EINFLUSS DES LOCKDOWNS AUF DAS BESUCHERAUFKOMMEN IM WILDNISPARK ZÜRICH

Abschlussbericht zur multivariaten Analyse

Fallstudie Research Methods MSc ENR HS20

Lea Kostiza, Nicole Mastai & Tobias Wildhaber

Gruppe 1

Zürcher Hochschule für Angewandte Wissenschaften
Research Methods / HS20

Bild Titelseite: Freizeitnutzung am Waldrand während des Lockdowns im Frühling 2020
Bildquelle: Meienberg (2020)

Modul: Research Methods HS 20 / Ecosystems & Biodiversity / Profil S

Korrektoren: Reto Rupf, Adrian Hochreutener

Abgabedatum: 08.01.2021

Inhaltsverzeichnis

1. Einleitung

Naherholungsräume ermöglichen Aktivitäten an der frischen Luft, bei der die Besuchenden sich von Stress erholen können. Dadurch haben sie einen positiven Einfluss auf die physische und psychische Gesundheit (Abraham, Sommerhalder, Bolliger-Salzmann & Abel, 2007). Ein solcher Naherholungsraum ist der Wildnispark Zürich, der sich in der Nähe der Stadt Zürich befindet.

Seit einigen Jahren wird dort ein automatisches Besuchermonitoring durchgeführt mittels Besucherzählstationen. Ein Teil dieser erhobenen Daten wurde ausgewertet, um herauszufinden, inwieweit der Lockdown einen Einfluss auf das Besucheraufkommen im Wildnispark Zürich hatte. In dieser Arbeit wird der Begriff Lockdown verwendet für die Zeit zwischen dem 6.3.-11.5. 2020, als in der Schweiz als Massnahme gegen die Ausbreitung von Covid-19 ein partieller Shutdown galt. In diesem Zeitrahmen waren alle nicht lebensnotwendigen Geschäfte geschlossen, die Schulen waren zu und viele Erwerbstätige arbeiteten von zuhause aus. Es gab in der Schweiz zwar keine Ausgangsbeschränkung, aber die Bevölkerung wurde dazu aufgerufen, zuhause zu bleiben. Einige sonst stark frequentierte Freiräume, wie beispielsweise die Seepromenade in Zürich, waren zudem abgesperrt.

Erste Studien deuten auf regionale Unterschiede hinsichtlich der Nutzung von Wäldern und anderen Naherholungsräumen während des Lockdowns in der Schweiz hin (Hunziker, Bauer, Salak & Hegetschweiler, 2020; Siegrist, Finger-Stich, Ketteter Bonnelame & Egeter, 2020). In der Deutschschweiz und in den Städten gaben während des Lockdowns deutlich mehr befragte Personen an, öfter in den Wald zu gehen, als noch vor dem Lockdown (Hunziker et al., 2020). Auswertungen von automatische Besucherzählungen in stadtnahen Wäldern in Deutschland (Kottenforst bei Bonn) und der Deutschschweiz (Badener Stadtwald) zeigen ebenso einen deutlichen Anstieg der Anzahl Besucher während des Lockdowns (Derks, Giessen & Winkel, 2020; Hunziker et al., 2020).

2. Fragestellung

In dieser Fallstudie wurde untersucht, wie sich der Lockdown auf das Besucherverhalten im Wildnispark Zürich ausgewirkt hat. Die konkreten Fragestellungen waren:

Welche Einflussfaktoren (Lockdown, Wetterparameter, Ferien, Wochentag, Kalenderwoche) erklären am besten das Besucheraufkommen?

Unterscheiden sich die Tages-, Wochen- und Saisongänge während des Lockdowns 2020 signifikant von denen im gleichen Zeitraum 2019?

3. Methoden

3.1. Untersuchungsgebiet

Das rund 1100 ha grosse Naturschutzgebiet Sihlwald, welches im periurbanen Raum südlich von Zürich liegt, gilt seit dem 1. Januar 2010 als erster national anerkannter Naturerlebnispark. Er ist Teil des Wildnisparks Zürich und seine Rolle als Naherholungsgebiet ist von grosser Bedeutung (Roth & Stauffer, 2010). Das untersuchte Besucheraufkommen bezieht sich auf die Zählstelle 502 am Sihluferweg, der von Sihlbrugg Richtung Gattikon verläuft.

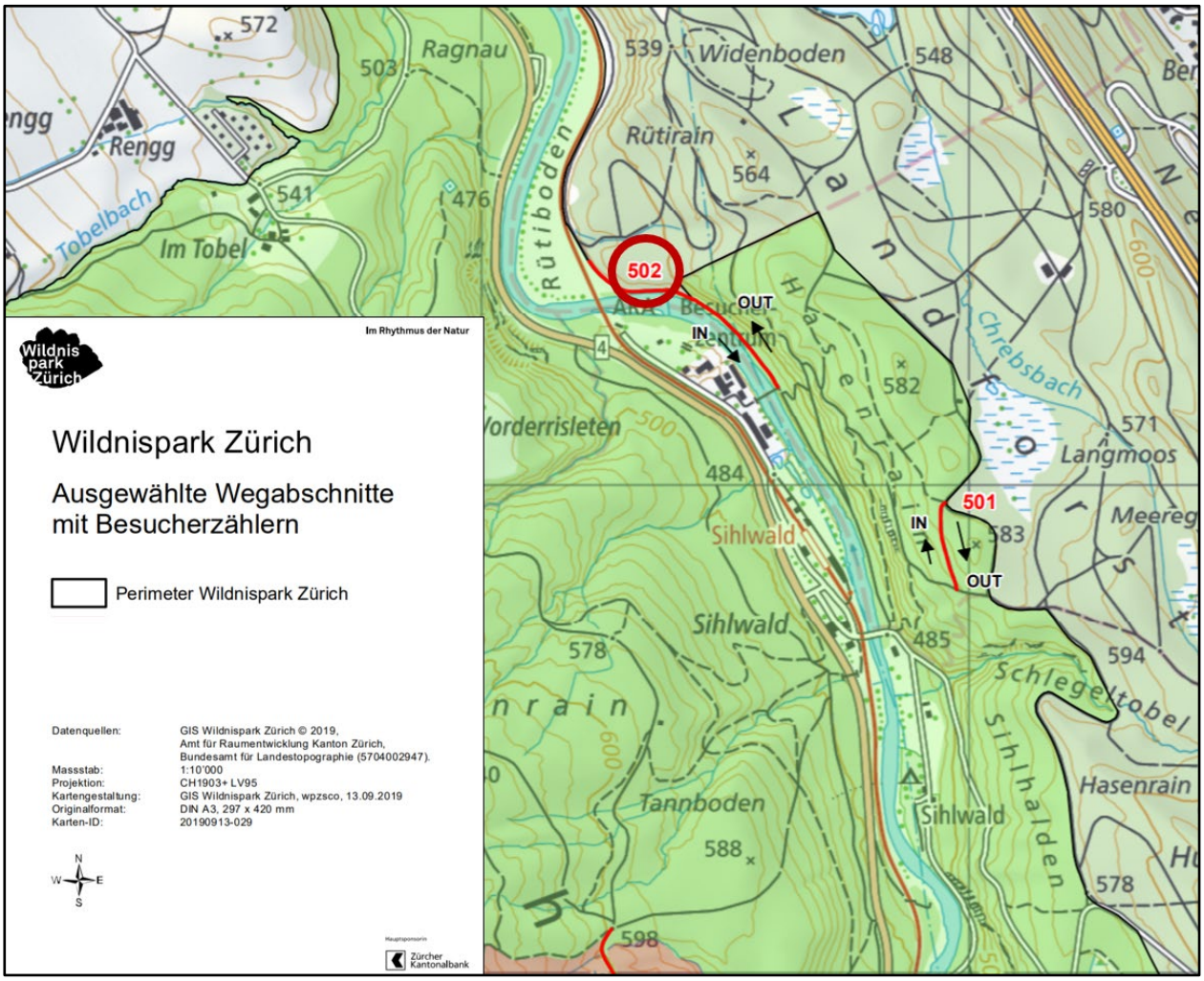

Abbildung 1: Übersicht einiger ausgewählten Wegabschnitte zur Erfassung der Besucherzahlen. Rot eingekreist der Standort der in diesem Bericht ausgewerteten Zählstelle 502 (verändert nach Stiftung Wildnispark Zürich, 2019)

3.2. Datenerhebung

Die Daten zu den Besucherzahlen wurden vom Wildnispark Zürich erfasst, kalibriert und zur Verfügung gestellt (Ronald Schmidt, Projektleiter Forschung, Monitoring und GIS, Stiftung Wildnispark Zürich). Die Analyse begrenzte sich auf Fahrradfahrende, die an der Zählstation 502 gezählt wurden (Abbildung 1). Diese Zählstation erfasst Personen mittels eines Pyrosensors, welcher auf den Temperaturunterschied einer Person zur Umgebung reagiert, sowie einer Induktionsschlaufe, welche die metallenen Felgen von Fahrrädern erkennt.

3.3. Datenanalyse

Um die Besucherzahlen in Bezug auf verschiedenen Einflussfaktoren zu untersuchen, wurde ein Generalized linear mixed effect model (GLMM) konstruiert. Als abhängige Variable wurden die täglichen Zählungen der Fahrräder der Zählstation 502 in beide Fahrtrichtungen (IN und OUT) für den Zeitraum vom 1. Jan. 2019 bis 31. Juni 2020 definiert. Als erklärenden Variablen wurden meteorologische und zeitbasierte Parameter verwendet (Tabelle 1).
Als Zufallsfaktoren (random factors) dienten die Variablen Kalenderwoche (1–52) und Jahr (2019/2020).

Quelle für die meteorologischen Daten (Niederschlag, Temperatur, Sonnenschein) war die Messstation Wädenswil (MeteoSchweiz, 2020). Die absoluten Messdaten für die Sonnenscheindauer wurden in Abhängigkeit von der Tageslänge (basierend auf Redwoods, 2020; bezogen auf die bürgerliche Dämmerung) als relative Werte in das Modell aufgenommen. Die Angaben zu den Schulferien stammen von Johannsen (2020).

