Abgabe 2 für Computergestützte Methoden

Gruppe 100

Abgabedatum: 30.01.2025

Name	**Matrikelnummer**
Shainthavi Suthakaran	4250125
Melissa Tursucu	4020830
Gözde Ünal	4250543

Das R-Markdown Dokument für die Abgabe 2 ist unter folgendem Link verfügbar:

https://github.com/melissatuu/group100/blob/main/Abgabe_2_COMET_WiSe2425_Gruppe100.pdf

Inhaltsverzeichnis

Aufgabe 3: Programmieren in R

3.1 Simulation in R

An Weihnachten wird häufig gewichtelt. Das Prinzip ist Ihnen bestimmt vertraut: Alle bringen ein Geschenk mit, diese werden durchnummeriert, dann zieht jede Person eine Zahl und wird so ganz zufällig beschenkt. Das macht besonders viel Spaß, wenn man das eigens mitgebrachte Geschenk nicht selbst wieder zugewiesen bekommt. Aber wie wahrscheinlich ist eigentlich ein solcher Unglücksfall? Bestimmen Sie durch Simulation in R die Wahrscheinlichkeit, dass unter $n = 8$ Personen keine Person das eigene Geschenk zurückerhält.

Im Folgenden wird diese Wahrscheinlichkeit für den Fall von n = 8 Teilnehmern mithilfe einer **Simulation** berechnet:

```
# Simulation einer Funktion zum Wichtelproblem
wichtel_simulation <- function(n) {

  # Generiert eine Liste mit n Geschenken
  geschenke <- 1:n

  # Mischt die Geschenke in zufälliger Reihenfolge
  zufaellige_verteilung <- sample(geschenke)

  # Prüft, ob niemand sein eigenes Geschenk erhält
  return(all(geschenke != zufaellige_verteilung))
}

# Simuliert das Spiel 1.000.000 Mal und speichert die Ergebnisse
wahrscheinlichkeit_berechnen <- function(n, anzahl_simulationen = 1000000) {
  ergebnisse <- replicate(anzahl_simulationen, wichtel_simulation(n))

  # Mittelwert der Ergebnisse
  return(mean(ergebnisse))
}

# Wir berechnen die Wahrscheinlichkeit,
# dass bei n = 8 Teilnehmern niemand sein eigenes Geschenk erhält

# Anzahl der Teilnehmer
n <- 8

# Wahrscheinlichkeit, dass niemand sein eigenes Geschenk erhält
wahrscheinlichkeit_andere_geschenke <- wahrscheinlichkeit_berechnen(n)

# Ausgabe des Ergebnisses
print(wahrscheinlichkeit_andere_geschenke)
```

```
## [1] 0.369455
```

Interpretation

Die berechnete Wahrscheinlichkeit liegt bei mehrmaliger Durchführung der Simulation zwischen 0.367 und 0.369. Das bedeutet, dass in etwa 36,7 % bis 36,9 % der Fälle bei einer zufälligen Zuteilung mit n = 8 Teilnehmern niemand sein eigenes Geschenk erhält. Die leichte Schwankung der Ergebnisse kann dadurch erklärt werden, dass die Reihenfolge der Zuteilung bei jeder Simulation zufällig generiert wird.

3.2 Verallgemeinerte Funktion

Wir wollen uns jetzt von dem Spezialfall n = 8 lösen. Schreiben Sie Ihren R Code aus Aufgabe 3.1 in eine Funktion namens wichtel ungluеck um. Dabei soll der Funktions- aufruf wichtel ungluеck(n, k, iterationen = 1e6) die Wahrscheinlichkeit ausgeben, mit der höchstens k unter n Personen ihr eigenes Geschenk zurückerhalten. Mit dem Argument iterationen soll eine positive ganze Zahlübergeben werden, mit der die Anzahl der Iterationen für die Simulation einstellbar ist. Dieser Wert ist standardmäßig auf 1e6 gesetzt.

Die folgende **Funktion wichtel_unglueck** berechnet die Wahrscheinlichkeit, dass höchstens k unter n Personen ihr eigenes Geschenk erhalten. Die Anzahl der Iterationen ist standardmäßig auf 1 Million gesetzt.

```r
wichtel_unglueck <- function(n, k, iterationen = 1e6) {

  # Funktion für die Simulation eines einzelnen Szenarios
  treffer_simulation <- function() {

    # Generiert eine Liste von Geschenken mit n Personen
    geschenke <- 1:n

    # Mischt die Geschenke zufällig
    zufaellige_verteilung <- sample(geschenke)

    # Prüft, ob höchstens k Personen ihr eigenes Geschenk erhalten
    return(sum(geschenke == zufaellige_verteilung) <= k)
  }

  # Wiederholt die Simulation 1e6-mal und berechnet die Trefferwahrscheinlichkeit
  ergebnisse <- replicate(iterationen, treffer_simulation())

  # Gibt die durchschnittliche Wahrscheinlichkeit zurück
  return(mean(ergebnisse))
}
```

```r
# Simulation eines Beispiels mit n = 8 und k = 0

# Definiert die Anzahl der Teilnehmer
n <- 8

# Legt die maximale Anzahl der Personen fest,
# die ihr eigenes Geschenk erhalten dürfen
k <- 0

# Anzahl der Iterationen für die Simulation
iterationen <- 1e6

# Berechnet die Wahrscheinlichkeit,
# dass höchstens k Personen ihr eigenes Geschenk erhalten
wahrscheinlichkeit <- wichtel_unglueck(n, k, iterationen)

# Gibt die berechnete Wahrscheinlichkeit aus
print(wahrscheinlichkeit)
```

```
## [1] 0.367962
```

Interpretation

Mithilfe der verallgemeinerten Funktion wichtel_unglueck wird die Wahrscheinlichkeit für beliebige Teilnehmerzahlen n und maximale Eigenzuteilungen k berechnet. Für den Beispielaufruf mit n = 8 und k = 0 liefert die Funktion eine Wahrscheinlichkeit zwischen 0.367 und 0.369, dies ist dieselbe Wahrscheinlichkeit wie in Aufgabe 3.1. Dies bedeutet, dass in etwa 36,7 % bis 36,9% der Fälle höchstens **keine** Person ihr eigenes Geschenk erhält. Wie bereits bei der speziellen Simulation in Aufgabe 3.1 gilt auch hier, dass die Ergebnisse aufgrund der zufälligen Zuteilung bei jeder Durchführung leicht schwanken können.

3.3 Dokumentation des Codes

Damit Sie Ihren Code auch noch nächstes Jahr Weihnachten verstehen und einsetzen können, ist es wichtig, dass er gut dokumentiert ist. Ergänzen Sie Kommentare in Ihrem Code, die insbesondere die einzelnen Schritte, die Eingaben und die Ausgabe verständlich beschreiben.

Die Kommentare in der Abgabe sind durchgehend vorhanden und erklären die einzelnen Schritte sowie die Eingabe- und Ausgabewerte der Funktionen.

3.4 Testfälle

Überlegen Sie sich vier Testfälle, die von Ihrer Funktion wichtel_unglueck erfüllt sein müssen und implementieren Sie diese, wobei das R Paket {testthat} zu verwenden ist.

