Zürcher Hochschule
für Angewandte Wissenschaften

Corona-Pandemie und Wetter – Auswirkungen auf die Besuchszahlen im Wildnispark Zürich

Fallstudie

ReMe HS21

ZÜRCHER HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN

DEPARTEMENT LIFE SCIENCES UND FACILITY MANAGEMENT

INSTITUT FÜR UMWELT UND NATÜRLICHE RESSOURCEN

Carina Kohler, Marigna Franck, Mirco Nietlisbach & Ursula Schöni

Bild Titelseite: Waldspaziergang während Lockdown

Bildquelle: © Everst

Modul: Research Methods HS21, Ecosystems & Biodiversity, Profil S

Fachkorrektoren: Adrian Hochreutener & Reto Rupf

Abgabedatum: 9. Januar 2022

1. Einleitung

Die Corona-Pandemie hat deutlich gezeigt, wie wichtig Wälder als Naherholungsgebiete sind. Weil aufgrund der partiellen Lockdowns im Frühling 2020 und im Winter 2020/21 andere Freizeitbeschäftigungen wegfielen, wurden die nahen Wälder für viele zum Sport- und Spielplatz (AfW 2021). Forschende der WSL kamen zum Schluss, dass während des ersten Lockdowns deutlich mehr Stadtbewohner:innen täglich in den Wald gingen, verglichen mit einem Frühling vor Corona-Zeiten (Hunziker et al. 2020). Andere Studien zeigten Ähnliches (Derks et al. 2020, Ramisch 2020). In der Umfrage von Suda et al. (2021) gaben Teilnehmende an, den Wald primär zur Erholung aufzusuchen. Seit der Corona-Pandemie ist die Bedeutung des Waldes als bewusst gewählter Rückzugsort gestiegen. Auch die Besuchszahlen des Wildnisparks Zürich zeigen in eine Richtung: nach oben (Wildnispark Zürich 2021a).

Um Massnahmen zur Besucherlenkung zu ergreifen und die Nutzung eines Gebietes natur- und sozialverträglich zu planen, dient Besuchermonitoring als hilfreiche Grundlage. Ein besseres Verständnis des Verhaltens der Besuchenden kann helfen, Managemententscheide in der Praxis zu treffen. Für die Verantwortlichen des Wildnisparks Zürich ist es deshalb wichtig, zu wissen, wie sich die Besuchszahlen seit Beginn der Corona-Pandemie verändert haben.

In dieser Fallstudie wurde neben dem Einfluss der Lockdowns auch derjenige von Wetterparametern, Wochentagen und Ferien auf die Besuchszahlen des Wildnisparks Zürich untersucht.

2. Fragestellungen

Folgende drei Forschungsfragen waren von Interesse:

- Welchen Einfluss haben neben den Lockdowns die Wetterparameter (Sonnenscheindauer, Tageshöchsttemperatur, Niederschlagssumme) sowie der Wochentag, die Ferien, die Kalenderwoche und das Jahr auf die Besuchszahlen?
- Wie stark sind die Einflüsse, welche Effektrichtungen sind beobachtbar und welche der untersuchten Parameter sind signifikant?
- Können deutliche Unterschiede zwischen den Vor-Covid-19-Jahren und danach bei Tages-, Wochen- oder Saisongang sowie den wichtigsten, deskriptiven Kennzahlen gefunden werden?

3. Methoden

3.1 Untersuchungsgebiet und Messzeitraum

Der Wildnispark Zürich liegt im periurbanen Raum südlich der Stadt Zürich, für deren Bevölkerung er von grosser Bedeutung ist (Wildnispark Zürich 2021b). Über den Zeitraum von Januar 2017 bis Juli 2021 wurden Daten der Zählstelle 211 (Abb. 1) erfasst und aufgezeichnet.

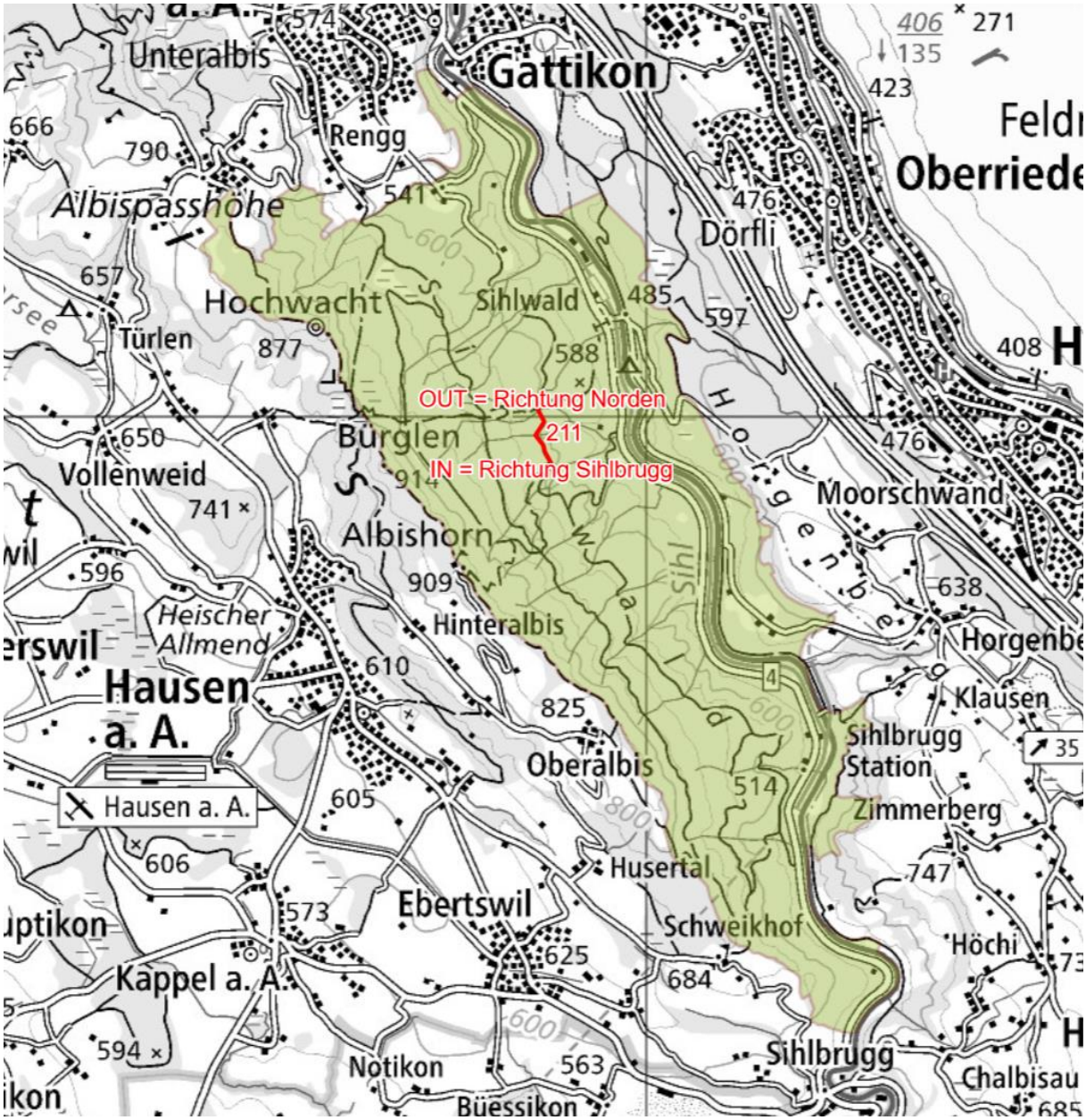

Abb. 1: Der Kartenausschnitt zeigt das Untersuchungsgebiet mit der automatischen Zählstelle 211, deren Daten im Rahmen der vorliegenden Untersuchung ausgewertet wurden. © swisstopo

3.2 Datenerhebung

Die Erfassung der Besuchszahlen erfolgte mittels Kombizählgerät der Firma Eco-Counter, das zwischen Fussgänger:innen und Fahrrädern unterscheidet. In der Fallstudie wurde nur die Anzahl Fussgänger:innen untersucht.

Die Zähldaten wurden durch den Wildnispark Zürich vorgängig bereinigt, validiert und kalibriert. Die Wetterdaten der Messstation Wädenswil konnten von MeteoSchweiz bezogen werden (MeteoSchweiz 2021).

3.3 Datenanalyse

Die Analyse erfolgte mit der Software R, Version 4.0.3 (R Core Team 2020), wobei unter anderem die R-Packages «tidyverse» (Wickham et al. 2019) und «lme4» (Bates et al. 2015) verwendet wurden. Nach dem Einlesen der Daten wurden Ferien, Lockdown-Phasen, Werktage und Wochenenden definiert (Tab. 1). Im Rahmen der deskriptiven Analyse wurden der Verlauf der Besuchszahlen visualisiert und die Phasen miteinander verglichen. Zudem wurden t-Tests und ANOVAs mit einer Signifikanzschwelle von $p \leq 0.05$ (signifikant) durchgeführt, um allfällige Unterschiede in den Besuchszahlen an verschiedenen Wochentagen zu identifizieren.

Der Zusammenhang zwischen den täglichen Besuchszahlen und den Einflussfaktoren wurde mittels Generalized Linear Mixed Effect Model (GLMM) aufgezeigt. Die Wetterparameter wurden skaliert, um die unterschiedlichen Einheiten vergleichbar zu machen. Mittels Korrelationsmatrix konnte geprüft werden, ob Variablen zu stark korrelierten (Schwellenwert>0.7). Da sich die Korrelationen nach Pearson (Freedman et al. 2007) zwischen den Variablen als unkritisch erwiesen (stärkste Korrelation zwischen Tagestemperatur und Sonnenscheindauer 0.55), konnten alle drei Wetterparameter beibehalten werden. Um saisonale Schwankungen weitestgehend auszuschalten, wurden die Variablen Kalenderwoche und Jahr als Random Factors definiert (Tab. 1).

In der Folge wurden diverse Modelle gerechnet, um unterschiedliche Verteilungen (exponentiell, negativ binomial, Poisson) zu prüfen. Die Modelldiagnostiken der drei Modelle wiesen auf eine Abweichung der Normalverteilung und der Varianzhomogenität bei den Residuen sowie teilweise auf eine Overdispersion hin, wodurch die Modellvoraussetzungen nicht erfüllt waren. Die Prüfung der Skewness zeigte eine rechtsschiefe Verteilung (Wert 2.479). Daher wurde eine log-10-Transformation der abhängigen Variable durchgeführt. Voraussetzung für die Transformation sind Werte >0, weshalb aufgrund einiger Nullwerte in der abhängigen Variable die Zähldaten +1 gerechnet wurden. Eine erneute Modellrechnung zeigte, dass das beste Modell keine Interaktion und keine quadratischen

Terme enthält. Zudem wurden die Ferien entfernt, da diese nicht signifikant waren und das Modell verbesserten (p=0.072, ΔAICc=4.28).

