmer hoert es sich
an

listen(res1)

listen(res2)

listen(res3)


#Exportieren

#save.wave(what=res1, where="sample/kull.wav")

#savewav(res1)


#Verschiedene Farbtypen

collevels=seq(-30,0,1)                    #1 Farbe von -30 bis 0

palette=reverse.gray.colors.2             #Farbpalette

(collevels=c(-exp(seq(log(30), 0, length=30)))) #krasse Farben
```

```
oscillo(silbe2)
par(new=TRUE)
env(silbe2, colwave=2)
```

AutomatischeErkennung.R

```
###Wichtige Packages fuer die Soundmanipulation###
###Falls untenstehende Packages nicht installiert, installieren.

###Packages laden###

library("seewave")
library("tuneR")
library("signal")
library("audio")
library("monitoR")
library("warbleR")
library("ggplot2")
library("dtw")
library("caTools")
library("phonTools")
library("tidyverse")
library("gibbonR")
library("viridis")
library("mixtools")

setwd("C:/Users/Felix/OneDrive - ZHAW/ZHAW/BA/R")
```

```
##Es beginnt ein interaktiver Teil. Die SOI (sound of interest) koennen manuell gesucht und markiert werden

## Anleitungen in R beachten!!


## Gebildetes csv laden

manual <- read.csv("data/KullernForm.csv", header=TRUE, sep=";")

manual1 <- read.csv("data/ZischenForm.csv", header=TRUE, sep=";")


head(manual)


##Zischen##

#Die markierten Stellen aus dem .wav koennen nun als Referenz dienen

template1 <- makeCorTemplate("sample/Form_2015-05-13_060600_CH.wav",
t.lim=c(31.0451,32.0668), frq.lim=c(1,2.2), name="t1")

template2 <- makeCorTemplate("sample/Form_2015-05-13_060600_CH.wav",
t.lim=c(43.3052,44.3966), frq.lim=c(1,2.2), name="t2")

template3 <- makeCorTemplate("sample/Form_2015-05-13_060600_CH.wav",
t.lim=c(54.6133,55.5654), frq.lim=c(1,2.2), name="t3")

template4 <- makeCorTemplate("sample/Form_2015-05-13_060600_CH.wav",
t.lim=c(175.496,176.727), frq.lim=c(1,2.2), name="t4")



##Kullern##

template1 <- makeCorTemplate("sample/Form_2015-05-13_060600_CH.wav", t.lim=c(100.556,
104.513), frq.lim=c(0.3,1), name="t1")

template2 <- makeCorTemplate("sample/Form_2015-05-13_060600_CH.wav", t.lim=c(111.34,
114.521), frq.lim=c(0.3,1), name="t2")

template3 <- makeCorTemplate("sample/Form_2015-05-13_060600_CH.wav", t.lim=c(131.634,
        133.631), frq.lim=c(0.3,1), name="t3")

template4 <- makeCorTemplate("sample/Form_2015-05-13_060600_CH.wav", t.lim=c(156.642,
160.311), frq.lim=c(0.3,1), name="t4")


# Templates kombinieren

templates <- combineCorTemplates(template1, template2, template3, template4)
```

```
## Threshold wird von 40% auf -0.1 heruntergesetzt, dass alle Treffer als positive Treffer gezählt werden.

templateCutoff(templates) <- rep(-0.1,4)


#Treffer finden lassen

scores <- corMatch( "sample/Test2_2015-05-13_071514_CH.wav", templates)


scores



#Uebersichtsplot  mit den ersten Ergebnissen

plot(scores@scores$t1$time, scores@scores$t1$score, type="l", col="blue", xlab="Time (s)", ylab="r")


# Wo liegt die hoechste Uebereinstimmung?

peaks <- findPeaks(scores)


peaks



#Plot  mit den Peaks der Uebereinstimmung der Templates und dem Original

plot(peaks, which.one="t1", t.each=3968, legend=FALSE, hit.marker="points", flim=c(0.5,2.5),

    ask = if(dev.list() == 2) TRUE else FALSE)



#wir koennen einen DataFrame erhalten

peaks.selected <- getPeaks(peaks)

head(peaks.selected)


write.csv(peaks.selected, file="scores.csv")


#Hier wurden alle peaksSelected als TrueEvents - also als Uebereinstimmugn gewertet, da teils mehrere Templates mit
```

```
#einer Stelle uebereinstimmen. Um nur die besten - also die Templates mit den besten Uebereinstimmungen zu behalten

#kann der Median ermittelt werden, um ihn schliesslich als Toleranzbereich zu waehlen.


duration <- median(manual$end.time - manual$start.time)

duration


#der Wert wird noch durch 2 geteilt, damit er als Toleranz verwendet werden kann

automatic <- timeAlign(peaks.selected, what="peaks", tol=duration/2)


#Um nun die wirklich relevanten Peaks zu finden, werden die False + (Falsche Uebereinstimmungen)

#und die True - (keine Uebereinstimmung) herausgefiltert

res <- eventEval(detections=automatic, standard=manual, what="peaks", score.cutoff=0.4, tol=duration/2)

head(res)


#Um eine Uebersicht zu erhalten, wieviele Punkte aussortiert wurden, benützen wir die Funktion table

table(res$outcome)



#Falls True+ > True-, folgenden Befehl ausführen:

theta <- seq(0, 1, by=0.01)