Um sicherzustellen, dass die Funktion wichtel_unglueck korrekt arbeitet, wurden vier Testfälle implementiert. Diese Testfälle müssen erfüllt werden, um verschiedene Szenarien zu überprüfen und die korrekte Funktionsweise der Simulation zu garantieren. Dafür wurde das R-Paket testthat verwendet.

```
# Installieren von testthat, falls nicht vorhanden
if (!require("testthat")) {
  install.packages("testthat")
}

library("testthat")

# 1. Test: n = k, höchstens alle Teilnehmer erhalten ihr eigenes Geschenk
# Dieses Szenario umfasst alle möglichen Zuteilungen
# (k = 0 + k = 1 + ... + k = n), daher ist die Wahrscheinlichkeit immer 1.

test_that("wichtel_unglueck mit n = k", (
  expect_equal(wichtel_unglueck(8, 8), 1)
))
```

```
## Test passed
```

```
# 2. Test: Monotonieprüfung, für eine gegebene Anzahl Personen n,
# sollte P(höchstens k) <= P(höchstens k+1) sein

# Da P(höchstens k+1) alle Fälle von P(höchstens k) enthält
# und zusätzlich weitere Fälle berücksichtigt,
# muss die Wahrscheinlichkeit monoton steigen oder gleich bleiben.

test_that("wichtel_unglueck monoton steigend mit k", {
  p_hoechstens_1 <- wichtel_unglueck(8, 1)
  p_hoechstens_2 <- wichtel_unglueck(8, 2)
  expect_true(p_hoechstens_1 <= p_hoechstens_2)
})
```

```
## Test passed
```

```r
# 3. Test: k < 0, unrealistischer Fall

# Da dies in der Realität nicht möglich ist,
# umfasst dieses Szenario keine gültigen Zuteilungen.
# Die Wahrscheinlichkeit muss daher immer 0 sein.

test_that("wichtel_unglueck mit k größer als n", (
  expect_equal(wichtel_unglueck(8, -3), 0)
))
```

```
## Test passed
```

```r
# 4. Test: Wahrscheinlichkeit liegt zwischen 0 <= p <= 1

# Da Wahrscheinlichkeiten per Definition niemals negativ sein
# oder einen Wert größer als 1 annehmen können,
# wird dieser Test sichergestellt,
# dass das Ergebnis der Funktion mathematisch korrekt bleibt.

test_that("wichtel_unglueck(n = 2, k = 1)", (
  expect_true(0 <= (result <- wichtel_unglueck(8, 5)) && result <= 1)
))
```

```
## Test passed
```

Ergebnis

Alle implementierten Testfälle wurden erfolgreich bestanden.

3.5 Vorbereitung des „bikeshare“-Datensatzes

Wir wenden uns nun dem “bikeshare” Datensatz zu, den Sie bereits von der ersten Abgabe kennen. Lesen Sie die Daten aus der .csv-Datei mit der Funktion read.csv() in R ein und speichern Sie ihn als data.frame ab. In der Aufgabe 4 werden Sie diese Daten auf verschiedene Arten visualisieren. Führen Sie vorher noch die folgenden Aufbereitungsschritte durch:

- *Filtern Sie die Daten nach der Station,die Ihrer Gruppe zugewiesen wurde. Die folgenden Schritte brauchen Sie nur für diesen Datenteil durchzuführen.*
- *Prüfen Sie, ob die Daten NAs enthalten. Falls ja, wenden Sie geeignete Imputationsverfahren an, sodass Ihr Datenteil für die Aufgabe 4 vollständig ist.*
- *LiegenDatenanomalien vor? Falls ja, überlegen Sie sich, wie Sie damit umgehen, sodass Ihr Datenteil für die Aufgabe 4 ausschließlich plausible Werte enthält. Insbesondere sollen Temperaturen in der Einheit Celsius anstatt Fahrenheit vorliegen.*
- *Bestimmen Sie den Monat mit der höchsten Gesamtanzahl ausgeliehener Fahrräder.*

In dieser Aufgabe wird der Datensatz[1] für die Analyse und Visualisierung vorbereitet. Dazu werden verschiedene Schritte durchgeführt, darunter das Einlesen und Filtern der Daten nach der zugewiesenen Station, das Identifizieren und Korrigieren fehlender oder fehlerhafter Werte sowie die Standardisierung der Einheiten. Ziel ist es, einen vollständigen und plausiblen Datensatz zu erstellen, der anschließend für weiterführende Analysen und Visualisierungen genutzt werden kann

Einlesen und Filtern des „bike_sharing"-Datensatzes

```r
# Working Direction setzen und CSV-Datei einlesen
setwd("/Users/shainthavisuthakaran/Downloads")
data <- read.csv("bike_sharing_data_(with_NAs).csv")

# Installiere und lade dplyr
if (!require("dplyr")) {install.packages("dplyr")}
library(dplyr)

# Station für Gruppe 100 definieren und Datensatz nach unserer Station filtern
gruppe_100 <- c("William St & Pine St")
gefilterter_datensatz <- data %>% filter(station%in% gruppe_100)

# Überprüfen auf fehlende Werte (NAs)
summary(gefilterter_datensatz)
```

```
##      group       station             date            day_of_year
##  Min.   :100   Length:364         Length:364         Min.   :  1.00
##  1st Qu.:100   Class :character   Class :character   1st Qu.: 92.25
##  Median :100   Mode  :character   Mode  :character   Median :182.50
##  Mean   :100                                         Mean   :182.81
##  3rd Qu.:100                                         3rd Qu.:272.75
##  Max.   :100                                         Max.   :365.00
##                                                      NA's   :2
##   day_of_week    month_of_year  precipitation       windspeed
##  Min.   :1.000   Min.   : 1.0   Min.   :-1.0000   Min.   :-1.00
##  1st Qu.:2.000   1st Qu.: 4.0   1st Qu.: 0.0000   1st Qu.: 8.28
##  Median :4.000   Median : 6.5   Median : 0.0000   Median :10.29
##  Mean   :3.986   Mean   : 6.5   Mean   : 0.1421   Mean   :10.89
##  3rd Qu.:6.000   3rd Qu.: 9.0   3rd Qu.: 0.0300   3rd Qu.:13.09
##  Max.   :7.000   Max.   :12.0   Max.   : 8.0500   Max.   :25.72
##  NA's   :2       NA's   :2      NA's   :1         NA's   :1
##  min_temperature average_temperature max_temperature     count
##  Min.   :-1.00   Min.   :-1.00       Min.   :-1.0    Min.   : -1.00
##  1st Qu.:37.00   1st Qu.:44.00       1st Qu.:52.0    1st Qu.: 59.50
##  Median :48.00   Median :55.00       Median :63.0    Median : 88.00
##  Mean   :48.98   Mean   :56.06       Mean   :63.6    Mean   : 86.06
##  3rd Qu.:61.50   3rd Qu.:68.50       3rd Qu.:77.5    3rd Qu.:114.00
##  Max.   :76.00   Max.   :83.00       Max.   :93.0    Max.   :175.00
##  NA's   :1       NA's   :1           NA's   :1       NA's   :1
```

[1] Vgl. Bikesharing-Daten (s. Quellenverzeichnis)

Identifikation fehlender Werte

```r
# Anzahl der Zeilen mit mindestens einem NA-Wert
anzahl_na_zeilen <- sum(!complete.cases(gefilterter_datensatz))
cat("Anzahl der Zeilen mit mindestens einem NA-Wert:", anzahl_na_zeilen, "\n")
```

```
## Anzahl der Zeilen mit mindestens einem NA-Wert: 12
```

Die folgende Tabelle zeigt klar, in welcher Zeile und in welcher Spalte ein fehlender Wert (NA) vorliegt.

Nummer	Zeile	Datum	Spalte
1	11	2023-01-11	min_temperature
2	22	2023-01-22	max_temperature
3	54	2023-02-23	day_of_year
4	140	2023-05-20	average_temperature
5	200	2023-07-19	month_of_year
6	244	2023-09-01	day_of_week
7	275	2023-10-02	day_of_year
8	276	2023-10-03	windspeed
9	290	2023-10-18	count
10	295	2023-10-23	precipitation
11	299	2023-10-27	day_of_week
12	351	2023-12-18	month_of_year

Korrektur und Imputation fehlender oder unplausibler Werte

Die Bearbeitung der NAs erfolgt systematisch von der linken zur rechten Spalte, wobei fehlende Werte Spalte für Spalte überprüft und imputiert werden.

