Weiner_Passanten01

```
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 begin unserer Explorativen Datenanlyse (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 refferenz als "passanten_raw". Dabei wurde der Dataframe passanten bereits in ein tibbel Format überführt (eine 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 sinvolle, descriptive oder weiterführend Explorative Datenanalyse durchzuführen sollten die Daten zuerst bereinigt werden. Als nächstes benennen wir die Spaltennamen in Snacke-Case um.

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

Anschließend wird die Anzahl an lehren fledern 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
jahr	0
monat	0
woche	0
tag	0
stunde	0
wochentag	0
tag_im_jahr	0

Es lässt sich entnehmen, das Wetter und temperatur jewails 112 NA einträge haben. Dies kann diverse gründe haben ist aber nicht relevant im moment. 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 der Daten

Der Datensatz enthält daten zu einer Passantenzählung in der Nürnberger 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 jewails eine 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.

```
kable(skim(passanten), digits = 0, caption = "Skim überblick der Daten pro Spalte")
```

Table 2: Skim überblick der Daten pro Spalte

skim_type	skim_variable	n_missing	complete_rate	POSIXct.min	POSIXct.max	POSIXct.median	POSIXct.n_unique	character.min	character.max	character.empty	character.n_unique	character.whitespace	numeric.mean	numeric.sd	numeric.p0	numeric.p25	numeric.p50	numeric.p75	numeric.p100	numeric.hist
POSIXct	zeitstempel	1	1	2024-01-01	2024-12-31 22:00:00	2024-06-30 16:00:00	8694	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	
character	wetter	1	12	NA	NA	NA	NA	3	19	0	9	0	NA	NA	NA	NA	NA	NA	NA	
character	location_name	0	1	NA	NA	NA	NA	12	15	0	3	0	NA	NA	NA	NA	NA	NA	NA	
character	geo_punkt	0	1	NA	NA	NA	NA	36	37	0	3	0	NA	NA	NA	NA	NA	NA	NA	
character	wochentag	0	1	NA	NA	NA	NA	6	10	0	7	0	NA	NA	NA	NA	NA	NA	NA	
numeric	temperatur	1	12	NA	NA	NA	NA	NA	NA	NA	NA	NA	11	8	-11	6	11	17	34	
numeric	passanten	0	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	840	997	0	53	467	1341	8075	
numeric	jahr	0	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	2024	0	2024	2024	2024	2024	2024	
numeric	monat	0	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	7	3	1	3	6	10	12	
numeric	woche	0	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	26	15	1	13	26	40	52	
numeric	tag	0	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	16	9	1	8	16	23	31	
numeric	stunde	0	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	11	7	0	5	11	17	23	
numeric	tag_im_jahr	0	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	183	106	1	91	182	276	366	

Der Datensatz besteht aus 26015 Zeile und 14 Spalten von denen 8 numerische, eine POSIXct und 4 Zeichenketten als Datentypen enthalten. Die Vollständigkeit ist bei allen auser 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 das es sich durch die gesamtzahl von 366 Tagen um ein Schaltjahr handeln muss.

Erhebung

Die Daten stamen von zählstationen an verschiedenen Punken aus der Nürnberger Innenstadt. Mit hilfe von Laserschranken zählen diese die Anzahl der Passanten welche gerade die Messtation Queren. Durch das Messen mit meheren Laserschranken pro Messtation 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 3: 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 4: 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 angegebenen Koordinaten und Messtellen anschaut sieht man, dass es jewails drei unterschiedliche werte in diesen Spalten gibt. Entscheident ist nun die anzahl der dopplungen dieser werte. Bei genauerer betrachtung sieht man, das die jewailigen dopplungen der menge: location und Geo Punkt gleich sind. Damit kann man davon ausgehen, das 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:

Die Koordinaten Stimmen mit der Jewailigen Straße und dem Name der Messtation überein. Man kann sehen, das die Stationen direkt in der Innenstad plaziert sind. Dabei ist jede Station in der Nähe einer Sehenswürdigkeit bzw. Öffentlichen Gebäude.

Die Messtation Schönbornstraße befindet sich nah an der Marienkapelle, die Messtation Spiegelstraße auf dem Weg zum Hofgarten und die Messtation Kaiserstraße ist in der nähe des Hauptbahnhof. Alle diese Straßen Hauptverkehrsstraßen auf denen mit vielen Passanten zu rechnen ist. Das Dreiecksmuster welche die Stationen Aufspannen bilden 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 Innenstad welche erkennbar sind? - Was ist die beliebteste Region in der Innenstad? - Woher kommt die beliebtheit? (Attraktionen, Wichtiger Teil des Altags, Veranstaltungsorte, …)

Aufgabe 2

```
#--- 1. Analyse aggregiert ---
# Funktion guppiert nach location_name und berechnet anschließend: passanten jahres Summe, erfasste Tag
```

Figure 1: Markierte Standorte der Messstationen in der Nürnberger Innenstadt

```
# 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 5: Jahressumme und Mittelwerte der Passantenanzahl nach Messstelle

location_name	passanten_jahr_summe	anzahl_tage_erfasst	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 6: Jahressumme der Passantenanzahl

passanten_jahr_summe	anzahl_tage_erfasst	durchschnitt_pro_monat	durchschnitt_pro_woche	durchschnitt_pro_tag	durchschnitt_pro_stunde
21864318	366	1822026	418170	59739	2489

Bereits durch die Einfache berechung 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 insgesammt gezählt wurden stellt die Schönbornstraße somit fast die Hälfte des gesamten Passanten verkehr. Den Daten nach ordnet sich die Kaiserstraße als zweit und die Spiegelstraße als am wenigsten ausgelastete Sraße an. Ebenfalls auffällig ist, dass die Schönbornstraße eine fast doppelt so hohe auslastung wie die Spiegelstraße hat. Somit ist hier eine ungeliche aufteilung der gesamtauslastung festzustellen.