Tab 1: Beschreibung der abhängigen Variable, der unabhängigen Variablen und der Zufallsfaktoren.

Variable	*Beschreibung*	*Datentyp*
Abhängige Variable		
Anzahl Fussgänger:innen intervall	Total (IN+OUT) der Fussgänger:innen, welche die Zählstation 211 passieren	Integer
Unabhängige Variablen		
Temperatur (tre200jx scaled) intervall	Tagesmaximum der Lufttemperatur, 2m über Boden [°C]	Double
Niederschlag (rre150j0 scaled) intervall	Halbtagessumme des Niederschlags [mm]	Double
Sonnenschein (sremaxdv scaled) ratio	Tagessumme der Sonnenscheindauer im Verhältnis zur Tageslänge [%]	Integer
Ferien binär	Schulferien im Kanton Zürich 1 = Ferien 0 = keine Ferien	Factor
Phasen nominal	Normal = Start der Erhebung bis Lockdown 1 (15.03.2020) Lockdown 1 = 16.03.2020-11.05.2020 Lockdown 2 = 22.12.2020-01.03.2021 Covid = seit Pandemiebeginn, aber nicht während eines Lockdowns	Factor
Wochentag nominal	Werktag = Montag-Freitag Wochenende = Samstag-Sonntag	Factor
Zufallsfaktoren (Random Factors)		
Kalenderwoche intervall	1-52	Factor
Jahr intervall	2017-2021	Factor

4. Resultate

Im Messzeitraum wurden 55‘637 Fussgänger:innen gezählt. Die Anzahl reichte von null bis zu 299 Fussgänger:innen pro Tag (Ø35). Der Vergleich der Tagesverläufe zeigt, dass in jeder Phase die Unterschiede in den Wochentagen höchst signifikant sind (ANOVA, $p<0.001$). Dabei gibt es v.a. zwischen den Werktagen und Wochenenden deutliche Unterschiede (Abb. 2). Dieses Ergebnis konnte mittels t-Test für alle vier Phasen bestätigt werden (Welch-Test, $p<0.001$).

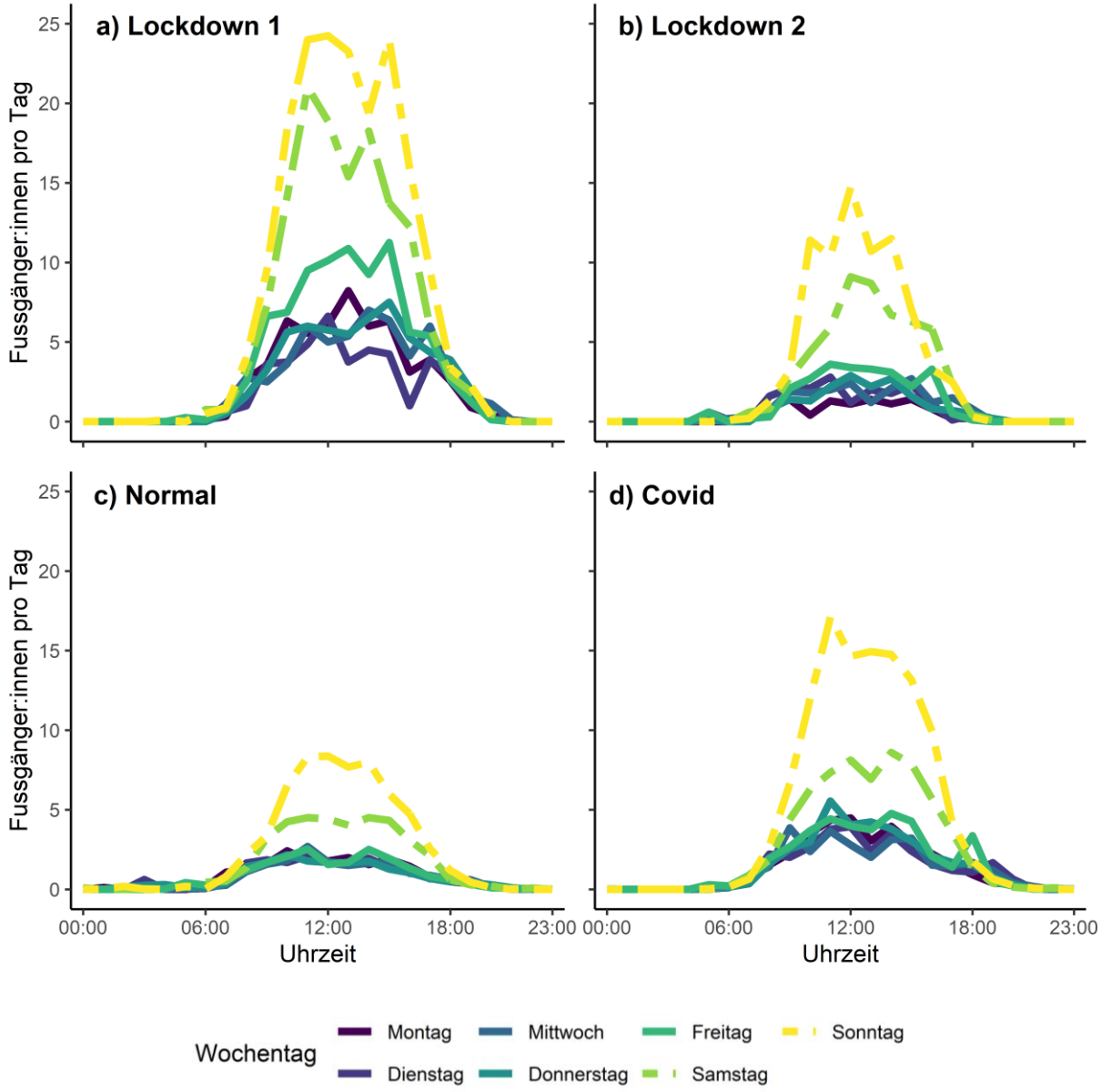

Abb. 2: Anzahl Fussgänger:innen, welche die Zählstelle 211 im Sihlwald während des Untersuchungszeitraums passierten. Vor allem während des ersten Lockdowns ist ein deutlicher Anstieg zu verzeichnen. Auffällig ist auch das erhöhte Besuchsaufkommen an Wochenenden.

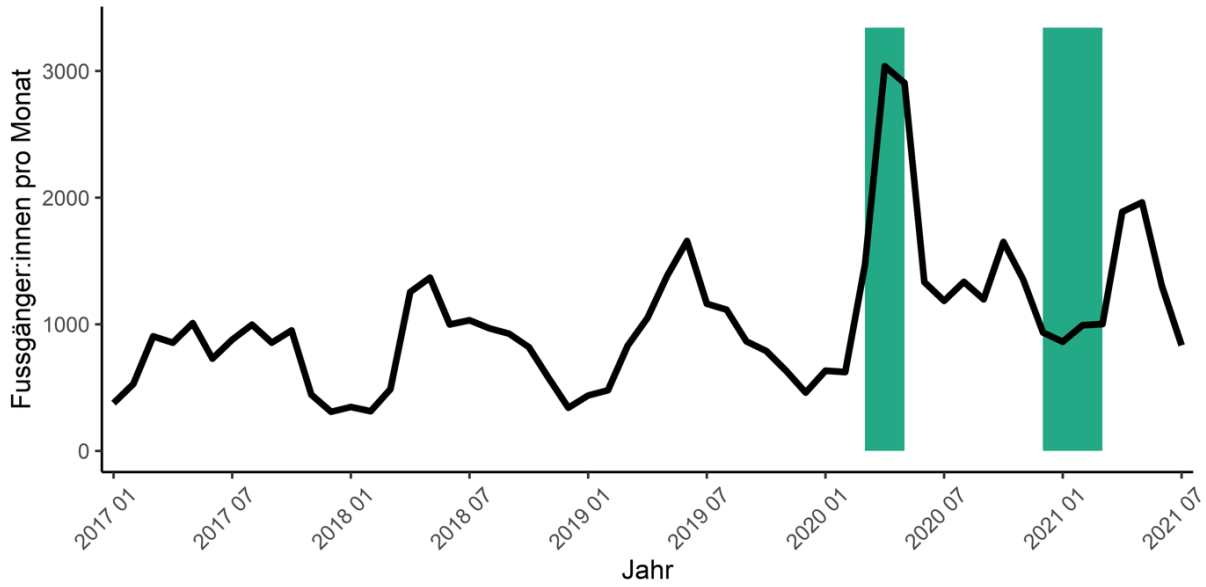

Abb. 3: Die beiden grün markierten Bereiche zeigen Lockdown 1 und Lockdown 2. Vor allem während des ersten Lockdowns besuchten deutlich mehr Leute den Sihlwald als vorher. Aber auch der zweite Lockdown, welcher in die Wintermonate fiel, zeigt, dass sich, verglichen mit einem Winter vor Corona, mehr Leute im Sihlwald aufhielten.

Die Auswertungen des GLMM haben ergeben, dass die Umweltparameter Temperatur, Niederschlag und Sonnenscheindauer, die Wochentage Freitag, Samstag und Sonntag sowie die einzelnen Phasen einen höchst signifikanten Einfluss auf das Besuchsaufkommen haben ($p<0.001$). Mit Ausnahme des Niederschlags wirken sich alle Parameter positiv auf die Besuchszahlen aus (Tab. 2). Auch Abb. 4 und 5 zeigen diese Zusammenhänge. Die Modellgüte (marginales R^2) liegt bei 58%.

Tab. 2: Die Ergebnisse des gerechneten Modells mit Angabe der unabhängigen Variablen (Predictors) und der jeweiligen geschätzten Werte (Estimates), Standardabweichungen (std. Error), Konfidenzintervalle (CI) und P-Werte (p). Die Berechnung erfolgte mit einer log-10-transformierten abhängigen Variable (Besuchsaufkommen). Drei Sterne kennzeichnen ein höchst signifikantes Ergebnis.