#Wir schreiben nun eine leere Confusion MAtrix um die Ergebnisse besser zu speichern.

categories <- matrix(NA, nrow=length(theta), ncol=5)

categories <- as.data.frame(categories)

colnames(categories) <- c("theta", "tp", "tn", "fp", "fn")

categories[,1] <- theta


#Jetzt benutzen wir einen for loop, um die richten Ergebnisse zu speichern

#What is a for loop? for (variable initialization-condition-variable update){ code to execute if the condition is true}

for(i in 1:length(theta)){

  tmp <- table(eventEval(detections=automatic, standard=manual,
```

```
                what="peaks", score.cutoff=theta[i],
                tol=duration/2)$outcome)
categories[i,2] <- tmp["TRUE +"]
categories[i,3] <- tmp["TRUE -"]
categories[i,4] <- tmp["FALSE +"]
categories[i,5] <- tmp["FALSE -"]
}

#Da wir am Anfang eine Matrix erstellt haben mit NA stellen, müssen wir diese jetzt durch 0 ersetzen.
categories[is.na(categories)] <- 0

#Struktur der Daten categories, das NA wurde nun durch 0 ersetzt.
head(categories)

#Nun koennen wir die TPR (True positive events) und FPR (False positive events) berechnen.
tpr <- with(categories, tp/(tp+fn))
fpr <- with(categories, fp/(fp+tn))

#Die AUC (Audio under Curve) koennen mit der Funktion trapz gefunden werden
auc <- trapz(rev(fpr), rev(tpr))
auc

#Plot von AUC in ROC (Receiver Operating characteristics)
plot(fpr, tpr, type="o", pch=19, cex=theta*2, xlab="false positive rate", ylab="true positive rate",
   main=paste("AUC=", round(auc,2)))

#High tpf, low fpr?
identify(fpr, tpr)

#In diesem Fall sind die Werte ziemlich gut, da sehr hohe tpr und sehr tiefe fpr
#Falls dies aber nicht so ist, koennen wir diese so verbessern
theta <- seq(0, 1, by=0.01)
tau <- seq(0, 0.2, by=0.01)
```

```
#Und bereiten wieer eine Leere MAtrix vor
auc <- rep (NA, times=length(tau))



#Jetzt ist ein doppelter loop notwendig, um die Ergebnisse zu verbessern
for(j in 1:length(tau)) {
## empty data frame for data storage
categories <- matrix(NA, nrow=length(theta), ncol=4)
categories <- as.data.frame(categories)
colnames(categories) <- c("tp", "tn", "fp", "fn")
## loop around theta
for(i in 1:length(theta))
{tmp <- table(eventEval(detections=automatic, standard=manual, what="peaks",
score.cutoff=theta[i], tol=tau[j])$outcome)
categories[i,1] <- tmp["TRUE +"]
categories[i,2] <- tmp["TRUE -"]
categories[i,3] <- tmp["FALSE +"]
categories[i,4] <- tmp["FALSE -"]
}

## replacement of NA values by 0 values
 categories[is.na(categories)] <- 0

 ## computation of metrics, TPR and FPR

 tpr <- with(categories, tp/(tp+fn))
 fpr <- with(categories, fp/(fp+tn))

 ## computation of AUC with trapezoid rule integration
 auc[j] <- trapz(rev(fpr), rev(tpr))}

#Manuell die Koordinaten für die hoechste Uebereinstimmung heraussuchen
```

```
tau.max <- tau[which.max(auc)]
tau.max

auc.max <- auc[which.max(auc)]
auc.max

col <- "red"
par (xpd=TRUE, las=1)
plot(tau,
    auc,
    type="o",
    yaxs="i",
    ylim=c(0,1.1),
    xlab=expression(paste("Tolerance (", tau, ")")), ylab="AUC")

points(tau.max,
      auc.max,
      pch=19,
      col=col)

segments(x0=tau.max,
        y0=0,
        x1=tau.max,
        y1=auc.max,
        lty=2, col=col)

text(tau.max, -0.03,
    label=tau.max,
    col=col)

templateCutoff(templates) <- rep(0.22,4)
scores <- corMatch( "sample/Test2_2015-05-13_071514_CH.wav", templates)
peaks <- findPeaks(scores)
```

```
peaks.selected <- getPeaks(peaks)
automatic <- timeAlign(peaks.selected, what="peaks", tol=duration/2)
res <- eventEval(detections=automatic, standard=manual,
        what="peaks", score.cutoff=0.22, tol=0)


plot(peaks, legend=TRUE, t.each=3600, hit.marker="points", flim=c(0.25,5), ask = if(dev.list() == 2)
TRUE else FALSE)


##Export in ein csv.
## Zuerst kann man noch mit dem gewählten Threshold filtern
df <- peaks.selected %>% filter(score >= 0.18)

## und anschliessend exportieren
write.table(df, file = "Test1.csv", sep = ";", row.names = FALSE)
```

BinAutoDec.R

```
library("seewave")
library("tuneR")
library("signal")
library("audio")
library("monitoR")
library("warbleR")
library("ggplot2")
library("dtw")
library("caTools")
library("phonTools")
library("tidyverse")
```

```
library("gibbonR")

library("viridis")

library("mixtools")

library("lubridate")