Einfügen Interpretation der Lage und der Verbindungen die die Staßen aufspannen

Aufgabe 3

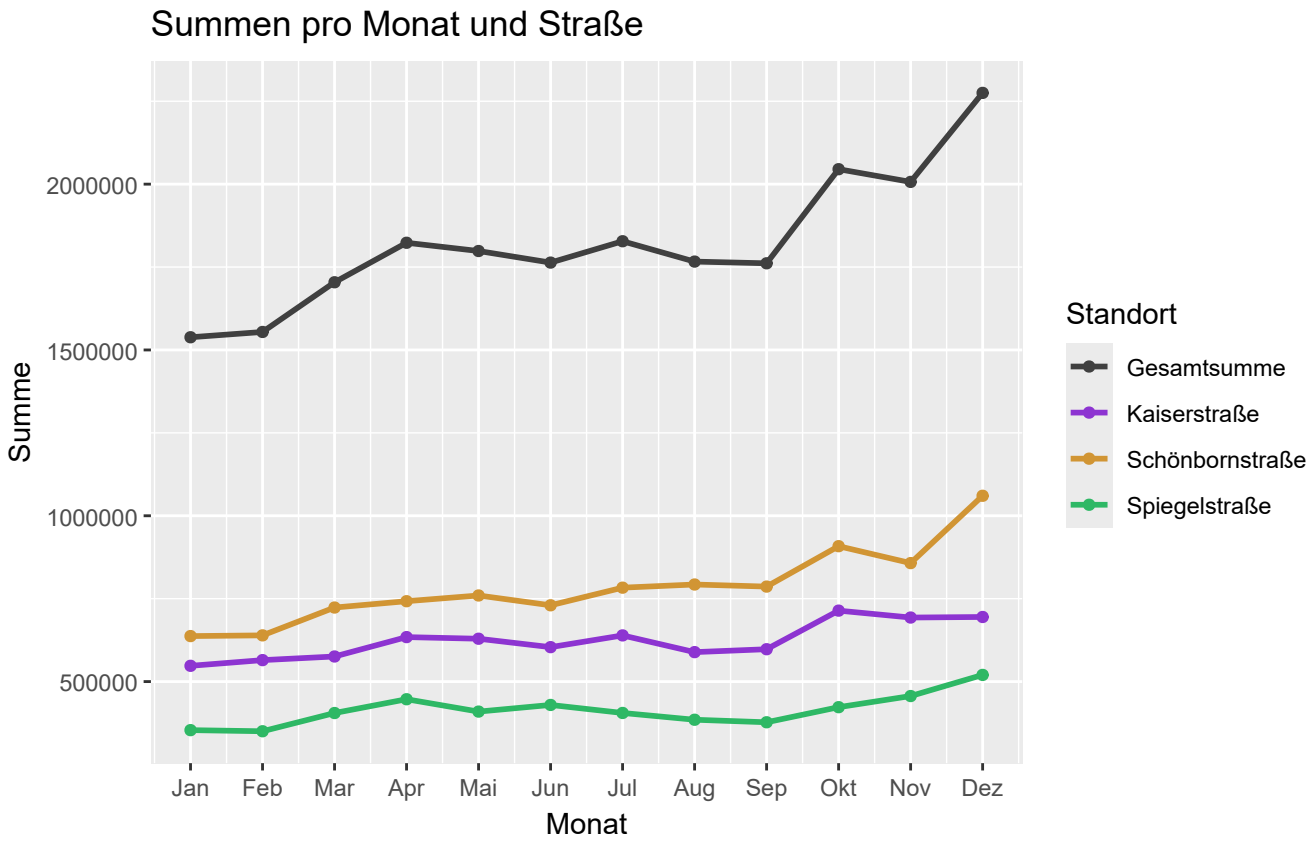
```r
#' [Monatssumme über das Jahr]
#' @description
#' [Die Funktion Berechnet die Monatsumme der Passaten, gruppiert nach der location_name pro jahr]
#'
#' @param passanten df
#'
#' @return
#' [Rückgabewert ist ein der df sume_monat. Das wide Format zeigt die Daten sortiert nach monat und Loc
#'
summe_monat <- 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 pro Monat und Sraße")
```

Table 7: Summe aller Passanten pro Monat und Sraße

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

```r
#' [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")
```

```r
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 pro Monat und Straße",
       x = "Monat",
       y = "Summe",
       color = "Standort")
plot_aufgabe3
```



Das Erstellte Diagramm bestätigt das aus Aufgabe2 bereits beschriebene Muster. Die Auslastung der Schönbornstraße ist konstant am höchsten gefolgt von der Kaiserstraße und der Spiegelstraße. Dise Reihenfolge ist über das Gesamte Jahr erkennbar was darauf schließen lässt das diese Auslastung nicht durch events oder ausergewöhnliche ereignisse, sondern eher den Alltag in der Nürnberger Innenstadt darstellt. In der Gesamtsummen 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 Innenstad erhöht zu habe. Das Liniendiagramm zeigt entsprechende ausreißer im September in der Gesamtsumme aber auch an den Jewailigen Stationen. Grund für die erhöhung von September auf Oktober sind mehrere Veranstaltungen. Zum einen das “Würzburger Stadfest” vom 14-15.Sep 2024 und die Würzburger Weinparade vom 29 August

bis 8. September.Daraus ergibt sich er Anstig in der Gesamtsumme von ca 500.000 Passanten. Speziell auf der Kaiserstraße und Scönbornstraße zeigt das Liniendiagrammist ein markanter anstig der Gesamtzahl pro Messtation. Hierzu betrachten wir den Ort an dem das Stadfest stattfand. Zum Stadfest ist zu sagen, das es eine dezentrale veranstaltung ist bei der "sich fast die gesamte Innenstadt in eine Vergnügungsmeile mit mehreren Bühnen, einem vielfältigen Programm" vgl. [2]. Mit dem Fehlenden Anstieg der Passantenzahl trifft die beschreibung "fast die gesamte Innenstad" sehr genau auf die tatsächlichen Daten zu. Somit trägt das Stadfest zur erhöhung der Passantenzahl bei ist aber nicht der grund für den erkennbaren anstieg auf der Kaiser und Schönborstraße. Die Weinparade hingegen fand für einen Längeren Zeitraum exklusiv auf und um den Unteren markplatz bei der Marienkapelle, in direkter nachberschaft zur Schönbornstraße statt [3]. Da die Kaiserstraße eine direkte verbindung von Haupbahnhof und Marienkapelle darstellt lässt sich so der explizite ansieg der Passanten für diesen Zeitraum erklären.

