Weiner_Passanten01

Contents


```
library(tidyverse)
library(readr)
library(broom)
library(gridExtra)
library(stats)
library(janitor)
library(skimr)
library(lubridate)
library(dplyr)
library(lubridate)
library(ggplot2)
library(fmsb)
library(corrplot)
```

```
getwd()
```

```
## [1] "C:/DHBW/Semester3/DataScience/Weiner_inf24_Passanten"
```

```
#Globale Variablen
tage_vektor <- c("Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag", "Sonntag")
monat_vektor <- c("Jan", "Feb", "Mar", "Apr", "Mai", "Jun", "Jul", "Aug", "Sep", "Okt", "Nov", "Dez" )
globale_farb_palette = c(
   "Kaiserstraße" = "#8D34D1",
   "Schönbornstraße" = "#D19534",
   "Spiegelstraße" = "#2EB865",

  # Die Aggregate
```

```
  "Gesamtdurchschnitt" = "#404040",
  "Gesamtsumme"        = "#404040"
)
```

Passanten in Würzburg

Import und Datenbereinigung

Am Beginn unserer Explorativen Datenanalyse (EDA) steht der Import der Rohdaten.

```
passanten_raw <- read_csv2("Datensatz/passanten_wuerzburg.csv")
passanten <- as_tibble(passanten_raw)

passanten <- passanten %>%
  select(Zeitstempel,Wetter,Temperatur,Passanten,'Location Name',GeoPunkt) %>%
  mutate(
    jahr = year(Zeitstempel),
    monat = month(Zeitstempel),
    woche = isoweek(Zeitstempel),
    tag = day(Zeitstempel),
    stunde = hour(Zeitstempel),
    wochentag = weekdays(Zeitstempel),
    tag_im_jahr = yday(Zeitstempel)
  ) %>%
  filter(jahr != 2023)
```

Nun besitzt man die Daten als Dataframe “passanten” und zur Referenz als “passanten_raw”. Dabei wurde der Dataframe passanten bereits in ein tibble Format überführt (ein moderner Dataframe) und bereits angepasst. So wurden weitere Spalten zum Dataframe hinzugefügt, die den Umgang mit den Daten später erleichtern, z. B. ist nun jede Woche über einen Index klar abfragbar. Auch wurden alle Daten aussortiert, die nicht aus dem Jahr 2024 stammen. Um nun eine sinnvolle, deskriptive oder weiterführende Explorative Datenanalyse durchzuführen, sollten die Daten zuerst bereinigt werden. Als Nächstes benennen wir die Spaltennamen in Snake-Case um.

```
passanten <- clean_names(passanten)
```

Anschließend wird die Anzahl an leeren Feldern in den Spalten gezählt.

```
kable(colSums(is.na(passanten)), digits = 0, caption = "Summe aller NA werte pro Spalte")
```

Table 1: Summe aller NA werte pro Spalte

	x
zeitstempel	0
wetter	112
temperatur	112
passanten	0
location_name	0
geo_punkt	0

	x
jahr	0
monat	0
woche	0
tag	0
stunde	0
wochentag	0
tag_im_jahr	0

Es lässt sich entnehmen, dass Wetter und Temperatur jeweils 112 NA-Einträge haben. Dies kann diverse Gründe haben. Nun weiß man, dass der Datensatz intakt ist und keine fehlenden Daten in wichtigen Spalten wie `zeitstempel` und `passanten` vorliegen. Nun haben wir einen Datensatz, mit dem man eine EDA durchführen kann.

Die folgende Beschreibung gibt bereits einen guten Überblick über die Daten und wie sie erfasst werden.

Aufgabe 1

Beschreibung des Datensatz

Der Datensatz enthält Daten zu einer Passantenzählung in der Würzburger Innenstadt aus dem Jahr 2024. Die wichtigste Spalte ist hierbei die Anzahl der Passanten, welche in `passanten` gemessen wird. Dabei wird die Anzahl der Passanten mit jeweils einem Zeitstempel und weiteren Metadaten wie Temperatur, Ort der Aufzeichnung und Koordinaten des Ortes zeilenweise angegeben. Man kann entnehmen, dass jede Stunde eine Aufzeichnung pro Location stattfindet.

```
skim(passanten)
```

Table 2: Data summary

Name	passanten
Number of rows	26015
Number of columns	13
Column type frequency:	
character	4
numeric	8
POSIXct	1
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
wetter	112	1	3	19	0	9	0
location_name	0	1	12	15	0	3	0
geo_punkt	0	1	36	37	0	3	0
wochentag	0	1	6	10	0	7	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
temperatur	112	1	11.26	7.92	-11	5.5	10.7	17.0	34	
passanten	0	1	840.45	996.55	0	53.0	467.0	1341.0	8075	
jahr	0	1	2024.00	0.00	2024	2024.0	2024.0	2024.0	2024	
monat	0	1	6.50	3.46	1	3.0	6.0	10.0	12	
woche	0	1	26.31	15.13	1	13.0	26.0	39.5	52	
tag	0	1	15.74	8.81	1	8.0	16.0	23.0	31	
stunde	0	1	11.44	6.90	0	5.0	11.0	17.0	23	
tag_im_jahr	0	1	183.12	105.99	1	91.0	182.0	276.0	366	

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
zeitstempel	0	1	2024-01-01	2024-12-31 22:00:00	2024-06-30 16:00:00	8694

Der Datensatz besteht aus 26015 Zeilen und 13 Spalten, von denen 8 numerische, eine POSIXct und 4 Zeichenketten als Datentypen enthalten. Die Vollständigkeit ist bei allen außer den bereits beschriebenen Spalten gegeben. Ebenfalls ist zu erkennen, dass man mit 366 Tagen, 52 Wochen und 12 Monaten ein vollständig abgebildetes Jahr hat. Auch zu erkennen ist, dass es sich durch die Gesamtzahl von 366 Tagen um ein Schaltjahr handeln muss.

Erhebung

Die Daten stammen von Zählstationen an verschiedenen Punkten aus der Würzburger Innenstadt. Mit Hilfe von Laserschranken zählen diese die Anzahl der Passanten, welche gerade die Messstation queren. Durch das Messen mit mehreren Laserschranken pro Messstation kann auch die Geh-Richtung des Passanten bestimmt werden [1]. Zur Konsistenz der Daten wird vermerkt: “Nach Herstellerangabe kann mit der verwendeten Technik bis zu einem Durchfluss von ca. 500 Personen pro Minute eine Zählgenauigkeit von 99% erreicht werden.” vgl. [1]. Dabei ist zu beachten, dass eine Zählstation eine Straße bis maximal 32m Breite abdecken kann. Die Erheber der Daten versichern jedoch, dass “Bei den veröffentlichten Daten handelt es sich immer um die Passantenfrequenz der gesamten Straßenbreite (außer es ist explizit anders angegeben).” vgl. [1].

Standorte

```
kable(table(passanten$location_name), digits = 0, caption = "Auflistung aller einzigartigen Location we
```

Table 6: Auflistung aller einzigartigen Location werte und wie of diese vorkommen

Var1	Freq
Kaiserstraße	8694
Schönbornstraße	8648
Spiegelstraße	8673

```
kable(table(passanten$geo_punkt), digits = 0, caption = "Auflistung aller einzigartigen Geo Punkte und
```

Table 7: Auflistung aller einzigartigen Geo Punkte und wie of diese vorkommen

Var1	Freq
49.79512717222457, 9.934114106308467	8673
49.795490162266525, 9.931060093195851	8648
49.798498976405355, 9.933887635731686	8694

Wenn man die angegebenen Koordinaten und Messstellen anschaut, sieht man, dass es jeweils drei unterschiedliche Werte in diesen Spalten gibt. Entscheidend ist nun die Anzahl der Dopplungen dieser Werte. Bei genauerer Betrachtung sieht man, dass die jeweiligen Dopplungen der Menge `location` und `Geo Punkt` gleich sind. Damit kann man davon ausgehen, dass jede Location einer eindeutigen Koordinate anhand der Anzahl der Dopplungen zugeordnet werden kann. Wenn man nun die Punkte auf einer Karte einträgt, erhält man folgende Übersicht:

Figure 1: Markierte Standorte der Messtationen in der Würzburger Innenstadt

Die Koordinaten stimmen mit der jeweiligen Straße und dem Namen der Messstation überein. Man kann sehen, dass die Stationen direkt in der Innenstadt platziert sind. Dabei ist jede Station in der Nähe einer Sehenswürdigkeit bzw. eines öffentlichen Gebäudes.