## Versuch imt binMatch


manual <- read.csv("data/KullernForm.csv", header=TRUE, sep=";")


#Kullern

template1 <- makeBinTemplate("sample/Form_2015-05-13_060600_CH.wav", buffer=0,
amp.cutoff = -30,

                t.lim=c(100.556, 104.513), frq.lim=c(0.3,1), name="t1")

template2 <- makeBinTemplate("sample/Form_2015-05-13_060600_CH.wav", buffer=1,
amp.cutoff = -30,

                t.lim=c(111.34, 114.521), frq.lim=c(0.3,1), name="t2")

template3 <- makeBinTemplate("sample/Form_2015-05-13_060600_CH.wav", buffer=1,
amp.cutoff = -30,

                t.lim=c(131.634,      133.631), frq.lim=c(0.3,1), name="t3")

template4 <- makeBinTemplate("sample/Form_2015-05-13_060600_CH.wav", buffer=1,
amp.cutoff = -30,

                t.lim=c(156.642, 160.311), frq.lim=c(0.3,1), name="t4")



templates <- combineBinTemplates(template1, template2, template3, template4)



templateCutoff(templates) <- rep(-0.1,4)


#Treffer finden lassen

scores <- binMatch( "sample/Form_2015-05-13_060600_CH.wav", templates)
```

```
peaks <- findPeaks(scores)

peaks

#Plot  mit den Peaks der Uebereinstimmung der Templates und dem Original
plot(peaks, which.one="t1", t.each=3968, legend=FALSE, hit.marker="points", flim=c(0.3,1),
    ask = if(dev.list() == 2) TRUE else FALSE)


peaks.selected <- getPeaks(peaks)
head(peaks.selected)

write.csv(peaks.selected, file="scores.csv")

#Hier wurden alle peaksSelected als TrueEvents - also als Uebereinstimmugn gewertet, da teils mehrere Templates mit
#einer Stelle uebereinstimmen. Um nur die besten - also die Templates mit den besten Uebereinstimmungen zu behalten
#kann der Median ermittelt werden, um ihn schliesslich als Toleranzbereich zu waehlen.

duration <- median(manual$end.time - manual$start.time)
duration

#der Wert wird noch durch 2 geteilt, damit er als Toleranz verwendet werden kann
automatic <- timeAlign(peaks.selected, what="peaks", tol=duration/2)

#Um nun die wirklich relevanten Peaks zu finden, werden die False + (Falsche Uebereinstimmungen)
#und die True - (keine Uebereinstimmung) herausgefiltert
```

```
res <- eventEval(detections=automatic, standard=manual, what="peaks", score.cutoff=0.4,
tol=duration/2)

head(res)


#Um eine Uebersicht zu erhalten, wieviele Punkte aussortiert wurden, benützen wir die
Funktion table

table(res$outcome)



#Falls True+ > True-, folgenden Befehl ausführen:

theta <- seq(0, 1, by=0.01)


#Wir schreiben nun eine leere Confusion MAtrix um die Ergebnisse besser zu speichern.

categories <- matrix(NA, nrow=length(theta), ncol=5)

categories <- as.data.frame(categories)

colnames(categories) <- c("theta", "tp", "tn", "fp", "fn")

categories[,1] <- theta


#Jetzt benutzen wir einen for loop, um die richten Ergebnisse zu speichern

#What is a for loop? for (variable initialization-condition-variable update){ code to execute if
the condition is true}

for(i in 1:length(theta)){
  tmp <- table(eventEval(detections=automatic, standard=manual,
                         what="peaks", score.cutoff=theta[i],
                         tol=duration/2)$outcome)
  categories[i,2] <- tmp["TRUE +"]
  categories[i,3] <- tmp["TRUE -"]
  categories[i,4] <- tmp["FALSE +"]
  categories[i,5] <- tmp["FALSE -"]
}
```

```
#Da wir am Anfang eine Matrix erstellt haben mit NA stellen, müssen wir diese jetzt durch 0 ersetzen.

categories[is.na(categories)] <- 0


#Struktur der Daten categories, das NA wurde nun durch 0 ersetzt.

head(categories)


#Nun koennen wir die TPR (True positive events) und FPR (False positive events) berechnen.

tpr <- with(categories, tp/(tp+fn))

fpr <- with(categories, fp/(fp+tn))


#Die AUC (Audio under Curve) koennen mit der Funktion trapz gefunden werden

auc <- trapz(rev(fpr), rev(tpr))

auc


#Plot von AUC in ROC (Receiver Operating characteristics)

plot(fpr, tpr, type="o", pch=19, cex=theta*2, xlab="false positive rate", ylab="true positive rate",

    main=paste("AUC=", round(auc,2)))


#High tpf, low fpr?

identify(fpr, tpr)


#In diesem Fall sind die Werte ziemlich gut, da sehr hohe tpr und sehr tiefe fpr

#Falls dies aber nicht so ist, koennen wir diese so verbessern

theta <- seq(0, 1, by=0.01)

tau <- seq(0, 0.2, by=0.01)