Die Informationen zur Imputation der Werte in den Spalten `day_of_year`, `day_of_week` und `month_of_year` wurden anhand eines Kalenders[2] bestimmt.

[2] Vgl. Schulferien.org (2023)

```r
# Zeile 54: day_of_year auf 54 setzen
gefilterter_datensatz$day_of_year[54] <- 54

# Zeile 275: day_of_year auf 275 setzen
gefilterter_datensatz$day_of_year[275] <- 275
```

```r
# Zeile 244: day_of_week auf Freitag = 5 setzen
gefilterter_datensatz$day_of_week[244] <- 5

# Zeile 299: day_of_week auf Freitag = 5 setzen
gefilterter_datensatz$day_of_week[299] <- 5
```

```r
# Zeile 200: month_of_year auf Juli = 7 setzen
gefilterter_datensatz$month_of_year[200] <- 7

# Zeile 351: month_of_year auf Dezember = 12 setzen
gefilterter_datensatz$month_of_year[351] <- 12
```

Die NAs der numerischen Spalten *precipitation* und *windspeed* werden durch das arithmetische Mittel des jeweiligen Monats ersetzt. Dies stellt sicher, dass die Werte repräsentativ für die Bedingungen der jeweiligen Monate sind, z. B. im Oktober typisch herbstliche Werte berücksichtigt werden.

```r
# Monat des fehlenden Wertes ermitteln
monat_precipitation <- gefilterter_datensatz$month_of_year[295]

# Mittelwert der Spalte "precipitation" für den entsprechenden Monat berechnen
mittelwert_monat_precipitation <- mean(gefilterter_datensatz$precipitation
                                       [gefilterter_datensatz$month_of_year
                                       == monat_precipitation], na.rm = TRUE)

# Mittelwert in Zeile 295 der Spalte "precipitation" einsetzen
gefilterter_datensatz$precipitation[295] <- mittelwert_monat_precipitation
```

```r
# Monat des fehlenden Wertes ermitteln
monat_windspeed <- gefilterter_datensatz$month_of_year[276]

# Mittelwert der Spalte "windspeed" für den entsprechenden Monat berechnen
mittelwert_monat_windspeed <- mean(gefilterter_datensatz$windspeed
                                   [gefilterter_datensatz$month_of_year
                                   == monat_windspeed], na.rm = TRUE)

# Mittelwert in Zeile 276 der Spalte "windspeed" einsetzen
gefilterter_datensatz$windspeed[276] <- mittelwert_monat_windspeed
```

```r
# Werte für max_temperature und average_temperature in Zeile 11 extrahieren
max_value <- gefilterter_datensatz$max_temperature[11]
average_value <- gefilterter_datensatz$average_temperature[11]

# Den geschätzten Wert für min_temperature berechnen
# Formel: min_temperature = 2 * average_temperature - max_temperature
min_value <- 2 * average_value - max_value

# Den berechneten Wert in die Spalte "min_temperature" der Zeile 11 einsetzen
gefilterter_datensatz$min_temperature[11] <- min_value
```

Wir nutzen die Formel $average\_temperature = (min\_temperature + max\_temperature) / 2$, um repräsentative Werte zu berechnen. Für fehlende Werte von max_temperature und min_temperature haben wir diese Formel entsprechend umgestellt, um plausible Ergebnisse zu erzielen. Auch wenn diese Methode nicht die exakten Werte liefert, ist sie dennoch besser geeignet als die Verwendung des Mittelwerts des Monats oder der gesamten Spalte.

```r
# Min und Max an dem Tag (Zeile 140) berechnen
min_value <- gefilterter_datensatz$min_temperature[140]
max_value <- gefilterter_datensatz$max_temperature[140]

# Durchschnitt aus Min und Max berechnen
average_value <- (min_value + max_value) / 2

# Berechneten Wert in die Spalte "average_temperature" der Zeile 140 einsetzen
gefilterter_datensatz$average_temperature[140] <- average_value
```

```r
# Werte für min_temperature und average_temperature in Zeile 22 extrahieren
min_value <- gefilterter_datensatz$min_temperature[22]
average_value <- gefilterter_datensatz$average_temperature[22]

# Den Wert für max_temperature berechnen
# Formel: max_temperature = 2 * average_temperature - min_temperature
max_value <- 2 * average_value - min_value

# Den berechneten Wert in die Spalte "max_temperature" der Zeile 22 einsetzen
gefilterter_datensatz$max_temperature[22] <- max_value
```

Als letztes haben wir die Zeilen entfernt, in denen die Spalte “count” NA-Werte enthält

```r
# Entfernung der Zeilen
gefilterter_datensatz <- gefilterter_datensatz[
                         !is.na(gefilterter_datensatz$count), ]
```

Kontrolle, ob die Imputation erfolgreich war

```
anzahl_na_zeilen <- sum(!complete.cases(gefilterter_datensatz))
cat("Anzahl der Zeilen mit mindestens einem NA-Wert:", anzahl_na_zeilen, "\n")
```

```
## Anzahl der Zeilen mit mindestens einem NA-Wert: 0
```

Plausibilitätsprüfung und Umrechnungen

In diesem Abschnitt haben wir Plausibilitätsprüfungen durchgeführt, um sicherzustellen, dass die Werte in den numerischen Spalten realistisch und konsistent sind. Wir haben die Minimal- und Maximalwerte jeder Spalte überprüft und alle negativen Werte entfernt, da diese für Variablen wie Niederschlag und Windgeschwindigkeit keinen Sinn ergeben. Bei den Temperaturen haben wir zusätzlich überprüft, ob die negativen Werte plausibel sein könnten. Da dies nicht der Fall war, wurden die entsprechenden Zeilen entfernt. Außerdem wurde die logische Bedingung *min_temperature <= average_temperature <= max_temperature* validiert, um sicherzustellen, dass die Temperaturen in einem realistischen Verhältnis stehen. Fehlerhafte Zeilen wurden identifiziert und gegebenenfalls korrigiert oder entfernt, um die Datenqualität für die weitere Analyse zu gewährleisten.

Wir beginnen mit der Überprüfung der Spannbreite der Werte in den Spalten *precipitation, windspeed, min_temperature, average_temperature, max_temperature und count*, um festzustellen, welche Werte diese annehmen können und wo sich die Werte innerhalb des Datensatzes befinden.

```
# Minimum und Maximum für die gewünschten Spalten berechnen und ausgeben
cat("precipitation: Min =", min(gefilterter_datensatz$precipitation,
                                na.rm = TRUE),
    ", Max =", max(gefilterter_datensatz$precipitation, na.rm = TRUE), "\n")
```

```
## precipitation: Min = -1 , Max = 8.05
```

Durch genauerer Beobachtung dieser Spalte fällt auf, dass ein maximaler Wert von 8.05 Zoll (200mm) auftaucht. Hier könnte es sich möglicherweise um eine Datenanomalie handeln. Eine Recherche im Internet jedoch stellt klar, dass sich am 29.09.2023 ein Starkregen in New York[3] ereignet hat, was den Vorfall erklärt.

```
cat("windspeed: Min =", min(gefilterter_datensatz$windspeed, na.rm = TRUE),
    ", Max =", max(gefilterter_datensatz$windspeed, na.rm = TRUE), "\n")
```

```
## windspeed: Min = -1 , Max = 25.72
```

```
cat("min_temperature: Min =", min(gefilterter_datensatz$min_temperature,
                                  na.rm = TRUE),
    ", Max =", max(gefilterter_datensatz$min_temperature, na.rm = TRUE), "\n")
```

```
## min_temperature: Min = -1 , Max = 76
```