Die Messstation Schönbornstraße befindet sich nah an der Marienkapelle, die Messstation Spiegelstraße auf dem Weg zum Hofgarten und die Messstation Kaiserstraße ist in der Nähe des Hauptbahnhofs. Alle diese Straßen sind Hauptverkehrsstraßen, auf denen mit vielen Passanten zu rechnen ist. Das Dreiecksmuster, welches die Stationen aufspannen, bildet somit eine Art Transitstrecke zwischen: Hauptbahnhof -> Hofgarten -> Marienkapelle -> Hauptbahnhof.

##Leitfragen Nach der grundlegenden Beschreibung der Daten und einer Einführung in den Datensatz sowie seinen Kontext folgen Leitfragen, welche helfen die EDA zu leiten. - Gibt es besondere Ereignisse in der Würzburger Innenstadt, welche erkennbar sind? - Was ist die beliebteste Region in der Innenstad? - Woher kommt die Beliebtheit? (Attraktionen, wichtiger Teil des Alltags, Veranstaltungsorte, …)

Aufgabe 2

```r
#--- 1. Analyse aggregiert ---
# Funktion guppiert nach location_name und berechnet anschließend: passanten jahres Summe, erfasste Tag
# Eingabe: df passanten
# Ausgabe: aggregiert_jahressumme_pro_location
aggregiert_jahressume_pro_location <- passanten %>%
  group_by(location_name) %>%
  summarise(

    # 1. Jahressumme pro Standort
    passanten_jahr_summe = sum(passanten, na.rm = TRUE),

    anzahl_tage_erfasst = n_distinct(as.Date(zeitstempel)),

    # 2. Mittelwert pro Monat
    durchschnitt_pro_monat = passanten_jahr_summe / 12,

    # 3. Mittelwert pro Woche
    durchschnitt_pro_woche = passanten_jahr_summe / (anzahl_tage_erfasst / 7),

    # 4. Mittelwert pro Tag
    durchschnitt_pro_tag = passanten_jahr_summe / anzahl_tage_erfasst,

    # Mittelwert pro Stunde
    durchschnitt_pro_stunde = durchschnitt_pro_tag / 24
  )

# --- 2. Analyse aggregiert (insgesamt) ---
# Funktion berechnet die Jahressumme aller passenten um damit den Mittelwert über alle Stationen zu ber
# Eingabe: df passanten
# Ausgabe: passanten_insgesamt
passanten_insgesamt <- passanten %>%
  summarise(

    # 1. Jahressumme (Gesamt)
    passanten_jahr_summe = sum(passanten, na.rm = TRUE),

    # Hilfsberechnung: Anzahl der einzigartigen Tage im gesamten Datensatz
    anzahl_tage_erfasst = n_distinct(as.Date(zeitstempel)),

    # 2. Mittelwert pro Monat (Gesamt)
    durchschnitt_pro_monat = passanten_jahr_summe / 12,

    # 3. Mittelwert pro Woche (Gesamt)
    durchschnitt_pro_woche = passanten_jahr_summe / (anzahl_tage_erfasst / 7),

    # 4. Mittelwert pro Tag (Gesamt)
    durchschnitt_pro_tag = passanten_jahr_summe / anzahl_tage_erfasst,

    # 5. Mittelwert pro Stunde (Gesamt)
    durchschnitt_pro_stunde = durchschnitt_pro_tag / 24
  )
```

```
kable(aggregiert_jahressume_pro_location, digits = 0, caption = "Jahressumme und Mittelwerte der Passant
```

Table 8: Jahressumme und Mittelwerte der Passantenanzahl nach Messstelle

location_name	passanten_jahressumme	anzahl_tage_erfasst	durchschnitt_pro_jahr	durchschnitt_pro_monat	durchschnitt_pro_woche	durchschnitt_pro_tag	durchschnitt_pro_stunde
Kaiserstraße	7482611	366	623551	143110	20444	852	
Schönbornstraße	9420052	366	785004	180165	25738	1072	
Spiegelstraße	4961655	366	413471	94895	13556	565	

```
kable(passanten_insgesamt, digits = 0, caption = "Jahressumme der Passantenanzahl")
```

Table 9: Jahressumme der Passantenanzahl

passanten_jahressumme	anzahl_tage_erfasst	durchschnitt_pro_jahr	durchschnitt_pro_monat	durchschnitt_pro_woche	durchschnitt_pro_tag	durchschnitt_pro_stunde
21864318	366	1822026	418170	59739	2489	

Bereits durch die einfache Berechnung der Mittelwerte pro Jahr, Monat, Woche und Tag lässt sich festhalten, dass die Schönbornstraße die meiste Auslastung, mit fast 10 Millionen Passanten 2024, erhalten hat. Mit 21864318 Passanten, die 2024 insgesamt gezählt wurden, stellt die Schönbornstraße somit fast die Hälfte des gesamten Passantenverkehrs dar. Den Daten nach ordnet sich die Kaiserstraße als zweite und die Spiegelstraße als am wenigsten ausgelastete Straße an. Ebenfalls auffällig ist, dass die Schönbornstraße eine fast doppelt so hohe Auslastung wie die Spiegelstraße hat. Somit ist hier eine ungleiche Aufteilung der Gesamtauslastung festzustellen.

Einfügen Interpretation der Lage und der Verbindungen, die die Straßen aufspannen

Aufgabe 3

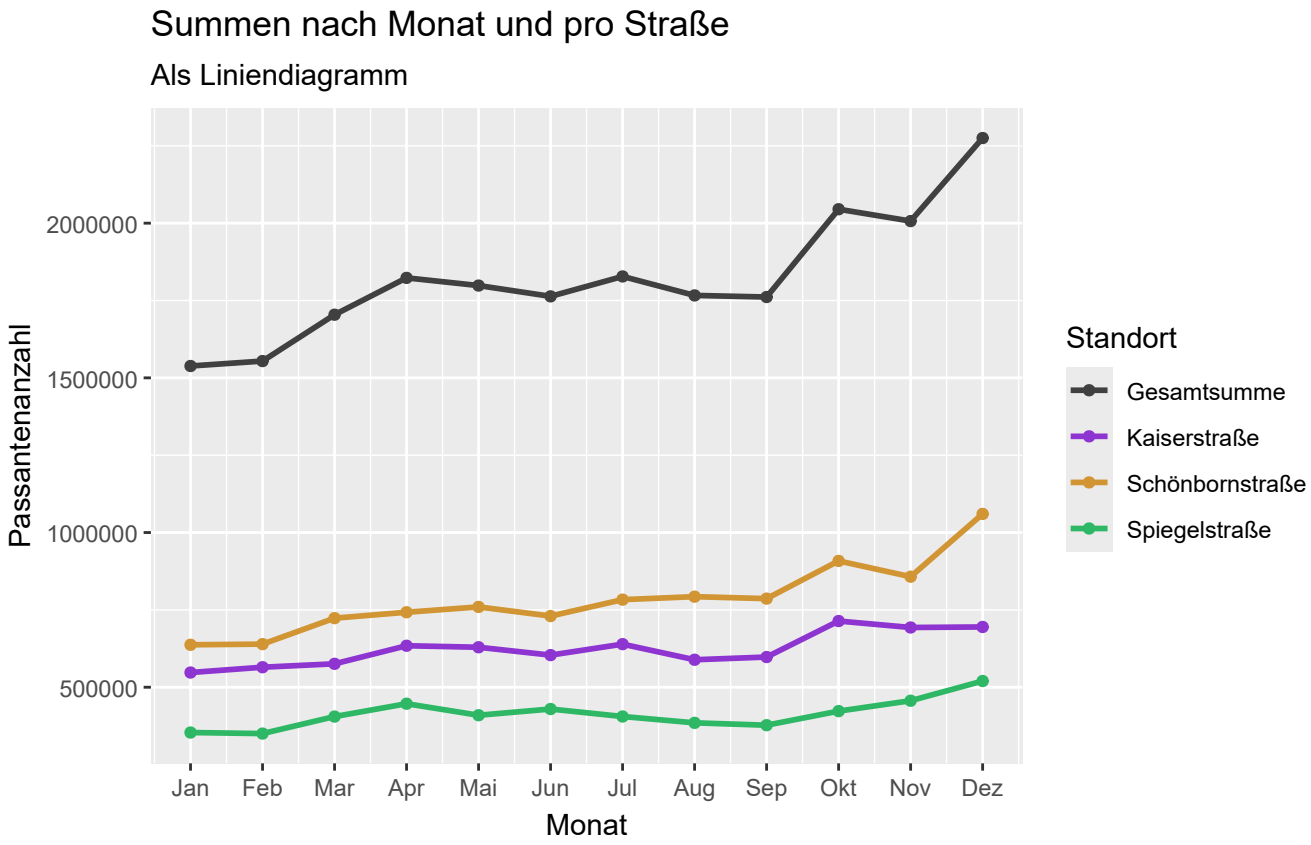
```
#' [Monatssumme über das Jahr]
#' @description
#' [Die Funktion Berechnet die Monatsumme der Passaten, gruppiert nach der location_name pro jahr]
#'
#' @param passanten df
#'
#' @return
#' [Rückgabewert ist ein der df sume_monat. Das wide Format zeigt die Daten sortiert nach monat und Loc
#'
summe_monat <- passanten %>%
   group_by(location_name, monat) %>%
   summarise(monatssumme = sum(passanten)) %>%
   ungroup() %>%
   #Vertauschen der Zeilen und Spalten
   pivot_wider(names_from = location_name,
               values_from = monatssumme)%>%
   mutate(Gesamtsumme = rowSums(across(where(is.numeric)), na.rm = TRUE)) %>%
   arrange(monat)

kable(summe_monat, digits = 0, caption = "Summe aller Passanten nach Monat und pro Straße wie über alle
```