Predictors	*Estimates*	*std. Error*	*CI*	*p*
(Intercept)	1.13 ***	0.03	1.07 – 1.19	**<0.001**
Temperatur (tre200jx scaled)	0.09 ***	0.01	0.07 – 0.12	**<0.001**
Niederschlag (rre150j0 scaled)	-0.08 ***	0.01	-0.10 – -0.07	**<0.001**
Sonnenschein (sremaxdv scaled)	0.09 ***	0.01	0.07 – 0.11	**<0.001**
Wochentag [Dienstag]	0.04	0.02	-0.00 – 0.09	0.069
Wochentag [Mittwoch]	0.01	0.02	-0.03 – 0.06	0.617

Wochentag [Donnerstag]	0.03	0.02	-0.02 – 0.07	0.228
Wochentag [Freitag]	0.09 ***	0.02	0.05 – 0.14	**<0.001**
Wochentag [Samstag]	0.40 ***	0.02	0.36 – 0.44	**<0.001**
Wochentag [Sonntag]	0.59 ***	0.02	0.55 – 0.64	**<0.001**
Phase [Covid]	0.19 ***	0.03	0.14 – 0.25	**<0.001**
Phase [Lockdown 1]	0.33 ***	0.04	0.24 – 0.41	**<0.001**
Phase [Lockdown 2]	0.28 ***	0.04	0.20 – 0.36	**<0.001**

Random Effects

σ^2	0.06
$\tau_{00\ Kalenderwoche}$	0.01
$\tau_{00\ Jahr}$	0.00
ICC	0.17
$N_{\ Kalenderwoche}$	53
$N_{\ Jahr}$	5
Observations	1610
Marginal R^2 / Conditional R^2	**0.581 / 0.653**

** p<0.05 ** p<0.01 *** p<0.001*

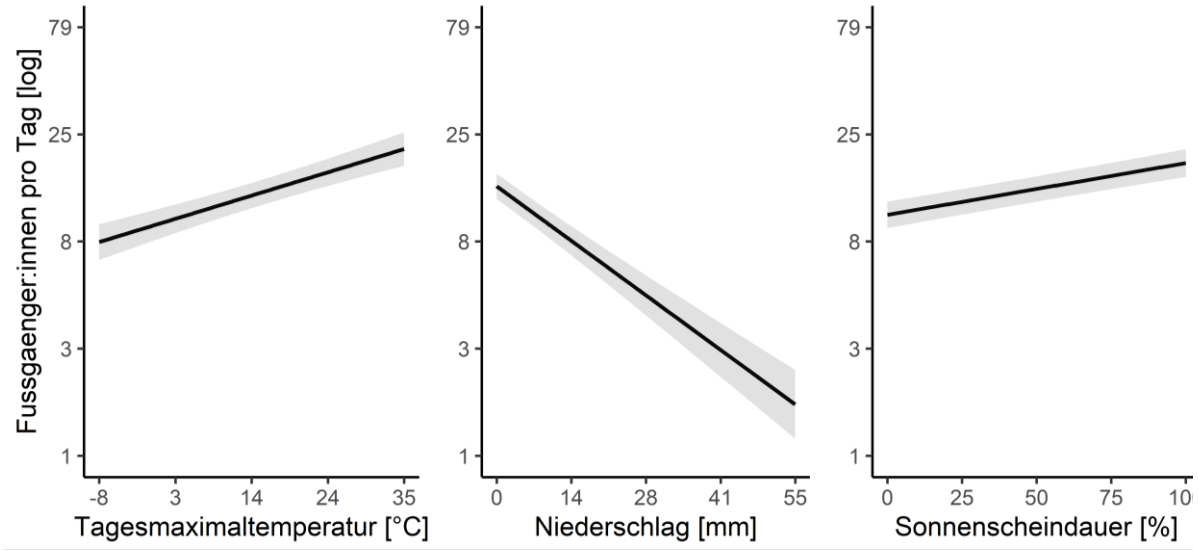

Abb. 4: Fussgänger:innen pro Tag in Abhängigkeit von Tagesmaximaltemperatur, Niederschlag und Sonnenscheindauer mit einem Konfidenzintervall von jeweils 95% ohne Interaktionen.

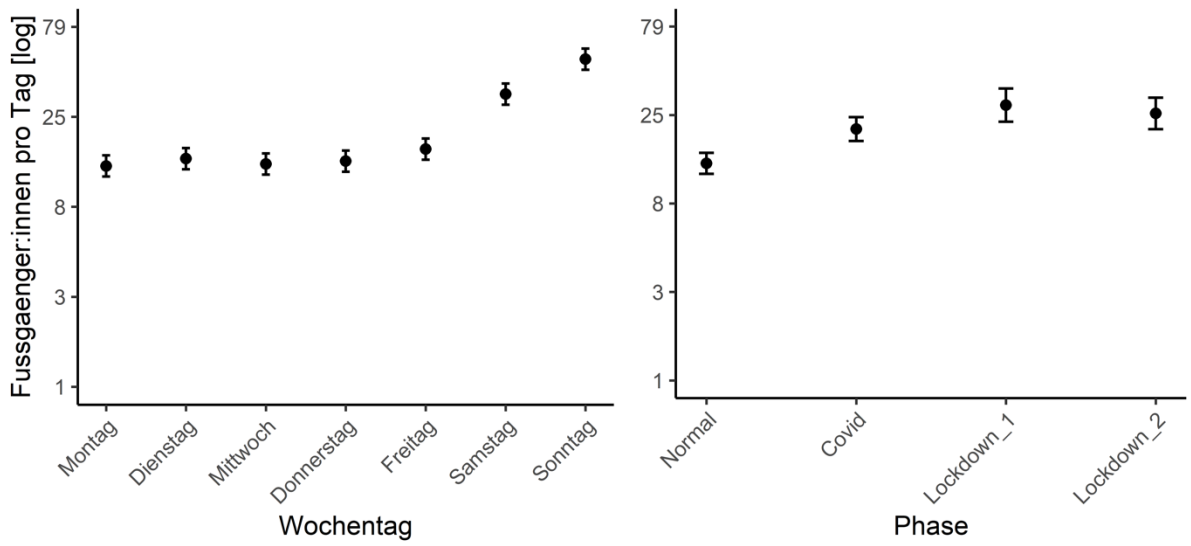

Abb. 5: Durchschnittliche Anzahl Fussgänger:innen pro Tag an den unterschiedlichen Wochentagen und in den verschiedenen Phasen mit einem Konfidenzintervall von 95% ohne Interaktionen.

5. Diskussion

Die Resultate bestätigen, dass Naherholungsgebiete für Stadtbewohnende wichtig sind und seit der Pandemie an Bedeutung gewonnen haben. Wie Studien zeigen, nutzte die Bevölkerung die Freiräume während der Lockdowns häufiger als sonst, wobei regionale Unterschiede festgestellt werden konnten (u.a. Derks et al. 2020, Egeter et al. 2020, Ramisch 2020, Schnabel-Jung & Wipf 2021, Suda et al. 2021). Das liegt daran, dass viele innerstädtische Grünräume während der Lockdowns gesperrt oder wegen des Social Distancings gemieden wurden (Hunziker et al. 2020).

Die dritte Erhebung des Waldmonitorings Schweiz endete kurz vor dem ersten Lockdown. Forschende der WSL erhielten so Gelegenheit, einen Teil der Befragten während des ersten Lockdowns erneut zu ihrem Waldbesuchsverhalten zu befragen. Hier zeigte sich, dass mehr Leute fast täglich den Wald aufsuchten (Wunderlich et al. 2021).

Entgegen den Befunden von Rupf et al. (2020) aus dem Badener Stadtwald während des ersten Lockdowns konnte in dieser Studie ein deutlicher Unterschied in den Besuchszahlen zwischen Werk- und Wochenendtagen festgestellt werden (Abb. 5), was darauf hindeutet, dass der Wildnispark Zürich primär am Wochenende aufgesucht wird.

Obwohl die Lockdowns einen Effekt auf die Besuchszahlen hatten und sich diese seither auf einem höheren Niveau eingependelt haben (Abb. 3), scheint auch das Wetter einen grossen Einfluss auf das Besuchsaufkommen zu haben. Während höhere Temperaturen und mehr Sonnenschein einen positiven Einfluss auf die Besuchszahlen haben, wirkt sich mehr Niederschlag negativ auf ebendiese

aus (Abb. 4). Das zeigt sich deutlich an den Zahlen für den Juli 2021, welche sich aufgrund des nasskalten Wetters kaum von denjenigen von Januar 2021 unterscheiden (Abb. 3). Millhäusler et al. (2016) konnten das Wetter in einer Studie im Schweizerischen Nationalpark ebenfalls als treibenden Faktor identifizieren.

Für das Management des Wildnisparks Zürich können folgende Empfehlungen abgeleitet werden:

- Ranger:innen sollten vermehrt an Wochenenden und bei schönem Wetter eingesetzt werden. Zu diesen Zeiten können am meisten Besuchende erreicht werden, um diese für die Bedürfnisse der Wildtiere und den Wald als wertvolles Ökosystem zu sensibilisieren.
- Während den Ferienzeiten drängen sich diesbezüglich keine verstärkten Bemühungen auf.
- Erhöhte Besuchszahlen sind gleichbedeutend mit einem höheren sozioökonomischen Wert, was als Argument für eine nachhaltige Erhöhung der Betriebsbeiträge genutzt werden kann.

In weiteren Studien wäre es spannend zu untersuchen, inwiefern sich die Profile der Erholungssuchenden seit der Pandemie verändert haben. Sollten sich unter den Besuchenden vermehrt «Wald-Novizen» befinden, könnte das Auswirkungen auf das Besuchermanagement haben. Verstärkte Sensibilisierungsarbeit wäre dann möglicherweise von Nöten, um die friedliche Koexistenz von Menschen und Wildtieren sicherzustellen.

Literaturverzeichnis

AfW. (2021). *Freizeit und Erholung im Wald in Zeiten von Covid-19*. Austausch von Forschenden zum Thema «Freizeit und Erholung im Wald in Zeiten von Covid-19». https://www.afw-ctf.ch/de/runde-waldtische/austausch-covid-19 (abgerufen am 04.12.2021)

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1). https://doi.org/10.18637/jss.v067.i01

Derks, J., Giessen, L., & Winkel, G. (2020). COVID-19-induced visitor boom reveals the importance of forests as critical infrastructure. *Forest Policy and Economics*, 118. https://doi.org/10.1016-/j.forpol.2020.102253

Egeter, M., Finger-Stich, A., Karn, S., Schellenberger, S., Siegrist, D., & Ketterer Bonnelame, L. (2020). Bleiben Sie zu Hause. Bitte. Alle. Zwei Befragungen zum Freizeitverhalten der Bevölkerung in Bezug auf Frei- und Grünräume während der Coronakrise in der Schweiz. *ILF Schriftenreihe,* 18. Rapperswil.

Freedman, D., Pisani, R., & Purves, R. (2007). *Statistics: Fourth International Student Edition*. W.W. Norton & Company. New York.

Hunziker, M., Bauer, N., Salak, B., & Hegetschweiler, K. T. (2020). Walderholung vor und während Corona-Lockdown: Ergebnisse zweier Befragungen derselben Personen im Rahmen von WaMos3. *Arbeitsgemeinschaft für den Wald AfW. Austausch von Forschenden zum Thema «Freizeit und Erholung im Wald in Zeiten von Covid-19»*, 5–8.

