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)
```

```
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", # z.B. Dunkelgrau
  "Gesamtsumme"        = "#404040"  # z.B. Schwarz
)
```

Passanten in Würzburg

Import und Datenbereinigung

zuerst importiert man die Daten und zeigt Sie an um zu überprüfen ob der import funktioniert hat.

```
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. Um mit ihnen sinvolle, descriptive oder weiterführend eine Explorative Datenanalyse durchzuführen sollten die Daten zuerst bereinigt werden. Als nächstes prüfen wir ob die Spalentnamen Aussagekräfig sind und ändern sie ab falls notwendig

```
#Formatieren Spalten Namen
passanten <- clean_names(passanten)
```

Somit können wir nun leer Spalten entfernen und nachsehen wie vollständig die Daten sind.

```
#Aufzählung der NA nach Spalte
colSums(is.na(passanten))
```

```
##   zeitstempel        wetter    temperatur     passanten location_name
##             0           112           112             0             0
##     geo_punkt          jahr         monat         woche           tag
##             0             0             0             0             0
##        stunde     wochentag   tag_im_jahr
##             0             0             0
```

Zum Schluss erhalten wir eine übersicht über die Daten

```
skim(passanten)
```

Table 1: 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

Aufgabe 1

Beschreibung der Daten

Der Datensatz besteht aus 26015 Zeile und 14 Spalten.Die in den Spalten aufgezeichneten werte sind:

```
skim(passanten)
```

Table 5: 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

Die interessantest Variable ist hierbei die anzahl der Passanten. Dabei werden hir die anzahl der Passanten mit jewails eine Zeitstempel und weiteren Metadaten wie: Temperatur, Ort der Aufzeichnung und Koordinaten des Ortes Zeilenweise angegeben. Die Zeitstempel sind dabei Stündlich für jede der drei Messstationen angegeben.

```
table(passanten$location_name)
```

```
##
##    Kaiserstraße Schönbornstraße   Spiegelstraße
##            8694            8648            8673
```

Somit kann man davon ausgehen das die anzahl der Passanten der anzahl der Passanten entspricht die inerhalb einer Stunde durch die Messtation erfasst wurden. Die Metadaten dienen dann zur Interpretation der Hauptdaten.

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. 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. Methodik_hystreet.com. 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)."

Standorte

```
anzahl_messpunkte_df <- passanten %>%
  summarise(
    Anzahl_Locations = n_distinct(location_name),
    Anzahl_GeoPunkte = n_distinct(geo_punkt)
  )

werte_messpunkte_df <- passanten %>%
   summarise(
      Locations  = unique(location_name),
      Geo_Punkte  = unique(geo_punkt)
   )

anzahl_messpunkte_df
```

```
## # A tibble: 1 x 2
##   Anzahl_Locations Anzahl_GeoPunkte
##              <int>            <int>
## 1                3                3
```

```
werte_messpunkte_df
```

```
## # A tibble: 3 x 2
##   Locations       Geo_Punkte
##   <chr>           <chr>
## 1 Kaiserstraße    49.798498976405355, 9.933887635731686
## 2 Schönbornstraße 49.795490162266525, 9.931060093195851
## 3 Spiegelstraße   49.79512717222457, 9.934114106308467
```

Der Datenstatz enthält Koordinaten in der Spalte geo_punkt und die orte der Messtationen in der Spalte location_name. Wenn man sich anschaut wie of eine einzelner Standort vorkommt sieht man, dass es jewail 3 Einzelne Standorte gibt und dazu passend die 3 Standorte in Koordinatenform. Die Summen (Wie oft ein Einzelner eintrag vorkommt) ist ebenfalls gleich. Somit ist eine eindeutige zuordung der Standorte zu den Koordinaten möglich. Gleichzeit können wir sicher sein das keine Falschen Standorte oder Koordinaten vorliegen und das jeder Standort auch mit den Korrekten Korrdinaten versehen sind.Wenn man nun die Koordinaten auf einer Karte anzeigen Lässt erhält man folgendes Bild.