Table 10: Summe aller Passanten nach Monat und pro Straße wie über alle Straßen summiert

monat	Kaiserstraße	Schönbornstraße	Spiegelstraße	Gesamtsumme
1	547543	637205	353571	1538320
2	564705	639348	350214	1554269
3	575659	723285	405180	1704127
4	634108	742347	446756	1823215
5	629219	759642	409499	1798365
6	603836	730054	429538	1763434
7	639235	783184	405390	1827816
8	588829	792729	384926	1766492
9	597582	786448	377293	1761332
10	713967	908278	422816	2045071
11	693123	857273	456390	2006797
12	694805	1060259	520082	2275158

```
#' [long plot format wandlung]
#' @description
#' [Formt den df summe_monat in ein long Format um]
#'
#' @param summe_monat
#'
#' @return
#' [Gib den df summe_monat_plot in einem long Format aus welches zur Graphischen Darstellung verwendet
mittelwert_monat_plot <- summe_monat %>%
   pivot_longer(cols = -monat,
                names_to = "location_name",
                values_to = "gesamtsumme")
```

```
plot_aufgabe3 <- ggplot(data = mittelwert_monat_plot, aes(x = monat, y = gesamtsumme, color = location_
  geom_line(linewidth = 1) +  # Zeichnet die Linien
  geom_point() +              # Fügt die Datenpunkte hinzu
   scale_color_manual(values = globale_farb_palette)+
  scale_x_continuous(breaks = 1:12, labels = monat_vektor)+

  # Titel und Achsenbeschriftungen
  labs(title = "Summen nach Monat und pro Straße",
       subtitle = "Als Liniendiagramm",
       x = "Monat",
       y = "Passantenanzahl",
       color = "Standort")
plot_aufgabe3
```



Das erstellte Diagramm bestätigt das aus Aufgabe 2 bereits beschriebene Muster. Die Auslastung der Schönbornstraße ist konstant am höchsten, gefolgt von der Kaiserstraße und der Spiegelstraße. Diese Reihenfolge ist über das gesamte Jahr erkennbar, was darauf schließen lässt, dass diese Auslastung nicht durch Events oder außergewöhnliche Ereignisse, sondern eher den Alltag in der Würzburger Innenstadt darstellt. In der Gesamtsumme kann man einen Anstieg ab April erkennen. Dies könnte auf den Beginn des Frühlings zurückzuführen sein.

Im September und Oktober scheinen einige Events die Anzahl der Passanten in der Innenstadt erhöht zu haben. Das Liniendiagramm zeigt entsprechende Ausreißer im September in der Gesamtsumme, aber auch an den jeweiligen Stationen. Grund für die Erhöhung von September auf Oktober sind mehrere Veranstaltungen. Zum einen das "Würzburger Stadtfest" vom 14.-15. Sep 2024 und die Würzburger Weinparade vom 29. August bis 8. September. Daraus ergibt sich der Anstieg in der Gesamtsumme von ca. 500.000 Passanten. Speziell auf der Kaiserstraße und Schönbornstraße zeigt das Liniendiagramm einen markanten Anstieg der Gesamtzahl pro Messstation. Hierzu betrachten wir den Ort, an dem das Stadtfest stattfand. Zum Stadtfest ist zu sagen, dass es eine dezentrale Veranstaltung ist, bei der "sich fast die gesamte Innenstadt in eine Vergnügungsmeile mit mehreren Bühnen, einem vielfältigen Programm" verwandelt, vgl. [2]. Mit dem fehlenden Anstieg der Passantenzahl trifft die Beschreibung "fast die gesamte Innenstadt" sehr genau auf die tatsächlichen Daten zu. Somit trägt das Stadtfest zur Erhöhung der Passantenzahl bei, ist aber nicht der Grund für den erkennbaren Anstieg auf der Kaiser- und Schönbornstraße. Die Weinparade hingegen fand für einen längeren Zeitraum exklusiv auf und um den Unteren Marktplatz bei der Marienkapelle, in direkter Nachbarschaft zur Schönbornstraße statt [3]. Da die Kaiserstraße eine direkte Verbindung von Hauptbahnhof und Marienkapelle darstellt, lässt sich so der explizite Anstieg der Passanten für diesen Zeitraum erklären.

Im Dezember gab es weitere Veranstaltungen in der Innenstadt. Hierbei zeigt das Liniendiagramm einen Anstieg in der Gesamtsumme, aber auch auf der Schönborn- und Spiegelstraße. Die Erklärung hierfür ist der Weihnachtsmarkt von Ende November bis zum 23. Dezember 2024 [4]. Veranstaltungsort ist hier ebenfalls der Obere und Untere Marktplatz und die Schönbornstraße, weshalb diese den höchsten Anstieg verzeichnet.

Aufgabe 4

```
tagessumme <- passanten %>%
   mutate(
      Datum = as_date(zeitstempel)
   ) %>%
   group_by(Datum, wochentag, location_name) %>%
   summarise(
    Tagessumme = sum(passanten, na.rm = TRUE)
    ) %>%
   ungroup()

gesamttagessumme <- passanten %>%
   mutate(
      Datum = as_date(zeitstempel)
   ) %>%
   group_by(Datum, wochentag) %>%
   summarise(
    Tagessumme_Gesamt = sum(passanten, na.rm = TRUE),
    .groups = 'drop' )


#' [Mittlerer Tageswert (Gegliedert, für Plot)]
#' @description
#' Berechnet den durchschnittlichen TAGESwert (mean(Tagessumme))
#' für jeden Wochentag, aufgeschlüsselt nach 'location_name'.
#' Sortiert zudem die Wochentage (als Faktor) in die korrekte Reihenfolge.
#' @param tagessumme [DataFrame] Der DF 'tagessumme' (Ergebnis von Block 1).
#' @return
#' [DataFrame] 'durchschnittlicher_tageswert_location_plot' (langes Format).
#'   Enthält den mittleren Tageswert pro Wochentag und Location (21 Zeilen).
durchschnittlicher_tageswert_location_plot <- tagessumme %>%
   mutate(
    wochentag = factor(wochentag, levels = tage_vektor)
   )%>%
   group_by(wochentag, location_name)%>%
   summarise(
      durchschnitt_wochentag_location = mean(Tagessumme)
   )%>%
   ungroup()


#' [Mittlerer Tageswert (Gesamt)]
#' @description
#' Berechnet den durchschnittlichen GESAMT-Tageswert (mean(Tagessumme_Gesamt))
#' für jeden Wochentag (summiert über alle Standorte).
#' Sortiert zudem die Wochentage (als Faktor) in die korrekte Reihenfolge.
#' @param gesamttagessumme [DataFrame] Der DF 'gesamttagessumme' (Ergebnis von Block 2).
#' @return
#' [DataFrame] 'durchschnittlicher_tageswert_gesamt' (langes Format).
#'   Enthält den mittleren Gesamt-Tageswert pro Wochentag (7 Zeilen).
durchschnittlicher_tageswert_gesamt <- gesamttagessumme %>%
   mutate(
```