#Und bereiten wieer eine Leere MAtrix vor

auc <- rep (NA, times=length(tau))
```

```
#Jetzt ist ein doppelter loop notwendig, um die Ergebnisse zu verbessern
for(j in 1:length(tau)) {
 ## empty data frame for data storage
 categories <- matrix(NA, nrow=length(theta), ncol=4)
 categories <- as.data.frame(categories)
 colnames(categories) <- c("tp", "tn", "fp", "fn")
 ## loop around theta
 for(i in 1:length(theta))
 {tmp <- table(eventEval(detections=automatic, standard=manual, what="peaks",
                  score.cutoff=theta[i], tol=tau[j])$outcome)
 categories[i,1] <- tmp["TRUE +"]
 categories[i,2] <- tmp["TRUE -"]
 categories[i,3] <- tmp["FALSE +"]
 categories[i,4] <- tmp["FALSE -"]
 }

 ## replacement of NA values by 0 values
 categories[is.na(categories)] <- 0

 ## computation of metrics, TPR and FPR

 tpr <- with(categories, tp/(tp+fn))
 fpr <- with(categories, fp/(fp+tn))

 ## computation of AUC with trapezoid rule integration
 auc[j] <- trapz(rev(fpr), rev(tpr))}

#Manuell die Koordinaten für die hoechste Uebereinstimmung heraussuchen
tau.max <- tau[which.max(auc)]
```

```
tau.max


auc.max <- auc[which.max(auc)]
auc.max


col <- "red"
par (xpd=TRUE, las=1)
plot(tau, auc, type="o", yaxs="i", ylim=c(0,1.1), xlab=expression(paste("Tolerance (", tau, ")")), ylab="AUC")
points(tau.max, auc.max, pch=19, col=col)
segments(x0=tau.max, y0=0, x1=tau.max, y1=auc.max, lty=2, col=col)
text(tau.max, -0.03, label=tau.max, col=col)


templateCutoff(templates) <- rep(5.2,4)
scores <- binMatch( "sample/Test2_2015-05-13_071514_CH.wav", templates)
peaks <- findPeaks(scores)
peaks.selected <- getPeaks(peaks)
automatic <- timeAlign(peaks.selected, what="peaks", tol=duration/2)
res <- eventEval(detections=automatic, standard=manual,
            what="peaks", score.cutoff=5.2, tol=0.17)



plot(peaks, legend=TRUE, t.each=3600, hit.marker="points", flim=c(0.25,5), ask = if(dev.list() == 2) TRUE else FALSE)



##Export in ein csv.
## Zuerst kann man noch mit dem gewählten Threshold filtern
df <- peaks.selected %>% filter(score >= 4.5)
```

```
## und anschliessend exportieren
write.table(df, file = "Test1.csv", sep = ";", row.names = FALSE)
```

AutomaticDetection.R

```
## Automatische Erkennung mit machine learning ###
# verwendete R Pakete. Am besten in dieser Reihenfolge installieren und von der Library laden

library("seewave")
library("tuneR")
library("signal")
library("audio")
library("monitoR")
library("warbleR")
library("ggplot2")
library("dtw")
library("caTools")
library("phonTools")
library("tidyverse")
library("gibbonR")
library("viridis")
library("mixtools")

#Um nicht wichtige errormeldungen zu unterdruecken, verwenden wir folgenden Code
options(error = NULL)

#Arbeitsumgebung festlegen
setwd("C:/Users/FailX/Desktop/BA/R")

#Outputordner festegen
output.dir <- ("C:/Users/FailX/Desktop/BA/R/sample/Output")
```

```
#Es gibt mehrere Möglichkeiten im Ablauf der Erkennung

#Falls ein Ordner mit verschiedenen .wav Dateien vorliegt, in dem sich die Referenzdateien befinden, die als

#die Grundwahrheit (ground truth) angesehen werden. --> Schritt 1

#Wenn ein .csv vorliegt, welches bereits berechnete MFCC Zahlen beinhaltet, können diese Schritte uebersprungen

#werden. Weiter bei Schritt XXXX.

#Falls keine Dateien oder .csv vorliegt --> R Skript "ReferenzdateienSammeln.R" verwenden!



#Schritt 1:

#working directory zum Ordner mit den Referenzdateien erstellen

#Im idealfall sind dort alle Dateien mit allen möglichen Balzgeräuschen vorhanden.

setwd("C:/Users/FailX/Desktop/BA/R/sample/Tetrao")

#Leere Liste kreiieren

list.of.tetrao <-list()

#Liste mit allen Dateien in diesem Ordner kreiieren

list.wav.file.names <- list.files()

#generierter Loop um jede einzelne Datei einzulesen und fuer R lesbar zu machen

for (x in 1:length(list.wav.file.names)) {

  tmp.wav <- tuneR::readWave(list.wav.file.names[x])

  list.of.tetrao[[x]] <- list(list.wav.file.names[x], tmp.wav)

}



#MFCC für das Kullern kreiieren
```

```
#Mel Frequency Cepstral Coefficients werden in der Spracherkennung verwendet und
#sorgen für eine kompakte Darstellung des Frequenzspektrums. Die automatisierte Erkennung basiert darauf.
Kullern.MFCC <- calcMFCC(
  list.wav.files = list.of.tetrao,
  n.window = 9,
  n.cep = 12,
  min.freq = 100,
  max.freq = 1000,
  feature.red = FALSE
  )

#Das selbe fuer das Zischen
Zischen.MFCC <- calcMFCC(
  list.wav.files = list.of.tetrao,
  n.window = 9,
  n.cep = 12,
  min.freq = 1200,
  max.freq = 2100,
  feature.red = FALSE
  )