MeteoSchweiz. (2021). *Qualitätsbezeichnung der Messwerte*. https://gate.meteoswiss.ch/idaweb/-text/datenqualitaet_legende_de.pdf (abgerufen am 06.12.2021)

Millhäusler, A., Anderwald, P., Haeni, M., & Haller, R. M. (2016). Publicity, economics and weather – Changes in visitor numbers to a European National Park over 8 years. *Journal of Outdoor Recreation and Tourism*, 16, 50–57. https://doi.org/10.1016/j.jort.2016.09.005

Ramisch, K. (2020). Der Wald ruft lauter seit Corona. Eine Online-Studie zum Erleben und Verhalten von Waldbesuchern währen der Covid-19-Pandemie. *Arbeitsgemeinschaft für den Wald AfW. Austausch von Forschenden zum Thema «Freizeit und Erholung im Wald in Zeiten von Covid-19»*, 17–18.

R Core Team. (2020). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/ (abgerufen am 29.11.2021)

Rupf, R., Riesen, M., Hochreutener, A., Bärlocher, B., & Wyttenbach, M. (2020). «Immer nur Wochenende» – erste Erkenntnisse zur Erholungsnutzung im Badener Stadtwald während des Covid-19-Lockdowns. *Arbeitsgemeinschaft für den Wald AfW. Austausch von Forschenden zum Thema «Freizeit und Erholung im Wald in Zeiten von Covid-19»*, 14–16.

Schnabel-Jung, U., & Wipf, S. (2021). Extreme Besucherströme bringen neue Herausforderungen für Schutzgebiete. In: M. Bürgi, S. Tobias, M. Hunziker, N. Bauer, P. Bebi, & F. Kienast (Hrsg.). *WSL Berichte,* 15, 57–61. Birmensdorf.

Suda, M., Gaggermeier, A., Ramisch, K., & Koller, N. (2021). Was Waldbesucher im Wald finden. *LWF aktuell*, *128*, 12–14.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., Francois, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T., Miller, E., Bache, S., Müller, K., Ooms, J., Robinson, D., Seidel, D., Spinu, V., & Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686.

Wildnispark Zürich. (2021a). *Jahresbericht 2020*. https://www.wildnispark.ch/de/allgemein/ueber-uns/publikationen/jahresbericht-2020-322 (abgerufen am 04.12.2021)

Wildnispark Zürich. (2021b). *Naturwald*. *Ursprünglicher Buchenwald.* https://www.wildnispark.ch/-de/der-park/naturerlebnispark-sihlwald/naturwald (abgerufen am 04.12.2021)