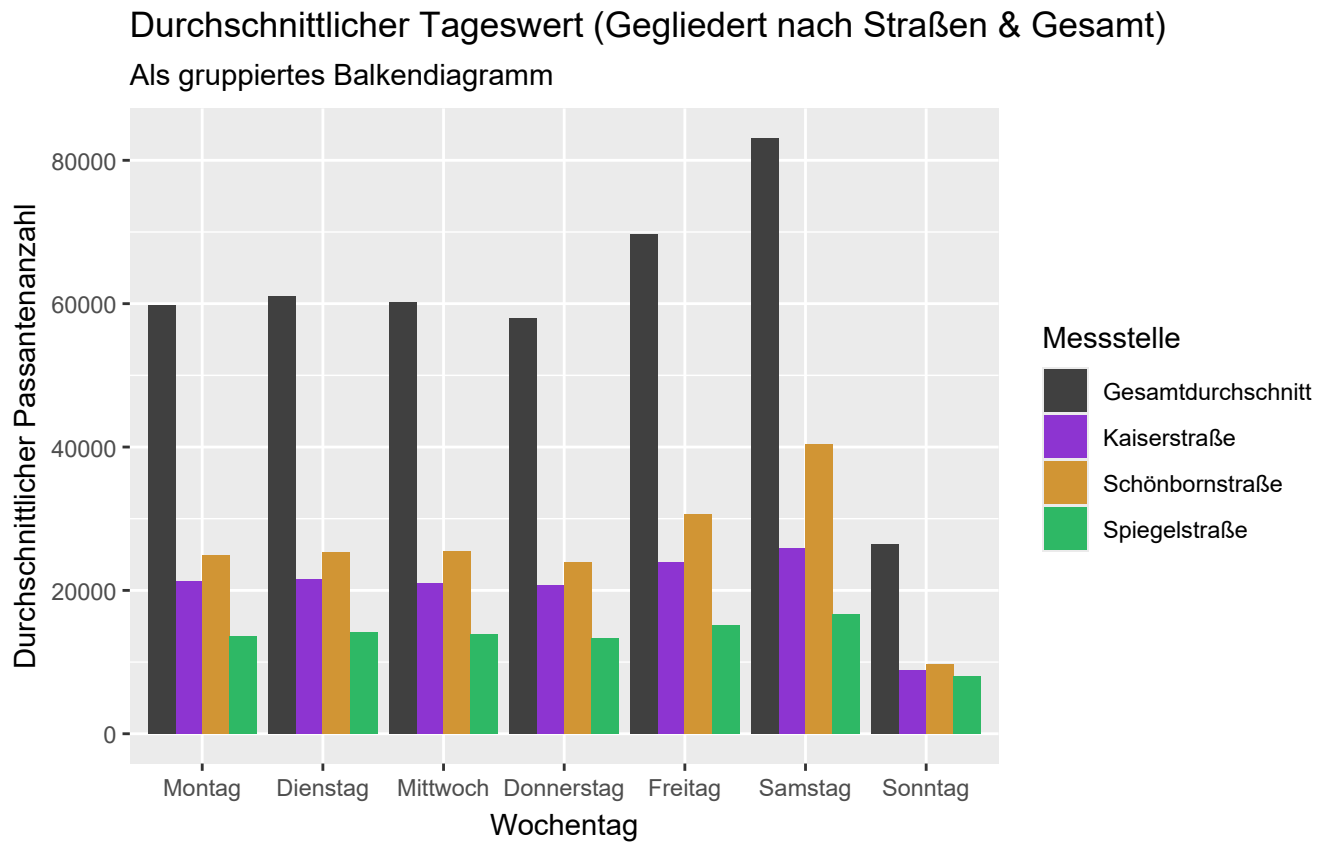
```
    wochentag = factor(wochentag, levels = tage_vektor)
   )%>%
   group_by(wochentag)%>%
   summarise(
      durchschnitt_wochentag = mean(Tagessumme_Gesamt)
   )%>%
   ungroup()

#' [Kombinierte Tabelle (Breites Format)]
#' @description
#' Kombiniert die gegliederten und die gesamten Mittelwerte in einer "breiten" Tabelle.
#' Diese Tabelle ist ideal für die tabellarische Darstellung in der Analyse.
#' @param durchschnittlicher_tageswert_location_plot [DataFrame] Der DF aus Block 3.
#' @param durchschnittlicher_tageswert_gesamt [DataFrame] Der DF aus Block 4.
#' @return
#' [DataFrame] 'durchschnittlicher_tageswert_location_wide'.
#'   Enthält 7 Zeilen (pro Wochentag) und Spalten für jeden Standort sowie die Gesamtsumme.
durchschnittlicher_tageswert_location_wide <- durchschnittlicher_tageswert_location_plot %>%
   pivot_wider(
      names_from = location_name,
      values_from = durchschnitt_wochentag_location
   ) %>%
   left_join(durchschnittlicher_tageswert_gesamt,by = "wochentag")
kable(durchschnittlicher_tageswert_location_wide, digits = 0, caption = "Durschnittlicher Tageswert pro
```

Table 11: Durschnittlicher Tageswert pro Straße

wochentag	Kaiserstraße	Schönbornstraße	Spiegelstraße	durchschnitt_wochentag
Montag	21221	24899	13606	59727
Dienstag	21485	25323	14172	60981
Mittwoch	20961	25391	13842	60194
Donnerstag	20706	23931	13377	58014
Freitag	23990	30563	15176	69729
Samstag	25893	40453	16735	83081
Sonntag	8818	9629	7974	26421

```
#Forme durchschnittlicher_tageswert_gesamt um, sodass man ihn mit durchschnittlicher_tageswert_gesamt v
durchschnittlicher_tageswert_gesamt_angepasst <- durchschnittlicher_tageswert_gesamt %>%
  rename(durchschnitt_wochentag_location = durchschnitt_wochentag) %>%
  mutate(location_name = "Gesamtdurchschnitt")

#Zusammenfügen der Beiden df's mit bind_rows
df_gesamt_long_plot <- bind_rows(
  durchschnittlicher_tageswert_location_plot,
  durchschnittlicher_tageswert_gesamt_angepasst
) %>%
  #Wochentage Richtig Sortieren
  mutate(
    wochentag = factor(wochentag, levels = tage_vektor)
  )

plot_aufgabe4<- ggplot(
```

```
  data = df_gesamt_long_plot,
  aes(x = wochentag, y = durchschnitt_wochentag_location, fill = location_name)
) +

  geom_col(position = "dodge") +

  # Füge deine 4 Custom-Farben hinzu
  scale_fill_manual(values = globale_farb_palette) +

  labs(
    title = "Durchschnittlicher Tageswert (Gegliedert nach Straßen & Gesamt)",
    subtitle = "Als gruppiertes Balkendiagramm",
    x = "Wochentag",
    y = "Durchschnittlicher Passantenanzahl",
    fill = "Messstelle"
  )
plot_aufgabe4
```



