R-Markdown zur Bearbeitung und Visualisierung der Daten

Jonas Zürcher, Stephan Wernli, Luca Casuscelli, Carole Mattmann

30.05.2020

Contents

1 Bearbeitung des Datensatzes in R

1.1 Laden der Bibliotheken und Import der Daten

Um diese Applikation zu erstellen wurden drei Bibliotheken benötigt. Mit Tidyverse wurde die Datenbearbeitung vereinfacht, mit Leaflet eine interaktive Karte erzeugt und mit Shiny die Applikation ermöglicht. Die ersten Schritte sind diese Bibliotheken sowie die CSV Datei zu laden.

1.2 Geodaten für die Zielbahnhöfe hinzufügen

Als erstes musste eine zusätzliche Spalte mit den Geodaten für die Zielbahnhöfe erstellt werden. Dazu wurde ein Vektor aus den Startbahnhöfen und ihrer Geodaten erstellt. Anschliessend wurde mittels der Bahnhofsnamen die Geodaten zum Zielbahnhof im Hauptdatensatz hinzugefügt.

```
# Datenformate anpassen
sbb$from <- as.character(sbb$from)
sbb$to <- as.character(sbb$to)

# Dataframe erstellen mit den Startorten und deren
# Geokoordinaten
sbb_geo <- sbb %>% group_by(from, lat, lon) %>% summarise()

# Umbennen der Spalte from zu to
sbb_geo <- sbb_geo %>% rename(to = from)

# Add geodatafrom
sbb2 <- sbb %>% rename(from_lat = lat, from_lon = lon)

# Erstellung eines neuen Dataframes mit den Geodaten der
# Ankunftsbahnhöfen
sbb2 <- left_join(sbb2, sbb_geo, by = "to")

sbb2 <- sbb2 %>% rename(to_lat = lat, to_lon = lon)
```

1.3 Erstellung einer Strecken ID

Dann wurde in einer zusätzlichen Spalte eine Strecken ID festgelegt. Diese ist eine Kombination aus dem Startbahnhof und Zielbahnhof.

```
sbb2 <- sbb2 %>% mutate(specialid = paste(from, "-", to))
```

1.4 Entfernen der “end-station” Einträge

Als nächstes wurden alle Einträge mit Zielbahnhof «end-station» entfernt. Diese Einträge konnten entfernt werden, da sie Artefakt der Datenextraktion darstellten und keine Strecke beschreiben.

```
sbb.m1 <- sbb2 %>% filter(to != "end-station")
```

1.5 Spalte mit Datum erstellen

Nun wurde eine weitere Spalte mit dem Datum erstellt. Dazu wurden geprüft ob eine Datumsangabe für die Abfahrt oder die Ankunft vorliegt. Für das Hinterlegen der Datumsangabe wurde die Abfahrtszeit priorisiert. Falls bei einem Eintrag keine Datumsangabe vorhanden ist, wird dieser herausgefiltert. Zum Schluss wird das Datum im Format «Jahr-Monat-Tag» abgespeichert.

```
# Datumseinträge anpassen
sbb.m1$arrival <- as.character(sbb.m1$arrival)
sbb.m1$departure <- as.character(sbb.m1$departure)

date <- 1

for (i in 1:length(sbb.m1$arrival)) {
    date[i] <- if (sbb.m1$arrival[i] != "None") {
        sbb.m1$arrival[i]
    } else {
        sbb.m1$departure[i]
    }
}
sbb.m1$date <- 1
sbb.m1$date <- date

# Neues Dataframe erstellen und Datum als Character
# formatieren
sbb.m2 <- sbb.m1
sbb.m2$date <- sbb.m2$date %>% as.character()

# Einträge ohne Datumsangabe herausfiltern
sbb.m2 <- sbb.m2 %>% filter(date != "None")

# Die Datenspalte auf das Format 'Jahr-Monat-Tag' begrenzen

sbb.m2$date <- substr(sbb.m2$date, 1, 10)
```

1.6 Einträge auf die gewählten Tage begrenzen

Als nächstes wurden nur noch Einträge mit den gewünschten Daten im Datensatz behalten.

```
sbb.m2 <- sbb.m2 %>% filter(date == "2020-03-09" | date == "2020-03-16" |
    date == "2020-03-23" | date == "2020-03-30" | date == "2020-04-06" |
    date == "2020-04-13" | date == "2020-04-20" | date == "2020-04-27" |
    date == "2020-05-04")
```