Man kann sehen, das die Stationen direkt in der Innenstad stationiert 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 in der nähe zum Hauptbahnhof. Dabei sind alle diese Straßen Hauptverkehrsstraßen auf welchen man mit gleichmäsiger auslastung rechnen kann. Das Dreiecksmuster welche die Stationen Aufspannen bilden somit eine Art Transitstrecke zwischen: Hauptbahnhof -> Hofgarten -> Marienkapelle -> Hauptbahnhof.

Aufgabe 2

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

Figure 1: Hier steht die Bildunterschrift.

```
# 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 = mean(passanten, na.rm = TRUE)
  )

# --- 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 = mean(passanten, na.rm = TRUE)
  )

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

Table 9: 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	861
Schönbornstraße	9420052	366	785004	180165	25738	1089
Spiegelstraße	4961655	366	413471	94895	13556	572

```
passanten_insgesamt
```

```
## # A tibble: 1 x 6
##   passanten_jahr_summe anzahl_tage_erfasst durchschnitt_pro_monat
##                  <dbl>               <int>                  <dbl>
## 1             21864318                 366                1822026.
## # i 3 more variables: durchschnitt_pro_woche <dbl>, durchschnitt_pro_tag <dbl>,
## #   durchschnitt_pro_stunde <dbl>
```

Aufgabe 3

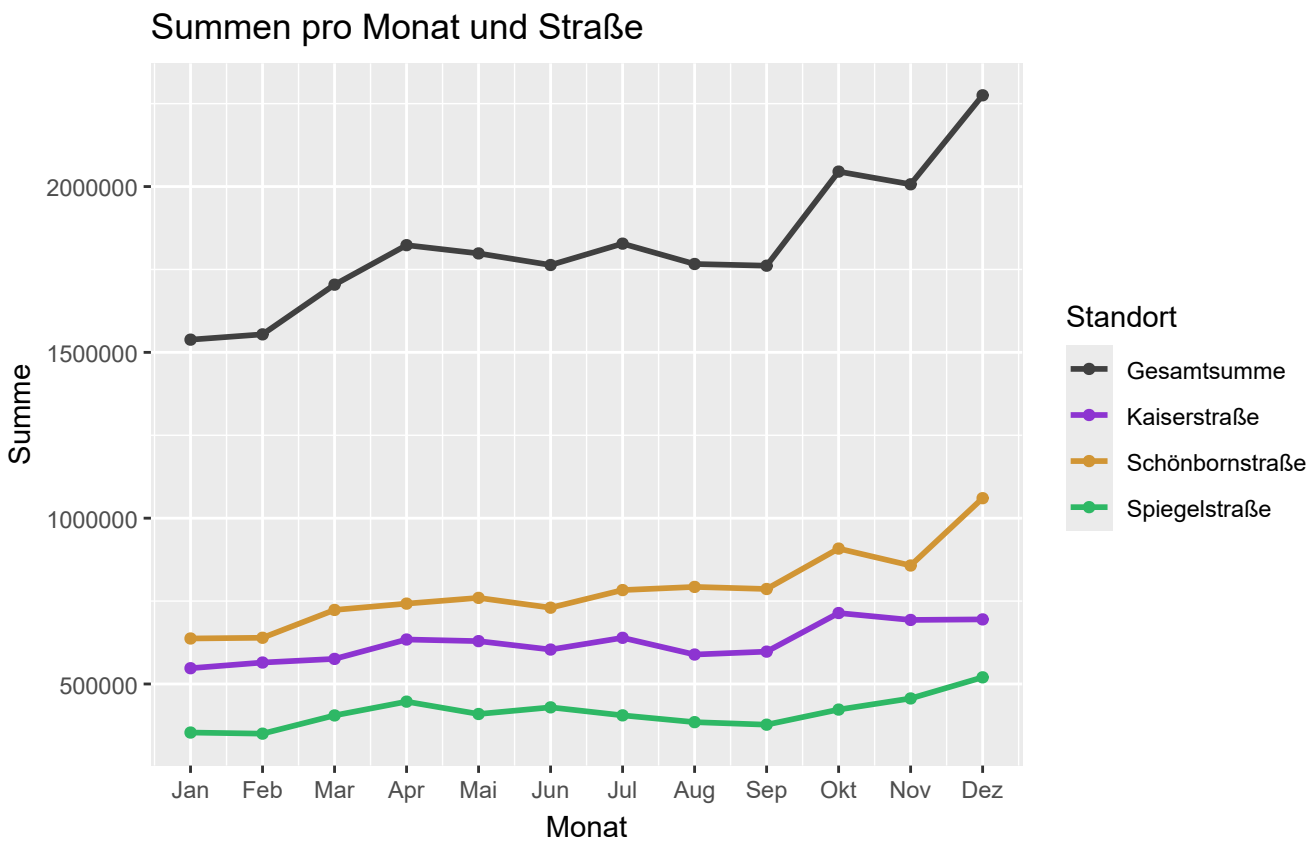
```
#' [Monatssumme über das Jahr]
#' @description
#' [Die Funktion Berechnet die Monatsumme der Passaten, gruppiert nach der location_name pro jahr]
#'
#' @param passanten df
#'
#' @return
#' [Rückgabewert ist ein der df sume_monat. Das wide Format zeigt die Daten sortiert nach monat und Loc
#'
```