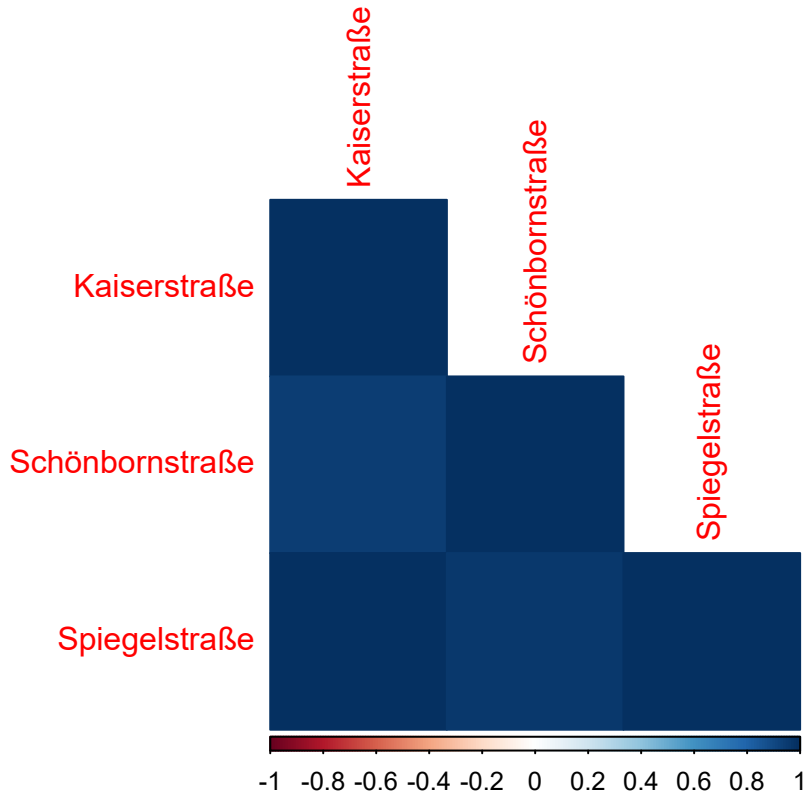

```
# kable(jahressumme, digits = 0,
#       caption = "Jahressumme und Mittelwerte der Passantenanzahl nach Messstelle")
```

Das Diagramm beschreibt die durchschnittliche Auslastung pro Tag, gegliedert nach dem Gesamtdurchschnitt und den einzelnen Locations. Auch in der reingezoomten Perspektive auf die Jahresdurchschnitte auf den

Wochentagen bleibt die Erkenntnis aus Aufgabe 2 bestehen. Die Rangliste der höchsten Auslastung beginnt erneut mit der Schönbornstraße, gefolgt von der Kaiser- und Spiegelstraße.

Die Grafik bestätigt den normalen Wochenrhythmus. Sichtbar wird das an den gleichbleibenden Durchschnitten von Montag bis Donnerstag, gefolgt von dem Anstieg am Freitag und Samstag und wenigen Passanten am Sonntag. Eine Begründung für diesen Wochenrhythmus ist das tägliche Geschäft in der Arbeitswoche, gefolgt vom Wochenende, an dem sich mehr Leute in der Innenstadt aufhalten. Da am Sonntag viele Geschäfte geschlossen haben, ist die Innenstadt auch nicht so stark besucht. Bemerkenswert ist hier die gleichmäßige Verteilung. Steigt der Gesamtdurchschnitt, so steigt auch der Durchschnitt jeder einzelnen Messstation. Das spricht für eine gewisse Korrelation zwischen den Auslastungen der einzelnen Messstationen.

```r
#' [1. Datenaufbereitung für Korrelation (Aufgabe 4)]
#' @description
#' [Erstellt die "breite" Tabelle (7 Zeilen x 3 Spalten),
#' die für die Korrelationsanalyse benötigt wird.]
#' @param durchschnittlicher_tageswert_location_plot [DataFrame] Dein "langer" DF
#' @return
#' [DataFrame] 'wide_data_task4'
wide_data_task4 <- durchschnittlicher_tageswert_location_plot %>%
  pivot_wider(
    names_from = location_name,
    values_from = durchschnitt_wochentag_location
  )



#' [2. Korrelationsmatrix (Der "Beweis")]
#' @description
#' [Berechnet die Korrelationskoeffizienten zwischen den
#' 7-Tage-Mustern der Standorte.]
#' @param wide_data_task4 [DataFrame] Der breite DF von oben
correlation_data_task4 <- wide_data_task4 %>%
  select(Kaiserstraße, Schönbornstraße, Spiegelstraße)

cor_matrix_task4 <- cor(correlation_data_task4)

#' [3. Correlogramm (Der "visuelle Beweis")]
#' @description
#' [Visualisiert die Korrelationsmatrix als Heatmap.]
corrplot(cor_matrix_task4, type = "lower", method = "color")
```



Über die Berechnung der Korrelation erkennt man eine starke korrelation. Ein Wert >0.5 gilt als starker Zusammenhang. Aus der Korrelationsmatrix lässt sich eine Korrelation >0.5 für alle Straßen ablesen. Die Kaiser und Spiegelstraße hängen dabei enger zusammen als die jeweilige Straße mit der Schönbornstraße.

Aufgabe 5

```r
#'[Stündl. Mittelwert (Gegliedert)]
#' @description
#' Berechnet den durchschnittlichen Passantenwert für jede Stunde (0-23)
#' und für jede einzelne Messstelle ('location_name').
#' @param passanten [DataFrame] Das Roh-DataFrame 'passanten'.
#'   Benötigt die Spalten 'stunde', 'location_name' und 'passanten'.
#' @return
#' [DataFrame] 'passantenanzahl_stunden_plot' (langes Format).
#'   Enthält 72 Zeilen (24h * 3 Messstellen) mit dem stündlichen Mittelwert.
passantenanzahl_stunden_plot <- passanten %>%
   group_by(stunde, location_name) %>%
   summarise(
      passantenanzahl = mean(passanten)
      )%>%
   ungroup()
```

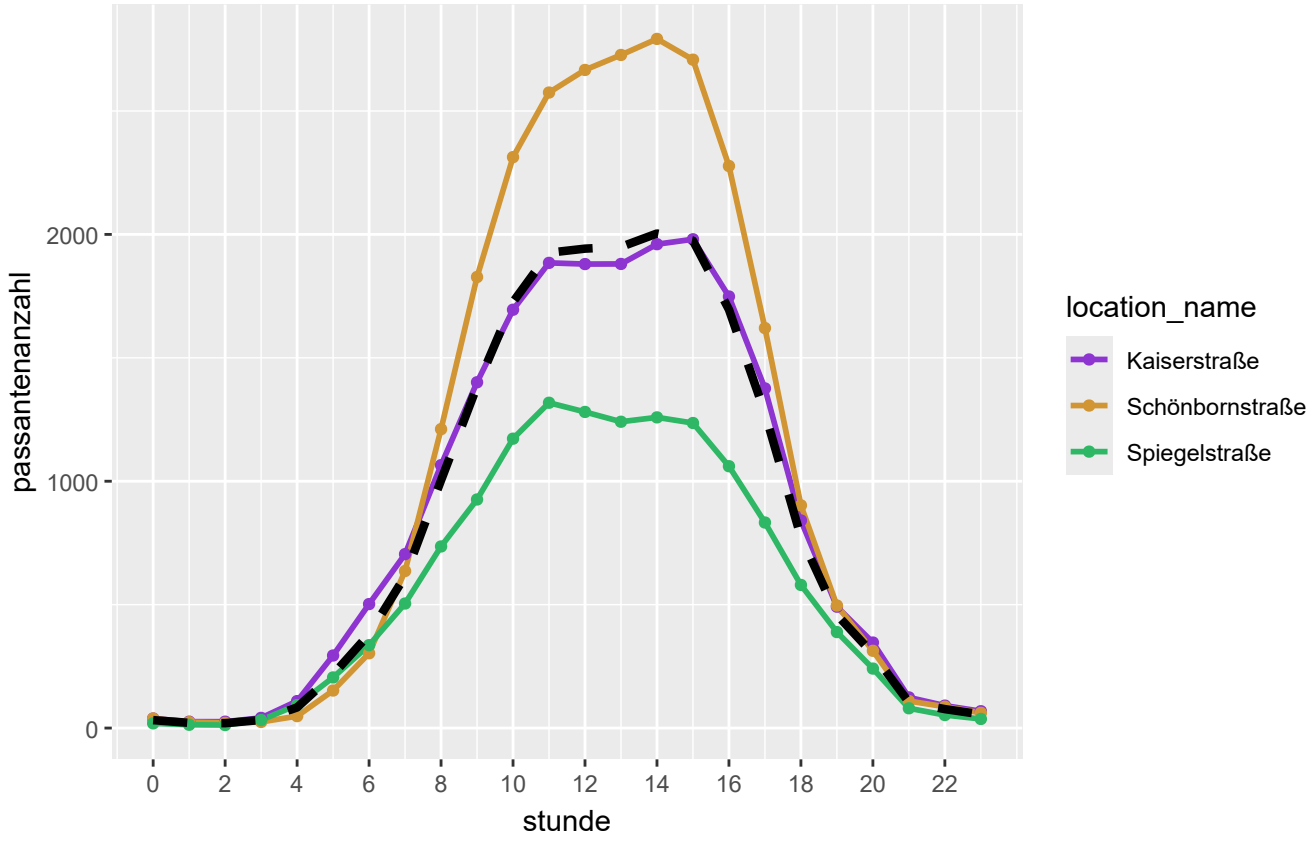
```r
#' [Stündl. Mittelwert (Summiert)]
#' @description
#' Berechnet den durchschnittlichen Passantenwert für jede Stunde (0-23)
#' über ALLE Messstellen hinweg (Gesamtdurchschnitt).
#' @param passanten [DataFrame] Das Roh-DataFrame 'passanten'.
#'   Benötigt die Spalten 'stunde' und 'passanten'.
#' @return
#' [DataFrame] 'avg_stunde_gesamt' (langes Format).
#'   Enthält 24 Zeilen (eine pro Stunde) mit dem Gesamt-Stundenmittelwert.
avg_stunde_gesamt <- passanten %>%
   group_by(stunde) %>%
   summarise(
      durchschnitt_passanten_gesamt = mean(passanten, na.rm = TRUE)
   )%>%
   ungroup()


#' [Stündl. Mittelwert (Breites Format)]
#' @description
#' Wandelt das "lange" Plot-Format in ein "breites" Tabellen-Format um.
#' Jede Messstelle wird zu einer eigenen Spalte.
#' @param passantenanzahl_stunden_plot [DataFrame] Das "lange" Ergebnis aus Block 1.
#' @return
#' [DataFrame] 'passantenanzahl_stunden_wide'.
#'   Enthält 24 Zeilen (eine pro Stunde) und Spalten für jede Messstelle.
passantenanzahl_stunden_wide <- passantenanzahl_stunden_plot %>%
   pivot_wider(
      names_from = location_name,
      values_from = passantenanzahl
      )%>%
   left_join(avg_stunde_gesamt, by = "stunde")
kable(passantenanzahl_stunden_wide, digits = 0, caption = "Durchschnittlicher Passantenwert jede Stunde
```