Im Dezember gab es weitere veranstaltungen in der Innenstad. Hiebei zeigt das Liniendiagramm eine Anstieg in der Gesamtsumme aber auch auf der Schönborn und Spiegelstraße.Die Erklärung hierführ ist der Weihnachtsmarkt vom 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' )


#' [durschnitt tageswert]
#' @description
#' [die Funktion gruppiert nach wochentag und location und berechnet dann den durchschnitt. gleichzeiti
#'
#' @param tagessumme
#' @return
#' [df durchnittlicher_tageswert der Passanten pro Messtation und Wochentag in long format]
durchschnittlicher_tageswert_location_plot <- tagessumme %>%
   mutate(
    wochentag = factor(wochentag, levels = tage_vektor)
   )%>%
   group_by(wochentag, location_name)%>%
   summarise(
      durchschnitt_wochentag_location = mean(Tagessumme)
```

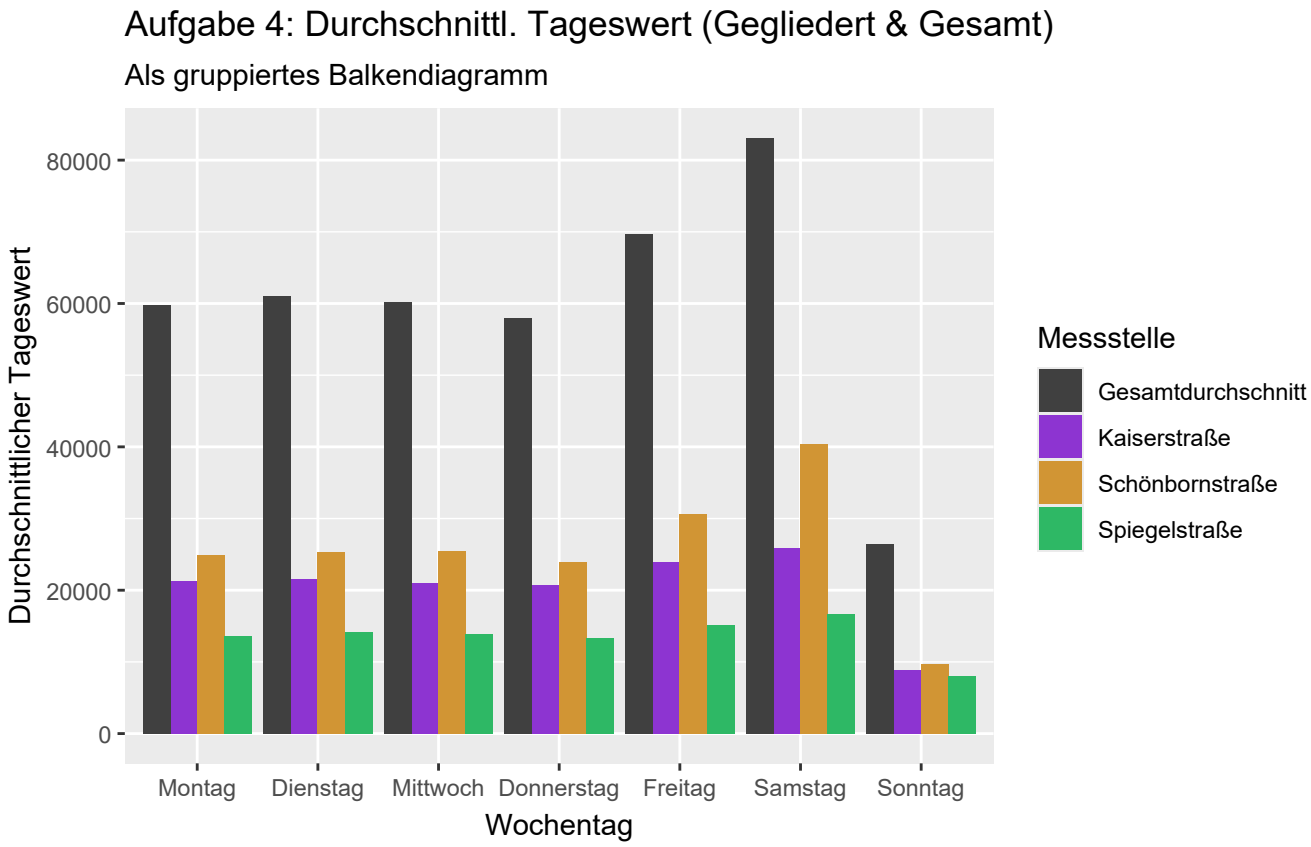
```
   )%>%
   ungroup()


#' [durschnitt tageswert]
#' @description
#' [die Funktion gruppiert nach wochentag und location und berechnet dann den durchschnitt. gleichzeiti
#' @param tagessumme
#' @return
#' [df durchnittlicher_tageswert der Passanten pro Messtation und Wochentag in long format]
durchschnittlicher_tageswert_gesamt <- gesamttagessumme %>%
   mutate(
    wochentag = factor(wochentag, levels = tage_vektor)
   )%>%
   group_by(wochentag)%>%
   summarise(
      durchschnitt_wochentag = mean(Tagessumme_Gesamt)
   )%>%
   ungroup()