```r
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)

summe_monat
```

```
## # A tibble: 12 x 5
##    monat Kaiserstraße Schönbornstraße Spiegelstraße Gesamtsumme
##    <dbl>        <dbl>           <dbl>         <dbl>       <dbl>
##  1     1       547543          637205        353571     1538320
##  2     2       564705          639348        350214     1554269
##  3     3       575659          723285        405180     1704127
##  4     4       634108          742347        446756     1823215
##  5     5       629219          759642        409499     1798365
##  6     6       603836          730054        429538     1763434
##  7     7       639235          783184        405390     1827816
##  8     8       588829          792729        384926     1766492
##  9     9       597582          786448        377293     1761332
## 10    10       713967          908278        422816     2045071
## 11    11       693123          857273        456390     2006797
## 12    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
ggplot(data = mittelwert_monat_plot, aes(x = monat, y = gesamtsumme, color = location_name)) +
  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")
```



Aufgabe 4

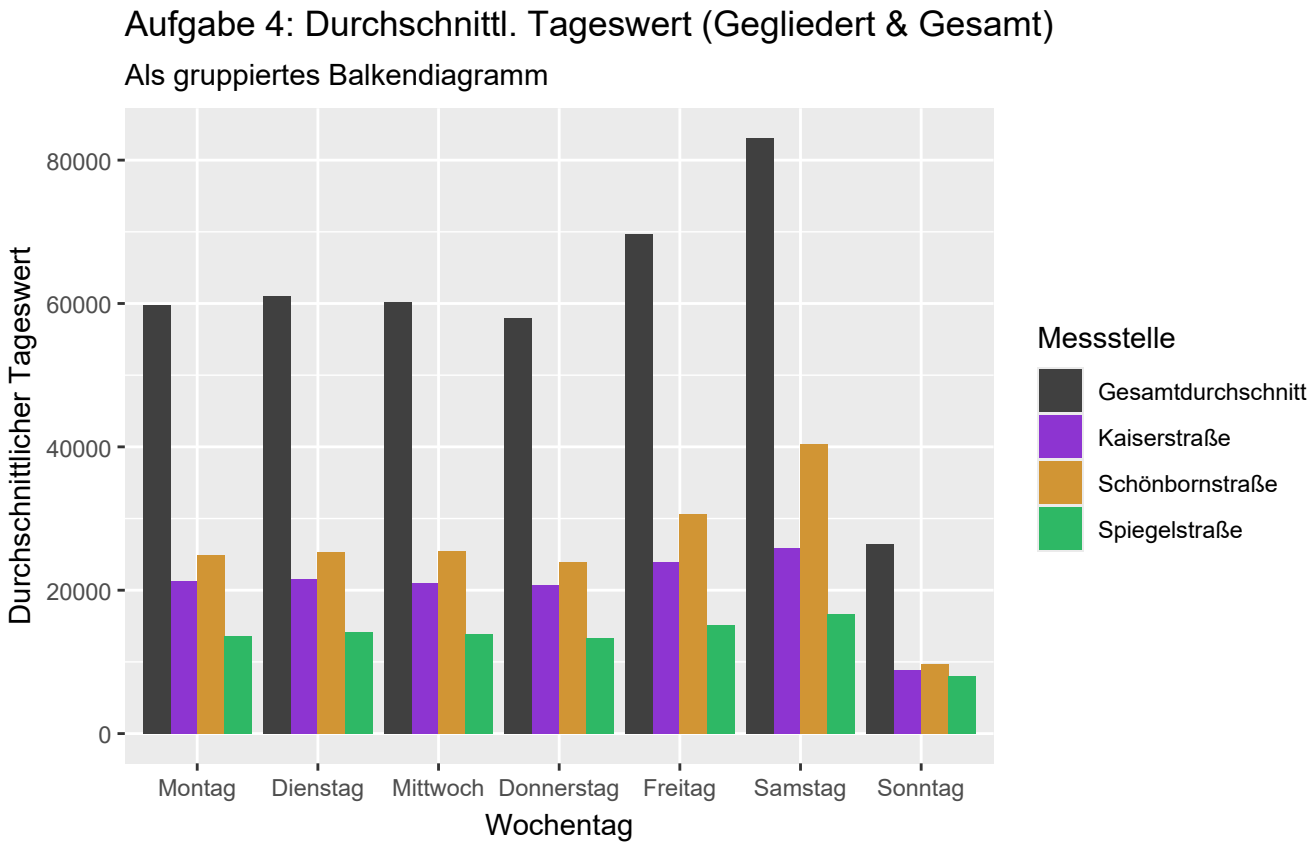
```
tagessumme <- passanten %>%
   mutate(
      Datum = as_date(zeitstempel)
   ) %>%
   group_by(Datum, wochentag, location_name) %>%
   summarise(
    Tagessumme = sum(passanten, na.rm = TRUE)
    ) %>%
   ungroup()

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


#' [durschnitt tageswert]
```

```r
#' @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")
durchschnittlicher_tageswert_location_wide
```

```
## # A tibble: 7 x 5
##   wochentag  Kaiserstraße Schönbornstraße Spiegelstraße durchschnitt_wochentag
##   <fct>             <dbl>           <dbl>         <dbl>                  <dbl>
## 1 Montag           21221.          24899.        13606                  59727.
## 2 Dienstag         21485.          25323.        14172.                 60981.
```

```
## 3 Mittwoch         20961.          25391.        13842.                60194.
## 4 Donnerstag       20706.          23931.        13377.                58014.
## 5 Freitag          23990.          30563.        15176.                69729.
## 6 Samstag          25893.          40453.        16735.                83081.
## 7 Sonntag           8818.           9629.         7974.                26421.
```