Table 12: Durchschnittlicher Passantenwert jede Stunde (0-23)

stunde	Kaiserstraße	Schönbornstraße	Spiegelstraße	durchschnitt_passanten_gesamt
0	38	39	19	32
1	26	24	13	21
2	26	21	12	20
3	41	25	32	33
4	109	48	93	84
5	294	152	205	217
6	502	303	336	380
7	705	636	505	615
8	1065	1211	736	1004
9	1401	1827	926	1385
10	1695	2313	1172	1726
11	1885	2575	1318	1926
12	1880	2666	1281	1942
13	1880	2727	1241	1950
14	1960	2792	1259	2004
15	1980	2708	1236	1975

stunde	Kaiserstraße	Schönbornstraße	Spiegelstraße	durchschnitt_passanten_gesamt
16	1749	2277	1061	1696
17	1376	1620	833	1276
18	841	902	580	774
19	492	496	389	459
20	346	313	241	301
21	124	109	80	105
22	91	87	53	77
23	68	62	37	56

```
ggplot() +
   # 1. Plot
  geom_line(data = passantenanzahl_stunden_plot,
            aes(x = stunde, y = passantenanzahl,
            color = location_name),
            linewidth = 1) +
  geom_point(data = passantenanzahl_stunden_plot,
             aes(x = stunde, y = passantenanzahl,
            color = location_name)) +

   # 2. Plot
  geom_line(
    data = avg_stunde_gesamt,
    aes(x = stunde, y = durchschnitt_passanten_gesamt, colour = NULL, group = 1),
    color = "black",
    linewidth = 1.5,
    linetype = "dashed"
  ) +

   scale_color_manual(values = globale_farb_palette)+

   scale_x_continuous(breaks = seq(0, 23, by = 2))
```




```
  labs(title = "Durchschnittlicher Tagesverlauf (Gegliedert nach Straße & Gesamt)",
       subtitle = "Als Liniendiagramm",
       x = "Uhrzeit (Stunde des Tages)",
       y = "Durchschnittliche Passanten pro Stunde",
       color = "Standort")
```

```
## <ggplot2::labels> List of 5
##  $ x       : chr "Uhrzeit (Stunde des Tages)"
##  $ y       : chr "Durchschnittliche Passanten pro Stunde"
##  $ colour  : chr "Standort"
##  $ title   : chr "Durchschnittlicher Tagesverlauf (Gegliedert nach Straße & Gesamt)"
##  $ subtitle: chr "Als Liniendiagramm"
```

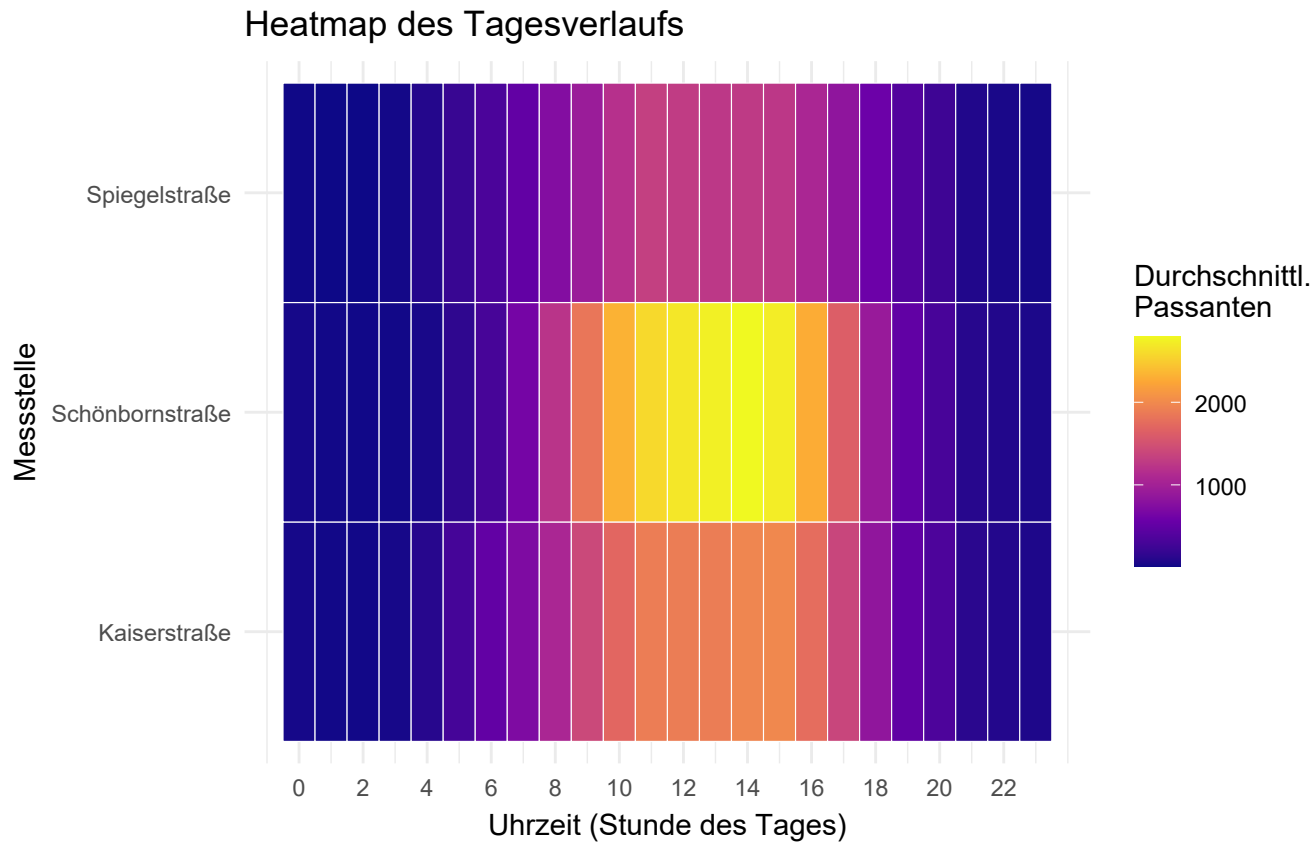
Das auf Grundlage der Tabelle erstellte Liniendiagramm zeigt den Durchschnitt der Passanten pro Stunde, aufgeteilt nach Straßen. Die gestrichelte Linie stellt dabei den Durchschnitt der Passanten pro Stunde insgesamt dar.