[3] Vgl. NPR (2023)

```r
cat("average_temperature: Min =", min(gefilterter_datensatz$average_temperature,
                                      na.rm = TRUE),
    ", Max =", max(gefilterter_datensatz$average_temperature, na.rm = TRUE),
    "\n")
```

```
## average_temperature: Min = -1 , Max = 83
```

```r
cat("max_temperature: Min =", min(gefilterter_datensatz$max_temperature,
                                  na.rm = TRUE),
    ", Max =", max(gefilterter_datensatz$max_temperature, na.rm = TRUE), "\n")
```

```
## max_temperature: Min = -1 , Max = 93
```

```r
cat("count: Min =", min(gefilterter_datensatz$count, na.rm = TRUE),
    ", Max =", max(gefilterter_datensatz$count, na.rm = TRUE), "\n")
```

```
## count: Min = -1 , Max = 175
```

Imputation und Eliminierung negativer Werte

```r
# Negative Werte in der Spalte "count" eliminieren
gefilterter_datensatz <- gefilterter_datensatz[
                         gefilterter_datensatz$count >= 0, ]
```

```r
# Zeilen mit negativen Werten in der Spalte "precipitation" finden
negative_precipitation <- which(gefilterter_datensatz$precipitation < 0)

# Schleife über alle Zeilen mit negativen Werten
for (i in negative_precipitation) {
  # Monat des negativen Wertes ermitteln
  monat_precipitation <- gefilterter_datensatz$month_of_year[i]

  # Mittelwert der Spalte "precipitation" für den entsprechenden Monat berechnen
  mittelwert_monat_precipitation <- mean(gefilterter_datensatz$precipitation[
    gefilterter_datensatz$month_of_year == monat_precipitation &
    gefilterter_datensatz$precipitation >= 0], na.rm = TRUE)

  # Den negativen Wert durch den berechneten Mittelwert ersetzen
  gefilterter_datensatz$precipitation[i] <- mittelwert_monat_precipitation
}
```

```r
# Sicherstellen, dass die Spalte "windspeed" numerisch ist
gefilterter_datensatz$windspeed <- as.numeric(gefilterter_datensatz$windspeed)

# Zeilen mit negativen Werten in der Spalte "windspeed" finden
negative_windspeed <- which(gefilterter_datensatz$windspeed < 0)
```

```r
# Schleife über alle Zeilen mit negativen Werten
for (i in negative_windspeed) {
  monat_windspeed <- gefilterter_datensatz$month_of_year[i]

  # Mittelwert der Spalte "windspeed" für den entsprechenden Monat berechnen
  mittelwert_monat_windspeed <- mean(gefilterter_datensatz$windspeed[
    gefilterter_datensatz$month_of_year == monat_windspeed &
    gefilterter_datensatz$windspeed >= 0], na.rm = TRUE)

  # Den negativen Wert durch den berechneten Mittelwert ersetzen
  gefilterter_datensatz$windspeed[i] <- mittelwert_monat_windspeed
}
```

```r
# Zeilen mit negativen Werten in der Spalte "min_temperature" finden
negative_min_temperature <- which(gefilterter_datensatz$min_temperature < 0)

# Schleife über alle Zeilen mit negativen Werten
for (i in negative_min_temperature) {
  # Werte für max_temperature und average_temperature
  # in der entsprechenden Zeile extrahieren
  max_value <- gefilterter_datensatz$max_temperature[i]
  average_value <- gefilterter_datensatz$average_temperature[i]

  # Den geschätzten Wert für min_temperature berechnen
  # Formel: min_temperature = 2 * average_temperature - max_temperature
  min_value <- 2 * average_value - max_value

  # Berechneten Wert in die Spalte "min_temperature"
  # der entsprechenden Zeile einsetzen
  gefilterter_datensatz$min_temperature[i] <- min_value
}
```

```r
# Zeilen mit negativen Werten in der Spalte "average_temperature" finden
negative_average_temperature <- which(
                                gefilterter_datensatz$average_temperature < 0)

# Schleife über alle Zeilen mit negativen Werten
for (i in negative_average_temperature) {
  # Werte für min_temperature und max_temperature
  # in der entsprechenden Zeile extrahieren
  min_value <- gefilterter_datensatz$min_temperature[i]
  max_value <- gefilterter_datensatz$max_temperature[i]

  # Durchschnitt aus Min und Max berechnen
  average_value <- (min_value + max_value) / 2

  # Den berechneten Wert in die Spalte "average_temperature"
  # der entsprechenden Zeile einsetzen
  gefilterter_datensatz$average_temperature[i] <- average_value}
```

```r
# Zeilen mit negativen Werten in der Spalte "max_temperature" finden
negative_max_temperature <- which(gefilterter_datensatz$max_temperature < 0)

# Schleife über alle Zeilen mit negativen Werten
for (i in negative_max_temperature) {
  # Werte für min_temperature und average_temperature
  # in der entsprechenden Zeile extrahieren
  min_value <- gefilterter_datensatz$min_temperature[i]
  average_value <- gefilterter_datensatz$average_temperature[i]


  # Den Wert für max_temperature berechnen
  # Formel: max_temperature = 2 * average_temperature - min_temperature
  max_value <- 2 * average_value - min_value

  # Den berechneten Wert in die Spalte "max_temperature"
  # der entsprechenden Zeile einsetzen
  gefilterter_datensatz$max_temperature[i] <- max_value
}
```

Nachdem wir alle negativen Werte ersetzt beziehungsweise eliminiert haben, kontrollieren wir die Spalten noch einmal.

```
## Summe der Zeilen mit negativen Werten in den Spalten:

##       precipitation           windspeed     min_temperature average_temperature
##                   0                   0                   0                   0
##     max_temperature
##                   0
```

Hier überprüfen wir, ob die logische Bedingung *min_temperature <= average_temperature <= max_temperature* gilt.

```r
# Überprüfung, ob Bedingung für jede Zeile gilt
check_condition <- gefilterter_datensatz$min_temperature <=
                   gefilterter_datensatz$average_temperature &
                   gefilterter_datensatz$average_temperature <=
                   gefilterter_datensatz$max_temperature

# Zeilen finden, in denen die Bedingung nicht erfüllt ist
invalid_rows <- which(!check_condition)
```

```r
# Ausgabe
if (length(invalid_rows) > 0) {
  cat("Die Bedingung wird in folgenden Zeilen nicht erfüllt:\n")
  print(invalid_rows)

  # Zeilen mit den fehlerhaften Werten anzeigen
  print(gefilterter_datensatz[invalid_rows, c("min_temperature",
                                              "average_temperature",
                                              "max_temperature")])
} else {
  cat("Die Bedingung ist in allen Zeilen erfüllt.\n")
}
```

```
## Die Bedingung ist in allen Zeilen erfüllt.
```

Anschließend wandeln wir die Einheit der Temperaturen von Fahrenheit in Celsius[4] um

```r
# Funktion zur Umrechnung von Fahrenheit in Celsius
fahrenheit_to_celsius <- function(f) {
  (f - 32) * 5 / 9
}

# Umrechnung für min_temperature, max_temperature und average_temperature
gefilterter_datensatz$min_temperature <- fahrenheit_to_celsius(
                                           gefilterter_datensatz$min_temperature)
gefilterter_datensatz$max_temperature <- fahrenheit_to_celsius(
                                           gefilterter_datensatz$max_temperature)
gefilterter_datensatz$average_temperature <- fahrenheit_to_celsius(
                                      gefilterter_datensatz$average_temperature)
```

Bezüglich der Aufgabe 4 haben wir uns entschieden die Niederschlagsmenge[5] und Windgeschwindigkeit[6] bereits hier umzuwandeln.

```r
# Umrechnungsfunktion für Inches in Millimeter
inches_to_mm <- function(inches) {
  inches * 25.4
}

# Umrechnung der Niederschlagsmenge (inches → mm)
gefilterter_datensatz$precipitation <- inches_to_mm(
                                      gefilterter_datensatz$precipitation)
```