Wunderlich, A. C., Salak, B., Hegetschweiler, T., Bauer, N., & Hunziker, M. (2021). The woods are calling: Auswirkungen der Corona-Pandemie auf die Schweizer Waldbesuche. In: M. Bürgi, S. Tobias, M. Hunziker, N. Bauer, P. Bebi, & F. Kienast (Hrsg.). *WSL Berichte,* 15, 49–52. Birmensdorf.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

```
#.####################################################################
# Corona -Pandemie und Wetter – Auswirkungen auf die Besuchszahlen im Wildnispark
Zürich. AutorInnen: Carina Kohler, Marigna Franck, Ursula Schoeni, Mirco
Nietlisbach ####
#.####################################################################

#.####################################################################
# METADATEN UND DEFINITIONEN ####
#.####################################################################

# Datenherkunft ####

# Wildnispark Sihlwald, Zürich
# Meteo Schweiz

# Benoetigte Bibliotheken ####

# nachfolgende Packages wurden bereits via install.packages("NAME") installiert ->
muessen jedoch immer wieder geladen werden:
library(tidyverse) # Data wrangling und piping
library(lubridate) # Arbeiten mit Datumsformaten
library(data.table)# schnelles Dateneinlesen
library(ggpubr)    # to arrange multiple plots in one graph
library(PerformanceAnalytics) # Plotte Korrelationsmatrix
library(MuMIn)     # Multi-Model Inference
library(AICcmodavg)# Modellaverageing
library(fitdistrplus)# Prueft die Verteilung in Daten
library(lme4)      # Multivariate Modelle
library(blmeco)    # Bayesian data analysis using linear models
library(sjPlot)    # Plotten von Modellergebnissen (tab_model)
library(lattice)   # einfaches plotten von Zusammenhängen zwischen Variablen
library(reshape2)
library(agricolae)

# Zeitliche Definitionen ####
# Die unterschiedlichen Zeitphasen (wie z.B. Ferien oder Lockdowns) definiert für
spätere Auswertungen.

# Start / Ende der Untersuchung
depo_start <- as.Date("2017-01-01")
depo_end <- as.Date("2021-07-31")

# Lockdowns 1 & 2
lock_1_start_2020 <- as.Date("2020-03-16")
lock_1_end_2020 <- as.Date("2020-05-11")

lock_2_start_2021 <- as.Date("2020-12-22")
lock_2_end_2021 <- as.Date("2021-03-01")

# Erste und letzte Kalenderwoche
woche_start <- week("2017-01-01")
woche_end <- week("2021-07-31")

# Ferienzeiten (https://www.schulferien.org/schweiz/ferien/2020/)
Winterferien_2016_start <- as.Date("2017-01-01")
Winterferien_2016_ende <- as.Date("2017-01-08")
```

```
Fruehlingsferien_2017_start <- as.Date("2017-04-15")
Fruehlingsferien_2017_ende <- as.Date("2017-04-30")
Sommerferien_2017_start <- as.Date("2017-07-15")
Sommerferien_2017_ende <- as.Date("2017-08-20")
Herbstferien_2017_start <- as.Date("2017-10-07")
Herbstferien_2017_ende <- as.Date("2017-10-22")
Winterferien_2017_start <- as.Date("2017-12-23")
Winterferien_2017_ende <- as.Date("2018-01-07")

Fruehlingsferien_2018_start <- as.Date("2018-04-21")
Fruehlingsferien_2018_ende <- as.Date("2018-05-06")
Sommerferien_2018_start <- as.Date("2018-07-14")
Sommerferien_2018_ende <- as.Date("2018-08-19")
Herbstferien_2018_start <- as.Date("2018-10-06")
Herbstferien_2018_ende <- as.Date("2018-10-21")
Winterferien_2018_start <- as.Date("2018-12-22")
Winterferien_2018_ende <- as.Date("2019-01-06")

Fruehlingsferien_2019_start <- as.Date("2019-04-20")
Fruehlingsferien_2019_ende <- as.Date("2019-05-05")
Sommerferien_2019_start <- as.Date("2019-07-13")
Sommerferien_2019_ende <- as.Date("2019-08-18")
Herbstferien_2019_start <- as.Date("2019-10-05")
Herbstferien_2019_ende <- as.Date("2019-10-20")
Winterferien_2019_start <- as.Date("2019-12-21")
Winterferien_2019_ende <- as.Date("2020-01-05")

Fruehlingsferien_2020_start <- as.Date("2020-04-11")
Fruehlingsferien_2020_ende <- as.Date("2020-04-26")
Sommerferien_2020_start <- as.Date("2020-07-11")
Sommerferien_2020_ende <- as.Date("2020-08-16")
Herbstferien_2020_start <- as.Date("2020-10-03")
Herbstferien_2020_ende <- as.Date("2020-10-18")
Winterferien_2020_start <- as.Date("2020-12-19")
Winterferien_2020_ende <- as.Date("2021-01-03")

Fruehlingsferien_2021_start <- as.Date("2021-04-24")
Fruehlingsferien_2021_ende <- as.Date("2021-05-09")
Sommerferien_2021_start <- as.Date("2021-07-17")
Sommerferien_2021_ende <- as.Date("2021-07-31")

#.####################################################################
# 1. DATENIMPORT ####
#.####################################################################

# Zähldaten
depo <- read_delim("_data/211_sihlwaldstrasse_2017_2021.csv", delim=";",
locale=locale(encoding="UTF-8"))

# Umweltdaten
meteo <- read_delim("_data/order_97149_data.csv", delim=";",
locale=locale(encoding="UTF-8"))

#.####################################################################
# 2. VORBEREITUNG ####
#.####################################################################

# Zähldaten pro Stunde: depo
############################################################
```

```
# Sichtung der Daten
str(depo)
head(depo)

#Datum und Uhrzeit sind im Moment in einer Spalte. Diese werden für die
Auswertungen getrennt
depo <- depo %>%
  mutate(Datum_Uhrzeit = as.character(DatumUhrzeit)) %>%
  separate(Datum_Uhrzeit, into = c("Datum", "Zeit"), sep = " ")%>%
  # mit separate() trennt man 1 Spalte in 2.
  mutate(Datum = as.Date(Datum, format = "%d.%m.%Y"))
  # hier wird Text zum Datum

# Auswahl der Fussgänger Daten (Fahrradfahrer / Zeit_Datum werden für die
Datenanalyse nicht mehr benötigt)
depo <- depo[,c(2,3,6,7)]

# Neue Spalte: Total / Ausschliessen von NA-Werten
depo <- depo %>%
  mutate(Total = Fuss_IN + Fuss_OUT) %>%
  na.omit(depo, invert = FALSE)

# .Convenience Variable Wochentag ####
depo$Wochentag <- weekdays(depo$Datum)

# Sortierung Levels der Convenience Variable: Wochentag
depo <- depo %>%
  mutate(Wochentag = base::factor(Wochentag, levels = c("Montag", "Dienstag",
"Mittwoch", "Donnerstag", "Freitag", "Samstag", "Sonntag")))

# .Convenience Variable: Wochenende ####
depo <- depo %>%
  mutate(Wochenende = if_else(Wochentag == "Montag" | Wochentag == "Dienstag" |
Wochentag == "Mittwoch" | Wochentag == "Donnerstag" | Wochentag == "Freitag",
"Werktag", "Wochenende"))

# .Convenience Variable: Kalenderwoche, Monat und Jahr ####
depo <- depo %>%
  mutate(KW = week(Datum)) %>%
  mutate(Monat = month(Datum)) %>%
  mutate(Jahr = year (Datum))

depo$Monat = format(depo$Datum, "%m")

# .Convenience Variable: Phase ####
depo <- depo %>%
  mutate(Phase = if_else(Datum >= lock_1_start_2020 & Datum <=
        lock_1_end_2020, "Lockdown_1", if_else(Datum >=
        lock_2_start_2021 & Datum <= lock_2_end_2021, "Lockdown_2",
        if_else(Datum < lock_1_start_2020, "Normal", "Covid"))))


# Sortierung Levels der Convenience Variable: Wochentag
depo <- depo %>%
  mutate(Phase = base::factor(Phase, levels = c("Normal", "Covid", "Lockdown_1",
"Lockdown_2")))

# Prüfen welche Werte die Variable "Phase" hat
```

```
unique(depo$Phase)

# Umwandlung in Factors
depo$Wochenende <- as.factor(depo$Wochenende)
depo$KW <- as.factor(depo$KW)
depo$Phase <- as.factor(depo$Phase)

# Convenience Variable: Stunde (als Integer)
depo$Stunde <- as.numeric(format(as.POSIXct(depo$Zeit,format="%H:%M"),"%H"))


# IN, OUT und Total müssen ganzzahlig sein
depo$Fuss_IN <- round(depo$Fuss_IN, digits = 0)
depo$Fuss_IN <- as.integer(depo$Fuss_IN)
depo$Fuss_OUT <- round(depo$Fuss_OUT, digits = 0)
depo$Fuss_OUT <- as.integer(depo$Fuss_OUT)
depo$Total <- round(depo$Total, digits = 0)
depo$Total <- as.integer(depo$Total)

# Zähldaten pro Tag: depo_d
############################################################

# Aggregierung der Stundendaten nach Datum
depo_d <- depo %>%
  group_by(Datum, Wochentag, Wochenende, KW, Monat, Jahr, Phase) %>%
  summarise(Total = sum(Fuss_IN + Fuss_OUT), Fuss_IN = sum(Fuss_IN), Fuss_OUT =
sum(Fuss_OUT))

summarize(depo_d)
head(depo_d)

# Zähldaten pro Monat: depo_m
############################################################

# Aggregierung der Stundendaten nach Monat
depo_m <- depo %>%
  group_by(Monat, Jahr) %>%
  summarise(Total = sum(Fuss_IN + Fuss_OUT))

# Neue Spalte "Ym" als Jahr + Monat
depo_m <- as.data.frame(depo_m)
# Sortierung anhand zwei Spalten aufsteigend (damit die Reihenfolge sicher stimmt)
depo_m[with(depo_m, order(Jahr, Monat)),]

depo_m <- depo_m %>%
  mutate(Jahr = as.factor(Jahr)) %>% # Jahr als Faktor
  mutate(Monat = as.factor(Monat)) %>% # Monat als Faktor
  mutate(Ym = paste(Jahr, Monat)) %>% # neue Variable
  mutate(Ym= factor(Ym, levels=unique(Ym))) # neue Variable als Faktor
# die Levels sind die einzelnen Einträge in der Spalte (welche ja bereits geordnet
sind)

# Umweltdaten: meteo ########################################################

# Datum von Integer zu Text umwandeln
meteo <- transform(meteo, time = as.Date(as.character(time), "%Y%m%d"))

# rre150j0 als numerisch
meteo$rre150j0 <- as.numeric(meteo$rre150j0)
```

```
# stn als faktor
meteo$stn <- as.factor(meteo$stn)

# Zuschneiden auf den gewünschten Zeitraum / das Komma hat die gleiche Funktion
wie ein &
meteo <- meteo %>%
  filter(time >= depo_start, time <=  depo_end)

# Ausschliessen der NA-Werte
meteo <- meteo %>%
  filter(!is.na(stn))%>%
  filter(!is.na(time))%>%
  filter(!is.na(tre200jx))%>%
  filter(!is.na(rre150j0))%>%
  filter(!is.na(sremaxdv))

# Überprüfen ob alle NA-Werte gelöscht wurden
(sum(is.na(meteo$time)))
(sum(is.na(meteo$tre200jx)))
(sum(is.na(meteo$rre150j0)))
(sum(is.na(meteo$sremaxdv)))
(sum(is.na(meteo$stn)))

# Struktur überprüfen
str(meteo)

#.####################################################################
# 3. DESKRIPTIVE ANALYSE UND VISUALISIERUNG ####
#.####################################################################

# Definition der Farben
pal <- hcl.colors(5,palette = "Heat")

# Besucheraufkommen pro Monat -> Facets Small Multiples
ggplot(depo_m, aes(Jahr,Total,group=1)) +
  geom_point() +
  geom_line() +
  facet_wrap(Monat~.)

# Verlauf Besucheraufkommen pro Monat
ggplot(depo_m, aes(Monat,Total,group=Jahr,colour=Jahr)) +