```r
#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 = "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")
```

Aufgabe 5

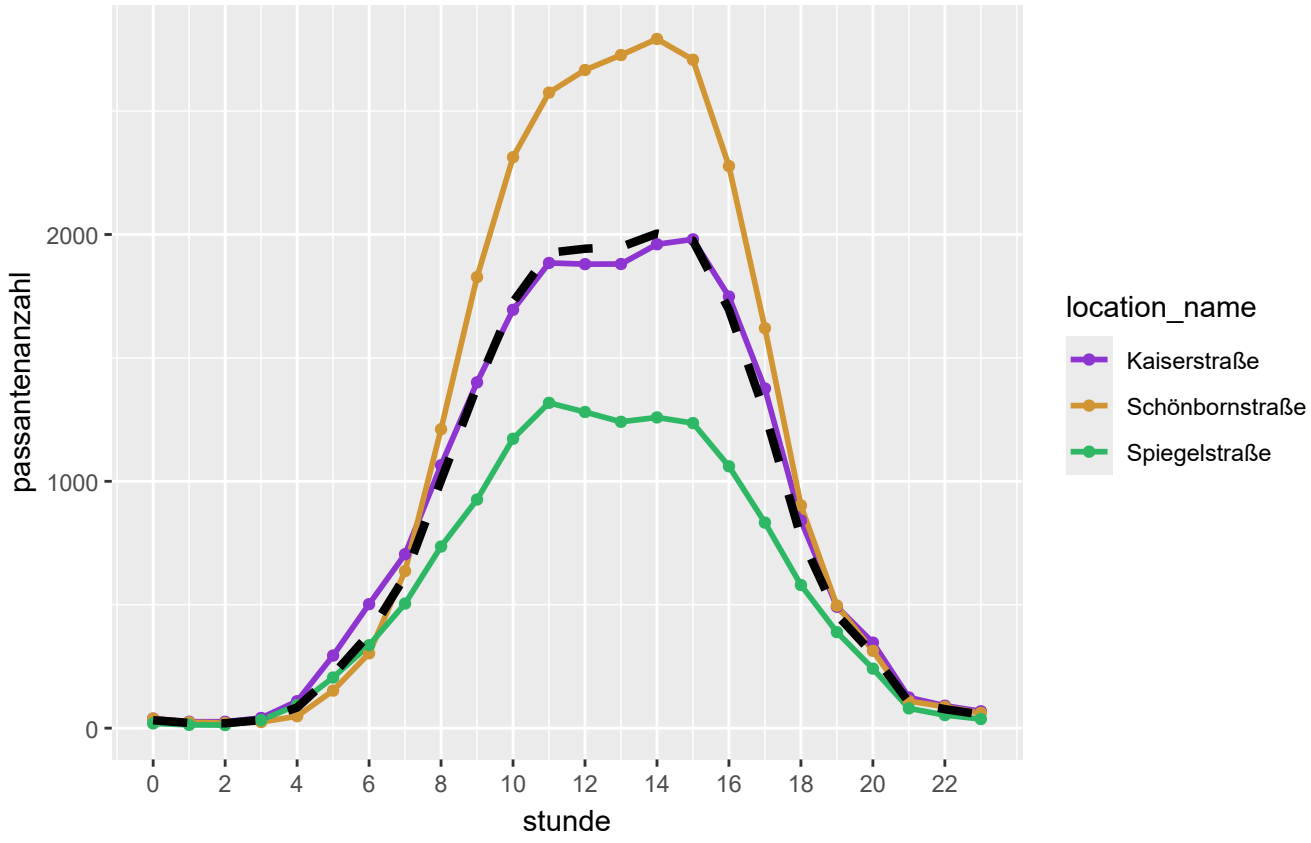
```
#'[Stündl. Mittelwert (Gegliedert)]
#' @description
#' Berechnet den durchschnittlichen Passantenwert für jede Stunde (0-23)