#Bei Bedarf kann ein .csv erstellt werden. Die MFCC werden gespeichert und haben eine geringe
#Dateigroesse und können leichter weitergegeben werden.
#Wichtig ist, dass sich folgende Argumente gemerkt werden
#c.nep, n.window, und min./max.freq muessen bei der Erkennung gleich sein!!
#output ort definieren
setwd("C:/Users/FailX/Desktop/BA/R")
#Fuer das Kullern
write.csv2(Kullern.MFCC, "Kullern.MFCC.csv", row.names = F)

#Fuer das Zischen
write.csv2(Zischen.MFCC, "Zischen.MFCC.csv", row.names = F)
```

```
#Schritt 2: .csv mit MFCC laden

#Ort wo sich die .csv dateien befinden
setwd("C:/Users/FailX/Desktop/BA/R")

#Welches Balzgeraeusch moechte ich finden?
#Zischen?
Zischen.MFCC <- read.csv2("Zischen.MFCC.csv")

#Kullern?
Kullern.MFCC <- read.csv2("Kullern.MFCC.csv")



#Wie gut koennen die verschiedenen Klassen unterschieden werden?
#Support Vector Machine
svm.output.kullern <- trainSVM(feature.df = Kullern.MFCC,
                    tune = TRUE,
                    train.n = 0.8,
                    test.n = 0.2,
                    cross = 5
                    )

svm.output.kullern$correct.prob

#Gaussian Mixture tools
gmm.output.kullern <- trainGMM(feature.df = Kullern.MFCC,
                      train.n = 0.8,
                      test.n = 0.2
                      )

gmm.output.kullern$correct.prob
```

```
#Linear discriminant function analysis
lda.output.kullern <- trainLDA(feature.df = Kullern.MFCC,
                    train.n = 0.8,
                    test.n = 0.2,
                    CV = FALSE
                    )

lda.output.kullern$correct.prob



#Neural networks
nn.output.kullern <- trainNN(feature.df = Kullern.MFCC,
                    train.n = 0.8,
                    test.n = 0.2)

nn.output.kullern$correct.prob



#Bestes Resultat = Methode fuer die Erkennung (SVM, GMM, NN)
#Im Falle des KullernMFCC ist SVM die beste Methode
#Darstellung mit einem biplot. Ein Unsicherheitskreis (uncertainty circle)
#Kullern
biplotGibbonR(Kullern.MFCC,
        classification.type = "PCA",
        class.labs = F)
#Zischen
biplotGibbonR(Zischen.MFCC,
        classification.type = "PCA",
        class.labs = F)

#Schritt 3: Automatisierte Erkennung
#Hier auf n.cep und n.window achten!! Muessen gleich sein wie calcMFCC
```

```
#.wav einlesen, welches durchsucht werden soll
wav.for.detection <- readWave("C:/Users/FailX/Desktop/BA/R/sam-
ple/SM02_0+1_20160522_040000.wav")

#Outputordner festegen
output.dir <- ("C:/Users/FailX/Desktop/BA/R/sample/Output")

#mit Kullern
detectionKullern <- detectGibbonR(feature.df = Kullern.MFCC,
                        model.type = "SVM",
                        tune = "FALSE",
                        wav.name = wav.for.detection,
                        which.quant = "low.quant",
                        min.freq = 0.1,
                        max.freq = 1,
                        low.quant.val = 0.1,
                        high.quant.val = 0.9,
                        n.windows = 9,
                        target.signal = "Kullern",
                        min.sound.event.dur = 1.25,
                        probability.thresh = 0.9,
                        density.plot = T,
                        output.dir = output.dir )

## wenn gwollt
#ergebnisse in einem Spektrogramm visualisieren
#Visu 1
temp.spec <- signal::specgram(wav.for.detection@left, Fs = wav.for.detection@samp.rate, n = 512,
                        overlap = 0)

#Visu 2

plot( temp.spec,
    xlab = "Time (s)",
```

```
    ylab = "Frequency (Hz)",
    col = viridis::viridis(512),
    useRaster = TRUE
    )

#Visu 3

# And add boxes for the identified sound events
for (x in 1:nrow(detectionKullern$timing.df)) {
  rect(detectionKullern$timing.df[x, 3],
      100,
      detectionKullern$timing.df[x, 4],
      1000,
      border = "red")
  vec <- c(detectionKullern$timing.df[x, 3], detectionKullern$timing.df[x, 4])
  x.val <- vec[-length(vec)] + diff(vec) / 2

  text(x.val, 2100, labels = round(detectionKullern$timing.df[x, 5], digits = 1))
}



#wiederholen mit zischen
#mit zischen
detectionZischen <- detectGibbonR(feature.df = Zischen.MFCC,
                          model.type = "SVM",
                          tune = "FALSE",
                          wav.name = wav.for.detection,
                          which.quant = "low.quant",
                          min.freq = 1.2,
                          max.freq = 2,
                          n.windows = 9,
                          high.quant.val = 0.05,
                          low.quant.val = 0.85,
```

```
                    target.signal = "Zischen",
                    min.sound.event.dur = 1,
                    max.sound.event.dur = 2,
                    probability.thresh = 0.99,
                    output.dir = output.dir )