Die meteorologischen Daten wurden vor der Aufnahme in das Modell skaliert, um deren Einfluss vergleichen zu können. Ein Korrelationstest nach Kendall (Kendall'sches Tau; 1938) der drei skalierten Variablen Niederschlag, Temperatur und Sonnenscheindauer mit einem Schwellenwert von 0.7 ergab eine unkritische Korrelation (0.5) zwischen der Temperatur und der Sonnenscheindauer. Alle drei Variablen wurden beibehalten.

Für das erste, globale Modell wurden alle Variablen ohne Interaktion aufgenommen. Die Variable Temperatur wurde als quadrierter Term implementiert, um einen unimodalen Einfluss zu berücksichtigen. In weiteren Modellen wurden auch Interaktionen zwischen den Variablen geprüft.

Die optische Modelldiagnostik zeigte eine Abweichung von einer Normalverteilung und eine Overdispersion in den Residuen. Eine logarithmische Transformation der Daten führte zu keiner Verbesserung, weshalb darauf verzichtet wurde. Um der Overdispersion Rechnung zu tragen, wurde ein GLMM mit einer negativen Binomialverteilung angewendet. Die definitive Modellauswahl fand mittels Multimodel Inference (Burnham & Anderson, 2004) statt. Die Modellgüte wurde mittels AICc und Vergleich des marginalen R^2 beurteilt.

Die Vergleiche zwischen dem Lockdown im Frühling 2020 und demselben Zeitraum 2019 wurden deskriptiv anhand verschiedener Visualisierungen (Tages-, Wochen- und Saisongang) durchgeführt.

Die Analyse erfolgte in der open-source Software R, Version 4.0.3 (R Core Team, 2020). Die multivariaten Modelle wurden mit den Packages lme4 (Bates, Mächler, Bolker & Walker, 2015) MuMIn (Barton, 2020) und blmeco (Korner-Nievergelt et al., 2015), die grafischen Darstellungen mit ggplot2 (Wickham, 2016), erstellt.

Tabelle 1: Auflistung der in die Modellberechnungen eingeflossenen abhängigen Variablen. Minimal- und Maximalwerte der meteorologischen Variablen beziehen sich auf die unskalierten Daten.

Variable	Beschreibung [Einheit]	Min – Max	Datentyp
Niederschlag	Halbtagessumme des Niederschlags; 6 UTC - 18 UTC [mm]	0–36.7	kontinuierlich
Temperatur	Lufttemperatur 2m über Boden; Tagmaximum; 6 UTC bis 18 UTC [°C]	-1.6–24.6	kontinuierlich
Sonnenscheindauer	Tagessumme der Sonnenscheindauer im Verhältnis zur Tageslänge [%]	0–94.4	kontinuierlich
Lockdown	Zeitraum des Lockdowns vom 16. März. bis zum 11. Mai 2020 (ja/nein) [-]	0–1	diskret (binär)
Ferien	Schulferien für den Kanton Zürich [-]	0–1	diskret (binär)
Wochentag	Montag bis Sonntag [-]	1–7	diskret

4. Resultate

Im Untersuchungszeitraum (578 Tage) wurden 116 243 Fahrräder gezählt. Der tiefste Wert lag bei 2, der höchste bei 1586 und der Median bei 101 Zählungen.
Das finale Modell erklärt 83% der Variabilität (marginales R^2) anhand der selektierten Variablen (Tabelle 2).

Von den getesteten Interaktionen wurde nur die zwischen den Variablen Wochentag und Ferien beibehalten. Zwar sind nur einzelne Terme dieser Interaktion signifikant, dennoch schneidet das Modell besser ab als ohne diese Interaktion (ΔAICc = 2.67).

Den höchsten Einfluss auf die Besucherzahl hatte die Variable Sonntag, gefolgt von den Variablen Samstag und Lockdown [ja] und Ferien [ja]. Den geringsten signifikanten Einfluss hatten der Sonnenschein und der Niederschlag (linear und quadriert).

Während des Lockdowns waren die Besucherzahlen höher als ausserhalb dieses Zeitraums (Abbildung 2a) sowie an den Wochenenden höher als an Werktagen (Abbildung 2b). Während der Schulferien waren die Zahlen ebenfalls höher (Abbildung 2c).

Mit dem Anstieg der Temperatur steigen auch die Besucherzahlen, wobei es bei hohen Temperaturen zu einer Abflachung kommt (Abbildung 3d; Man beachte den breiten Konfidenzintervall am oberen Ende der Regressionslinie). Der Niederschlag hat einen negativen Einfluss auf die Besucherzahlen, (Abbildung 3e). Der Einfluss der Sonnenscheindauer ist positiv (Abbildung 3f).

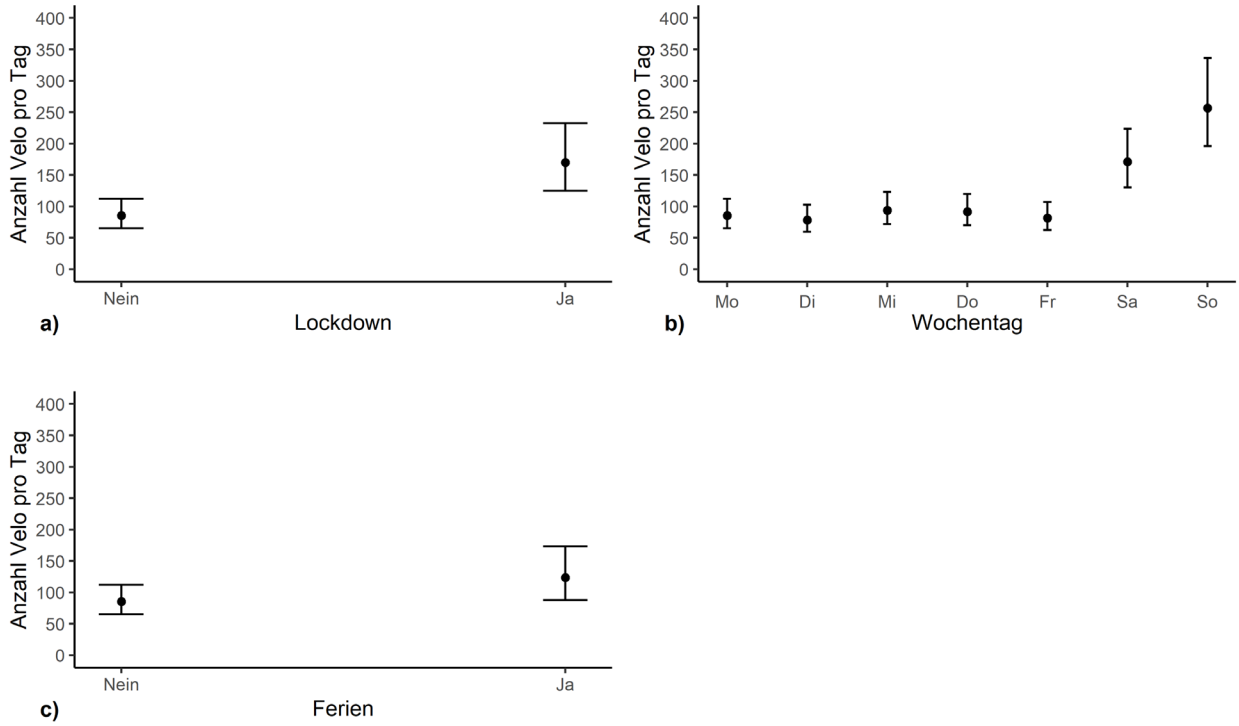

Abbildung 2: Einfluss der Zeitvariablen auf das finale Modell mit 95% Konfidenzintervall.

Tabelle 2: Übersicht des finalen Modells mit Interaktionen. Estimates (Log-Mean) = Schätzer (untransformiert), std. Error = Standardfehler, CI = Konfidenzintervall. Die Estimates für die Wochentage verwenden den Montag als Referenzwert.

Variablen	Estimates (Log-Mean)	std. Error	CI	p
(Intercept)	4.45	0.14	4.18 – 4.72	<0.001
Lockdown [ja]	0.69	0.09	0.51 – 0.86	<0.001
Dienstag	-0.09	0.08	-0.25 – 0.07	0.282
Mittwoch	0.09	0.08	-0.07 – 0.25	0.253
Donnerstag	0.07	0.08	-0.09 – 0.23	0.401
Freitag	-0.04	0.08	-0.20 – 0.11	0.580
Samstag	0.69	0.08	0.53 – 0.85	<0.001
Sonntag	1.10	0.08	0.94 – 1.26	<0.001
Ferien [ja]	0.37	0.14	0.10 – 0.64	0.008
Temperatur	0.69	0.04	0.61 – 0.77	<0.001
Temperatur^2	-0.19	0.03	-0.24 – -0.14	<0.001
Niederschlag	-0.26	0.02	-0.30 – -0.22	<0.001
Sonnenscheindauer	0.33	0.03	0.28 – 0.39	<0.001
Dienstag * Ferien [1]	-0.01	0.18	-0.36 – 0.34	0.957
Mittwoch * Ferien [1]	-0.30	0.18	-0.65 – 0.05	0.091
Donnerstag * Ferien [1]	-0.18	0.18	-0.52 – 0.17	0.314
Freitag * Ferien [1]	-0.07	0.18	-0.41 – 0.28	0.705
Samstag * Ferien [1]	-0.31	0.17	-0.64 – 0.01	0.059
Sonntag * Ferien [1]	-0.52	0.17	-0.85 – -0.19	0.002
Random Effects				
σ^2	0.17			
$\tau_{00\ KW}$	0.08			
$\tau_{00\ Jahr}$	0.03			
ICC	0.37			
N_{KW}	53			
N_{Jahr}	2			
Observations	578			
Marginal R^2 / Conditional R^2	0.827 / 0.891			