[4] Vgl. Metric Conversions (2025)
[5] Vgl. Inch Calculator (2025)
[6] Vgl. Fahrschulsuche.at (2025)

```r
# Umrechnungsfaktor von mph in km/h
mph_to_kmh <- function(mph) {
  mph * 1.60934
}

# Umrechnung der Windgeschwindigkeit
gefilterter_datensatz$windspeed <- mph_to_kmh(gefilterter_datensatz$windspeed)
```

Bestimmung des Monats mit der höchsten Gesamtanzahl ausgeliehener Fahrräder

```r
# Summieren der Gesamtanzahl ausgeliehener Fahrräder pro Monat
monatliche_summen <- aggregate(count ~ month_of_year,
                               data = gefilterter_datensatz, sum, na.rm = TRUE)

# Die Summen für alle Monate anzeigen
print(monatliche_summen)
```

```
##    month_of_year count
## 1              1  1550
## 2              2  1625
## 3              3  1926
## 4              4  2654
## 5              5  3231
## 6              6  3135
## 7              7  3383
## 8              8  3949
## 9              9  2966
## 10            10  2712
## 11            11  2428
## 12            12  1683
```

```r
# Monat mit der höchsten Anzahl finden
max_monat <- monatliche_summen[which.max(monatliche_summen$count), ]
max_monat
```

```
##   month_of_year count
## 8             8  3949
```

Antwort: Der Monat mit der höchsten Gesamtanzahl ausgeliehener Fahrräder ist: August (= 8) mit insgesamt 3949 ausgeliehenen Fahrrädern.

Aufgabe 4: Visualisieren in R

4.1 Problemematische Darstellung der Grafik

Identifizieren Sie in der folgenden Grafik1 3 Dinge, die den in der Vorlesung[7] *besprochenen Grundsätzen der Datenvisualisierung widersprechen:*

Das grüne Smiley- und rote Sadface-Symbol stellen keine zusätzliche Informationen dar und sind daher überflüssig. Zudem beinhaltet die Grafik übermäßig viele Textboxen, die den Chartjunk vergrößern und dadurch entsteht eine größere Unübersichtlichkeit.

Die farbliche Darstellung der Grafik ist ebenfalls nicht ideal, da sie Grün- und Rottöne verwendet, die für Personen mit einer Rot-Grün-Schwäche schwer zu unterscheiden sind.

Die Grafik besitzt keine y-Achse und liefert somit keine Informationen über die Einheiten, in denen die Balken gemessen werden. Zwar wird in der Grafik das Ziel von 4 Mrd.€ für 2020 erwähnt, aber ohne eine durchgehende Skala ist die genaue Zuordnung schwierig.

Zudem fehlt die Beschriftung für die x-Achse mit den konkreten Einheiten (z.B. Jahre), welcher möglicherweise auch erklären könnte, weshalb nur die Zeitpunkte 2014 und 2020 beobachtet werden.

4.2 Zusammenhänge zwischen Fahrradnutzung und Einflussfaktoren

Stellen Sie den Zusammenhang zwischen der Anzahl ausgeliehener Fahrräder und:

- *der Temperatur (in Grad Celsius);*
- *der Niederschlagsmenge;*
- *der Windgeschwindigkeit (in Kilometer pro Stunde);*
- *der Zeit*

grafisch dar, die in der Vorlesung besprochenen Grundsätze der Datenvisualisierung befolgend (diese sind auch in den nachfolgenden Aufgaben zu beachten). Wählen Sie eine geeignete Darstellungsform, wobei das R-Paket ggplot2 zu verwenden ist. (Hinweis: es sind 4 Grafiken zu erstellen.)

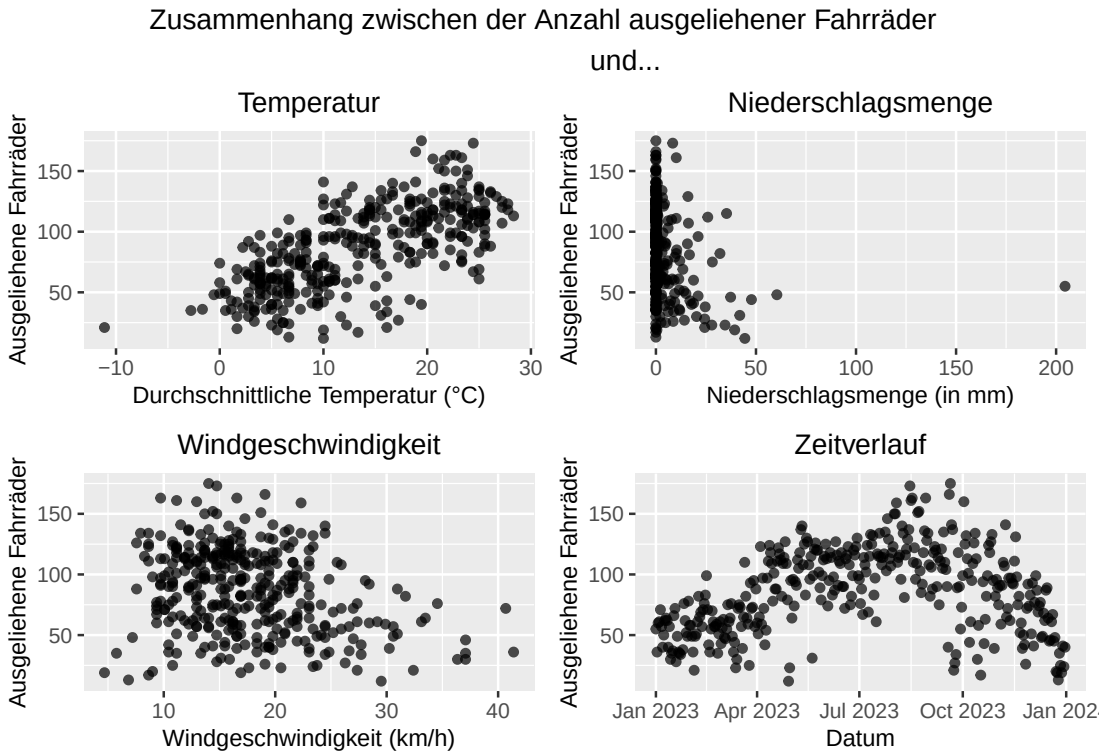
Die Abbildung stellt den Zusammenhang zwischen der Anzahl ausgeliehener Fahrräder und der mittleren Temperatur, der Niederschlagsmenge, der Windgeschwindigkeit und der Zeit grafisch dar.

```r
# Bibliotheken installieren und laden
if (!requireNamespace("ggplot2", quietly = TRUE)) {install.packages("ggplot2")}
library(ggplot2)

if (!requireNamespace("gridExtra", quietly = TRUE)) {
  install.packages("gridExtra")}
```