#' [df in Tabellenformat ]
#' @description
#' [Formatiert einen df in eine Aussagekräftige Tabelle]
#'
#' @param durschnittlicher_tageswert_plot]
#' @return
#' []
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 8: Durschnittlicher Tageswert pro Location

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
```

```r
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 = "Aufgabe 4: Durchschnittl. Tageswert (Gegliedert & Gesamt)",
    subtitle = "Als gruppiertes Balkendiagramm",
    x = "Wochentag",
    y = "Durchschnittlicher Tageswert",
    fill = "Messstelle"
  )
plot_aufgabe4
```



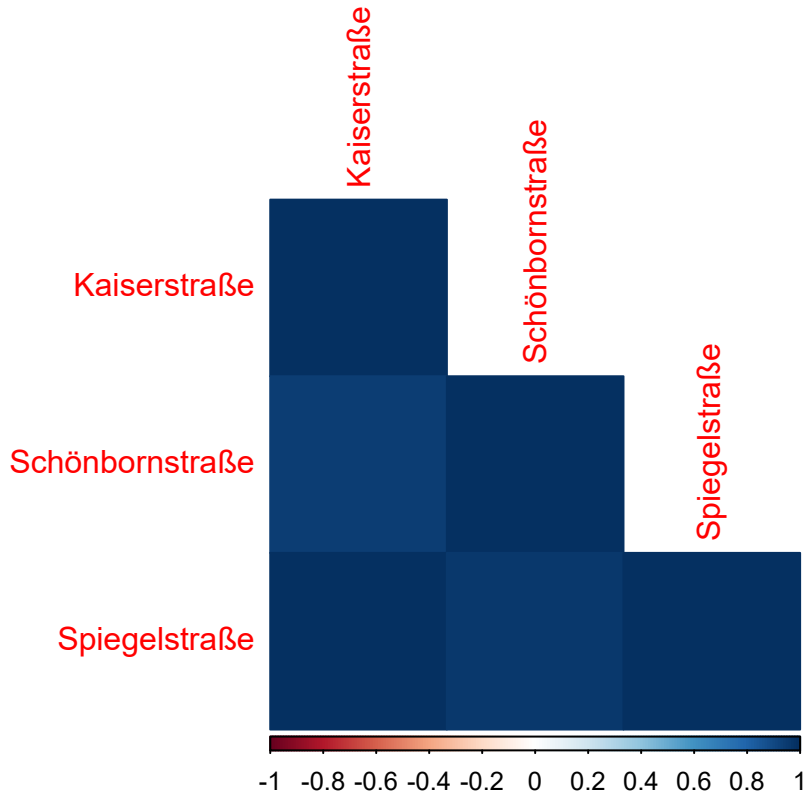

```
# kable(jahressumme, digits = 0,
#       caption = "Jahressumme und Mittelwerte der Passantenanzahl nach Messstelle")
```

Das Diagramm beschreibt die Durschnittliche Auslastung pro Tag, gegliedert nach dem Gesamtdurschnitt und den einzelnen Locations. Auch in der reingezoomten perspektive auf die Jahresdurschnitte auf den Wochentagen bleibt die Erkentniss aus Aufgabe 2 bestehen. Die Rangliste der der höchsten auslastung beginnt erneut mit der Schönbornstraße, gefolgt von der Kaiser und Spiegelstraße.

Die Graphik bestätigt den normalen Wochenrythmus. Sichtbar wird das an den Gelichbleiben Durchschnitten von Montag bis Donnerstag gefolgt von dem Anstieg an Freitag und Samstag und wenige Passanten am Sonntag. Ein Begründung für diesen Wochenrythmus ist Tägliche Geschäft in der Arbeitswoche gefolgt vom Wochenende an dem sich mehr Leute in der Innenstad aufhalten. Da am Sonntag viele Geschäfte geschlossen haben ist die Innenstad auch nicht so stark besucht. Bemerkenswert ist hier die gleichmäsige verteilung. Steigt der Gesamtdurchschnitt so steigt auch der durchschnitt jeder einzelenen Messtation. Das Spricht für eine gewisse Korrelation zwischen den Auslastungen der eizelnen Messtationen.