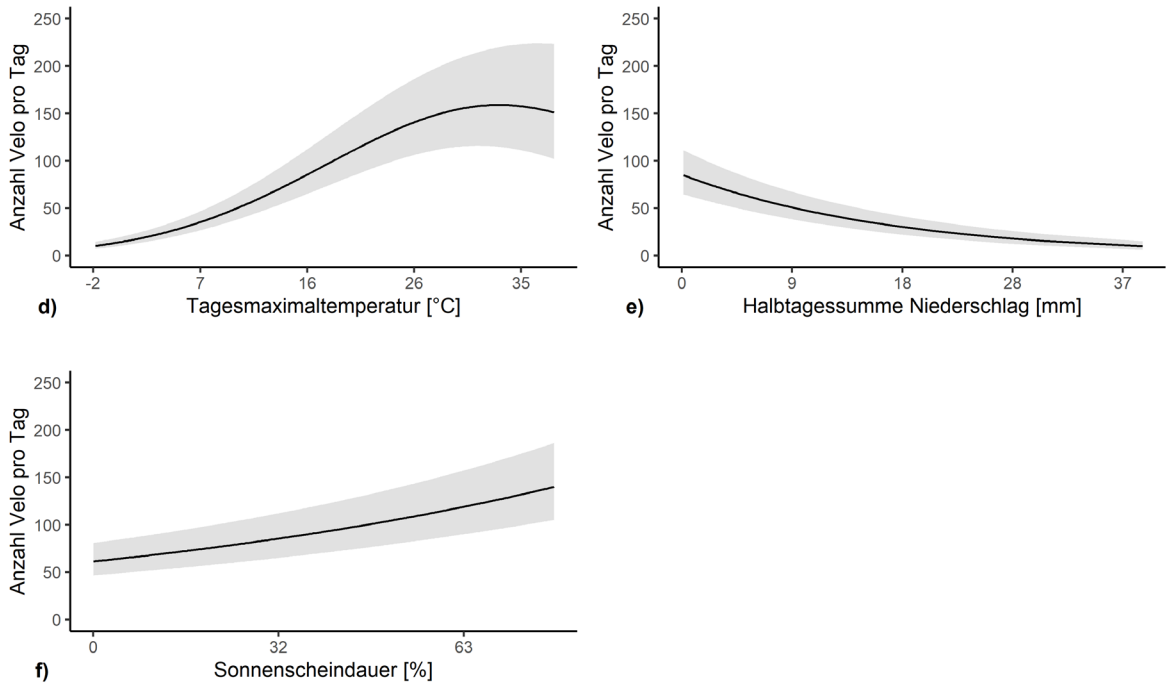

Abbildung 3: Einfluss der meteorologischen Variablen auf das finale Modell mit 95% Konfidenzintervall.

Während des Lockdowns wurden total 31 581 Fahrräder gezählt, im selben Zeitraum 2019 waren es 9 865. Die Unterschiede der beiden Zeiträume sind hochsignifikant, mit einem Mittelwert von 173 für 2019 und 554 für 2020 (Welch t-Test, $t = 5.85$, $p < 0.001$).

Der Unterschied zwischen den Besucherzahlen an Werktagen und den Wochenenden ist in beiden Zeiträumen signifikant. Für den Lockdown betrug der Mittelwert an den Werktagen 385 und an den Wochenenden 988, (Welch t-Test, $t = -4.77$, $p < 0.001$). Für die Periode 2019 waren es im Mittel 126 an Werktagen und 284 an Wochenenden (Welch t-Test, $t = -2.27$, $p = 0.03$).

Der visuelle Vergleich der Saisongänge zeigt eine stärkere Streuung für den Lockdown (Abbildung 4h). Zudem sind die Unterschiede zwischen den Werktagen und den Wochenenden in der Periode 2019 klarer (Abbildung 4g).

Die gemittelten Tagesgänge (Abbildung 5) zeigen für den Lockdown tendenziell höhere Besucherzahlen in den Abendstunden (nach 18 Uhr).

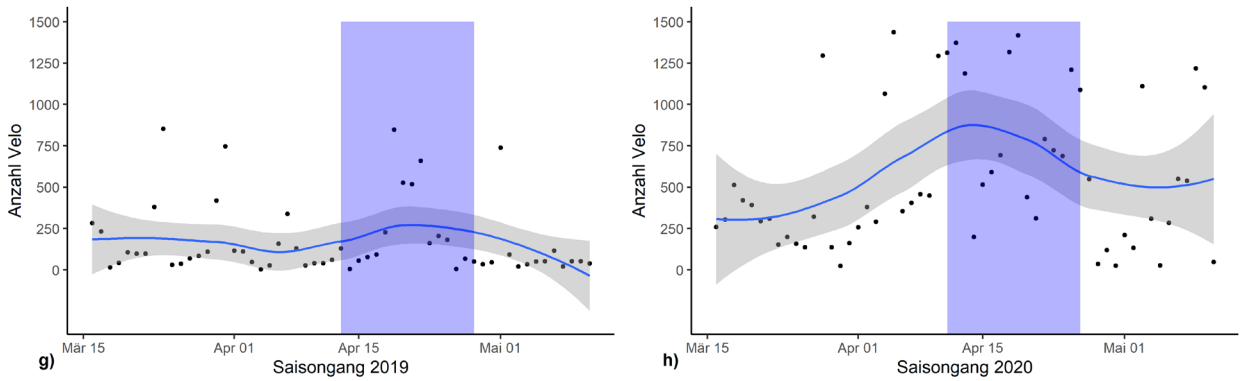

Abbildung 4: Saisongänge für den Lockdown 2020 (h) und denselben Zeitraum 2019 (g) mit 95% Konfidenzintervall. Blau hinterlegt ist der Zeitraum der Frühlingsferien für den Kanton Zürich.

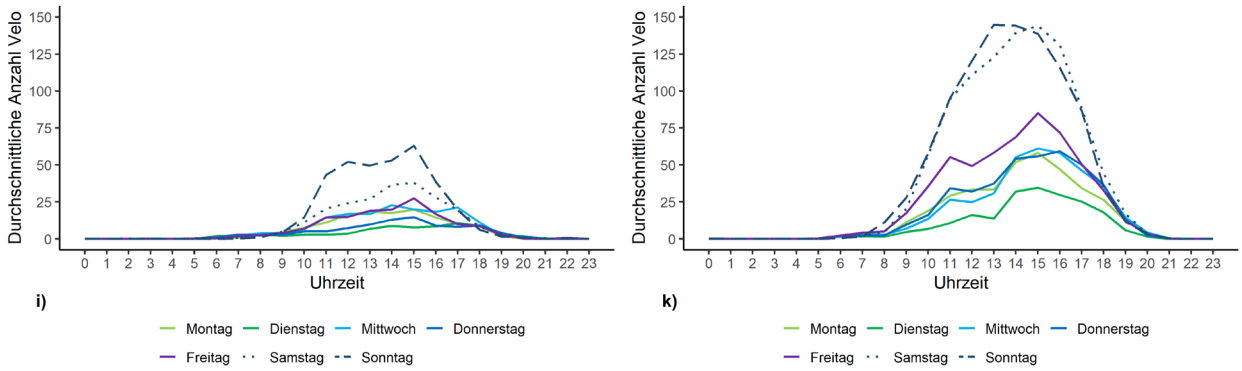

Abbildung 5: Gemittelte Tagesgänge nach Wochentagen für den Lockdown 2020 (k) und denselben Zeitraum 2019 (i).

5. Diskussion

Während des Lockdowns wurden etwa dreimal so viele Velofahrer gezählt wie im gleichen Zeitraum 2019. Diese starke Zunahme an Waldbesuchern während des Lockdowns deckt sich mit den Ergebnissen verschiedener Studien (Derks et al., 2020; Hunziker et al., 2020; Rupf, Riesen, Hochreutener, Bärlocher & Wyttenbach, 2020).

Der Sihluferweg, an dem sich die Zählstelle 502 befindet, ist eine regionale Veloroute und wird auch von Pendlern befahren. Es ist daher unklar, welchen Anteil die Velofahrerinnen ausmachen, die den Wildnispark Zürich bewusst als Naherholungsraum aufsuchten. Zu berücksichtigen ist auch das möglicherweise veränderte Pendleraufkommen während des Lockdowns[1]. Interessant wäre dabei ein Vergleich mit anderen Besucherzählstationen des Wildnispark Zürich, die nicht von Pendlern befahren werden.

Aus dem finalen Modell ergeben sich der Sonntag, gefolgt vom Samstag und dem Lockdown als die Variablen mit dem grössten Einfluss auf die Besucherzahlen. Rupf et al. (2020) haben in ihrem Modell im Badener Stadtwald ebenfalls den Sonntag als die Variable mit dem grössten Einfluss identifiziert, gefolgt vom Lockdown und dem Samstag.

Im finalen Modell hatte der Niederschlag einen eher geringen, aber signifikanten Einfluss auf die Besucherzahlen. Brandenburg et al. (2007) haben in ihrer Studie festgestellt, dass Velofahrer stärker auf schlechtes Wetter reagieren als Fussgänger. In einer weiteren Studie wäre es daher interessant, die Velofahrerinnen und Fussgänger an der Zählstelle 502 miteinander zu vergleichen.

Die Dämmerungsstunden sind besonders relevant für Wildtiere wie beispielsweise Rehe. Die gemittelten Tagesgänge zeigten für den Lockdown tendenziell höhere Besucherzahlen in den Abendstunden. Diese Tendenz sollte in einem weiteren Schritt ebenfalls noch statistisch ausgewertet werden.

In weiteren Studien sollte das Besucheraufkommen im Wildnispark Zürich über längere Zeiträume der Pandemie und darüber hinaus und daraus folgende Auswirkungen auf die Wildtiere analysiert werden.

[1] Durch vermehrtes Arbeiten im Home-Office und dem Meiden des öffentlichen Verkehrs, eventuell zu Gunsten des Fahrrads, wären sowohl eine Abnahme wie auch eine Zunahme des Pendlerverkehrs im Wildnispark Zürich plausibel.

Abbildungsverzeichnis

Literaturverzeichnis

Abraham, A., Sommerhalder, K., Bolliger-Salzmann, H. & Abel, T. (2007). *Landschaft und Gesundheit: Das Potential einer Verbindung zweier Konzepte*. Bern: Universität Bern.