[7] Vgl. Computergestützte Methoden - Block 4, Kapitel 1, Folie 14

```r
library(gridExtra)

# Daten in ggplot2-konformem Format vorbereiten
gefilterter_datensatz$date <- as.Date(gefilterter_datensatz$date,
                                      format = "%Y-%m-%d")

# Temperatur (in °C) vs. Anzahl ausgeliehener Fahrräder
grafik_temperatur <- ggplot(data = gefilterter_datensatz) +
  geom_point(aes(x = average_temperature, y = count),
             color = "black", alpha = 0.7) +
  xlab("Durchschnittliche Temperatur (°C)") +
  ylab("Ausgeliehene Fahrräder") +
  ggtitle("Temperatur") +
  theme(
    axis.title = element_text(size = 10),
    plot.title = element_text(hjust = 0.5, size = 12))

# Niederschlagsmenge (in mm) vs. Anzahl ausgeliehener Fahrräder
grafik_niederschlag <- ggplot(data = gefilterter_datensatz) +
  geom_point(aes(x = precipitation, y = count), color = "black", alpha = 0.7) +
  xlab("Niederschlagsmenge (in mm)") +
  ylab("Ausgeliehene Fahrräder") +
  ggtitle("Niederschlagsmenge") +
  theme(
    axis.title = element_text(size = 10),
    plot.title = element_text(hjust = 0.5, size = 12))

# Windgeschwindigkeit (in km/h) vs. Anzahl ausgeliehener Fahrräder
grafik_windgeschwindigkeit <- ggplot(data = gefilterter_datensatz) +
  geom_point(aes(x = windspeed, y = count), color = "black", alpha = 0.7) +
  xlab("Windgeschwindigkeit (km/h)") +
  ylab("Ausgeliehene Fahrräder") +
  ggtitle("Windgeschwindigkeit") +
  theme(
    axis.title = element_text(size = 10),
    plot.title = element_text(hjust = 0.5, size = 12))

# Datum (Zeit) vs. Anzahl ausgeliehener Fahrräder
grafik_zeit <- ggplot(data = gefilterter_datensatz) +
  geom_point(aes(x = date, y = count), color = "black", alpha = 0.7) +
  xlab("Datum") +
  ylab("Ausgeliehene Fahrräder") +
  ggtitle("Zeitverlauf") +
  theme(
    axis.title = element_text(size = 10),
    plot.title = element_text(hjust = 0.5, size = 12))
```

```r
# Plots in einem Grid arrangieren
grid.arrange(grafik_temperatur, grafik_niederschlag,
             grafik_windgeschwindigkeit, grafik_zeit,
             ncol = 2, nrow = 2,
             top = "Zusammenhang zwischen der Anzahl ausgeliehener Fahrräder
                    und...")
```

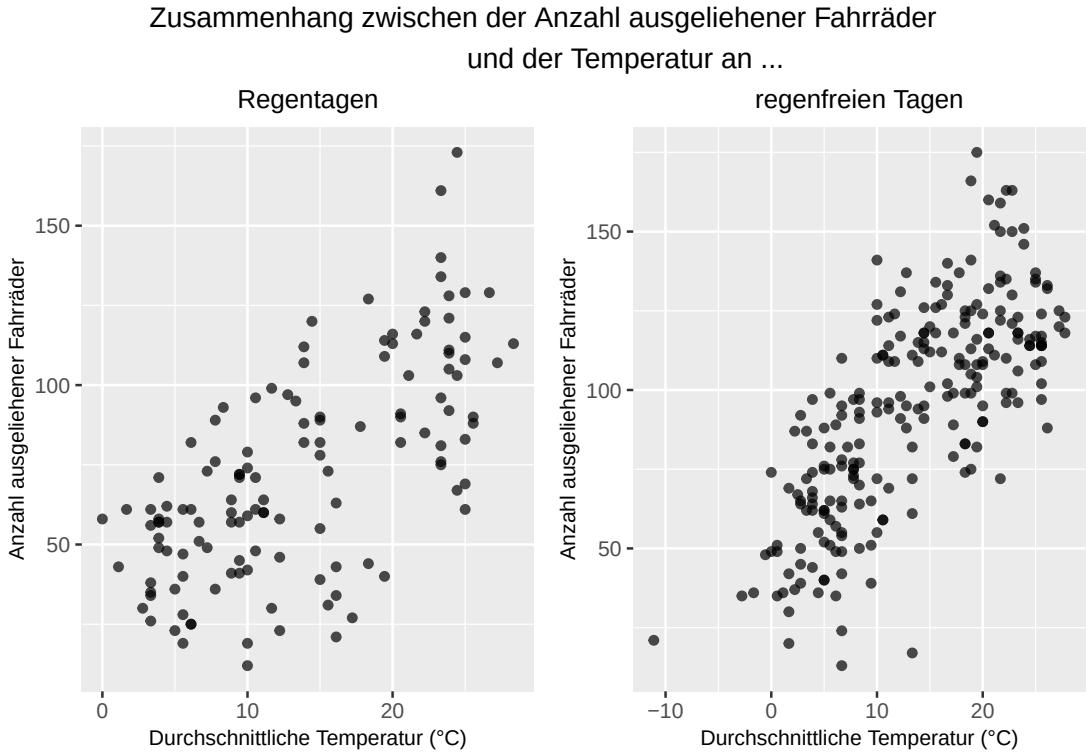


4.3 Temperatur vs. Fahrradnutzung an Regen- und regenfreien Tagen

- *Stellen Sie den Zusammenhang zwischen der Anzahl ausgeliehener Fahrräder und der Temperatur dar, und zwar getrennt für:*
- *Tage, an denen es geregnet hat;*
- *Tage, an denen es nicht geregnet hat.*

In diesem Abschnitt wird der Zusammenhang zwischen der Temperatur und der Anzahl ausgeliehener Fahrräder visualisiert, getrennt nach Tagen mit und ohne Regen.

```r
# Bibliotheken laden
library(ggplot2)

# Grafik für Regentage
grafik_regentage <- ggplot(data = filter(gefilterter_datensatz,
                                         precipitation > 0)) +
  geom_point(aes(x = average_temperature, y = count), color = "black",
                 alpha = 0.7) +
  xlab("Durchschnittliche Temperatur (°C)") +
  ylab("Anzahl ausgeliehener Fahrräder") +
  ggtitle("Regentagen") +
  theme(
    axis.title = element_text(size = 9),
    plot.title = element_text(hjust = 0.5, size = 11)
  )

# Grafik für regenfreie Tage
grafik_regenfrei <- ggplot(data = filter(gefilterter_datensatz,
                                         precipitation == 0)) +
  geom_point(aes(x = average_temperature, y = count), color = "black",
                 alpha = 0.7) +
  xlab("Durchschnittliche Temperatur (°C)") +
  ylab("Anzahl ausgeliehener Fahrräder") +
  ggtitle("regenfreien Tagen") +
  theme(
    axis.title = element_text(size = 9),
    plot.title = element_text(hjust = 0.5, size = 11)
  )

# Beide Plots nebeneinander anzeigen
grid.arrange(grafik_regentage, grafik_regenfrei, nrow = 1, ncol = 2,
             top = "Zusammenhang zwischen der Anzahl ausgeliehener Fahrräder
                    und der Temperatur an ...")
```