```
#' [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,
```

```r
    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")
```



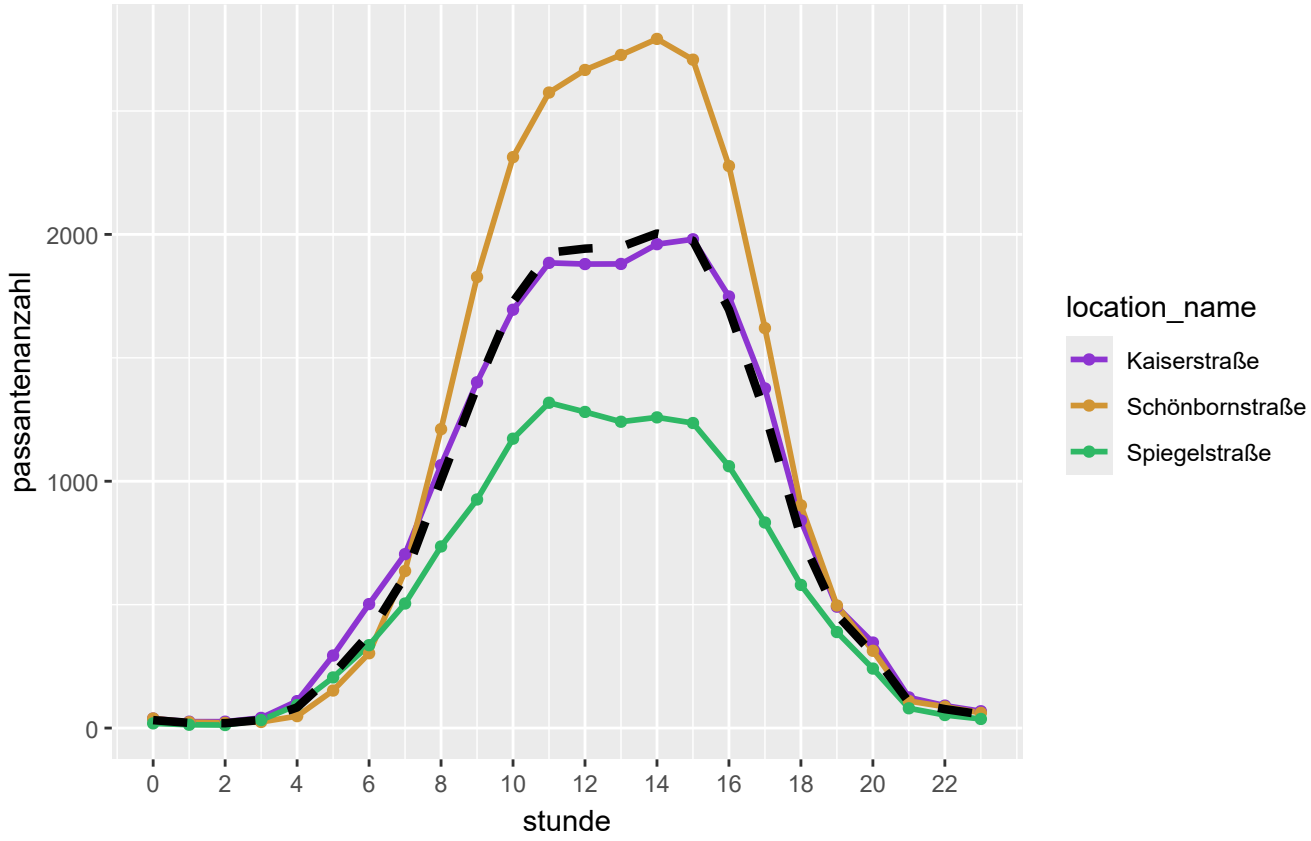
Über die Berechnung der Korrelation erkennt man das klar den zusammenhang der Messtationen untereinander wobei Kaiser und Spieglstaße enger zusammenhängen als die jewailige straße mit der Schönbornestraße. Das habe die Schönbornstraße weiter hervor wenn auch nicht sehr stark.

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()



#' [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")
passantenanzahl_stunden_wide
```

```
## # A tibble: 24 x 5
##    stunde Kaiserstraße Schönbornstraße Spiegelstraße durchschnitt_passanten_ge~1
##     <int>        <dbl>           <dbl>         <dbl>                         <dbl>
##  1      0         38.4            38.8          19.0                          32.1
##  2      1         25.9            23.9          13.3                          21.0
##  3      2         25.9            21.1          12.2                          19.7
##  4      3         41.2            24.7          32.3                          32.8
##  5      4        109.             48.2          93.1                          83.5
##  6      5        294.            152.          205.                          217.
##  7      6        502.            303.          336.                          380.
##  8      7        705.            636.          505.                          615.
##  9      8       1065.           1211.          736.                         1004.
## 10      9       1401.           1827.          926.                         1385.
## # i 14 more rows
## # i abbreviated name: 1: durchschnitt_passanten_gesamt
```

```r
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", # <-- DIESE ANWEISUNG HINZUFÜGEN
    linewidth = 1.5,
    linetype = "dashed"
  ) +

   scale_color_manual(values = globale_farb_palette)+

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



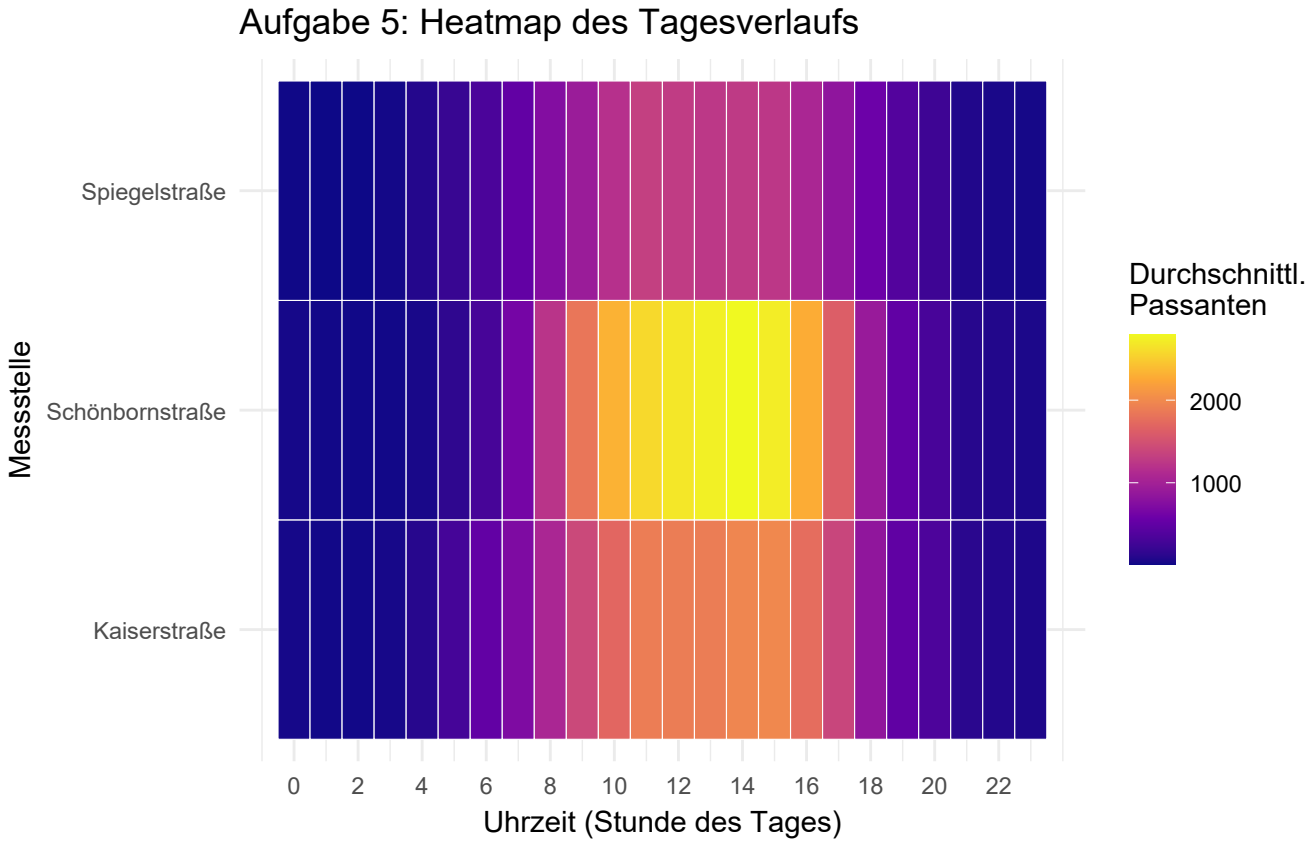