Die bereits festgestellte Rangliste nach Auslastung bestätigt sich auch in dieser Ansicht.

Morgens pendeln mehr Leute über die Kaiserstraße. Das ist im Bereich zwischen 4 und 7 Uhr auf dem Graphen abzulesen. Die Kaiserstraße liegt dabei deutlich über dem Gesamtdurchschnitt, der Schönborn- und Spiegelstraße. Eine Begründung ist die Lage der Kaiserstraße. Sie ist die direkte Verbindung vom Hauptbahnhof in die Innenstadt. Da die Anzahl der Werktage die Anzahl der Feiertage + gesetzlich vorgegebener Urlaubstage übertrifft, kann man davon ausgehen, dass der Durchschnitt stark durch den morgendlichen Verkehr zum Hauptbahnhof beeinflusst ist.

Die Kaiserstraße scheint demnach den Gesamtdurchschnitt sehr genau abzubilden. Ablesbar ist dies aus dem Diagramm, da die Kaiserstraße und der Gesamtdurchschnitt einen ähnlichen Verlauf haben.

```
#' [Aufgabe 5: Heatmap des Tagesverlaufs]
#' @description
#' [Zeigt die Passantenanzahl als farbige Kacheln.
#' Ideal, um "Hot Spots" (Peaks) schnell zu identifizieren.]
library(ggplot2)
# library(viridis) # Für eine farbenblinde-sichere Palette (optional)

ggplot(passantenanzahl_stunden_plot,
       aes(x = stunde, y = location_name, fill = passantenanzahl)) +

  geom_tile(color = "white") + # 'color="white"' fügt dünne weiße Ränder hinzu

  # Wir brauchen eine sequentielle Farbskala (kontinuierlich)
  scale_fill_viridis_c(option = "C") + # "viridis_c" ist eine gute Standard-Palette
  # Oder: scale_fill_gradient(low = "lightblue", high = "darkblue")

  scale_x_continuous(breaks = seq(0, 23, by = 2)) +

  labs(
    title = "Heatmap des Tagesverlaufs",
    x = "Uhrzeit (Stunde des Tages)",
    y = "Messstelle",
    fill = "Durchschnittl.
Passanten"
  ) +
  theme_minimal()
```



Die Heatmap dient als effektive, komplementäre Visualisierung der stündlichen Passantenfrequenzen. Durch die Darstellung der Passantenanzahl mittels Farbintensität ermöglicht sie eine schnelle Identifizierung von "Hot Spots" und bestätigt die bereits im Liniendiagramm gewonnenen Erkenntnisse. Dies zeigt sich beispielsweise in der deutlich intensiveren Färbung der Kaiserstraße in den frühen Morgenstunden sowie den hellsten Bereichen für alle Standorte am Nachmittag, was die zuvor beschriebenen Muster visuell unterstreicht. ## Vorhersagemodell

Durch die in Aufgabe 4 beschreiben Korrelation zwischen den Straßen in der durchschnittlichen Tageswert stellt sich nun die frage ob man die auslastung einer Straße durch die werte der beiden anderen vorhersagen kann. Aus Aufgabe 5 Wissen wir, das die Kaiserstaße sehr ähnlich zur durchschnittlichen Gesamtauslastung ist. Daher versucht man die Kaiserstraße durch die Spiegel und Schönborstraße vorherzusagen.

```
#' [3. Multiple Lineare Regression]
#' @description
#' [Modelliert einen Standort (Kaiserstraße) als Funktion
#' der BEIDEN anderen Standorte, um deren gemeinsamen
#' und individuellen Einfluss zu quantifizieren.]
#' @param passantenanzahl_stunden_wide [DataFrame] Dein "breiter" DF.
#'   Benötigt die Spalten 'Kaiserstraße', 'Schönbornstraße', 'Spiegelstraße'.
#' @return
#' [lm-Objekt] 'model' enthält das trainierte Regressionsmodell.
#' [Text-Output] 'summary(model)' zeigt die Ergebnisse (wie in deinem Screenshot).

correlation_data <- passantenanzahl_stunden_wide %>%
  select(Kaiserstraße, Schönbornstraße, Spiegelstraße)
```

```r
model <- lm(Kaiserstraße ~ Schönbornstraße + Spiegelstraße, data = correlation_data)
zusammenfassung <- summary(model)
zusammenfassung
```

```
##
## Call:
## lm(formula = Kaiserstraße ~ Schönbornstraße + Spiegelstraße,
##     data = correlation_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -85.651 -21.882  -7.153  11.705 123.797
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      23.5321    19.6732   1.196  0.24497
## Schönbornstraße   0.2189     0.0689   3.177  0.00454 **
## Spiegelstraße     1.0495     0.1509   6.953 7.22e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 53.35 on 21 degrees of freedom
## Multiple R-squared:  0.9956, Adjusted R-squared:  0.9952
## F-statistic:  2382 on 2 and 21 DF,  p-value: < 2.2e-16
```

```r
#kable(zusammenfassung, caption= "Zusammenfassung der Regressionsberechnung.")
```

Die erstellte lineare Regression liefert ein Modell, um die Passantenfrequenz der Kaiserstraße auf Basis der Frequenzen der Schönbornstraße und Spiegelstraße vorherzusagen. Die Zusammenfassung des Modells (`summary(model)`) gibt Aufschluss über die Güte und die Zusammenhänge:

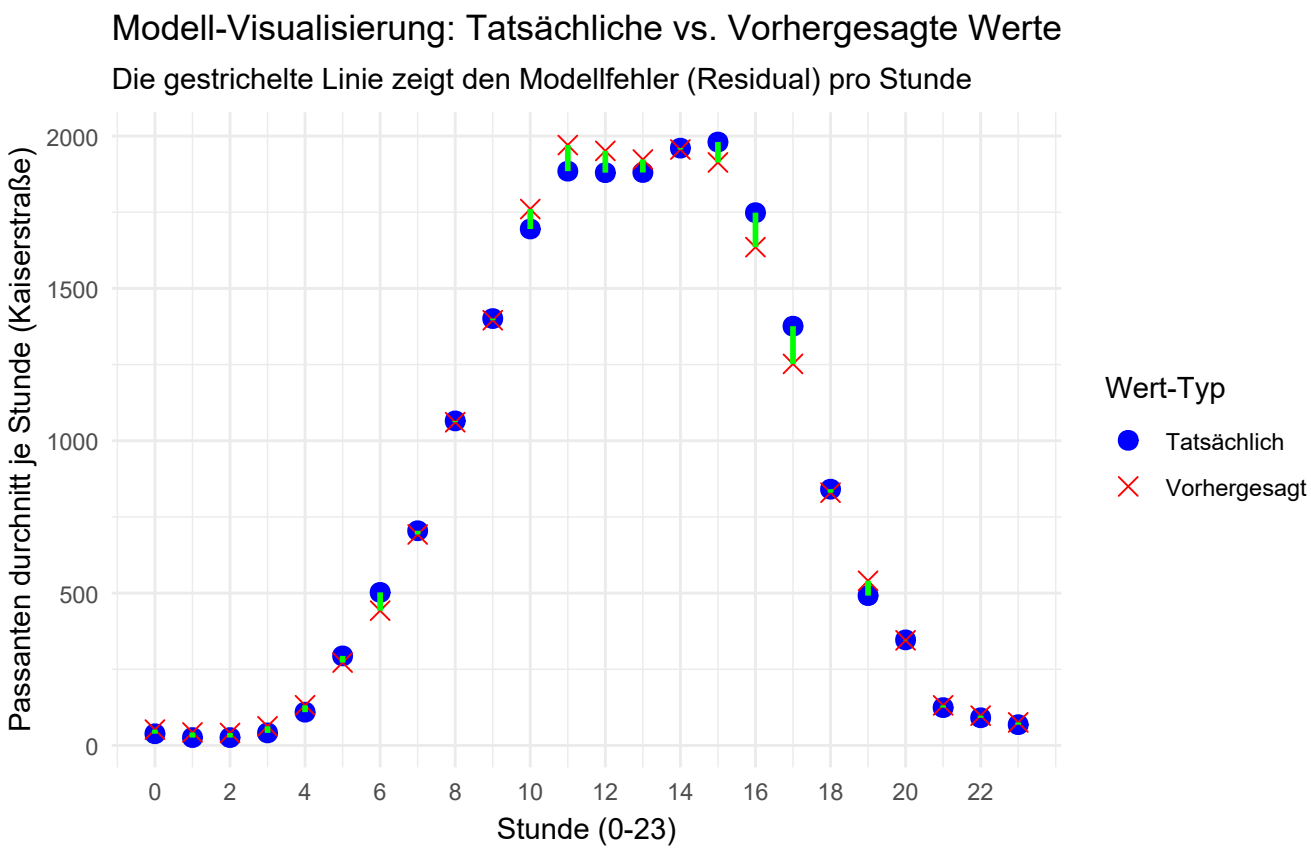
Statistische Signifikanz p-value: Die erste und wichtigste Zeile welche man für eine Auswertung betrachtet ist die unterste: F-statistic: 2382 on 2 and 21 DF, p-value: $<$ 2.2e-16. Das p-value ist hier mit 2.2e-16 angegeben. Dieser sehr kleinen p-Werte deuten darauf hin, dass der Einfluss beider Straßen auf die Kaiserstraße hoch signifikant ist. Es ist also äußerst unwahrscheinlich, dass dieser starke Zusammenhang nur ein Zufallsprodukt in den Daten ist.