#' und für jede einzelne Messstelle ('location_name').
#' @param passanten [DataFrame] Das Roh-DataFrame 'passanten'.
#'   Benötigt die Spalten 'stunde', 'location_name' und '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")
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))
```

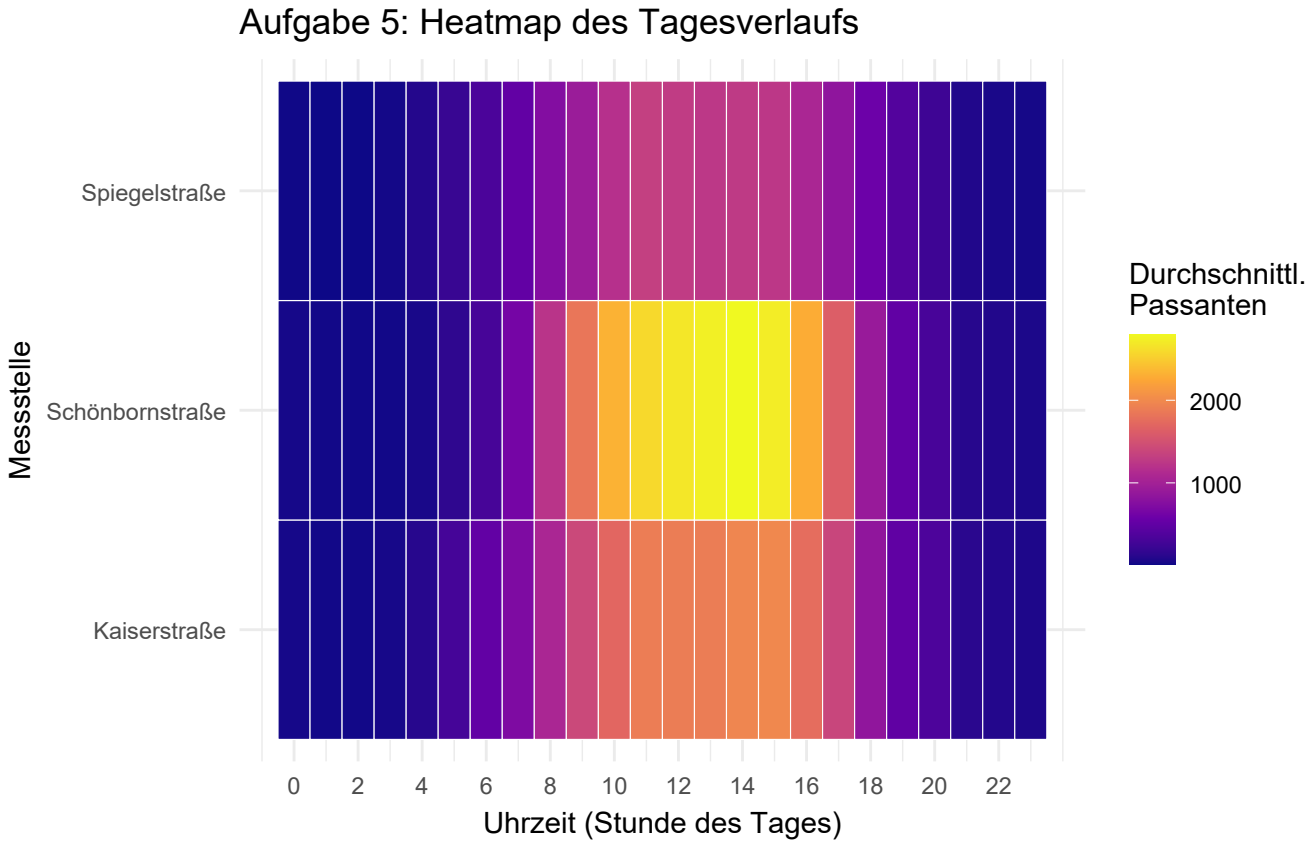