?detectGibbonR
#Falls beide Balzgerauesche in einem Plot
#Visu 1 wiederholen plus Visu 4

#Visu 1
temp.spec <- signal::specgram(wav.for.detection@left, Fs = wav.for.detection@samp.rate, n = 512,
                    overlap = 0)

#Visu 4

# And add boxes for the identified sound events
for (x in 1:nrow(detectionZischen$timing.df)) {
  rect(detectionZischen$timing.df[x, 3],
      1100,
      detectionZischen$timing.df[x, 4],
      2100,
      border = "blue")
  vec <- c(detectionZischen$timing.df[x, 3], detectionZischen$timing.df[x, 4])
  x.val <- vec[-length(vec)] + diff(vec) / 2

  text(x.val, 2100, labels = round(detectionZischen$timing.df[x, 5], digits = 1))
}

#Ergebnisse abspeichern und zur Auswertung vorbereiten
#Wo das .csv gepeichert werden soll
setwd("C:/Users/FailX/Desktop/BA/R/sample/Output")
```

```
write.csv2(detectionKullern, "Kullern_22.5.2016.csv", row.names = F)

write.csv2(detectionZischen, "Zischen_22.5.2016.csv", row.names = F)



## Durchsuchen mehrerer .wav files gleichzeitig

#Im idealfall sind dort alle Dateien die durchsucht werden sollen vorhanden.
setwd("C:/Users/FailX/Desktop/Test")

#Leere Liste kreiieren
list2Det <-list()

#Liste mit allen Dateien in diesem Ordner kreiieren
list.wav.file.names <- list.files()

#generierter Loop um jede einzelne Datei einzulesen und fuer R lesbar zu machen
for (x in 1:length(list.wav.file.names)) {
  tmp.wav <- tuneR::readWave(list.wav.file.names[x])
  list2Det[[x]] <- list(list.wav.file.names[x], tmp.wav)
}

#List emit MFCC lade - alternativ erstellen - siehe weiter oben
#Ort wo sich die .csv dateien befinden
setwd("C:/Users/FailX/Desktop/BA/R")

#Welches Balzgeraeusch moechte ich finden?
#Zischen?
Zischen.MFCC <- read.csv2("ZischenMFCC.csv")

#Kullern?
Kullern.MFCC <- read.csv2("KullernMFCC.csv")
```

```
#Detect multiple files

detect.mf <- batchDetectGibbonR(input = list2Det,
                        feature.df = Kullern.MFCC,
                        model.type = "SVM",
                        min.freq = 0.1,
                        max.freq = 1,
                        which.quant = "low.quant",
                        n.windows = 9,
                        min.sound.event.dur = 1.5,
                        wav.output = TRUE,
                        target.signal = "Kullern",
                        probability.thresh = 0.9,
                        output.dir = output.dir
                        )
```

DarstellungErgebnisse.R

```
## Versuche des Plottens
library(ggplot2)
library(tidyverse)
library(lubridate)
library(ggpubr)
library(cowplot)

setwd("C:/Users/Felix/OneDrive - ZHAW/ZHAW/BA/R")

#Datei einlesen----
```

```
af1<-read_delim("data/PL2Man.csv",delim=";")



#Zeit in richtiges Format umwandeln----
af1$start.time<-as.numeric(as.character(af1$start.time))
af1$end.time<-as.numeric(as.character(af1$end.time))

#Sekunden auf 0 Dezimalstellen runden----
af1$start.time<-round(af1$start.time,digits=0)
af1$end.time<-round(af1$end.time,digits=0)
af1

#Sekunden in HMS umwandeln----
af1$start.time<-seconds_to_period(af1$start.time)
af1$end.time<-seconds_to_period(af1$end.time)

#Richtig umformen----
af1$start.time<-sprintf('%02d:%02d:%02d', af1$start.time@hour, minute(af1$start.time), second(af1$start.time))
af1$end.time<-sprintf('%02d:%02d:%02d', af1$end.time@hour, minute(af1$end.time), second(af1$end.time))

af1
#Als POSICct umwandeln----
af1$start.time<-as.POSIXct(af1$start.time,format="%H:%M:%S")
af1$end.time<-as.POSIXct(af1$end.time,format="%H:%M:%S")



#Plot erstellen----
```

```
p<-ggplot(af1) +
  geom_linerange(aes(x = af1$date,
                     ymin = af1$start.time,
                     ymax = af1$end.time),
                 alpha = 1, position = position_dodge(width = 0.3), size = 5, color= "limegreen") +
  coord_flip() +
  labs(y="Zeit [h]",
       x="",
       title = "Plotwatcher manuell")+
  scale_y_time(labels=c("04:00", "06:00","08:00", "10:00")) +
  theme_bw()
p




#Plot für lange Aufzeichnungen-----
ggplot(af1)+
  geom_point(mapping=aes(x=af1$date,y=af1$start.time,color=af1$event),pch="|",cex=3,position = position_dodge(width = 0.4), size = 2)+
coord_flip()+
  labs(y="Zeit [h]",
       x="",
       title = "Auswertung 'Testlang' ")+
  theme_bw()




#Datei einlesen----
af2<-read_delim("data/SM3Auto.csv",delim=";")
```

```
#Zeit in richtiges Format umwandeln----
af2$start.time<-as.numeric(as.character(af2$start.time))
af2$end.time<-as.numeric(as.character(af2$end.time))