Barton, K. (2020). *MuMIn: Multi-Model Inference*. Verfügbar unter: https://CRAN.R-project.org/package=MuMIn

Bates, D., Mächler, M., Bolker, B. & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, *67*(1), 1–48. https://doi.org/10.18637/jss.v067.i01

Brandenburg, C., Matzarakis, A. & Arnberger, A. (2007). Weather and cycling—a first approach to the effects of weather conditions on cycling. *Meteorological Applications*, *14*(1), 61–67. https://doi.org/10.1002/met.6

Burnham, K. P. & Anderson, D. R. (Hrsg.). (2004). *Model Selection and Multimodel Inference*. New York, NY: Springer New York. https://doi.org/10.1007/b97636

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*, 102253. https://doi.org/10.1016/j.forpol.2020.102253

Hunziker, M., Bauer, N., Salak, B. & Hegetschweiler, K. (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*.

Johannsen, M. (2020). Ferien Schweiz. *schulferien.org*. Zugriff am 30.11.2020. Verfügbar unter: https://www.schulferien.org/schweiz/ferien/

Kendall, M. G. (1938). A NEW MEASURE OF RANK CORRELATION. *Biometrika*, *30*(1–2), 81–93. https://doi.org/10.1093/biomet/30.1-2.81

Korner-Nievergelt, F., Roth, T., Felten, S. von, Guelat, J., Almasi, B. & Korner-Nievergelt, P. (2015). *Bayesian Data Analysis in Ecology using Linear Models with R, BUGS and Stan*. Elsevier.

Meienberg, D. (2020, April 5). Partys gabs nur drinnen. Zugriff am 29.12.2020. Verfügbar unter: https://www.tagesanzeiger.ch/partys-gabs-nur-drinnen-975221021307

MeteoSchweiz. (2020). Bundesamt für Meteorologie und Klimatologie, MeteoSchweiz. *Bundesamt für Meteorologie und Klimatologie*. Zugriff am 30.11.2020. Verfügbar unter: https://www.meteoschweiz.admin.ch/home/service-und-publikationen/beratung-und-service.html

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

Redwoods. (2020). Sonnenaufgang und Sonnenuntergang Zürich City. *sunrisesunset.de*. Zugriff am 30.11.2020. Verfügbar unter: https://sunrisesunset.de/sonne/schweiz/zurich-kreis-1-city/

Roth, I. & Stauffer, C. (2010). *Charta Wildnispark Zürich Sihlwald 2009-2018*. Sihlwald: Stiftung Wildnispark Zürich.

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*.

Siegrist, D., Finger-Stich, A., Ketteter Bonnelame, L. & Egeter, M. (2020). Bleiben Sie zu Hause. Bitte. Alle. Das Freizeitverhalten der Bevölkerung in Bezug auf Frei- und Grünräume während der Coronakrise in den Kantonen Genf und Zürich. *Arbeitsgemeinschaft für den Wald AfW - Austausch von Forschenden zum Thema «Freizeit und Erholung im Wald in Zeiten von Covid-19*.

Stiftung Wildnispark Zürich. (2019). *GIS Wildnispark Zürich*. Sihlwald: Stiftung Wildnispark Zürich.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. Verfügbar unter: https://ggplot2.tidyverse.org