Modellgüte (`Adjusted R-squared`): Das adjustierte R-Quadrat (`Adjusted R-squared`) ist das wichtigste Maß für die Güte des Modells. Der Wert von 0.9956 bedeuten, dass das Modell **99 %** der Varianz (der Schwankungen) in den Passantenzahlen der Kaiserstraße allein durch die Daten der beiden anderen Straßen erklären kann. Ein derart hoher Wert zeigt eine exzellente Passform und bestätigt, dass die Passantenströme der drei Straßen systemisch stark miteinander verknüpft sind. Die Güte des berechneten modell ist hier sehr hoch.

Interpretation der Koeffizienten (`Coefficients`): Die Koeffizienten zeigen, wie stark die beiden anderen Straßen die Kaiserstraße beeinflussen. Der positiver Koeffizient von 0.21 bei der Schönbornsprache liefert das modell folgende berechnung: ein Passatn mehr in der Schönbornstraße fürhr zu 0.2 Passanten mehr in der Kaiserstraße.

```r
#' [1. "Intuitiver" Modell-Plot (Dumbbell-Plot)]
#' @description
#' [Visualisiert die Güte des Modells, indem TATSÄCHLICHE Werte
#' (aus der Tabelle) direkt den VORHERGESAGTEN Werten (aus dem Modell)
#' für jede Stunde gegenübergestellt werden.]
```

```r
#' @param model [lm-Objekt] Dein Regressionsmodell ('model')
#' @param passantenanzahl_stunden_wide [DataFrame] Dein "breiter" DF,
#'   der 'stunde' und die Prädiktoren (Schönbornstraße etc.) enthält.
model_data_augmented <- augment(model, newdata = passantenanzahl_stunden_wide)

ggplot(model_data_augmented, aes(x = stunde)) +

  geom_point(
    aes(y = Kaiserstraße, color = "Tatsächlich"), # y = Deine echte Spalte
    size = 3
  ) +

  geom_point(
    aes(y = .fitted, color = "Vorhergesagt"), # y = Die Modell-Prognose
    size = 3,
    shape = 4 # 'shape = 4' ist ein 'X'
  ) +

  geom_segment(
    aes(xend = stunde, y = Kaiserstraße, yend = .fitted),
    color = "green",
    linewidth = 1
  ) +

  # --- 4. Manuelle Farb- und Legendensteuerung ---
  scale_color_manual(
    name = "Wert-Typ",
    values = c("Tatsächlich" = "blue", "Vorhergesagt" = "red")
  ) +

  scale_x_continuous(breaks = seq(0, 23, by = 2)) +

  labs(
    title = "Modell-Visualisierung: Tatsächliche vs. Vorhergesagte Werte",
    subtitle = "Die gestrichelte Linie zeigt den Modellfehler (Residual) pro Stunde",
    x = "Stunde (0-23)",
    y = "Passanten durchnitt je Stunde (Kaiserstraße)"
  ) +
  theme_minimal()
```



Das Diagramm stellt die tatsächlichen Werte (blaue Punkte) den vom Modell vorhergesagten Werten (rote Kreuze) für jede Stunde gegenüber. - Die grünen Verbindungslinien visualisieren den Vorhersagefehler (die Residuen) für jede Stunde. - Man erkennt, dass die Linien insgesamt sehr kurz sind, was die hohe Genauigkeit des Modells visuell bestätigt. Die Vorhersagen liegen sehr nah an den echten Werten.

Eine Abweichungen bei der das Modell weniger vorhersagt (Tatsächlicher Wert liegt über der Vorhersage) ist um 6 Uhr morgens und im Zeitraum 15-19 Uhr. Diese Abweichung könnte durch das erhöhte Pendleraufkommen am Hauptbahnhof zu beginn und ende eines Arbeitstages entstehen. Aufgrund des unmittelbaren Anschluss der Kaiserstraße an den Hauptbahnhof ist diese spezifische Event nicht auf den anderen straßen anzutreffen. So können sie die Auslasung nicht vorhersagen und Residuen entstehen.

Im Zeitraum von 10 bis 12 Uhr sagt das Modell einen höheren wert voraus. Das bedeutet, das auf annhame der Auslasutng Schönborn und Spiegelstraße die Kaiserstraße voller sien müsst. Aus dem Liniendiagramm der Aufgabe 5 ist zu erkenne, das dei Spiegelstraße um 10Uhr einen “Mittagspeak” hat. Die Kaiserstaße hingehen Stagniert um 10 Uhr befor sie gegen 15 Uhr ihren Peak erreicht. Der Peak der Spiegelstraße könnte hier die erklärung für das “überschätzen” des modell sein.

Zusammenfassend lässt sich sagen, dass das lineare Regressionsmodell hervorragend geeignet ist, die Passantenfrequenz der Kaiserstraße vorherzusagen. Dies untermauert die bereits in der Korrelationsanalyse gezeigte starke systemische Verbindung der Passantenströme in diesem Bereich der Innenstadt.

Fazit

Die explorative Datenanalyse der Würzburger Passantenzahlen für das Jahr 2024 hat erfolgreich grundlegende Muster und Zusammenhänge aufgedeckt. Als Antwort auf die Leitfragen wurde die Schönbornstraße eindeutig als belebtester Standort identifiziert, deren hohe Frequenz auf ihre zentrale Lage als Einkaufsstraße

und die Nähe zu Sehenswürdigkeiten zurückzuführen ist. Die Analyse bestätigte klare Wochen- und Tagesrhythmen, die dem typischen Stadtleben mit Pendler- und Einkaufsverkehr entsprechen. Zudem konnten saisonale Ereignisse wie die Weinparade und der Weihnachtsmarkt als deutliche Ausreißer in den Daten nachgewiesen werden.

Die starke Korrelation und das erfolgreiche Regressionsmodell untermauern, dass die drei Messstationen als vernetztes System agieren. Eine Limitation der Arbeit ist die Nichtberücksichtigung der Wetterdaten. Zukünftige Analysen könnten daher untersuchen, welchen quantitativen Einfluss Temperatur und Wetter auf die Passantenfrequenz haben, um die Vorhersagemodelle weiter zu verfeinern und ein noch umfassenderes Bild der urbanen Mobilität zu zeichnen.

Literatur

[1] "Methodik - hystreet.com."

[2] W. erleben, "Stadtfest Würzburg 2025," *Würzburg erleben.* Sep-2025.

[3] weinparadenwirt, "Das Fest," *Weinparade Würzburg.*

[4] K. Kraus, "Das ist 2024 auf dem Würzburger Weihnachtsmarkt geboten," *Würzburg erleben.* Nov-2024.