  scale_color_manual(values = pal) +
  geom_point() +
  geom_line() +
  labs(
    x = "Monat",
    y = "Besucherzahlen Total"
  )

# Histogramm Small Multiples für Besucheraufkommen pro Wochentage und Jahr
ggplot(depo_d) +
  geom_col(mapping = aes(x = Wochentag, y = Total)) +
  facet_wrap(Jahr~.)

# Histogramm Small Multiples für Wochentag/Wochenende pro Jahr
ggplot(depo_d) +
  geom_col(mapping = aes(x = Wochenende, y = Total)) +
  facet_wrap(Jahr~.)
```

```
## Fussgänger:innen pro Monat im Jahresverlauf ####
depo_m$Ym <- as.character(depo_m$Ym)

ggplot(depo_m, aes(Ym, Total, group=1)) +
  geom_rect(mapping = aes(xmin="2020 03", xmax="2020 05",
                          ymin =0, ymax=max(Total+(Total/100*10))),
            fill = "#22A884FF", alpha = 0.4, colour = NA) +
  geom_rect(mapping = aes(xmin="2020 12", xmax="2021 03",
                          ymin =0, ymax=max(Total+(Total/100*10))),
            fill = "#22A884FF", alpha = 0.4, colour = NA) +
  scale_x_discrete(breaks = c("2017 01", "2017 07", "2018 01", "2018 07", "2019
01", "2019 07", "2020 01", "2020 07", "2021 01", "2021 07"),
                   labels = c("2017 01", "2017 07", "2018 01", "2018 07", "2019
01", "2019 07", "2020 01", "2020 07", "2021 01", "2021 07"))+
  theme_classic(base_size = 10)+
  geom_line(size=1.2) + #Linie erst nach Rechteck, damit Linie im Vordergrund ist
  labs(
    x = "Jahr",
    y = "Fussgänger:innen pro Monat"
  ) +
  theme(axis.text.x=element_text(angle=45, hjust=0.7, vjust = 0.7))

ggsave("_results/Zahl über Jahre.png", width=16, height=8, units="cm", dpi=1000)

# Gruppieren nach Wochentag und Phase
mean_phase_wd <- depo_d %>%
  group_by(Wochentag, Phase) %>%
  summarise(Total = sum(Fuss_IN + Fuss_OUT))

# neues Dataframe wird als .csv gespeichert
write.csv(mean_phase_wd, "_results/mean_phase_wd.csv")


## Boxplot Wochengang ####
ggplot(data = depo_d) +
  geom_boxplot(mapping = aes(x= Wochentag, y = Total, fill = Phase)) +
  labs(
    x = "Wochentag",
    y = "Fussgänger:innen pro Tag"
  )

# Speichern des Resultats
ggsave("_results/FussgängerInnen pro Wochentag_Phase.png", width = 20, height =
10, units = "cm", dpi = 1000)

# Alternativhypothese: Das Besucheraufkommen am Wochenende und an Wochentagen ist
unterschiedlich
# Nullhypothese: Das Besucheraufkommen am Wochenende und an Wochentagen ist nicht
unterschiedlich oder nur zufällig unterschiedlich.

# Statistik: Unterschied WE und WO waehrend Lockdown 1
t.test(depo_d$Total [depo_d$Phase == "Lockdown_1" & depo_d$Wochenende=="Werktag"],
depo_d$Total [depo_d$Phase == "Lockdown_1" & depo_d$Wochenende=="Wochenende"])
# t = -6.2742, df = 19.345, p-value = 4.643e-06
# p-Wert < 0.05 -> Unterschied höchst signifikant

# Statistik: Unterschied WE und WO waehrend Lockdown 2
```

```
t.test(depo_d$Total [depo_d$Phase == "Lockdown_2" & depo_d$Wochenende=="Werktag"],
depo_d$Total [depo_d$Phase == "Lockdown_2" & depo_d$Wochenende=="Wochenende"])
# t = -6.732, df = 21.912, p-value = 9.322e-07
# p-Wert < 0.05 -> Unterschied höchst signifikant

# Statistik: Unterschied WE und WO waehrend Covid
t.test(depo_d$Total [depo_d$Phase == "Covid" & depo_d$Wochenende=="Werktag"],
depo_d$Total [depo_d$Phase == "Covid" & depo_d$Wochenende=="Wochenende"])
# t = -9.8826, df = 124.09, p-value < 2.2e-16
# p-Wert < 0.05 -> Unterschied höchst signifikant

# Statistik: Unterschied WE und WO waehrend Normal
t.test(depo_d$Total [depo_d$Phase == "Normal" & depo_d$Wochenende=="Werktag"],
depo_d$Total [depo_d$Phase == "Normal" & depo_d$Wochenende=="Wochenende"])
# t = -16.877, df = 406.36, p-value < 2.2e-16
# p-Wert < 0.05 -> Unterschied höchst signifikant

## Tagesgang mit Mittelwert pro Stunde ####

# Gruppieren nach Wochentag, Zeit und Phase
Mean_h <- depo %>%
  group_by(Wochentag, Zeit, Phase) %>%
  summarise(Total = mean(Fuss_IN + Fuss_OUT))

## ANOVA-Test ####
# Prüfen, ob es signifikante Unterschiede zwischen den einzelnen Wochentagen in
den unterschiedlichen Phasen

# Normal-Phase
aov.Normal <- aov(Total[Phase=="Normal"]~Wochentag[Phase=="Normal"], data=depo)
summary(aov.Normal)
# p-Wert 2*10^16
# Post-Hoc Test
HSD.test(aov.Normal, "Wochentag", group=T, console=T)

# Covid-Phase
aov.Covid <- aov(Total[Phase=="Covid"]~Wochentag[Phase=="Covid"], data=depo)
summary(aov.Covid)
# p-Wert 2*10^16
# Post-Hoc Test
HSD.test(aov.Covid, "Wochentag", group=T, console=T)

# Lockdown_1
aov.Lockdown1 <- aov(Total[Phase=="Lockdown_1"]~Wochentag[Phase=="Lockdown_1"],
data=depo)
summary(aov.Lockdown1)
# p-Wert 2*10^16
# Post-Hoc Test
HSD.test(aov.Lockdown1, "Wochentag", group=T, console=T)

# Lockdown_2
aov.Lockdown2 <- aov(Total[Phase=="Lockdown_2"]~Wochentag[Phase=="Lockdown_2"],
data=depo)
summary(aov.Lockdown2)
# p-Wert 2*10^16
# Post-Hoc Test
HSD.test(aov.Lockdown2, "Wochentag", group=T, console=T)

## Tagesgang unterteilt nach Wochentagen ####
```

```
## .Normal ####
tag_norm <- ggplot(subset(Mean_h, Phase %in% c("Normal")),
                   mapping=aes(x = Zeit, y = Total, colour = Wochentag, linetype =
Wochentag, group=Wochentag)) +
  geom_line(size=1.5)+
  scale_y_continuous(limits=c(0,25))+
  scale_colour_viridis_d()+
  scale_linetype_manual(values = c(rep("solid", 5), "twodash", "twodash"))+
  scale_x_discrete(breaks=c("00:00", "06:00" , "12:00", "18:00", "23:00"))+
  theme(plot.margin=unit(c(1,0.5,0.0,0.5), "cm"))+
  labs(y = "Fussgänger:innen pro Tag", x="Uhrzeit")+
  theme_classic(base_size = 10)

## .Covid ####
tag_Covid <- ggplot(subset(Mean_h, Phase %in% c("Covid")),
                    mapping=aes(x = Zeit, y = Total, colour = Wochentag, linetype
= Wochentag, group=Wochentag)) +
  geom_line(size=1.5)+
  scale_y_continuous(limits=c(0,25))+
  scale_colour_viridis_d()+
  scale_linetype_manual(values = c(rep("solid", 5), "twodash", "twodash"))+
  scale_x_discrete(breaks=c("00:00", "06:00" , "12:00", "18:00", "23:00"))+
  theme(plot.margin=unit(c(1,0.5,0.0,0.5), "cm"))+
  labs(y = "Fussgänger:innen pro Tag", x="Uhrzeit")+
  theme_classic(base_size = 10)

## .Lockdown 1 ####
tag_Lockdown1 <- ggplot(subset(Mean_h, Phase %in% c("Lockdown_1")),
                        mapping=aes(x = Zeit, y = Total, colour = Wochentag,
linetype = Wochentag, group=Wochentag)) +
  geom_line(size=1.5)+
  scale_y_continuous(limits=c(0,25))+
  scale_colour_viridis_d()+
  scale_linetype_manual(values = c(rep("solid", 5), "twodash", "twodash"))+
  scale_x_discrete(breaks=c("00:00", "06:00" , "12:00", "18:00", "23:00"))+
  theme(plot.margin=unit(c(1,0.5,0.0,0.5), "cm"))+
  labs(y = "Fussgänger:innen pro Tag", x="Uhrzeit")+
  theme_classic(base_size = 10)

## .Lockdown 2 ####
tag_Lockdown2 <- ggplot(subset(Mean_h, Phase %in% c("Lockdown_2")),
                        mapping=aes(x = Zeit, y = Total, colour = Wochentag,
linetype = Wochentag, group=Wochentag)) +
  geom_line(size=1.5)+
  scale_y_continuous(limits=c(0,25))+
  scale_colour_viridis_d()+
  scale_linetype_manual(values = c(rep("solid", 5), "twodash", "twodash"))+
  scale_x_discrete(breaks=c("00:00", "06:00" , "12:00", "18:00", "23:00"))+
  theme(plot.margin=unit(c(1,0.5,0.0,0.5), "cm"))+
  labs(y = "Fussgänger:innen pro Tag", x="Uhrzeit")+
  theme_classic(base_size = 10)

tag_norm
tag_Lockdown1
tag_Lockdown2
tag_Covid
```

```
#Die vier Plots in einem arrangieren
ggarrange(tag_Lockdown1+ # plot 1 aufrufen
            rremove("x.text")+ # plot 1 braucht es nicht alle Achsenbeschriftungen
            rremove("x.title"),
          tag_Lockdown2+ # plot 2 aufrufen
            rremove("y.text")+ # bei plot 2 brauchen wir keine Achsenbeschriftung
            rremove("y.title")+
            rremove("x.text")+
            rremove("x.title"),
          tag_norm,
          tag_Covid+
            rremove("y.text")+
            rremove("y.title"),
          ncol = 2, nrow = 2, # definieren, wie die plots angeordnet werden
          heights = c(0.9, 1), # beide plots sind wegen der fehlenden Beschriftung
nicht gleich hoch
          widths = c(1,0.9),
          labels = c("a) Lockdown 1", "b) Lockdown 2", "c) Normal", "d) Covid"),
          font.label = list(size=11),
          label.x = c(0.01,-0.1,0.06,-0.04), # wo stehen die Plottitel
          label.y = 0.97,
          common.legend = TRUE, legend = "bottom") # wir brauchen nur eine
Legende, unten


ggsave("_results/Tagesgang.png", width=16, height=16, units="cm", dpi=1000)

## Kennzahlen berechnen ####

# Summe IN und OUT berechnen
sum_phase <- depo_d %>%
  group_by(Phase) %>%
  summarise(Total = sum(Total), Fuss_IN = sum(Fuss_IN), Fuss_OUT = sum(Fuss_OUT))

# als .csv speichern
write.csv(sum_phase, "_results/sum_phase.csv")

# Mittelwerte IN und OUT berechnen
mean_phase_d <- depo_d %>%
  group_by(Phase) %>%
  summarise(Total = mean(Total), Fuss_IN = mean(Fuss_IN), Fuss_OUT =
mean(Fuss_OUT))

# Prozentualen Anteil auf eine Nachkommastelle gerundet anfügen
mean_phase_d <- mean_phase_d %>%
  mutate(Proz_IN = round((Fuss_IN/Total)*100, 1)) %>%
  mutate(Proz_OUT = round((Fuss_OUT/Total)*100, 1))
  # berechnen und auf eine Nachkommastelle runden

write.csv(mean_phase_d, "_results/mean_phase_d.csv")

# Spalten filtern
mean_phase_d_abs <- mean_phase_d[,-c(2,5,6), drop=FALSE]
mean_phase_d_proz <- mean_phase_d[,-c(2,3,4), drop=FALSE]

# In eine Long-Table umwandeln
mean_phase_d_abs <- reshape2::melt(mean_phase_d_abs, measure.vars = c("Fuss_IN",
"Fuss_OUT"), value.name = "Durchschnitt",variable.name = "Gruppe")
```

```
mean_phase_d_proz <- reshape2::melt(mean_phase_d_proz, measure.vars =
c("Proz_IN","Proz_OUT"), value.name = "Durchschnitt",variable.name = "Gruppe")

# Balkendiagramme erstellen
ggplot(data = mean_phase_d_abs) +
  geom_bar(mapping = aes(x = Fuss_IN, fill = Phase)) +
  facet_wrap(Fuss_IN~Fuss_OUT) +
  labs(
  x = "Bewegungsrichtung",
  y = "Durchschnitt [mean]"
  )

abs <- ggplot(data = mean_phase_d_abs, mapping = aes(x = Gruppe, y = Durchschnitt,
fill = Phase))+
  geom_col(position = "dodge", width = 0.8)+
  scale_fill_manual(values = c("royalblue", "red4", "orangered", "gold2"), name =