4.4 Verteilungen von Einflussfaktoren und Fahrradnutzung

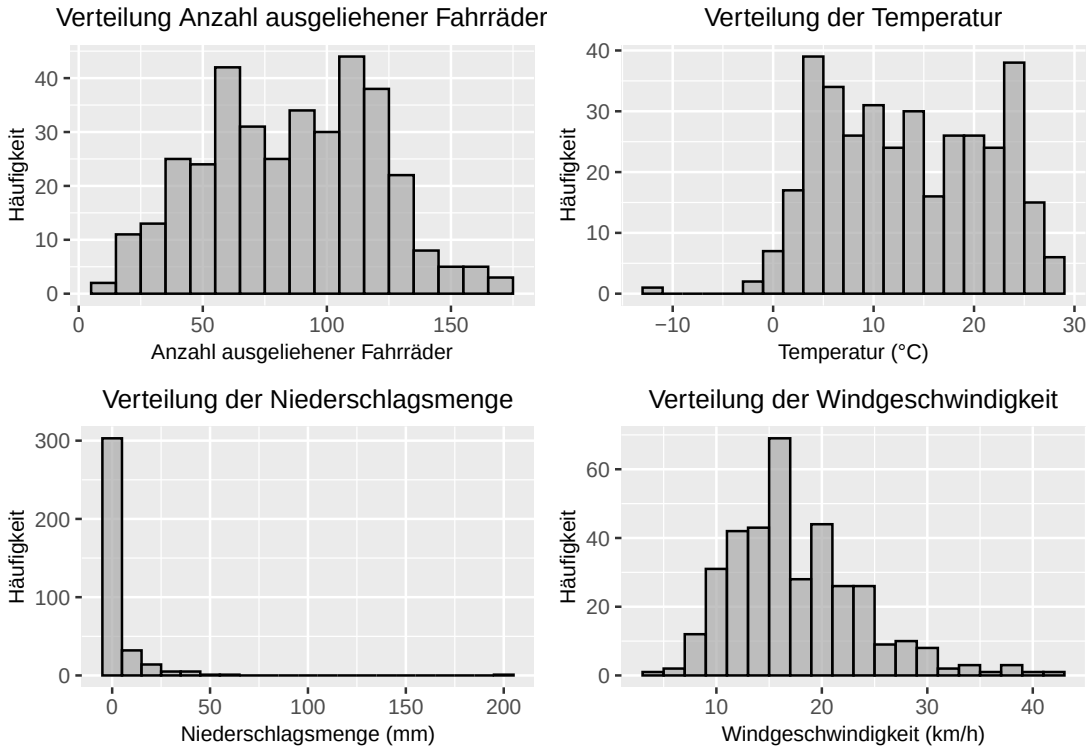
Stellen Sie die Verteilung

- *der Anzahl ausgeliehener Fahrräder;*
- *der Temperatur (in Grad Celsius);*
- *der Niederschlagsmenge;*
- *der Windgeschwindigkeit (in Kilometer pro Stunde)*

grafisch dar. Wählen Sie eine geeignete, in der Vorlesung vorgestellte Darstellungsform, wobei das R-Paket ggplot2 zu verwenden ist. (Hinweis: es sind 4 Grafiken zu erstellen.)

In diesem Abschnitt werden die Verteilungen der Anzahl ausgeliehener Fahrräder, der Temperatur (in °C), der Niederschlagsmenge und der Windgeschwindigkeit (in km/h) grafisch dargestellt.

```r
library(ggplot2)

# Verteilung der Anzahl ausgeliehener Fahrräder
grafik_count <- ggplot(data = gefilterter_datensatz) +
  geom_histogram(aes(x = count), binwidth = 10, fill = "darkgrey",
                 color = "black", alpha = 0.7) +
  xlab("Anzahl ausgeliehener Fahrräder") +
  ylab("Häufigkeit") +
  ggtitle("Verteilung Anzahl ausgeliehener Fahrräder") +
  theme(plot.title = element_text(hjust = 0.5, size = 11.3),
        axis.title.x = element_text(size = 9),
        axis.title.y = element_text(size = 9))

# Verteilung der Temperatur
grafik_temperature <- ggplot(data = gefilterter_datensatz) +
  geom_histogram(aes(x = average_temperature), binwidth = 2, fill = "darkgrey",
                 color = "black", alpha = 0.7) +
  xlab("Temperatur (°C)") +
  ylab("Häufigkeit") +
  ggtitle("Verteilung der Temperatur") +
   theme(plot.title = element_text(hjust = 0.5, size = 11.3),
        axis.title.x = element_text(size = 9),
        axis.title.y = element_text(size = 9))

# Verteilung der Niederschlagsmenge
grafik_precipitation <- ggplot(data = gefilterter_datensatz) +
  geom_histogram(aes(x = precipitation), binwidth = 10, fill = "darkgrey",
                 color = "black", alpha = 0.7) +
  xlab("Niederschlagsmenge (mm)") +
  ylab("Häufigkeit") +
  ggtitle("Verteilung der Niederschlagsmenge") +
  theme(plot.title = element_text(hjust = 0.5, size = 11.3),
        axis.title.x = element_text(size = 9),
        axis.title.y = element_text(size = 9))

# Verteilung der Windgeschwindigkeit
grafik_windspeed <- ggplot(data = gefilterter_datensatz) +
  geom_histogram(aes(x = windspeed), binwidth = 2, fill = "darkgrey",
                 color = "black", alpha = 0.7) +
  xlab("Windgeschwindigkeit (km/h)") +
  ylab("Häufigkeit") +
  ggtitle("Verteilung der Windgeschwindigkeit") +
  theme(plot.title = element_text(hjust = 0.5, size = 11.3),
        axis.title.x = element_text(size = 9),
        axis.title.y = element_text(size = 9))

# Alle Plots in einer 2x2-Anordnung anzeigen
grid.arrange(grafik_count, grafik_temperature, grafik_precipitation,
             grafik_windspeed, nrow = 2, ncol = 2)
```



4.5 Verteilung der Anzahl ausgeliehener Fahrräder nach Jahreszeiten

Erweitern Sie Ihre Grafiken aus 4.4, indem Sie die Verteilung der Anzahl ausgeliehener Fahrräder getrennt nach Jahreszeit, also für:

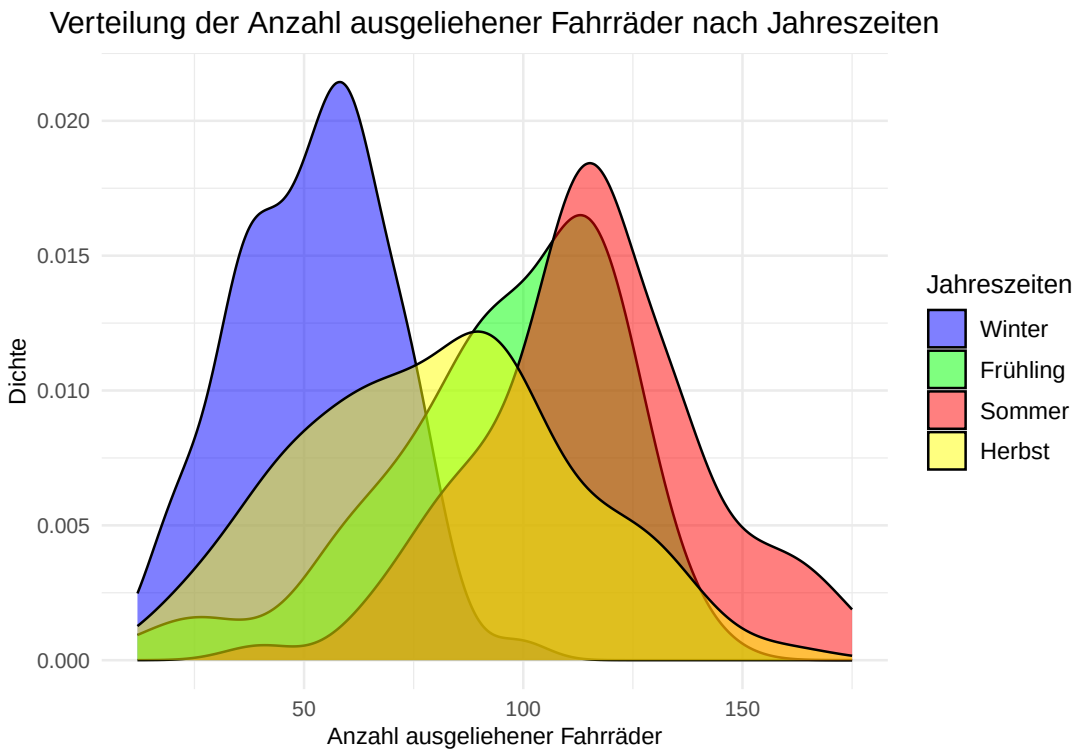
- *den Frühling;*
- *den Sommer;*
- *den Herbst;*
- *den Winter,*

in einer Grafik darstellen. Nutzen Sie dafür sich überlagernde Kerndichteschätzer und verschiedene, transparente Farben. (Hinweis: es ist eine Grafik zu erstellen).

In dieser Abbildung wird die Verteilung der Anzahl ausgeliehener Fahrräder erweitert, indem sie für die vier Jahreszeiten[8] Frühling, Sommer, Herbst und Winter getrennt dargestellt wird.