Anhang

R-Code: Analyse und Modell 2020

```
# FALLSTUDIE MODUL RESEARCH METHODS, HS20 ###################################
# AUSWERTUNG VON AUTOMATISCH GENERIERTEN BESUCHERZAHLEN #####################
# AUTOR: TOBIAS WILDHABER ####
# .###########################################################################


# 0. METADATA UND DEFINITIONEN ####
## .##########################################################################


# Datenherkunft  ####
# Wildnispark Sihlwald
# Meteo Schweiz
## .##########################################################################


# Ordnerstruktur  ####
# R_data
#   Scripts
#   Data
#   Results
# .###########################################################################


# Bibliotheken ###############################################################################################
#####

library(arsenal) # Dataframes vergleichen
library(lubridate) # Arbeiten mit Datumsformaten
library(magrittr) # erweitertes Piping
library(data.table) # schnelles Dateneinlesen
library(GGally) # plotten von mehreren Plots gleichzeitig
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 Zusammenhaengen zwischen Variablen
library(tidyverse) # Data wrangling und piping
library(moments) # Skewness berechnen
library(suncalc) # Relative Sonnenscheindauer berechnen
library(cowplot) # Grid-Plots exportieren


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

# Zaehldaten einlesen ####
depo <- fread("Data/zaehldaten_502_20160101_20200831.csv")
str(depo)

# Zeit und Datum trennen und richtig formatieren ####
depo <- depo %>%
  mutate(Datum_Uhrzeit = as.character(Datum_Uhrzeit)) %>%
  separate(Datum_Uhrzeit, into = c("Datum", "Zeit"), sep = " ") %>%
  mutate(Datum = as.Date(Datum, format = "%d.%m.%Y"))

# Daten der Fussgaenger entfernen ####
depo <- depo %>%
  dplyr::select(-c("502_F_IN_cal", "502_F_OUT_cal"))
```

```r
# Total IN und OUT berechnen und in neue Spalte schreiben

depo <- depo %>%
  mutate(Total = rowSums(.[, 3:4]))

# oder alternativ mit select
depo$Total <- depo %>%
  dplyr::select("502_V_IN_cal":"502_V_OUT_cal") %>%
  rowSums(na.rm = TRUE) # NA-Werte entfernen, hier nicht noetig

# Spalten umbenennen
colnames(depo) <- c("Datum", "Zeit", "IN", "OUT", "Total")

# Meteodaten einlesen ####
meteo <- fread("./Meteo/Meteo_data.txt")

# alternativ geht das Einlesen auch mit diplyr
meteo <- read_csv2("./Meteo/Meteo_data.txt")

# Datum korrekt formatieren
meteo <- transform(meteo, time = as.Date(as.character(time), "%Y%m%d"))

# Datentyp der Werte ebenfalls korrigieren
meteo$tre200jx <- meteo$tre200jx %>% as.double()
meteo$rre150j0 <- meteo$rre150j0 %>% as.double()
meteo$sre000d0 <- meteo$sre000d0 %>% as.double()


# oder in einer Funktion
meteo <- meteo %>% mutate(across(c(tre200jx, rre150j0), as.double))

# NA-Werte entfernen

meteo <- meteo %>% na.omit() # datatable
meteo <- meteo %>% drop_na() # tidyverse >> beides funktioniert


# 2. VORBEREITUNG UND EXPLORATIVE VISUALISIERUNG ##############################

# Start und Ende definieren ####

# Untersuchungszeitraum
# der Installations- sowie Deinstallationstag werden aufgrund unvollstaendiger
# oder beeinflusster Messungen i.d.R. weggelassen.
depo_start <- as.Date("2019-01-01")
depo_end <- as.Date("2020-7-31")

# Start und Ende Lockdown
lock_start <- as.Date("2020-03-16")
lock_end <- as.Date("2020-05-11")

# Ebenfalls muessen die erste und letzte Kalenderwoche der Untersuchungsfrist definiert werden
# Diese werden bei Wochenweisen Analysen ebenfalls ausgeklammert da sie i.d.R. unvollstaendig
sind
KW_start <- week(depo_start)
KW_end <- week(depo_end)

# Erster und letzter Tag der Ferien
# je nach Untersuchungsdauer muessen hier weitere oder andere Ferienzeiten ergaenzt werden
# (https://www.schulferien.org/schweiz/ferien/2020/)
Fruehlingsferien_2019_start <- as.Date("2019-04-13")
Fruehlingsferien_2019_ende <- as.Date("2019-04-28")
Sommerferien_2019_start <- as.Date("2019-07-6")
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("2020-01-02")
```

```
Fruehlingsferien_start <- as.Date("2020-04-11")
Fruehlingsferien_ende <- as.Date("2020-04-26")
Sommerferien_start <- as.Date("2020-07-11")

# Einteilung Tag und Nacht ####
# Einteilung gemaess der Buergerlichen Daemmerung fuer Zuerich
# (https://sunrisesunset.de/sonne/schweiz/zurich-kreis-1-city/)
Latitude <- 47.38598
Longitude <- 8.50806


# Berechnung Tageslaenge ####
# Ende der Sommerzeit Schweiz: 27.10.2019, Start 29.3.2020

So_end <- as.Date("2019-10-27") # Ende Sommerzeit
Wi_start <- as.Date("2019-10-28")
Wi_end <- as.Date("2020-03-29")

lumidata_So <-
  getSunlightTimes(
    date = seq.Date(depo_start, So_end, by = 1),
    keep = c("sunrise", "sunset"),
    lat = Latitude,
    lon = Longitude,
    tz = "UTC"
  )

lumidata_Wi <-
  getSunlightTimes(
    date = seq.Date(Wi_start, Wi_end, by = 1),
    keep = c("sunrise", "sunset"),
    lat = Latitude,
    lon = Longitude,
    tz = "UTC"
  )

lumidata_So_20 <-
  getSunlightTimes(
    date = seq.Date(Wi_end, depo_end, by = 1),
    keep = c("sunrise", "sunset"),
    lat = Latitude,
    lon = Longitude,
    tz = "UTC"
  )

# CH ist im Winter UTZ + 1. Darum auf alle Angaben eine Stunde draufschlagen
# Im Sommer + 2
lumidata_So <- lumidata_So %>%
  mutate(
    sunrise = sunrise + hours(2),
    sunset = sunset + hours(2)
  )

lumidata_Wi <- lumidata_Wi %>%
  mutate(
    sunrise = sunrise + hours(1),
    sunset = sunset + hours(1)
  )

lumidata_So_20 <- lumidata_So_20 %>%
  mutate(
    sunrise = sunrise + hours(2),
    sunset = sunset + hours(2)
  )

# Verbinde alle Teildatensaetze
lumidata <- rbind(lumidata_So, lumidata_Wi, lumidata_So_20)
```

```r
# Berechne Tageslaenge
lumidata$Tageslaenge <- lumidata$sunset - lumidata$sunrise

# Wandle in min um
lumidata$Tageslaenge <- as.numeric(lumidata$Tageslaenge * 60)

# behalte relevante Spalten
lumidata <- dplyr::select(lumidata, "date", "Tageslaenge")

str(lumidata)

## 'data.frame':    579 obs. of  2 variables:
##  $ date       : Date, format: "2019-01-01" "2019-01-02" ...
##  $ Tageslaenge: num  512 513 514 515 516 ...

# Datenssatz auf Untersuchungszeitraum zuschneiden ####
depo <- depo %>% filter(Datum >= depo_start & Datum <= depo_end)

# Wochentage einfuegen ####

depo$Wochentag <- depo$Datum %>% lubridate::wday(label = T, abbr = F) # Paket (lubridate) angeben, da wday auch in datatable vorhanden ist



lvl <- c("Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag", "Sonntag")

depo <- depo %>%
mutate(Wochentag = factor(Wochentag, levels = lvl))

# oder einfacher, ohne mutate
# levels(depo$Wochentag) <- lvl # funktioniert nicht und verschiebt die Levels!

# Stunden einfuegen ####
depo$Stunde <- as.numeric(format(as.POSIXct(depo$Zeit, format = "%H:%M"), "%H"))

# Werte runden ####

depo <- depo %>%
  mutate(across(where(is.numeric), round, 0)) %>%
  mutate(across(where(is.numeric), as.integer)) # Datentyp aller numerischen Werte als integer speichern

# Explorativer Plot Datum und Total ####
plot(depo$Datum, depo$Total)

# Stundendaten aggregieren ####

# Gemaess angaben in der Uebung einzelen aggregieren (IN, OUT und Total) und dann wieder zusammensetzen
# Day <- aggregate(depo$Total,by=list(depo$Datum),sum)
# colnames(DayAll) <- c("Datum", "Anzahl_Total")

# einfacher: alles auf einmal zusammenfassen mit tidyverse
Day <- depo %>%
  group_by(Datum) %>%
  summarise_if(is.numeric, sum)
# depo_date <- depo %>% group_by(Datum) %>% summarise(across(.cols = c(IN, OUT, Total), sum))
#oder mit Angabe der Spalten

# Convenience Variabeln einfuegen ####
Day$Wochentag <- Day$Datum %>% lubridate::wday(label = T, abbr = F)

# levels(Day$Wochentag) <- c("Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag", "Sonntag") #Achtung, dieser Code ist falscht und andert die Wochentage!
Day <- Day %>% mutate(Wochentag = factor(Wochentag,
  levels = lvl
))
```

```
Day$KW <- Day$Datum %>% lubridate::week()
Day$KW <- as.factor(Day$KW)
Day <- Day %>%
  mutate(Woche = if_else(Wochentag == "Montag" | Wochentag == "Dienstag" |
    Wochentag == "Mittwoch" | Wochentag == "Donnerstag" |
    Wochentag == "Freitag", "Werktag", "Wochenende"))

# Wetterdaten auf Untersuchungszeitraum zuschneiden ####
meteo <- meteo %>% filter(time >= depo_start & time <= depo_end)
# meteo und Day sind nun gleich lang (578 Zeilen)

# Wetterdaten explorativ plotten ####
ggplot(data = meteo, aes(time, tre200jx)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Datum", y = "Lufttemperatur [°C]")

ggplot(data = meteo, aes(time, rre150j0)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Datum", y = "Niederschlag [mm]")

ggplot(data = meteo, aes(time, sre000d0)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Datum", y = "Sonnenscheindauer [min]")



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

# Daten auf Lockdown zuschneiden ####

lock <- depo %>% filter(Datum >= lock_start & Datum <= lock_end)
lock_day <- Day %>% filter(Datum >= lock_start & Datum <= lock_end)
lock_meteo <- meteo %>% filter(time >= lock_start & time <= lock_end)

# Besucherzahlen summieren ####
SumBesuchende <- lock %>% summarise(across("IN":"Total", sum)) %>% # Summen fuer alle Spalten
  `colnames<-`(c("sumVeloIN", "sumVeloOUT", "sumVelo"))

# Prozentwerte berechnen ####
ProzBesuchende <- SumBesuchende %>%
  transmute(
    ProzVeloIN = sumVeloIN / sumVelo, # transmute loescht alte Werte
    ProzVeloOUT = sumVeloOUT / sumVelo
  ) %>%
  mutate(across(everything(), round, 2)) # alles auf zwei Stellen runden

# Tabellenformat aendern ####
SumBesuchende <- gather(SumBesuchende)
ProzBesuchende <- gather(ProzBesuchende)

# Visualisierung Richtungsverteilung (Pie Chart) ####

palette <- c("grey90", "grey60")

pie(ProzBesuchende$value, labels = c("Velo IN; 56%", "Velo OUT; 43%"), col = palette)
title("Prozentuales Verhältnis aller Velofahrer*innen")

dev.copy(png, "./Results/Richtungsverteilung.png", width = 800, height = 500) # Bild speichern

dev.off()
```

```
# Visualisierung Ganglinien ####

# Mittelwerte berechnen fuer IN und OUT ####
mean_besuchende <- lock %>%
  group_by(Stunde) %>%
  summarise(across(3:4, mean)) %>% # Berechnet Mittelwerte fuer die angegebenen Spalten und gr
uppiert sie nach Stunden
  pivot_longer(2:3) %>% # in langes Format umwandeln, Spalten 2 und drei pivotieren (aus zwei
mach eine)
  `colnames<-`(c("Stunde", "Gruppe", "Durchschnitt")) # Spalten gemaess Vorgaben umbenennen

## `summarise()` ungrouping output (override with `.groups` argument)

# Mittelwerte berechnen fuer Ganglinien nach Wochentagen ####
mean_besuchende_tage <- lock %>%
  group_by(Stunde, Wochentag) %>%
  dplyr::summarise(across(Total, mean)) # Berechnet Mittelwerte fuer die angegebenen Spalten u
nd gruppiert sie nach Stunden

## `summarise()` regrouping output by 'Stunde' (override with `.groups` argument)

# Ganglinien visualisieren (Barplot) ####
ggplot(data = mean_besuchende, aes(Stunde, Durchschnitt, fill = Gruppe)) +
  geom_bar(position = "dodge", stat = "identity") +
  labs(x = "Uhrzeit [h]", y = "Durchschnittliche Anzahl Velo") +
  scale_fill_grey(start = 0.7, end = 0.4) +
  theme_bw(base_size = 15) +
  scale_x_continuous(limits = c(0, 23), breaks = seq(0, 23, by = 2)) +
  scale_y_continuous(limits = c(0, 50), breaks = seq(0, 50, by = 5))

ggsave("Ganglinien.png", path = "Results/") # Bild speichern

## Saving 5 x 4 in image

## Warning: Removed 2 rows containing missing values (geom_bar).

# Ganglinien nach Wochentagen visualisieren (Lineplot) ####

palette <- c("#92D24F", "#01AF50", "#02AFEF", "#026CBF", "#712EA1", "#1D507A", "#1F4D77")
ltype <- c(
  "Montag" = "solid", "Dienstag" = "solid", "Mittwoch" = "solid",
  "Donnerstag" = "solid", "Freitag" = "solid",
  "Samstag" = "dotted", "Sonntag" = "longdash"
)

ggplot(data = mean_besuchende_tage, aes(Stunde, Total, color = Wochentag, linetype = factor(Wo
chentag))) +
  geom_line(size = 1) +
  labs(x = "Uhrzeit", y = "Durchschnittliche Anzahl Velo") +
  scale_x_continuous(limits = c(0, 23), breaks = seq(0, 23, by = 1)) +
  scale_y_continuous(limits = c(0, 150), breaks = seq(0, 150, by = 25)) +
  scale_color_manual(breaks = lvl, labels = lvl, values = palette) +