```
labs(title = "Aufgabe 5: Durchschnittlicher Tagesverlauf (Gegliedert & Gesamt)",
     x = "Uhrzeit (Stunde des Tages)",
     y = "Durchschnittliche Passanten pro Stunde",
     color = "Standort")
```

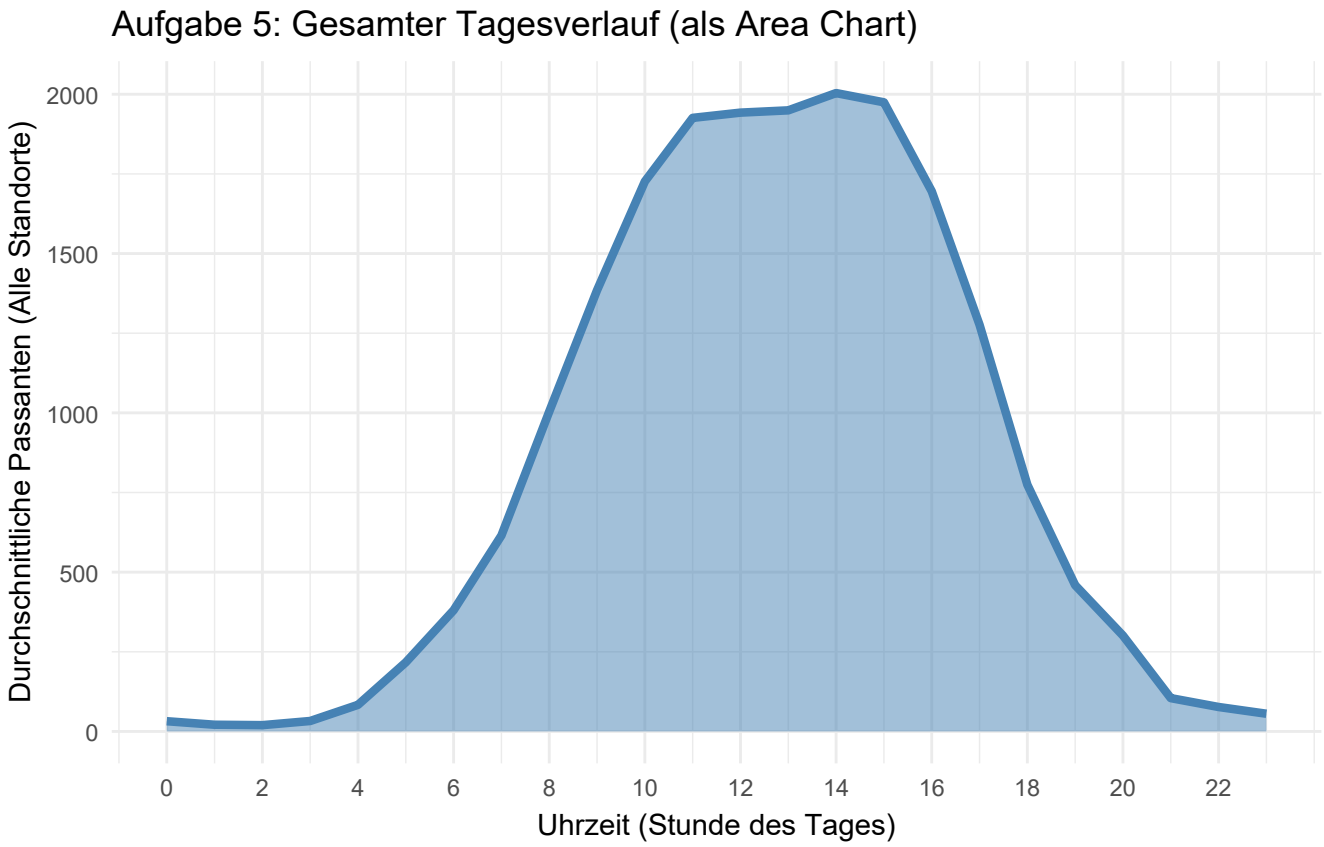
```
## <ggplot2::labels> List of 4
##  $ x     : chr "Uhrzeit (Stunde des Tages)"
##  $ y     : chr "Durchschnittliche Passanten pro Stunde"
##  $ colour: chr "Standort"
##  $ title : chr "Aufgabe 5: Durchschnittlicher Tagesverlauf (Gegliedert & Gesamt)"
```

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)) +

  # 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()
```