#Sekunden auf 0 Dezimalstellen runden----
af2$start.time<-round(af2$start.time,digits=0)
af2$end.time<-round(af2$end.time,digits=0)
af2

#Sekunden in HMS umwandeln----
af2$start.time<-seconds_to_period(af2$start.time)
af2$end.time<-seconds_to_period(af2$end.time)

#Richtig umformen----
af2$start.time<-sprintf('%02d:%02d:%02d', af2$start.time@hour, minute(af2$start.time), second(af2$start.time))
af2$end.time<-sprintf('%02d:%02d:%02d', af2$end.time@hour, minute(af2$end.time), second(af2$end.time))

af2
#Als POSICct umwandeln----
af2$start.time<-as.POSIXct(af2$start.time,format="%H:%M:%S")
af2$end.time<-as.POSIXct(af2$end.time,format="%H:%M:%S")


#Plot erstellen----

p1<-ggplot(af2) +
  geom_linerange(aes(x = af2$date,
```

```
                ymin = af2$start.time,
                ymax = af2$end.time),
              alpha = 1, position = position_dodge(width = 0.3), size = 5, color = "limegreen") +
 coord_flip() +
 labs(y="Zeit [h]",
     x="",
     title = "Songmeter automatisch")+
 scale_y_time(labels=c("04:00", "06:00","08:00", "10:00")) +
 theme_bw()
p1




#Plot für 2h Aufzeichnung-----
p2 <- ggplot(af2)+
 geom_point(mapping=aes(x=af2$date,y=af2$start.time),pch="|",cex=3,position = posi-
tion_dodge(width = 0.4),
          color = "limegreen", size = 5) +
 coord_flip()+
 labs(y="Zeit [h]",
     x="",
     title = "Songmeter automatisch")+
 scale_y_time(labels=c("04:00", "06:00","08:00", "10:00")) +
 theme_bw()


p2



plot_grid(p, p2+ rremove("y.text"),
        ncol = 2, rel_widths = c(1.15,1))
```

```
###########???

library(data.table)

#subsets der daten erstellen
#zuerst nach den Programmen

manuell<-af1%>%
  filter(date=="manuell")

CC<-af1%>%
  filter(date=="CC Seewave")

gg<-af1%>%
  filter(date=="gibbonR")

bs<-af1%>%
  filter(date=="Bin Seewave")
#nun jeweils ein subset mit kullern und zischen erstellen und dann nur
#df mit end und start zeit als spalten. den rest wollen wir nicht

manuell_zischen<-manuell%>%
  filter(event=="Zischen")%>%
  select("start.time","end.time")

manuell_kullern<-manuell%>%
```

```
  filter(event=="Kullern")%>%
  select("start.time","end.time")

CC_zischen<-CC%>%
  filter(event=="Zischen")%>%
  select("start.time","end.time")

CC_kullern<-CC%>%
  filter(event=="Kullern")%>%
  select("start.time","end.time")

gg_zischen<-gg%>%
  filter(event=="Zischen")%>%
  select("start.time","end.time")

gg_kullern<-gg%>%
  filter(event=="Kullern")%>%
  select("start.time","end.time")

bs_zischen<-bs%>%
  filter(event=="Zischen")%>%
  select("start.time","end.time")

bs_kullern<-bs%>%
  filter(event=="Kullern")%>%
  select("start.time","end.time")



#alle data frames in data.tables umwandeln
```

```
setDT(bs_kullern)
setDT(bs_zischen)
setDT(gg_kullern)
setDT(gg_zischen)
setDT(CC_kullern)
setDT(CC_zischen)
setDT(manuell_kullern)
setDT(manuell_zischen)

###
#Kullern-----
####

#binSeewave
#ol=overlaps

setkey(bs_kullern, start.time,end.time)
bs_kullern_ol<-foverlaps(manuell_kullern,bs_kullern,which=TRUE)



#gibbonR
setkey(gg_kullern, start.time,end.time)
gg_kullern_ol<-foverlaps(manuell_kullern,gg_kullern,which=TRUE)



#CC seawave

setkey(CC_kullern, start.time,end.time)
```

```
CC_kullern_ol<-foverlaps(manuell_kullern,CC_kullern,which=TRUE)




###
#Zischen----
###

#binSeewave
#ol=overlaps

setkey(bs_zischen, start.time,end.time)
bs_zischen_ol<-foverlaps(manuell_zischen,bs_zischen,which=TRUE)
bs_zischen_ol


#gibbonR
setkey(gg_zischen, start.time,end.time)
gg_zischen_ol<-foverlaps(manuell_zischen,gg_zischen,which=TRUE)
gg_zischen_ol


#CC seawave

setkey(CC_zischen, start.time,end.time)
CC_zischen_ol<-foverlaps(manuell_zischen,CC_zischen,which=TRUE)
CC_zischen_ol

###
#Lösche NA und count den rest -> ergibt die Anzahl Uebereinstimmungen
```

```
#kullern

CC_kullern_ol%>%
  filter(yid!="NA")%>%
  count()


gg_kullern_ol%>%
  filter(yid!="NA")%>%
  count()

bs_kullern_ol%>%
  filter(yid!="NA")%>%
  count()