"Phase")+
  scale_x_discrete(labels = c("IN", "OUT"))+
  labs(y = "Durchschnitt [mean]", x= "Bewegungsrichtung")+
  theme_classic(base_size = 15)+
  theme(legend.position = "bottom")

proz <- ggplot(data = mean_phase_d_proz, mapping = aes(x = Gruppe, y =
Durchschnitt, fill = Phase))+
  geom_col(position = "dodge", width = 0.8)+
  scale_fill_manual(values = c("royalblue", "red4", "orangered", "gold2"), name =
"Phase")+
  scale_x_discrete(labels = c("IN", "OUT"))+
  labs(y = "Durchschnitt [%]", x= "Bewegungsrichtung")+
  theme_classic(base_size = 15)+
  theme(legend.position = "bottom")

# Arrange und Export Verteilung
ggarrange(abs,            # plot 1 aufrufen
          proz,            # plot 2 aufrufen
          ncol = 2, nrow = 1,    # definieren, wie die plots angeordnet werden
          heights = c(1),        # beide sind bleich hoch
          widths = c(1,0.95),    # plot 2 ist aufgrund der fehlenden y-
achsenbesch. etwas schmaler
          labels = c("a) Absolute Verteilung", "b) Relative Verteilung"),
          label.x = 0,        # wo stehen die labels
          label.y = 1.0,
          common.legend = TRUE, legend = "bottom") # wir brauchen nur eine
Legende, unten

ggsave("Verteilung.png", width=20, height=15, units="cm", dpi=1000, path =
"_results/")

#.####################################################################
# 4. MULTIFAKTORIELLE ANALYSE UND VISUALISIERUNG ####
#.####################################################################

# Mittels Join werden die Datensätze zusammengeführt
umwelt <- left_join(depo_d, meteo, by = c("Datum" = "time"))

# nicht benötigte Spalten entfernen
umwelt <- umwelt %>%
```

```
  dplyr::select(-c(Fuss_IN, Fuss_OUT, stn)) # dplyr braucht es, weil es mehrere
selects sind.

# Wieviele NA-Werte?
sum(is.na(umwelt))

# NA-Werte entfernen
umwelt <- na.omit(umwelt)
# 10 Zeilen werden entfernt

#Ferienzeit als Convenience Variable hinzufügen -> Ferien=1, sonst=0
umwelt <- umwelt %>%
  mutate(Ferien = if_else(
    Datum >= Winterferien_2016_start & Datum <= Winterferien_2016_ende |
    Datum >= Fruehlingsferien_2017_start & Datum <= Fruehlingsferien_2017_ende |
    Datum >= Sommerferien_2017_start & Datum <= Sommerferien_2017_ende |
    Datum >= Herbstferien_2017_start & Datum <= Herbstferien_2017_ende |
    Datum >= Winterferien_2017_start & Datum <= Winterferien_2017_ende |
    Datum >= Fruehlingsferien_2018_start & Datum <= Fruehlingsferien_2018_ende |
    Datum >= Sommerferien_2018_start & Datum <= Sommerferien_2018_ende |
    Datum >= Herbstferien_2018_start & Datum <= Herbstferien_2018_ende |
    Datum >= Winterferien_2018_start & Datum <= Winterferien_2018_ende |
    Datum >= Fruehlingsferien_2019_start & Datum <= Fruehlingsferien_2019_ende |
    Datum >= Sommerferien_2019_start & Datum <= Sommerferien_2019_ende |
    Datum >= Herbstferien_2019_start & Datum <= Herbstferien_2019_ende |
    Datum >= Winterferien_2019_start & Datum <= Winterferien_2019_ende |
    Datum >= Fruehlingsferien_2020_start & Datum <= Fruehlingsferien_2020_ende |
    Datum >= Sommerferien_2020_start & Datum <= Sommerferien_2020_ende |
    Datum >= Herbstferien_2020_start & Datum <= Herbstferien_2020_ende |
    Datum >= Winterferien_2020_start & Datum <= Winterferien_2020_ende |
    Datum >= Fruehlingsferien_2021_start & Datum <= Fruehlingsferien_2021_ende |
    Datum >= Sommerferien_2021_start,"1", "0"))

# Ferien und Jahr in Faktoren umwandeln
umwelt$Ferien <- as.factor(umwelt$Ferien)
umwelt$Jahr <- as.factor(umwelt$Jahr)

# Abfrage KW
is.factor(umwelt$KW) # [1] TRUE

umwelt <- as.data.frame(umwelt) # Umwandlung in dataframe (Achtung: mit Abfrage
is.dataframe(umwelt), gibt es ein TRUE, doch es stellt sich heraus dass es doch
ein Tibble ist)

# Umweltvariablen skalieren
umwelt <- umwelt %>%
  mutate(tre200jx_scaled = scale(tre200jx)) %>%
  mutate(rre150j0_scaled = scale(rre150j0)) %>%
  mutate(sremaxdv_scaled = scale(sremaxdv))

## Korrelationsmatrix ####
cor <- cor(umwelt[,13:15])
#                   tre200jx_scaled rre150j0_scaled sremaxdv_scaled
#tre200jx_scaled      1.00000000     -0.06381999       0.5469517
#rre150j0_scaled     -0.06381999      1.00000000      -0.3199216
#sremaxdv_scaled      0.54695170     -0.31992158       1.0000000

cor[abs(cor) < 0.7] <-  0 #Setzt alle Werte kleiner 0.7 auf 0
cor
```

```
?cor
# mit Schwellenwert 0.7 ergeben sich keine kritischen Korrelationen
# -> entsprechend muss keine Variable ausgeschlossen werden.
# bei Schwellenwert 0.5 würde es jedoch eine kritische Korrelation geben:
# Lufttemperatur und Sonnenscheindauer.

# Korrelationsgrafik
chart.Correlation(umwelt[,13:15], histogram=TRUE, pch=19)

## Variablenselektion mit dredge-Funktion ####
f <- Total ~ Wochentag + Ferien + Phase + tre200jx_scaled + rre150j0_scaled +
sremaxdv_scaled
# Jetzt kommt der Random-Factor hinzu und es wird eine Formel daraus gemacht
# Random-Factors: Kalenderwochen & Jahr (diese Effekte untersuchen wir nicht)
f_dredge <- paste(c(f, "+ (1|KW)", "+ (1|Jahr)"), collapse = " ") %>%
  as.formula()

f_dredge
# Das Modell mit dieser Formel ausführen
m <- glmer(f_dredge, data = umwelt, family = poisson, na.action = "na.fail")
# Das Modell in die dredge-Funktion einfügen (siehe auch ?dredge)
all_m <- dredge(m)
# suche das beste Modell
print(all_m)
# Importance values der Variablen
# hier wird die wichtigkeit der Variablen in den verschiedenen Modellen abgelesen
MuMIn::importance(all_m)

# Model selection table
# (Int) Frn Phs r15_scl srm_scl t20_scl Wch df     logLik    AICc    delta weight
#  64 3.116 +   + -0.21  0.18  0.16     + 16  -9932.464 19897.3     0.00  0.999
#  63 3.136 +   -0.21  0.18   0.16      + 15  -9940.261 19910.8    13.55  0.001
# -> eindeutiges Modell?

# Schliesslich wird ein Modelaverage durchgeführt
# Schwellenwert für das delta-AIC = 2
avgmodel <- model.avg(all_m, rank = "AICc", subset = delta < 2)
summary(avgmodel)


## Verteilung der abhängigen Variable prüfen ####

f1 <- fitdist(umwelt$Total,"norm")  # Normalverteilung
f2 <- fitdist(umwelt$Total,"pois")  # Poisson
f3 <- fitdist(umwelt$Total,"nbinom")  # negativ binomial
f4 <- fitdist(umwelt$Total,"exp")  # exponentiell
f6 <- fitdist(umwelt$Total,"logis")  # logistisch
f7 <- fitdist(umwelt$Total,"geom")  # geometrisch

gofstat(list(f1,f2,f3,f4,f6,f7),
        fitnames = c("Normalverteilung", "Poisson","negativ
binomial","exponentiell","logistisch","geometrisch"))

# die 4 besten (gemaess Akaike's Information Criterion) als Plot,
plot.legend <- c("exponentiell", "negativ binomial", "geometrisch", "logistisch")
# vergleicht mehrere theoretische Verteilungen mit den empirischen Daten
cdfcomp(list(f4, f3, f7, f6), legendtext = plot.legend)

# zum Vergleich mit Poisson-Verteilung
```

```
plot.legend <- c("exponentiell", "negativ binomial", "geometrisch", "logistisch",
"poisson")
cdfcomp(list(f4, f3, f7, f6, f2), legendtext = plot.legend)

## Modelle berechnen #####

## Poisson Modell
tagesmodel_pois <- glmer(Total ~ Wochentag + Ferien + Phase + tre200jx_scaled +
rre150j0_scaled + sremaxdv_scaled + (1 | KW) + (1 | Jahr), family = poisson, data
= umwelt)

summary(tagesmodel_pois) # Zeigt das Resultat des Modells

# Modelldiagnostik tagesmodel_pois
plot(tagesmodel_pois, type = c("p", "smooth")) # Verteilung der Residuen
qqmath(tagesmodel_pois) # Pruefen auf Normalverteilung

# Overdispersion describes the observation that variation is higher than would be
expected.
dispersion_glmer(tagesmodel_pois) #it shouldn't be over 1.4

# zeige die erklaerte Varianz (je hoeher r2m ist, desto besser!)
r.squaredGLMM(tagesmodel_pois) # nicht relevant da overdispersion

## negativ binomial Modell
tagesmodel_nb <- glmer.nb(Total ~ Wochentag + Ferien + Phase + tre200jx_scaled +
rre150j0_scaled + sremaxdv_scaled + (1 | KW) + (1 | Jahr), data = umwelt)

summary(tagesmodel_nb) # Zeigt das Resultat des Modells

# Modelldiagnostik tagesmodel_nb
plot(tagesmodel_nb, type = c("p", "smooth")) # Verteilung der Residuen
qqmath(tagesmodel_nb) # Pruefen auf Normalverteilung

# Overdispersion describes the observation that variation is higher than would be
expected.
dispersion_glmer(tagesmodel_nb) #it shouldn't be over 1.4

# zeige die erklaerte Varianz (je hoeher r2m ist, desto besser!)
r.squaredGLMM(tagesmodel_nb)

## exponentielles Modell
tagesmodel_exp <- glmer((Total + 1) ~ Wochentag + Ferien + Phase + tre200jx_scaled
+ rre150j0_scaled + sremaxdv_scaled + (1 | KW) + (1 | Jahr), family = Gamma(link =
"log"), data = umwelt)

summary(tagesmodel_exp) # Zeigt das Resultat des Modells

# Modelldiagnostik tagesmodel_exp
plot(tagesmodel_exp, type = c("p", "smooth")) # Verteilung der Residuen
qqmath(tagesmodel_exp) # Pruefen auf Normalverteilung

# Overdispersion describes the observation that variation is higher than would be
expected.
dispersion_glmer(tagesmodel_exp) #it shouldn't be over 1.4

# zeige die erklaerte Varianz (je hoeher r2m ist, desto besser!)
r.squaredGLMM(tagesmodel_exp)
```

```
## Transformation prüfen ####
library(moments)
skewness(umwelt$Total)
# 2.478808 --> rechtsschiefe Verteilung


## log10-Transformation der abhängigen Variable ####
tagesmodel_log <- glmer(log10(Total + 1) ~ Wochentag + Ferien + Phase +
tre200jx_scaled + rre150j0_scaled + sremaxdv_scaled + (1 | KW) + (1 | Jahr), data
= umwelt)

summary(tagesmodel_log) # Zeigt das Resultat des Modells

# Verteilung der Residuen
plot(tagesmodel_log, type = c("p", "smooth"))

# Pruefen auf Normalverteilung
qqmath(tagesmodel_log)

# Overdispersion describes the observation that variation is higher than would be
expected.
dispersion_glmer(tagesmodel_log) #it shouldn't be over 1.4

# zeige die erklaerte Varianz (je hoeher r2m ist, desto besser!)
r.squaredGLMM(tagesmodel_log) # R2m = 0.582915

## Alternative Modelle ####

# Annahme: "Es gehen weniger Leute in den Wald, wenn es zu heiss ist."
# Quadratischer Term: Temperatur
# Testen

tagesmodel_log_quad <- lmer(log10(Total + 1) ~ Wochentag + Ferien + Phase +