1.7 Ein Dataframe für jeden Tag erstellen

Nun wird der Datensatz in mehrere Datensätze aufgeteilt. Die Aufteilung erfolgt nach Datum.

```
sbb.09.3.2020 <- sbb.m2 %>% filter(date == "2020-03-09")

sbb.16.3.2020 <- sbb.m2 %>% filter(date == "2020-03-16")
```

```r
sbb.23.3.2020 <- sbb.m2 %>% filter(date == "2020-03-23")

sbb.30.3.2020 <- sbb.m2 %>% filter(date == "2020-03-30")

sbb.06.4.2020 <- sbb.m2 %>% filter(date == "2020-04-06")

sbb.13.4.2020 <- sbb.m2 %>% filter(date == "2020-04-13")

sbb.20.4.2020 <- sbb.m2 %>% filter(date == "2020-04-20")

sbb.27.4.2020 <- sbb.m2 %>% filter(date == "2020-04-27")

sbb.04.5.2020 <- sbb.m2 %>% filter(date == "2020-05-04")
```

1.8 Summierung der gefahrenen Strecken

Der nächste Schritt ist für jeden Datensatz und deren einzigartigen Strecken die Anzahl Fahrten zu berechnen.

```r
sbb.09.3.2020.count <- sbb.09.3.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)

sbb.16.3.2020.count <- sbb.16.3.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)

sbb.23.3.2020.count <- sbb.23.3.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)

sbb.30.3.2020.count <- sbb.30.3.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)

sbb.06.4.2020.count <- sbb.06.4.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)

sbb.13.4.2020.count <- sbb.13.4.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)

sbb.20.4.2020.count <- sbb.20.4.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)

sbb.27.4.2020.count <- sbb.27.4.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)

sbb.04.5.2020.count <- sbb.04.5.2020 %>% group_by(specialid,
    from, from_lat, from_lon, to, to_lat, to_lon, date) %>% count(specialid)
```

1.9 Berechnung der Verkehrsabnahme

Mit Hilfe der berechneten Anzahl Fahrten kann nun die Verkehrsabnahme in Prozent berechnet werden. Als Vergleichswerte wurden die Anzahl Fahrten vom ersten Erfassungstag (09.03.2020) verwendet. Ein positiver Wert bedeutet eine Abnahme und ein negativer Wert eine Zunahme.

```r
# 1
sbb.09.3.2020.count$change <- 1
for (i in 1:length(sbb.09.3.2020.count$specialid)) {
    sbb.09.3.2020.count$change[i] <- if (sbb.09.3.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.09.3.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.09.3.2020.count$specialid[i])) *
            100), digits = 1)

    } else {
        0
    }
}

# 2
sbb.16.3.2020.count$change <- 1
for (i in 1:length(sbb.16.3.2020.count$specialid)) {
    sbb.16.3.2020.count$change[i] <- if (sbb.16.3.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.16.3.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.16.3.2020.count$specialid[i])) *
            100), digits = 1)

    } else {
        0
    }
}

# 3
sbb.23.3.2020.count$change <- 1
for (i in 1:length(sbb.23.3.2020.count$specialid)) {
    sbb.23.3.2020.count$change[i] <- if (sbb.23.3.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.23.3.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.23.3.2020.count$specialid[i])) *
            100), digits = 1)

    } else {
        0
    }
}

# 4
sbb.30.3.2020.count$change <- 1
for (i in 1:length(sbb.30.3.2020.count$specialid)) {
    sbb.30.3.2020.count$change[i] <- if (sbb.30.3.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.30.3.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.30.3.2020.count$specialid[i])) *
            100), digits = 1)

    } else {
        0
```

```
    }
}

# 5
sbb.06.4.2020.count$change <- 1
for (i in 1:length(sbb.06.4.2020.count$specialid)) {
    sbb.06.4.2020.count$change[i] <- if (sbb.06.4.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.06.4.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.06.4.2020.count$specialid[i])) *
            100), digits = 1)

    } else {
        0
    }
}

# 6
sbb.13.4.2020.count$change <- 1
for (i in 1:length(sbb.13.4.2020.count$specialid)) {
    sbb.13.4.2020.count$change[i] <- if (sbb.13.4.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.13.4.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.13.4.2020.count$specialid[i])) *
            100), digits = 1)

    } else {
        0
    }
}

# 7
sbb.20.4.2020.count$change <- 1
for (i in 1:length(sbb.20.4.2020.count$specialid)) {
    sbb.20.4.2020.count$change[i] <- if (sbb.20.4.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.20.4.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.20.4.2020.count$specialid[i])) *
            100), digits = 1)

    } else {
        0
    }
}

# 8
sbb.27.4.2020.count$change <- 1
for (i in 1:length(sbb.27.4.2020.count$specialid)) {
    sbb.27.4.2020.count$change[i] <- if (sbb.27.4.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.27.4.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.27.4.2020.count$specialid[i])) *
            100), digits = 1)
```

```
    } else {
        0
    }
}

# 9

sbb.04.5.2020.count$change <- 1
for (i in 1:length(sbb.04.5.2020.count$specialid)) {
    sbb.04.5.2020.count$change[i] <- if (sbb.04.5.2020.count$specialid[i] %in%
        sbb.09.3.2020.count$specialid) {
        round(100 - (sbb.04.5.2020.count$n[i]/(subset(sbb.09.3.2020.count$n,
            sbb.09.3.2020.count$specialid == sbb.04.5.2020.count$specialid[i])) *
            100), digits = 1)

    } else {
        0
    }
}
```