#zischenn

CC_zischen_ol%>%
  filter(yid!="NA")%>%
  count()

gg_zischen_ol%>%
  filter(yid!="NA")%>%
  count()

bs_zischen_ol%>%
  filter(yid!="NA")%>%
  count()
```

Anhang B: Aufgabenstellung

Bachelor-Arbeit		
Studienjahrgang		UI 16
Titel		**Monitoring der Birkhuhnbalz mit Flächenkamerasund Bio-akustik**
7.1.1.1 VERTRAULICH		X **ja** **nein**
Fachgebiet		Wildtiermanagement
Namen	StudentIn	Felix Konrad Burkhalter Seestrasse 251 8804 Au ZH +41 77 401 87 90 burkhfel@students.zhaw.ch
	1. KorrektorIn	Stefan Suter ZHAW Wädenswil +41 58 934 53 88 suts@zhaw.ch
	2. KorrektorIn	Annette Stephani ZHAW Wädenswil +41 58 934 55 99 stpi@zhaw.ch

Aufgabenstellung • **Ausgangslage** • **Zielsetzungen** • **Zusätzliche Auftrags-modalitäten**	**Ausgangslage:** In manchen Regionen der Schweiz sind die Bestände und die Verbreitung des Birkhuhns geschrumpft. Als Gründe werden die zunehmende Beanspruchung des Habitats durch verschiedene Freizeitaktivitäten genannt. Über deren Einfluss auf die Balzaktivität der Birkhähne ist noch wenig bekannt. **Zielsetzungen:**

	Den Einfluss menschlicher Störung auf die Balzaktivität soll mittels einer Auswertung von Akustik- und Bildaufnahmen besser untersucht werden. - Haben menschliche Störungen Einfluss auf die Balzaktivität? - Wie werden Balzplätze nach einer menschlichen Störung weiter genutzt? - Wann sind die Birkhennen auf dem Balzplatz? Zurzeit wird Akustikmaterial manuell ausgewertet. Das bedeutet, dass jede Aufnahme einzeln in einem Spektrogramm begutachtet wird. Stellen, an denen eine Balzaktivität zu erkennen ist, müssen so herausgesucht werden. Dies ist mit hohem zeitlichem und somit finanziellem Aufwand verbunden. Im Rahmen des methodischen Teils dieser Arbeit soll geklärt werden, wie sich in Zukunft akustische Aktivität automatisch erfassen lässt. **Zusätzliche Auftragsmodalitäten:** Provisorisches Inhaltsverzeichnis Zusammenfassung Abstract Inhaltsverzeichnis Liste der Abkürzungen 1 Einleitung 2 Literaturübersicht 2.1 Ablauf des Balzgeschehens beim Birkhuhn 2.2 Einfluss menschlicher Störung auf die Balzaktivität 2.3 Analysemethoden von Akustikaufnahmen 2.4 Problematik 3 Material und Methoden 3.1 Verwendete Materialien für die Akustikauswertung 3.2 Analyseparameter 3.3 Anwendung einer automatisierten Akustikauswertung 4 Ergebnisse 4.1 Statistische Auswertung der Akustikaufnahmen in Bezug auf die Wahl eines Balzgebietes 4.1.1 Erkenntnisse 4.1.2 Schwierigkeiten bei der Erhebung 4.2 Erfolgskontrolle der automatisierten Akustikauswertung 4.2.2 Chancen und Grenzen 4.3 Ablauf des Balzgeschehens 4.4 Einfluss menschlicher Störung auf die Birkhuhnbalz 5 Diskussion 6 Literaturverzeichnis

	Verzeichnis der Bilder Verzeichnis der Tabellen Anhang
Formale Anforderungen	Alle relevanten Merkblätter zu studentischen Arbeiten
Zeitplan	*Siehe Anhang*
Abgabetermin (12.00 Uhr)	**08.08.2019**
Bemerkungen	**Abgabeform:** Arbeit: Abgabe in elektronischer Form auf Complesis
Arbeitsort	ZHAW Wädenswil, (Luzern)

Plagiate verstossen gegen die Urheberrechte, eine Verletzung dieser Rechte wird gemäss der Studien- und Prüfungsordnung für die Bachelorstudiengänge der Hochschule Wädenswil vom 01.09.2006 in § 38, 39 geregelt. Diese Studien- und Prüfungsordnung gilt für alle Bachelorstudienjahrgänge bis und mit Studienstart 2009**.**

Für Bachelorstudienjahrgänge mit Studienbeginn ab 2010 und die Masterstudiengänge mit Studienbeginn ab 2009 gilt § 39 der Rahmenprüfungsordnung für Bachelor- und Masterstudiengänge an der Zürcher Hochschule für Angewandte Wissenschaften vom 29.01.2008**.**

Anhang C: Plagiatserklärung

PLAGIATSERKLÄRUNG

Zürcher Hochschule
für Angewandte Wissenschaften

zh aw Life Sciences und Facility Management

Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit im Departement Life Sciences und Facility Management

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat.

Der/die unterzeichnende Studierende erklärt, dass alle verwendeten Quellen (auch Internetseiten) im Text oder Anhang korrekt ausgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten Paragraph 39 und Paragraph 40 der Rahmenprüfungsordnung für die Bachelor- und Masterstudiengänge an der Zürcher Hochschule für Angewandte Wissenschaften vom 29. Januar 2008 sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum Unterschrift der/die Studierende

…… ……