  scale_linetype_manual(values = ltype) +
  guides(linetype = FALSE, color = guide_legend(
    nrow = 2, byrow = TRUE,
    override.aes = list(linetype = c(rep("solid", 5), "dotted", "twodash"))
  )) +
  theme_classic(base_size = 15) +
  theme(legend.position = "bottom", legend.title = element_blank(), legend.box = "vertical")

ggsave("Ganglinien_Tage.png", path = "Results/", scale = 1.5, dpi = 300) # Bild speichern

## Saving 7.5 x 6 in image

# Wochengang visualisieren (Boxplot) ####
ggplot(data = lock_day, aes(x = Wochentag, y = Total)) +
  geom_boxplot(size = 1) +
```

```
  labs(x = "Wochentag", y = "Anzahl Velo") +
  theme_classic(base_size = 15) +
  scale_y_continuous(limits = c(0, 1500), breaks = seq(0, 1500, by = 250)) +
  theme(axis.line = element_line(size = 1), axis.ticks = element_line(size = 1))

ggsave("Wochengang.png", path = "Results/") # Bild speichern

## Saving 5 x 4 in image

# Saisongang visualisieren (Scatterplot) ####
season <- ggplot(data = lock_day, aes(x = Datum, y = Total)) +
  geom_point() +
  geom_smooth() +
  annotate(
    geom = "rect", xmin = Fruehlingsferien_start, xmax = Fruehlingsferien_ende,
    ymin = -Inf, ymax = 1500, fill = "blue", alpha = 0.3
  ) + # zeichnet Rechteck
  scale_y_continuous(limits = c(-300, 1500), breaks = seq(0, 1500, by = 250)) +
  # scale_x_date(limits = as.Date(c("2020-03-16","2020-05-11"))) +
  labs(x = "Datum", y = "Anzahl Velo") +
  theme_classic(base_size = 15)


ggsave("Saisongang.png", path = "Results/") # Bild speichern

## Saving 5 x 4 in image

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

# Bestbesuchte Tage herausfiltern ####

# unlist(lock_day$Total) %>% sort(decreasing = T) %>% .[1:10] # ergibt nur die besten Werte
max_day <- lock_day %>% slice_max(Total, n = 10) # selektiert die gesamten Zeilen mit den best
en Werten

write.csv2(max_day, "./Results/bestbesuchte_tage.csv")

# Nutzung Werktage und Wochenende ####

mean_werktag <- lock_day %>%
  filter(Woche == "Werktag") %>%
  summarize(werktag = mean(Total)) # Mittelwert Werktage berechnen

mean_wochenende <- lock_day %>%
  filter(Woche == "Wochenende") %>%
  summarize(Wochenende = mean(Total)) # Mittelwert Wochenende berechnen

mean_wochentage <- as.tibble(c(mean_werktag, mean_wochenende)) # Werte zusammenfuegen

## Warning: `as.tibble()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

write.csv2(mean_wochentage, "Results/mittelwerte_biker_wochentage.csv") # Werte als csv speich
ern

# Plot

ggplot(lock_day, aes(Woche, Total)) +
  geom_boxplot() +
  scale_y_continuous(limits = c(0, 1500), breaks = seq(0, 1500, by = 250)) +
  labs(x = "", y = "Anzahl Velo pro Tag") +
  theme_classic(base_size = 15)

ggsave("Tagesnutzung-19.png", path = "Results/")
```

```r
# Statistik: T-Test Unterscheid Werktage und Wochenende ####

t.test(
  lock_day$Total [lock_day$Woche == "Werktag"],
  lock_day$Total [lock_day$Woche == "Wochenende"]
)

##
##  Welch Two Sample t-test
##
## data:  lock_day$Total[lock_day$Woche == "Werktag"] and lock_day$Total[lock_day$Woche == "Wo
chenende"]
## t = -4.7676, df = 19.285, p-value = 0.0001289
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -868.7533 -339.0486
## sample estimates:
## mean of x mean of y
##  384.5366  988.4375

# Es besteht ein höchstsignifikanter Unterschied zwischen den taeglichen Besuchszahlen an Werk
tagen und am Wochenende
# (Welch t-Test, t = -4.77, p < 0.001)

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

# Join der Datensaetze Day und meteo ####

umwelt <- left_join(Day, meteo, by = c("Datum" = "time")) %>% # fuegt die beiden Tabellen mit
Join zusammen
  drop_na() # wuerde Beobachtungen mit NA-Werten entfernen, falls vorhanden

# Join der Datensaetze umwelt und lumidata ####

# umwelt und lumidata sind unterschiedlich lang, dataframes verlgeichen und Unterschiede suche
n
comparedf(umwelt, lumidata, tol.vars = c(Datum = "date"))

# doppleten Wert fuer 2020-03-29 loeschen
lumidata <- lumidata[-454, ]

# Join
umwelt <- left_join(umwelt, lumidata, by = c("Datum" = "date")) %>% # fuegt die beiden Tabelle
n mit Join zusammen
  drop_na() # wuerde Beobachtungen mit NA-Werten entfernen, falls vorhanden

# neue Spalte mit relativer Sonnenscheindauer erstellen ####

umwelt$sre000d0_relative <- (100 / umwelt$Tageslaenge * umwelt$sre000d0)

# Faktoren erstellen und definieren ####

# Kalenderwoche
umwelt$KW <- as_factor(umwelt$KW)
umwelt$Jahr <- year(umwelt$Datum) %>% as_factor()

# Lockdown
umwelt <- umwelt %>%
  mutate(Lockdown = if_else(Datum >= lock_start & Datum <= lock_end, "1", "0"))
```

```
# Ferien
umwelt <- umwelt %>% mutate(Ferien = if_else
(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_start & Datum <= Fruehlingsferien_ende |
    Datum >= Sommerferien_start, "1", "0"))

umwelt$Ferien <- as_factor(umwelt$Ferien)
umwelt$Lockdown <- as_factor(umwelt$Lockdown)

# Kontraste der Wochentage anpassen ####

contrasts(umwelt$Wochentag) <- contr.treatment(7) # Montag als Referenzwert
# contrasts(umwelt$Wochentag) <- contr.sum        # Sonntag als Referenzwert


# Saklierung der Wetterdaten ####

umwelt <- umwelt %>%
  mutate(
    tre200jx_scaled = scale(tre200jx),
    rre150j0_scaled = scale(rre150j0),
    sre000d0_scaled = scale(sre000d0_relative)
  )

# auf Korrelation testen

cor <- cor(umwelt[, 18:20], method ="kendall") # Meteodaten als erklaerende Variablen korrelie
ren
predictors <- (umwelt[, 18:20]) # Meteodaten als erklaerende Variablen selektieren fuer Korrel
ationsmatrix
cor[abs(cor) < 0.7] <- 0
cor

print(cor, digits = 3)

# mit Schwellenwert 0.7 ergeben sich keine kritischen Korrelationen,
# es muss also keine Variable ausgeschlossen werden (anders waere es mit
# Schwellenwert 0.5, dann muesste man Lufttemperatur oder Sonnenscheindauer ausschliessen)


# Korrelation visualisieren

chart.Correlation(predictors, histogram = TRUE, pch = 19) # Korrelationsgrafik der Messwerte

# alternative Visualisierung (psych-Package)
# library(psych)
# pairs.panels(cor, scale=TRUE)

# Verteilung der abhaengigen Variabel pruefen ####

f1 <- fitdist(umwelt$Total, "norm") # Normalverteilung
f1_1 <- fitdist(umwelt$Total, "lnorm") # log-Normalvert.
f2 <- fitdist(umwelt$Total, "pois") # Poisson
f3 <- fitdist(umwelt$Total, "nbinom") # negativ binomial
f4 <- fitdist(umwelt$Total, "exp") # exponentiell
f5 <- fitdist(umwelt$Total, "gamma") # gamma
f6 <- fitdist(umwelt$Total, "logis") # logistisch
f7 <- fitdist(umwelt$Total, "geom") # geometrisch
f8 <- fitdist(umwelt$Total, "weibull") # Weibull
```

```
gof <- gofstat(list(f1, f1_1, f2, f3, f4, f5, f6, f7, f8),
  fitnames = c(
    "Normalverteilung", "log-Normalverteilung", "Poisson",
    "negativ binomial", "exponentiell", "gamma", "logistisch",
    "geometrisch", "weibull"
  )
)
gof

# Sortiert AIC-Werte aufsteigend. Beste Modelle sind also log-Norm, weibull, gamma und neg.bin
om
sort(gof$aic)

# die 4 besten (gemaess Akaike's Information Criterion AIC) als Plot -> je tiefer der AIC, des
to besser
# dazu zum Vergleich Poisson (mit sehr hohem AIC)
plot.legend <- c("log norm", "weibull", "gamma ", "negativ binomial", "poisson")


# vergleicht mehrere theoretische Verteilungen mit den empirischen Daten
cdfcomp(list(f1_1, f8, f5, f3, f2), legendtext = plot.legend)


# Modell berechnen ####

# neg.bin-Modell
Tages_Model_nb <- glmer.nb(Total ~ Lockdown + tre200jx_scaled +
  rre150j0_scaled + sre000d0_scaled +
  Wochentag + Ferien +
  (1 | KW) + (1 | Jahr), data = umwelt)

summary(Tages_Model_nb) # Zeigt das Resultat des Modells

# Modelldiagnostik ####

# glmer bringt einige eigene Funktionen mit, mit denen sich testen laesst,
# ob das Modell valide ist. Untenstehend sind sie aufgefuehrt.
# (--> analog zu den Funktionen aus der Vorlesung, aber halt für glmer)

# Verteilung der Residuen

plot(Tages_Model_nb, type = c("p", "smooth"))

# Pruefen auf Normalverteilung

qqmath(Tages_Model_nb)

# Overdispersion describes the observation that variation is higher than would be expected.

dispersion_glmer(Tages_Model_nb) # it shouldn't be over 1.4 >> 1.02, also soweit ok

## [1] 1.015515

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

# Alternative Modelle ####

# Quadriertes Modell ####

Tages_Model_nb_quad <- glmer.nb(Total ~ Lockdown + Wochentag + Ferien +
  tre200jx_scaled + I(tre200jx_scaled^2) +
  rre150j0_scaled + sre000d0_scaled +
  (1 | KW) + (1 | Jahr), data = umwelt)
```

```r
summary(Tages_Model_nb_quad)

plot(Tages_Model_nb_quad, type = c("p", "smooth"))

qqmath(Tages_Model_nb_quad)

dispersion_glmer(Tages_Model_nb_quad) # it shouldn't be over 1.4 >> 1.02, also soweit ok

r.squaredGLMM(Tages_Model_nb_quad) # besser als nb-Modell

# Interaktionsmodel ####

Tages_Model_nb_int <- glmer.nb(Total ~ Lockdown + Wochentag * Ferien +
  tre200jx_scaled + I(tre200jx_scaled^2) *
    rre150j0_scaled + sre000d0_scaled +
  (1 | KW) + (1 | Jahr), data = umwelt)


summary(Tages_Model_nb_int)

# summary(Tages_Model_nb_int2)

plot(Tages_Model_nb_int, type = c("p", "smooth"))

qqmath(Tages_Model_nb_int)

dispersion_glmer(Tages_Model_nb_int) # it shouldn't be over 1.4 >> 1.02, also soweit ok

## [1] 1.019567

r.squaredGLMM(Tages_Model_nb_int) # besser als nb- und quad-Modell

## Warning: The null model is correct only if all variables used by the original
## model remain unchanged.

##                 R2m       R2c
## delta     0.8200803 0.8834402
## lognormal 0.8281797 0.8921653
## trigamma  0.8106140 0.8732425

# Interaktionsmodel 2 (ohne Interaktion Ferien * Wochentag) ####

Tages_Model_nb_int2 <- glmer.nb(Total ~ Lockdown + Wochentag + Ferien +
  tre200jx_scaled + I(tre200jx_scaled^2) *
    rre150j0_scaled + sre000d0_scaled +
  (1 | KW) + (1 | Jahr), data = umwelt)

summary(Tages_Model_nb_int2)


plot(Tages_Model_nb_int2, type = c("p", "smooth"))

qqmath(Tages_Model_nb_int2)

dispersion_glmer(Tages_Model_nb_int2) # it shouldn't be over 1.4 >> 1.02, also soweit ok

r.squaredGLMM(Tages_Model_nb_int2) # besser als nb- und quad-Modell

# Interaktionsmodel 3 (ohne Interaktion Temperatur * Niederschlag) ####

Tages_Model_nb_int3 <- glmer.nb(Total ~ Lockdown + Wochentag * Ferien +
  tre200jx_scaled + I(tre200jx_scaled^2) +
  rre150j0_scaled + sre000d0_scaled +
  (1 | KW) + (1 | Jahr), data = umwelt)
```

```r
summary(Tages_Model_nb_int3)

plot(Tages_Model_nb_int3, type = c("p", "smooth"))

qqmath(Tages_Model_nb_int3)

dispersion_glmer(Tages_Model_nb_int3) # it shouldn't be over 1.4 >> 1.02, also soweit ok

## [1] 1.020928

r.squaredGLMM(Tages_Model_nb_int3) # besser als nb- und quad-Modell

# Vergleich der Modellguete mittels AICC ####

cand.models <- list()
# cand.models[[1]] <- Tages_Model
cand.models[[1]] <- Tages_Model_nb
cand.models[[2]] <- Tages_Model_nb_int
cand.models[[3]] <- Tages_Model_nb_int2
cand.models[[4]] <- Tages_Model_nb_int3
cand.models[[5]] <- Tages_Model_nb_quad

Modnames <- c(
  "Tages_Model_nb", "Tages_Model_nb_int", "Tages_Model_nb_int2", "Tages_Model_nb_int3",
  "Tages_Model_nb_quad"
)

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

# K = Anzahl geschaetzter Parameter (2 Funktionsparameter und die Varianz)
# Delta_AICc <2 = Statistisch gleichwertig
# AICcWt = Akaike weight in %

# Negativ-Binominales-Modell mit Interaktionen ist das beste Modell >> int3
# (regualaeres Tages-Modell mit Poisson wurde vorgaengig schon ausgeschlossen)

tab_model(Tages_Model_nb_int3, transform = NULL, show.se = TRUE)

# Transformation ####

skewness(umwelt$Total)

## [1] 2.498919

# stark positiv (2.5), daher log10
# (https://www.datanovia.com/en/lessons/transform-data-to-normal-distribution-in-r/)
# Transformation macht Interpretation der Modelle jedoch komplex, Ergebnisse von
# transformierten Daten lassen sich nicht auf untransformierte Daten uebertragen
# Nach Moeglichkeit daher von Transformation absehen

# Modellvisualisierung ####

# Tagestemperatur ####

t <- plot_model(Tages_Model_nb_int3,
  type = "pred", terms = "tre200jx_scaled [all]",
  title = "", axis.title = c("Tagesmaximaltemperatur [°C]", "Anzahl Velo pro Tag")
)

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