```
  labs(title = "Aufgabe 5: Durchschnittlicher Tagesverlauf (Gegliedert & Gesamt)",
       x = "Uhrzeit (Stunde des Tages)",
       y = "Durchschnittliche Passanten pro Stunde",
       color = "Standort")
```

```
## <ggplot2::labels> List of 4
##  $ x     : chr "Uhrzeit (Stunde des Tages)"
##  $ y     : chr "Durchschnittliche Passanten pro Stunde"
##  $ colour: chr "Standort"
##  $ title : chr "Aufgabe 5: Durchschnittlicher Tagesverlauf (Gegliedert & Gesamt)"
```

Extra Plot

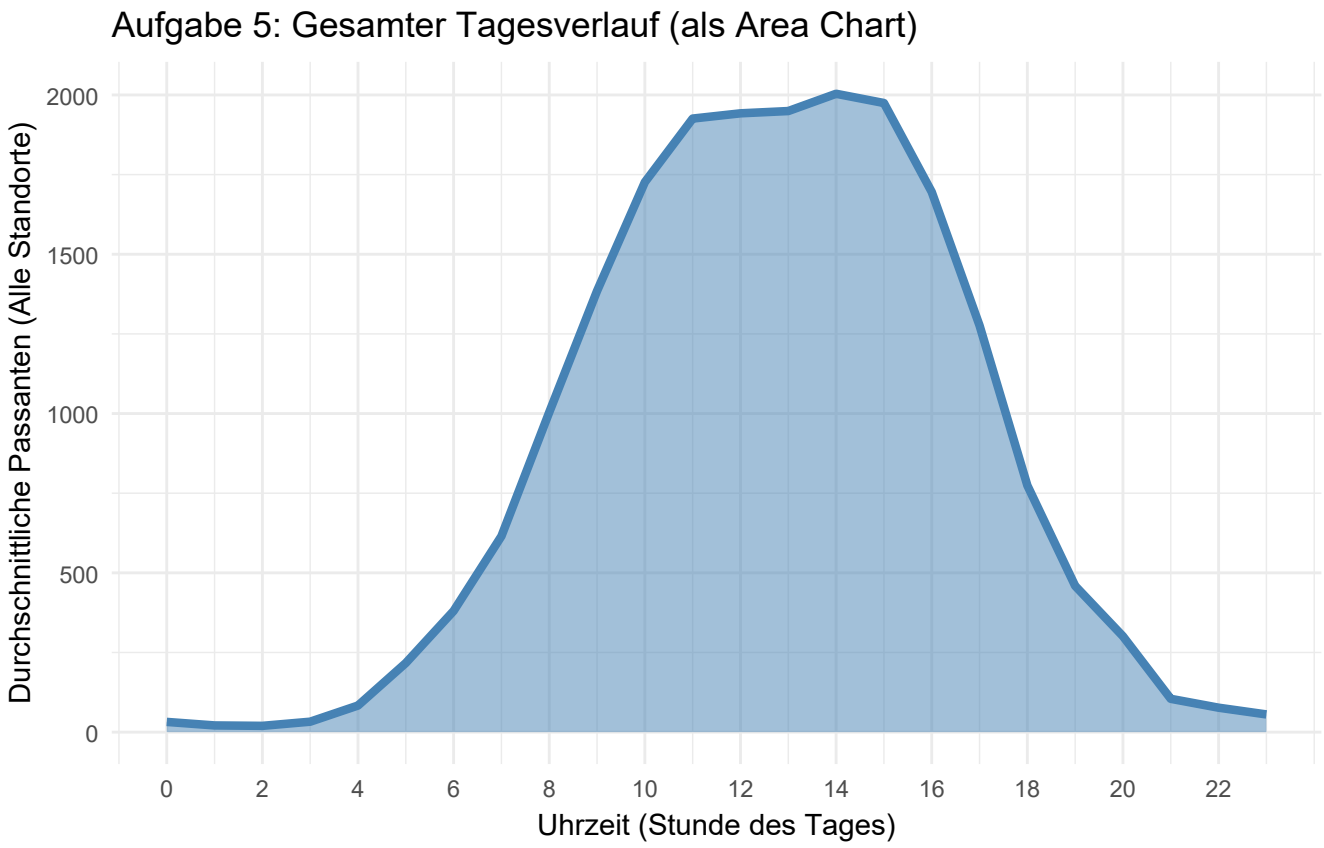
```
#' [Aufgabe 5: Heatmap des Tagesverlaufs]
#' @description
#' [Zeigt die Passantenanzahl als farbige Kacheln.
#' Ideal, um "Hot Spots" (Peaks) schnell zu identifizieren.]
library(ggplot2)
# library(viridis) # Für eine farbenblinde-sichere Palette (optional)