tre200jx_scaled + I(tre200jx_scaled^2) + rre150j0_scaled + sremaxdv_scaled + (1 |
KW) + (1 | Jahr), data = umwelt)

summary(tagesmodel_log_quad) # Zeigt das Resultat des Modells

# Modelldiagnostik tagesmodel_log_quad
plot(tagesmodel_log_quad, type = c("p", "smooth")) # Verteilung der Residuen
qqmath(tagesmodel_log) # Pruefen auf Normalverteilung
dispersion_glmer(tagesmodel_log_quad) #it shouldn't be over 1.4 -> 0.239
r.squaredGLMM(tagesmodel_log_quad) # R2m = 0.5839901 -> besser


## Interaktionsmodelle ####

# Interaktionsmodell 1: Temperatur * Regen
# Annahme: Im Winter hat Niederschlag einen negativeren Effekt als im Sommer
# Ausgangslage: tagesmodel_log_squad

tagesmodel_log_int1 <- lmer(log10(Total + 1) ~ Wochentag + Ferien + Phase +
tre200jx_scaled + I(tre200jx_scaled^2) * rre150j0_scaled + sremaxdv_scaled + (1 |
KW) + (1 | Jahr), data = umwelt)

summary(tagesmodel_log_int1)

# Modelldiagnostik tagesmodel_log_int1
plot(tagesmodel_log_int1, type = c("p", "smooth")) # Verteilung der Residuen
```

```
qqmath(tagesmodel_log_int1) # Pruefen auf Normalverteilung
dispersion_glmer(tagesmodel_log_int1) #it shouldn't be over 1.4 -> 0.239
r.squaredGLMM(tagesmodel_log_int1) # R2m = 0.5836229 -> schlechter

# Interaktionsmodell 2: Wochentag * Ferien
# Annahme: An den Wochentagen in den Ferien sind Allgemein mehr Menschen unterwegs
# Ausgangslage: tagesmodel_log_squad

tagesmodel_log_int2 <- lmer(log10(Total + 1) ~ Wochentag * Ferien + Phase +
tre200jx_scaled + I(tre200jx_scaled^2) + rre150j0_scaled + sremaxdv_scaled + (1 |
KW) + (1 | Jahr), data = umwelt)

summary(tagesmodel_log_int2)

# Interaktionsmodell 3: Temperatur * Regen / Wochentag * Ferien
# Annahme: beide obigen Hypothesen kombiniert
# Ausgangslage: tagesmodel_log_squad

tagesmodel_log_int3 <- lmer(log10(Total + 1) ~ Wochentag * Ferien + Phase +
tre200jx_scaled + I(tre200jx_scaled^2) * rre150j0_scaled + sremaxdv_scaled + (1 |
KW) + (1 | Jahr), data = umwelt)

summary(tagesmodel_log_int3)


# Vergleich der Modellguete mittels AICC ####

cand.models <- list()
# cand.models[[1]] <- Tages_Model
cand.models[[1]] <- tagesmodel_log
cand.models[[2]] <- tagesmodel_log_quad
cand.models[[3]] <- tagesmodel_log_int1
cand.models[[4]] <- tagesmodel_log_int2
cand.models[[5]] <- tagesmodel_log_int3

Modnames <- c("tagesmodel_log", "tagesmodel_log_quad", "tagesmodel_log_int1",
"tagesmodel_log_int2", "tagesmodel_log_int3")

aictab(cand.set = cand.models, modnames = Modnames)

# Model selection based on AICc:

#  K   AICc Delta_AICc AICcWt Cum.Wt Res.LL
# tagesmodel_log      17 191.42       0.00   0.99   0.99 -78.52
# tagesmodel_log_quad 18 201.38       9.96   0.01   1.00 -82.48
# tagesmodel_log_int1 19 209.36      17.94   0.00   1.00 -85.44
# tagesmodel_log_int2 24 222.57      31.15   0.00   1.00 -86.91
# tagesmodel_log_int3 25 230.10      38.67   0.00   1.00 -89.64


tab_model(tagesmodel_log, transform = NULL, show.se = TRUE)
# Fazit: Variable "Ferien" nicht signifikant -> prüfen via AICc

tagesmodel_log_oF <- lmer(log10(Total + 1) ~ Wochentag + Phase + tre200jx_scaled +
rre150j0_scaled + sremaxdv_scaled + (1 | KW) + (1 | Jahr), data = umwelt)

summary(tagesmodel_log_oF) # Zeigt das Resultat des Modells

# Verteilung der Residuen
```

```
plot(tagesmodel_log_oF, type = c("p", "smooth"))

# Pruefen auf Normalverteilung
qqmath(tagesmodel_log_oF)

# Overdispersion describes the observation that variation is higher than would be
expected.
dispersion_glmer(tagesmodel_log_oF) #it shouldn't be over 1.4

# zeige die erklaerte Varianz (je hoeher r2m ist, desto besser!)
r.squaredGLMM(tagesmodel_log_oF) # R2m = 0.582915

# Vergleich der Modellguete mittels AICC ####

cand.models <- list()
cand.models[[1]] <- tagesmodel_log
cand.models[[2]] <- tagesmodel_log_quad
cand.models[[3]] <- tagesmodel_log_int1
cand.models[[4]] <- tagesmodel_log_int2
cand.models[[5]] <- tagesmodel_log_int3
cand.models[[6]] <- tagesmodel_log_oF

Modnames <- c("tagesmodel_log", "tagesmodel_log_quad", "tagesmodel_log_int1",
"tagesmodel_log_int2", "tagesmodel_log_int3", "tagesmodel_log_oF")

aictab(cand.set = cand.models, modnames = Modnames)

# Model selection based on AICc:

#  K   AICc Delta_AICc AICcWt Cum.Wt Res.LL
# tagesmodel_log_oF   16 187.14       0.00   0.89   0.89 -77.40
# tagesmodel_log      17 191.42       4.28   0.10   1.00 -78.52
# tagesmodel_log_quad 18 201.38      14.24   0.00   1.00 -82.48
# tagesmodel_log_int1 19 209.36      22.23   0.00   1.00 -85.44
# tagesmodel_log_int2 24 222.57      35.43   0.00   1.00 -86.91
# tagesmodel_log_int3 25 230.10      42.96   0.00   1.00 -89.64

# Modell ohne Ferien hat den besten AIC Wert


## Modellresultate ####

min(umwelt$tre200jx_scaled)
max(umwelt$tre200jx_scaled)

labels.t <- round(seq(floor(min(umwelt$tre200jx)), ceiling(max(umwelt$tre200jx)),
length.out = 5), 0)

Tempplot <- plot_model(tagesmodel_log_oF, type = "pred", terms = "tre200jx_scaled
[all]", title = "", axis.title = c("Tagesmaximaltemperatur [°C]",
"Fussgaenger:innen pro Tag [log]")) +
  scale_x_continuous(breaks = c(-2.47,-1.3,-0.13,1.04,2.21),
                     labels = c(labels.t))+
  scale_y_continuous(breaks = c(-0.1,0.4,0.9,1.4,1.9),
                     labels = round(c(10^-0.1, 10^0.4, 10^0.9, 10^1.4, 10^1.9),0),
                     limits = c(-0.1, 1.9))+
  theme_classic(base_size = 10)

Tempplot
```

```
ggsave("_results/temp.png", width=15, height=15, units="cm", dpi=1000)

## .Niederschlag ####

min(umwelt$rre150j0_scaled)
max(umwelt$rre150j0_scaled)

labels.r <- round(seq(floor(min(umwelt$rre150j0)), ceiling(max(umwelt$rre150j0)),
length.out = 5 ), 0)

Regenplot <- plot_model(tagesmodel_log_oF, type = "pred", terms = "rre150j0_scaled
[all]", title = "", axis.title = c("Niederschlag [mm]", "Fussgaenger:innen pro Tag
[log]")) +
  scale_x_continuous(breaks = c(-0.37,2.63,5.63,8.63,11.63),
                     labels = c(labels.r))+
  scale_y_continuous(breaks = c(-0.1,0.4,0.9,1.4,1.9),
                     labels = round(c(10^-0.1, 10^0.4, 10^0.9, 10^1.4, 10^1.9),0),
                     limits = c(-0.1, 1.9))+
  theme_classic(base_size = 10)

Regenplot
ggsave("_results/Regen.png", width=15, height=15, units="cm", dpi=1000)

## .Sonnenscheindauer ####

min(umwelt$sremaxdv_scaled)
max(umwelt$sremaxdv_scaled)

labels.s <- round(seq(floor(min(umwelt$sremaxdv)), ceiling(max(umwelt$sremaxdv)),
length.out = 5), 0)

Sonnenplot <- plot_model(tagesmodel_log_oF, type = "pred", terms =
"sremaxdv_scaled [all]", title = "", axis.title = c("Sonnenscheindauer [%]",
"Fussgaenger:innen pro Tag [log]")) +
  scale_x_continuous(breaks = c(-1.15,-0.47,0.21,0.89,1.57), labels =
c(labels.s))+
  scale_y_continuous(breaks = c(-0.1,0.4,0.9,1.4,1.9), labels = round(c(10^-0.1,
10^0.4, 10^0.9, 10^1.4, 10^1.9),0),
                     limits = c(-0.1, 1.9))+
  theme_classic(base_size = 10)

Sonnenplot
ggsave("_results/Sonne.png", width=15, height=15, units="cm", dpi=1000)

## .Covid Phasen ####

Phasenplot <- plot_model(tagesmodel_log_oF, type = "pred", terms = "Phase", title
= "", axis.title = c("Phase", "Fussgaenger:innen pro Tag [log]"), dot.size = 1.5,
line.size = 0.5) +
  scale_y_continuous(breaks = c(-0.1,0.4,0.9,1.4,1.9),
                     labels = round(c(10^-0.1, 10^0.4, 10^0.9, 10^1.4, 10^1.9),0),
                     limits = c(-0.1, 1.9))+
  theme_classic(base_size = 10)+
  theme(axis.text.x=element_text(angle=45, hjust=1, vjust =1))

Phasenplot
ggsave("_results/Phase.png", width=15, height=15, units="cm", dpi=1000)
```

```
## .Wochentag ####

Tagplot <- plot_model(tagesmodel_log_oF, type = "pred", terms = "Wochentag", title
= "", axis.title = c("Wochentag", "Fussgaenger:innen pro Tag [log]"),
colors="blue", dot.size = 1.5, line.size = 0.5 ) +
  scale_y_continuous(breaks = c(-0.1,0.4,0.9,1.4,1.9),
                     labels = round(c(10^-0.1, 10^0.4, 10^0.9, 10^1.4, 10^1.9),0),
                     limits = c(-0.1, 1.9))+
  theme_classic(base_size = 10, base_rect_size = 1)+
  theme(axis.text.x=element_text(angle=45, hjust=1, vjust =1))

Tagplot
ggsave("_results/Tag.png", width=15, height=15, units="cm", dpi=1000)

#Die drei Wetterplots in einem arrangieren

ggarrange(Tempplot,
          Regenplot+
            rremove("y.title"),
          Sonnenplot+
            rremove("y.title"),
          ncol = 3, nrow = 1, # definieren, wie die plots angeordnet werden
          heights = c(1, 1,1),
          widths = c(1.1,1,1))

ggsave("_results/Umweltparameter.plot.png", width=16, height=8, units="cm",
dpi=1000)

#Die beiden Plots der deskriptiven Variabeln werden in einem Plot arragniert

ggarrange(Tagplot,
          Phasenplot+
            rremove("y.title"),
          ncol = 2, nrow = 1, # definieren, wie die plots angeordnet werden
          heights = c(1), # beide plots sind wegen der fehlenden Beschriftung
nicht gleich hoch
          widths = c(1.1,1))

ggsave("_results/TagePhasen.plot.png", width=16, height=8, units="cm", dpi=1000)


#Als nächstes wurde ein Interaktionsplot durchgeführt, um zu schauen,
#ob sich der Einfluss der Ferien durch Covid verändert hat

interaction.plot(umwelt$Phase, umwelt$Ferien, umwelt$Total)
?interaction.plot
#Dieser Interaktionsplot zeigt deutlich, dass während dem 1. Lockdown die Ferien
#einen stärkeren Einfluss haben
#Dies kann jedoch auch daher führen, dass während dem 1. Lockdown verhältnismässig
viele Ferientage vorkommen

#Aus diesem Grund wurde als nächstes die drei Covid-Phasen in eine zusammengefasst
#Nur der Unterschied zwischen vor Covid und nach Covid wird betrachtet

umwelt$Wann[umwelt$Phase == "Lockdown_1"] <- "nach_Covid"
umwelt$Wann[umwelt$Phase == "Lockdown_2"] <- "nach_Covid"
umwelt$Wann[umwelt$Phase == "Covid"] <- "nach_Covid"
umwelt$Wann[umwelt$Phase == "Normal"] <- "vor
```

```
_Covid"

interaction.plot(umwelt$Wann, umwelt$Ferien, umwelt$Total)
interaction.plot(umwelt$Ferien, umwelt$Wann, umwelt$Total)
#Der Interaktionsplot zeigt, dass kaum eine Interaktion der Ferien vor und nach
Covid erkennbar ist
```