```
temp <- t + scale_x_continuous(breaks = c(-2, -1, 0, 1, 2), limits = c(-2, NA), labels = c(lab
els)) +
  scale_y_continuous(limits = c(0, 250), breaks = seq(0, 250, by = 50)) +
  #  ylim(0,250) +
  theme_classic(base_size = 20)

ggsave("Tagestemperatur_int.png", path = "Results/") # Bild speichern

# Sonnenscheindauer ####

t <- plot_model(Tages_Model_nb_int3,
  type = "pred", terms = "sre000d0_scaled [all]",
  title = "", axis.title = c("Sonnenscheindauer [%]", "Anzahl Velo pro Tag")
)

labels <- round(seq(floor(min(umwelt$sre000d0_relative)), ceiling(max(umwelt$sre000d0_relative
)),
  length.out = 4
), 0)

sun <- t + scale_x_continuous(breaks = c(-1, 0, 1, 2), limits = c(-1, NA), labels = c(labels))
+
  theme_classic(base_size = 20) +
  scale_y_continuous(limits = c(0, 250), breaks = seq(0, 250, by = 50))

ggsave("Sonnenscheindauer_int.png", path = "Results/") # Bild speichern

# Niederschlagssumme ####

t <- plot_model(Tages_Model_nb_int3,
  type = "pred", terms = "rre150j0_scaled [all]",
  title = "", axis.title = c("Halbtagessumme Niederschlag [mm]", "Anzahl Velo pro Tag")
)

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

rain <- t + scale_x_continuous(breaks = c(0, 2, 4, 6, 8), limits = c(0, NA), labels = c(labels
)) +
  scale_y_continuous(limits = c(0, 250), breaks = seq(0, 250, by = 50)) +
  theme_classic(base_size = 20)

ggsave("Niederschlag_int.png", path = "Results/") # Bild speichern

# Plotgrid Meteodaten ####

# ggsave("Meteodaten_grid.png", path = "Results/", grid)
cowplot::save_plot("Results/Meteodaten_grid.png", grid, ncol = 2, nrow = 2)

# Wochentage ####

t <- plot_model(Tages_Model_nb_int3,
  type = "pred", terms = "Wochentag",
  title = "", axis.title = c("Wochentag", "Anzahl Velo pro Tag")
)

week_int <- t + theme_classic(base_size = 20) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.2, hjust = 1)) +
  scale_y_continuous(limits = c(0, 350), breaks = seq(0, 350, by = 50)) +
  ylim(0, 400)

ggsave("Wochentage_int.png", path = "Results/") # Bild speichern
```

```
# Ferien ####

t <- plot_model(Tages_Model_nb_int3,
  type = "pred", terms = "Ferien",
  title = "", axis.title = c("Ferien", "Anzahl Velo pro Tag")
)

ferien_int <- t + theme_classic(base_size = 20) +
  scale_x_continuous(breaks = c(0, 1), labels = c("Nein", "Ja")) +
  scale_y_continuous(limits = c(0, 250), breaks = seq(0, 250, by = 50))

ggsave("Ferien_int.png", path = "Results/") # Bild speichern

# Lockdown ####

t <- plot_model(Tages_Model_nb_int3,
  type = "pred", terms = "Lockdown",
  title = "", axis.title = c("Lockdown", "Anzahl Velo pro Tag")
)

lock_int <- t + theme_classic(base_size = 20) +
  scale_x_continuous(breaks = c(0, 1), labels = c("Nein", "Ja")) +
  scale_y_continuous(limits = c(0, 250), breaks = seq(0, 250, by = 50))

ggsave("Lockdown_int.png", path = "Results/") # Bild speichern

## Saving 5 x 4 in image

# Plotgrid Zeitdaten ####

grid <- cowplot::plot_grid(week_int, ferien_int, lock_int, labels = "auto", ncol = 2, nrow = 2
)

grid

# ggsave("Meteodaten_grid.png", path = "Results/", grid)
cowplot::save_plot("Results/Zeitdaten_grid.png", grid, ncol = 2, nrow = 2)
```

R-Code: Deskriptiver Vergleich 2019

```
VORBEREITUNG UND EXPLORATIVE VISUALISIERUNG ##################################

# Start und Ende definieren ####

# Untersuchungszeitraum
# der Installations- sowie Deinstallationstag werden aufgrund unvollstaendiger
# oder beeinflusster Messungen i.d.R. weggelassen.
depo_start <- as.Date("2019-01-01")
depo_end <- as.Date("2020-7-31")

# Start und Ende Lockdown
lock_start <- as.Date("2020-03-16")
lock_end <- as.Date("2020-05-11")

# Start und Ende Zeitraum Lockdown 2019
lock_19_start <- as.Date("2019-03-16")
lock_19_end <- as.Date("2019-05-11")

# Ebenfalls muessen die erste und letzte Kalenderwoche der Untersuchungsfrist definiert werden
# Diese werden bei Wochenweisen Analysen ebenfalls ausgeklammert da sie i.d.R. unvollstaendig
sind
KW_start <- week(depo_start)
KW_end <- week(depo_end)

# Erster und letzter Tag der Ferien
# je nach Untersuchungsdauer muessen hier weitere oder andere Ferienzeiten ergaenzt werden
# (https://www.schulferien.org/schweiz/ferien/2020/)

Fruehlingsferien_2019_start <- as.Date("2019-04-13")
Fruehlingsferien_2019_ende <- as.Date("2019-04-28")
Sommerferien_2019_start <- as.Date("2019-07-6")
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("2020-01-02")
Fruehlingsferien_start <- as.Date("2020-04-11")
Fruehlingsferien_ende <- as.Date("2020-04-26")
Sommerferien_start <- as.Date("2020-07-11")

# Datenssatz auf Untersuchungszeitraum zuschneiden ####
depo <- depo %>% filter(Datum >= depo_start & Datum <= depo_end)

# Wochentage einfuegen ####

depo$Wochentag <- depo$Datum %>% lubridate::wday(label = T, abbr = F) # Paket (lubridate) ange
ben, da wday auch in datatable vorhanden ist

lvl <- c("Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag", "Sonntag")

depo <- depo %>%
  mutate(Wochentag = factor(Wochentag, levels = lvl))

# oder einfacher, ohne mutate
# levels(depo$Wochentag) <- lvl # funktioniert nicht und verschiebd die Levels!

# Stunden einfuegen ####
depo$Stunde <-
  as.numeric(format(as.POSIXct(depo$Zeit, format = "%H:%M"), "%H"))
```

```r
# Werte runden ####

depo <- depo %>%
  mutate(across(where(is.numeric), round, 0)) %>%
  mutate(across(where(is.numeric), as.integer)) # Datentyp aller numerischen Werte als integer
speichern

# Exploativer Plot Datum und Total ####
plot(depo$Datum, depo$Total)

# Stundendaten aggregieren ####

# Gemaess angaben in der Uebung einzelen aggregieren (IN, OUT und Total) und dann wieder zusam
mensetzen
# Day <- aggregate(depo$Total,by=list(depo$Datum),sum)
# colnames(DayAll) <- c("Datum", "Anzahl_Total")

# einfacher: alles auf einmal zusammenfassen mit tidyverse
Day <- depo %>%
  group_by(Datum) %>%
  summarise_if(is.numeric, sum)
# depo_date <- depo %>% group_by(Datum) %>% summarise(across(.cols = c(IN, OUT, Total), sum))
#oder mit Angabe der Spalten

# Convenience Variabeln einfuegen ####
Day$Wochentag <- Day$Datum %>% lubridate::wday(label = T, abbr = F)

# levels(Day$Wochentag) <- c("Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samst
ag", "Sonntag") #Achtung, dieser Code ist falscht und andert die Wochentage!
Day <- Day %>% mutate(Wochentag = factor(Wochentag,
  levels = lvl
))