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 = "Aufgabe 5: Heatmap des Tagesverlaufs",
    x = "Uhrzeit (Stunde des Tages)",
    y = "Messstelle",
    fill = "Durchschnittl.\nPassanten" # \n für Zeilenumbruch
  ) +
  theme_minimal()
```




```
#' [Aufgabe 5: (Einzel-) Flächendiagramm für Gesamtsumme]
#' @description
#' [Zeigt den Gesamtdurchschnitt über den Tag als gefüllte Fläche.
#' Betont das Gesamtvolumen.]

ggplot(avg_stunde_gesamt,
       aes(x = stunde, y = durchschnitt_passanten_gesamt)) +
```

```r
  # Erst die Fläche (mit Transparenz)
  geom_area(fill = "steelblue", alpha = 0.5) +

  # Dann die Linie (im selben Farbton, aber dunkler/solide)
  geom_line(color = "steelblue", linewidth = 1.5) +

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

  labs(
    title = "Aufgabe 5: Gesamter Tagesverlauf (als Area Chart)",
    x = "Uhrzeit (Stunde des Tages)",
    y = "Durchschnittliche Passanten (Alle Standorte)"
  ) +
  theme_minimal()
```



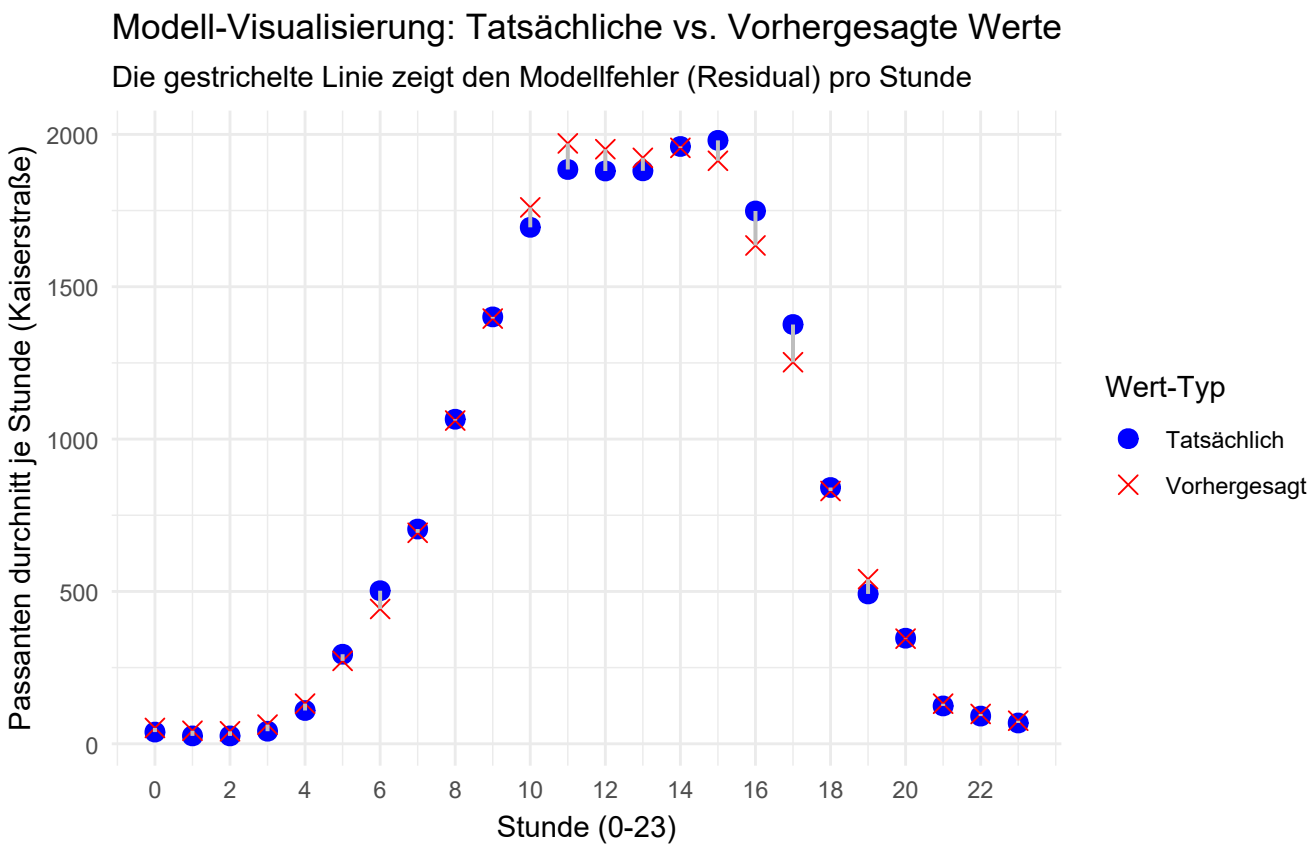

```r
#' [3. Multiple Lineare Regression]
#' @description
#' [Modelliert einen Standort (Kaiserstraße) als Funktion
#' der BEIDEN anderen Standorte, um deren gemeinsamen
#' und individuellen Einfluss zu quantifizieren.]
#' @param passantenanzahl_stunden_wide [DataFrame] Dein "breiter" DF.
#'   Benötigt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

```r
#' [Text-Output] 'summary(model)' zeigt die Ergebnisse (wie in deinem Screenshot).

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

model <- lm(Kaiserstraße ~ Schönbornstraße + Spiegelstraße, data = correlation_data)

summary(model)
```