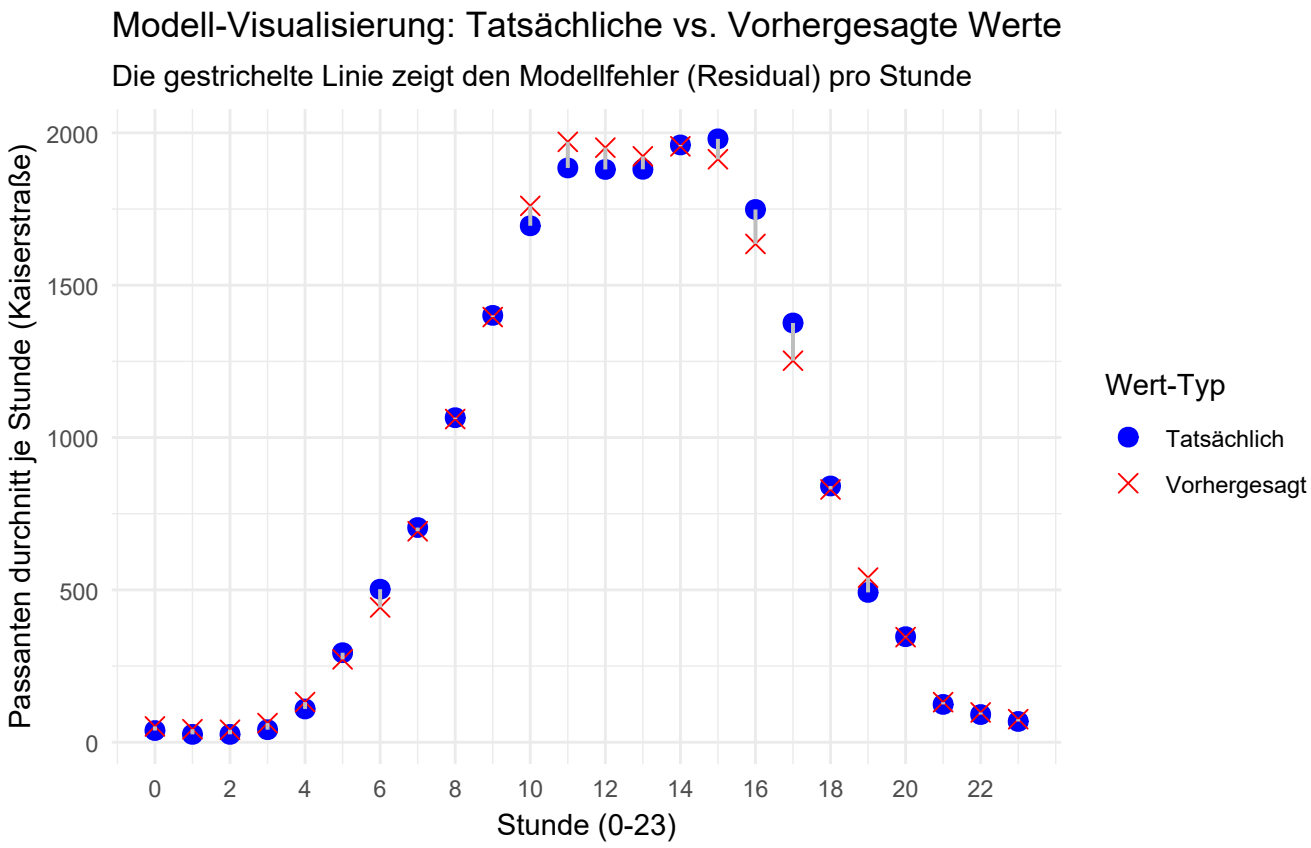

```
#' [3. Multiple Lineare Regression]
#' @description
#' [Modelliert einen Standort (Kaiserstraße) als Funktion
#' der BEIDEN anderen Standorte, um deren gemeinsamen
#' und individuellen Einfluss zu quantifizieren.]
#' @param passantenanzahl_stunden_wide [DataFrame] Dein "breiter" DF.
#'   Benötigt 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)

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
```

```
#' [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'
  ) +

  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()
```



Literatur

https://studyflix.de/statistik/mittelwert-6133