[8] Vgl. Deutscher Wetterdienst (2025)

```r
# Bibliotheken laden
library(ggplot2)

# Jahreszeiten erstellen (astronomische Grenzen)
breaks <- c(as.Date("2023-01-01"), as.Date("2023-03-20"), as.Date("2023-06-20"),
            as.Date("2023-09-22"), as.Date("2023-12-20"), as.Date("2023-12-31"))
labels <- c("Winter", "Frühling", "Sommer", "Herbst", "Winter")

# Umwandlung der "date"-Spalte in Date-Typ
gefilterter_datensatz$date <- as.Date(gefilterter_datensatz$date)

# Jahreszeiten-Spalte direkt im Datensatz erstellen
gefilterter_datensatz$Jahreszeiten <- cut(gefilterter_datensatz$date,
                                          breaks = breaks, labels = labels,
                                          include.lowest = TRUE, right = TRUE)

# Eine Grafik mit überlagernden Kerndichteschätzern für alle Jahreszeiten
ggplot_count <- ggplot(data = gefilterter_datensatz,
                       aes(x = count, fill = Jahreszeiten)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(values = c("Winter" = "blue", "Frühling" = "green",
                               "Sommer" = "red", "Herbst" = "yellow")) +
  scale_x_continuous(limits = c(min(gefilterter_datensatz$count, na.rm = TRUE),
                                max(gefilterter_datensatz$count, na.rm = TRUE)))+
  labs(
    title = "Verteilung der Anzahl ausgeliehener Fahrräder nach Jahreszeiten",
    x = "Anzahl ausgeliehener Fahrräder",
    y = "Dichte"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 13),
    axis.title.x = element_text(size = 10),
    axis.title.y = element_text(size = 10),
    legend.title = element_text(size = 11),
    legend.text = element_text(size = 10)
  )

# Grafik anzeigen
print(ggplot_count)
```



4.6 Erstellung eines 3D-Scatterplots mit Plotly

Erstellen Sie mithilfe des R-Pakets plotly einen 3D-Scatterplot. Auf der x-Achse soll die Temperatur, auf der y-Achse die Windgeschwindigkeit und auf der z-Achse die Anzahl ausgeliehener Fahrräder abgebildet sein. Nutzen Sie für die Punkte des 3D-Scatterplots eine geeignete Farbskala, wobei die Farbe von der Anzahl ausgeliehener Fahrräder abhängen soll. (Hinweis: es ist eine Grafik zu erstellen.)

Diese Visualisierung ist ein interaktiver 3D-Scatterplot, der den Zusammenhang zwischen der durchschnittlichen Temperatur, der Windgeschwindigkeit und der Anzahl ausgeliehener Fahrräder veranschaulicht. Da Plotly in RMarkdown bei der Erstellung eines PDF-Dokuments nicht unterstützt wird, können wir leider nur eine statische 2D-Ansicht in Form eines gespeicherten PNG-Bildes anbieten, anstatt der interaktiven 3D-Visualisierung.

```r
library(plotly)

# 3D-Scatterplot erstellen
figure <- plot_ly(
  data = gefilterter_datensatz,
  x = ~windspeed,
  y = ~average_temperature,
  z = ~count,
  type = "scatter3d",
  mode = "markers",
  marker = list(
    size = 5,
    color = ~count,
    colorscale = "Viridis",
    opacity = 0.8
  )
)

# Layout-Einstellungen mit größeren Achsenbeschriftungen & Titel weiter unten
figure <- figure %>% layout(
  title = list(
    text = "3D Scatterplot: Temperatur, Windgeschwindigkeit
            und ausgeliehene Fahrräder",
    font = list(size = 26),
    yref = "container",
    y = 0.95
  ),
  scene = list(
    xaxis = list(title = list(text = "Temperatur (°C)",
                              font = list(size = 23))),
    yaxis = list(title = list(text = "Windgeschwindigkeit (km/h)",
                              font = list(size = 23))),
    zaxis = list(title = list(text = "Anzahl ausgeliehener Fahrräder",
                              font = list(size = 23)))
  )
)

# Grafik anzeigen
figure
```

Die folgende Abbildung zeigt die gespeicherte Plotly-Grafik als PNG-Datei:

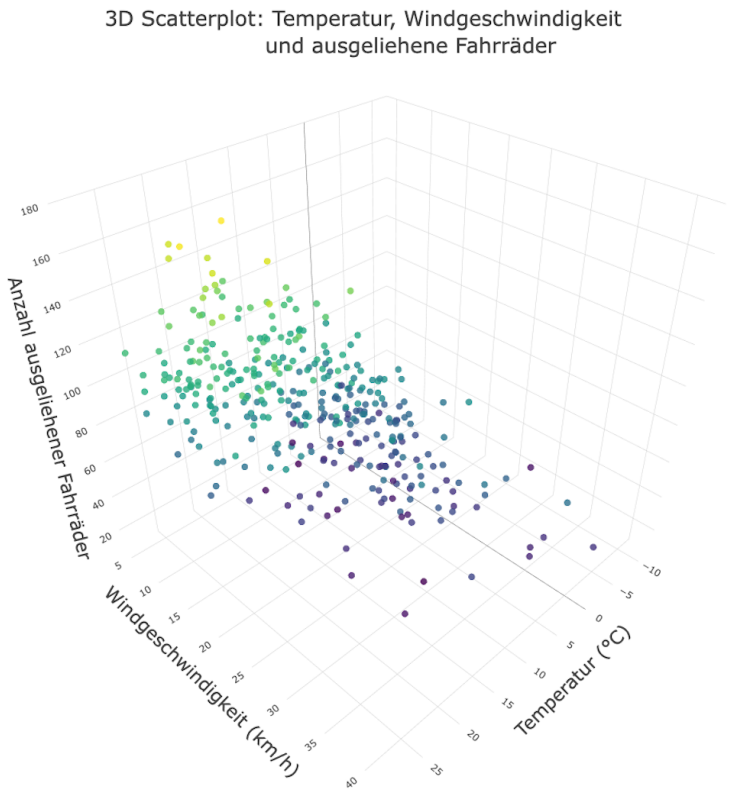


Figure 1: 3D Scatterplot: Temperatur, Windgeschwindigkeit und ausgeliehene Fahrräder

Quellenverzeichnis

- Deutsche Wetterdienst (DWD). (2025). *Bestimmung der Jahreszeiten.* Abrufbar unter: https://www.dwd.de/DE/service/lexikon/Functions/glossar.html?lv3=101324&lv2=101304 (Zugriff: 28. Januar 2025).
- Fahrschulsuche.at. (2025). *MPH in KMH umrechnen.* Abrufbar unter: https://www.fahrschulsuche.at/service/mph-in-kmh— (Zugriff: 28. Januar 2025).
- Inch Calculator. (2025). *Zoll in Millimeter umrechnen.* Abrufbar unter: https://www.inchcalculator.com/convert/inch-to-millimeter/ (Zugriff: 28. Januar 2025).
- Metric Conversions. (2025). *Fahrenheit in Celsius umrechnen.* Abrufbar unter: https://www.metric-conversions.org/de/temperatur/fahrenheit-in-celsius.htm (Zugriff: 28. Januar 2025).
- NPR. (2023). *New York swamped by record-breaking rainfall as more downpours expected Saturday.* Abrufbar unter: https://www.npr.org/2023/09/30/1202824340/new-york-swamped-by-record-breaking-rainfall-as-more-downpours-expected-saturday (Zugriff: 28. Januar 2025).
- Schulferien.org. (2023). *Kalender der Schulferien in Deutschland 2023.* Abgerufen am 20. Januar 2025, von https://www.schulferien.org/deutschland/kalender/2023/
- Universität Bielefeld. (2025). *Datensatz "bike_sharing_data_(with_NAs).csv" für die Abgabe der Veranstaltung "Computergestützte Methoden".*
- Universität Bielefeld. (2025). *Skript zur Vorlesung "Computergestützte Methoden".* Unveröffentlichte Vorlesungsunterlagen.