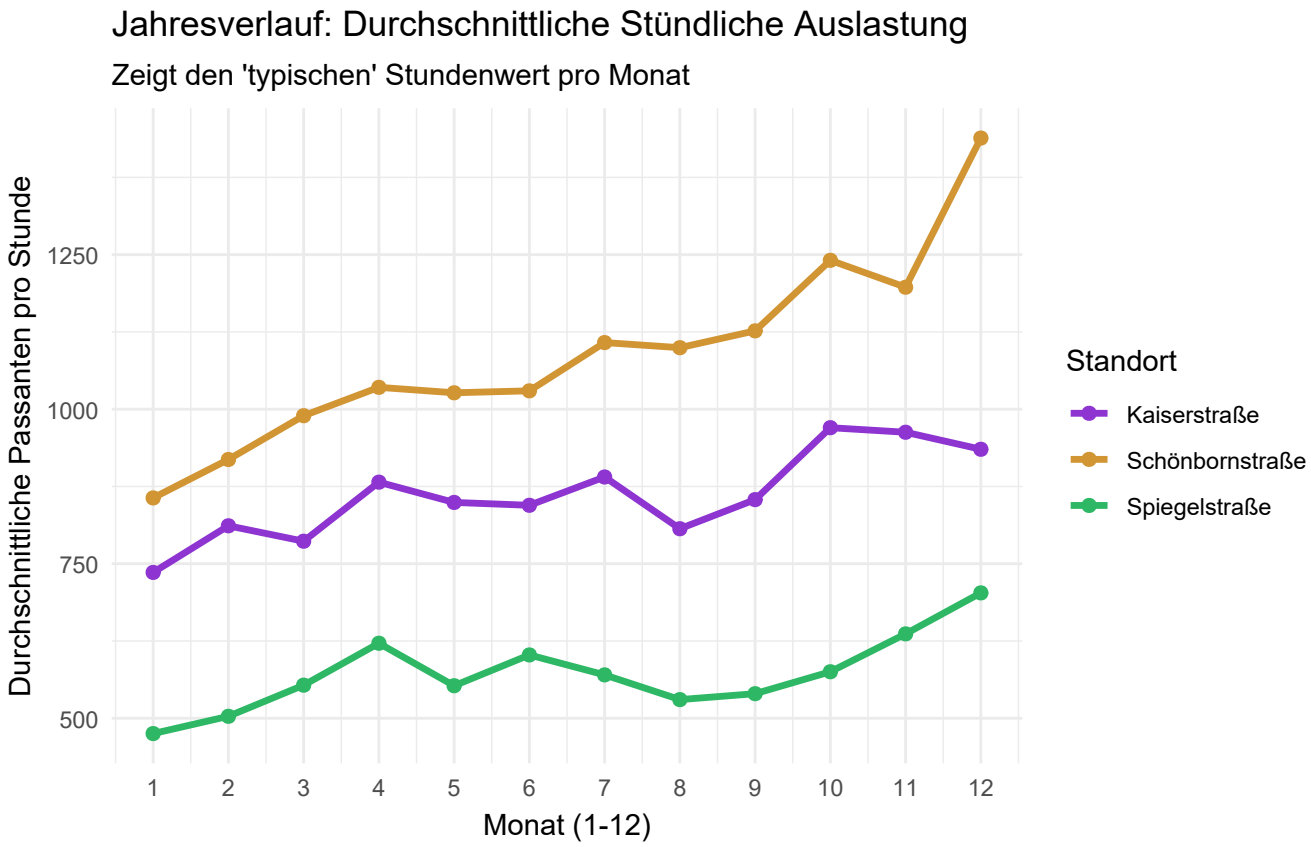
```
##
## Call:
## lm(formula = Kaiserstraße ~ Schönbornstraße + Spiegelstraße,
##     data = correlation_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -85.651 -21.882  -7.153  11.705 123.797
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      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
#' [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.]
#' @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'
  ) +
```

```r
geom_segment(
  aes(xend = stunde, y = Kaiserstraße, yend = .fitted),
  color = "grey",
  linewidth = 0.7
) +

# --- 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()
```



```r
library(lubridate) # Falls noch nicht geladen

#' [Monat extrahieren]
#' @description
#' [Stellt sicher, dass die 'monat'-Spalte (numerisch 1-12)
#' für die Aggregation verfügbar ist.]
passanten_mit_monat <- passanten %>%
  mutate(
    # (Stelle sicher, dass 'zeitstempel_dt' existiert,
    # siehe Code von Aufgabe 4/5 'Datum extrahieren')
    monat = month(zeitstempel, label = FALSE)
  )

#' [Stündl. Mittelwert pro Monat (Gegliedert)]
#' @description
#' [Berechnet den durchschnittlichen Passantenwert PRO STUNDE
#' für jeden Monat (1-12) und jede Messstelle.
#' (Logik von A5, angewandt auf A3)]
avg_hourly_by_month_plot <- passanten_mit_monat %>%
   group_by(monat, location_name) %>%
   summarise(
      # Das ist die Logik aus Aufgabe 5: mean(passanten)
      durchschnitt_pro_stunde = mean(passanten, na.rm = TRUE)
      ) %>%
   ungroup()

#' [Plot: Jahresverlauf der stündl. Mittelwerte]
#' @description
#' [Zeigt den Jahresverlauf der durchschnittlichen stündlichen
#' Auslastung pro Standort.]
ggplot(avg_hourly_by_month_plot,
       aes(x = monat, y = durchschnitt_pro_stunde, color = location_name)) +

  geom_line(linewidth = 1.2) +
  geom_point(size = 2) +

  # Verwende deine globale Palette
  scale_color_manual(values = globale_farb_palette) +

  # Zeige alle 12 Monate auf der Achse
  scale_x_continuous(breaks = 1:12) +

  labs(
    title = "Jahresverlauf: Durchschnittliche Stündliche Auslastung",
    subtitle = "Zeigt den 'typischen' Stundenwert pro Monat",
    x = "Monat (1-12)",
    y = "Durchschnittliche Passanten pro Stunde",
    color = "Standort"
  ) +
  theme_minimal()
```



Fazit

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.