Day$KW <- Day$Datum %>% lubridate::week()
Day$KW <- as.factor(Day$KW)
Day <- Day %>%
  mutate(
    Woche = if_else(
      Wochentag == "Montag" | Wochentag == "Dienstag" |
        Wochentag == "Mittwoch" |
        Wochentag == "Donnerstag" |
        Wochentag == "Freitag",
      "Werktag",
      "Wochenende"
    )
  )

# Wetterdaten auf Untersuchungszeitraum zuschneiden ####
meteo <- meteo %>% filter(time >= depo_start & time <= depo_end)
# meteo und Day sind nun gleich lang (578 Zeilen)

# Wetterdaten explorativ plotten ####
ggplot(data = meteo, aes(time, tre200jx)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Datum", y = "Lufttemperatur [°C]")

ggplot(data = meteo, aes(time, rre150j0)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Datum", y = "Niederschlag [mm]")

ggplot(data = meteo, aes(time, sre000d0)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Datum", y = "Sonnenscheindauer [min]")
```

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


# Daten auf Lockdown zuschneiden ####

lock.19 <-
  depo %>% filter(Datum >= lock_19_start & Datum <= lock_19_end)
lock_day.19 <-
  Day %>% filter(Datum >= lock_19_start & Datum <= lock_19_end)
lock_meteo.19 <-
  meteo %>% filter(time >= lock_19_start & time <= lock_19_end)

# Besucherzahlen summieren ####
SumBesuchende <-
  lock.19 %>% summarise(across("IN":"Total", sum)) %>% # Summen fuer alle Spalten
  `colnames<-`(c("sumVeloIN", "sumVeloOUT", "sumVelo"))

# Prozentwerte berechnen ####
ProzBesuchende <- SumBesuchende %>%
  transmute(
    ProzVeloIN = sumVeloIN / sumVelo,
    # transmute loescht alte Werte
    ProzVeloOUT = sumVeloOUT / sumVelo
  ) %>%
  mutate(across(everything(), round, 2)) # alles auf zwei Stellen runden

# Tabellenformat aendern ####
SumBesuchende <- gather(SumBesuchende)
ProzBesuchende <- gather(ProzBesuchende)

# Viualisierung Richtungsverteilung (Pie Chart) ####

palette <- c("grey90", "grey60")

pie(
  ProzBesuchende$value,
  labels = c("Velo IN; 56%", "Velo OUT; 43%"),
  col = palette
)
title("Prozentuales Verhältnis aller Velofahrer*innen")

dev.copy(
  png,
  "./Results-compare/Richtungsverteilung-19.png",
  width = 800,
  height = 500
) # Bild speichern

## png
##   3

dev.off()

## png
##   2

# Visualisierung Ganglinien ####

MeanVeloIN <-
  aggregate(
    lock.19$IN,
    by = list(lock.19$Stunde),
    FUN = mean,
    na.rm = TRUE
  )
```

```r
# Mittelwerte berechnen fuer IN und OUT ####
mean_besuchende19 <-
  lock.19 %>%
  group_by(Stunde) %>%
  summarise(across(3:4, mean)) %>% # Berechnet Mittelwerte fuer die angegebenen Spalten und gr
uppiert sie nach Stunden
  pivot_longer(2:3) %>% # in langes Format umwandeln, Spalten 2 und drei pivotieren (aus zwei
mach eine)
  `colnames<-`(c("Stunde", "Gruppe", "Durchschnitt")) # Spalten gemaess Vorgaben umbenennen

# Mittelwerte berechnen fuer Ganglinien nach Wochentagen ####
mean_besuchende19_tage <- lock.19 %>%
  group_by(Stunde, Wochentag) %>%
  summarise(across(5, mean)) # Berechnet Mittelwerte fuer die angegebenen Spalten und gruppier
t sie nach Stunden

# Ganglinien visualisieren (Barplot) ####
ggplot(data = mean_besuchende19, aes(Stunde, Durchschnitt, fill = Gruppe)) +
  geom_bar(position = "dodge", stat = "identity") +
  labs(x = "Uhrzeit [h]", y = "Durchschnittliche Anzahl Velo") +
  scale_fill_grey(start = 0.7, end = 0.4) +
  theme_bw(base_size = 15) +
  scale_x_continuous(limits = c(0, 23), breaks = seq(0, 23, by = 2)) +
  scale_y_continuous(limits = c(0, 50), breaks = seq(0, 50, by = 5))

ggsave("Ganglinien-19.png", path = "Results-compare/") # Bild speichern

# Ganglinien nach Wochentagen visualisieren (Lineplot) ####

palette <- c("#92D24F", "#01AF50", "#02AFEF", "#026CBF", "#712EA1", "#1D507A", "#1F4D77")
ltype <- c(
  "Montag" = "solid", "Dienstag" = "solid", "Mittwoch" = "solid",
  "Donnerstag" = "solid", "Freitag" = "solid",
  "Samstag" = "dotted", "Sonntag" = "longdash"
)

day.lines.19 <- ggplot(data = mean_besuchende19_tage, aes(Stunde, Total, color = Wochentag, li
netype = factor(Wochentag))) +
  geom_line(size = 1) +
  labs(x = "Uhrzeit", y = "Durchschnittliche Anzahl Velo") +
  scale_x_continuous(limits = c(0, 23), breaks = seq(0, 23, by = 1)) +
  scale_y_continuous(limits = c(0, 150), breaks = seq(0, 150, by = 25)) +
  scale_color_manual(breaks = lvl, labels = lvl, values = palette) +
  scale_linetype_manual(values = ltype) +
  guides(linetype = FALSE, color = guide_legend(
    nrow = 2, byrow = TRUE,
    override.aes = list(linetype = c(rep("solid", 5), "dotted", "twodash"))
  )) +
  theme_classic(base_size = 15) +
  theme(legend.position = "bottom", legend.title = element_blank(), legend.box = "vertical")

ggsave("Ganglinien-Tage-19.png", path = "Results-compare/", scale = 1.5, dpi = 300) # Bild spe
ichern

# Wochengang visualisieren (Boxplot) ####
ggplot(data = lock_day.19, aes(x = Wochentag, y = Total)) +
  geom_boxplot(size = 1) +
  labs(x = "Wochentag", y = "Anzahl Velo") +
  theme_classic(base_size = 15) +
  scale_y_continuous(limits = c(0, 1500), breaks = seq(0, 1500, by = 250)) +
  theme(axis.line = element_line(size = 1), axis.ticks = element_line(
    size =
      1
  ))

ggsave("Wochengang-19.png", path = "Results-compare/") # Bild speichern
```

```
# Saisongang visualisieren (Scatterplot) ####
season.19 <- ggplot(data = lock_day.19, aes(x = Datum, y = Total)) +
  geom_point() +
  geom_smooth() +
  annotate(
    geom = "rect", xmin = Fruehlingsferien_2019_start,
    xmax = Fruehlingsferien_2019_ende, ymin = -Inf,
    ymax = 1500, fill = "blue",
    alpha = 0.3
  ) + # zeichnet Rechteck
  scale_y_continuous(
    limits = c(-300, 1500),
    breaks = seq(0, 1500, by = 250)
  ) +
  labs(x = "Saisongang 2019", y = "Anzahl Velo") +
  theme_classic(base_size = 15)

season.19

ggsave("Saisongang-19.png", path = "Results-compare/") # Bild speichern

# Bestbesuchte Tage herausfiltern ####

# unlist(lock_day$Total) %>% sort(decreasing = T) %>% .[1:10] # ergibt nur die besten Werte
max_day <-
  lock_day.19 %>% slice_max(Total, n = 10) # selektiert die gesamten Zeilen mit den besten Wer
ten

write.csv2(max_day, "./Results-compare/bestbesuchte_tage_2019.csv")

# Nutzung Werktage und Wochenende ####

mean_werktag <- lock_day.19 %>%
  filter(Woche == "Werktag") %>%
  summarize(werktag = mean(Total)) # Mittelwert Werktage berechnen

mean_wochenende <-
  lock_day.19 %>%
  filter(Woche == "Wochenende") %>%
  summarize(Wochenende = mean(Total)) # Mittelwert Wochenende berechnen

mean_wochentage <-
  as.tibble(c(mean_werktag, mean_wochenende)) # Werte zusammenfuegen

write.csv2(
  mean_wochentage,
  "Results-compare/mittelwerte_biker_wochentage_2019.csv"
) # Werte als csv speichern

# Plot

ggplot(lock_day.19, aes(Woche, Total)) +
  geom_boxplot() +
  scale_y_continuous(limits = c(0, 1500), breaks = seq(0, 1500, by = 250)) +
  labs(x = "", y = "Anzahl Velo pro Tag") +
  theme_classic(base_size = 15)



ggsave("Tagesnutzung-19.png", path = "Results-compare/")

# Plotgrid Saisongang ####

grid <- cowplot::plot_grid(
   season.19, season.20,
   labels = c("g)", "h)"),
   ncol = 2, nrow = 1,
   label_y = 0.1, label_x = 0.05, label_size = 15, align = "v"
)
```

```
grid
cowplot::save_plot("Results-compare/Saisongang_grid.png", grid, ncol = 2.5, nrow = 1.25)

# Plotgrid Tagesgang ####

 grid <- cowplot::plot_grid(
   day.lines.19, day.lines.20,
   labels = c("i)", "k)"),
   ncol = 2, nrow = 1,
   label_y = 0.25, label_x = 0.05, label_size = 15, align = "v"
 )

 grid
 cowplot::save_plot("Results-compare/Tagesgang_grid.png", grid, ncol = 2.5, nrow = 1.25)


# Statistik: T-Test Unterscheid Werktage und Wochenende ####

 t.test(
   lock_day.19$Total [lock_day.19$Woche == "Werktag"],
   lock_day.19$Total [lock_day.19$Woche == "Wochenende"]
 )


# Statistik: T-Test Unterscheid 2019 und 2020 ####

 t.test(lock_day$Total, lock_day.19$Total)
```