1.10 Vereinigung der Datensätze

Nun können die Datensätze wieder zu einem einzigen Datensatz verbunden werden.

```
sbb_complete <- sbb.09.3.2020.count %>% bind_rows(sbb.16.3.2020.count) %>%
    bind_rows(sbb.23.3.2020.count) %>% bind_rows(sbb.30.3.2020.count) %>%
    bind_rows(sbb.06.4.2020.count) %>% bind_rows(sbb.13.4.2020.count) %>%
    bind_rows(sbb.20.4.2020.count) %>% bind_rows(sbb.27.4.2020.count) %>%
    bind_rows(sbb.04.5.2020.count)
```

1.11 Erstellung der Funktionen für die Farbwahl, Strichdicke und Gruppierung in Leaflet

Für die Farbwahl, Strichdicke und Gruppierung in der Visualisierung muss noch jeweils eine Funktion definiert werden, welche diese Attribute je nach Anzahl Fahrstrecken und prozentualer Verkehrsveränderung auswählt. Für die automatische Farbwahl wurden 6 Farben gewählt. Zwei verschiedene Rottöne für die Verkehrsabnahme, schwarz für neue Strecken oder keine Veränderung und zwei Grüntöne für Verkehrszunahmen. Zusätzlich wurde noch die Farbe Blau definiert, um allfällige Programmierfehler zu erkennen.

```
# Funktion für die dynamische Farbgebung der Strecke

getcolor <- function(p) {
    if (p == 0) {
        "black"
    } else if (p > 0 & p <= 50) {
        "#fb6a4a"  #lightred
    } else if (p > 50) {
        "#a50f15"  #darkred
    } else if (p < 0 & p >= -50) {
        "#74c476"  #lightgreen
    } else if (p <- 50) {
```

```
        "#006d2c"  #darkgreen
    } else {
        "blue"
    }
}
```

Die Strichdicke wurde anhand der Verkehrszahl festgelegt. Damit soll stark befahrene Strecken besser erkennbar sein. Die Gewichtung willkürlich gewählt.

```
# Funktion für die dynamische Strichdicke der Strecke

getweight <- function(n) {
    if (n < 10) {
        1
    } else if (n <= 100) {
        2
    } else {
        4
    }
}
```

Die Gruppierung wird später verwendet, um verschieden stark befahrene Strecken ein- und auszublenden in der interaktiven Karte. Es wurden dieselbe Verkehrszahlen für die Einteilung verwendet wie für die Strichdicke.

```
# Funktion für die dynamische Gruppeneinteilung der Strecke

getgroup <- function(n) {
    if (n < 10) {
        "Tiefe Verkehrsfrequenz (weniger als 10 Fahrten)"
    } else if (n <= 100) {
        "Mittlere Verkehrsfrequenz (10-100 Fahrten)"
    } else {
        "Hohe Verkehrsfrequenz (über 100 Fahrten)"
    }
}
```

1.12 Anwendung der Funktionen für die Farbwahl, Strichdicke und Gruppierung

Nun wurden diese drei Funktionen auf das komplette Datenset angewendet und jeweils eine neue Spalte mit den Attributen erstellt. Man könnte die Funktion auch direkt im Visualisierungscode ausführen, dies würde aber zu längeren Ausführungszeiten führen und wäre visuell je nach Datenmenge für den Nutzer ersichtlich.

```
# Anwendung der Funktionen zur Erstellung der Spalten:
# 'Color', 'Weight' und 'Group'

sbb_complete$color <- sapply(sbb_complete$change, getcolor)

sbb_complete$weight <- sapply(sbb_complete$n, getweight)

sbb_complete$group <- sapply(sbb_complete$n, getgroup)
```

```r
# Umformatierung der Datumsspalte von Character zu Date
sbb_complete$date <- as.Date(sbb_complete$date)
```

2 Visualisierung der Daten in der Applikation

Mit dem bearbeiteten Datensatz können nun die Daten leicht in der Applikation visualisiert werden.

2.1 Erstellung eines individuellen Diagramm-Layouts

Für die Visualisierung des Liniendiagramms wurden noch zuerst ein zur Verfügung gestelltes Diagramm-Layout leicht individualisiert.

```
theme_eldorado <- function() {
    theme_classic() %+replace% theme(panel.grid.minor.y = element_line(color = "black",
        size = 0.05, linetype = "dashed"))
}
```

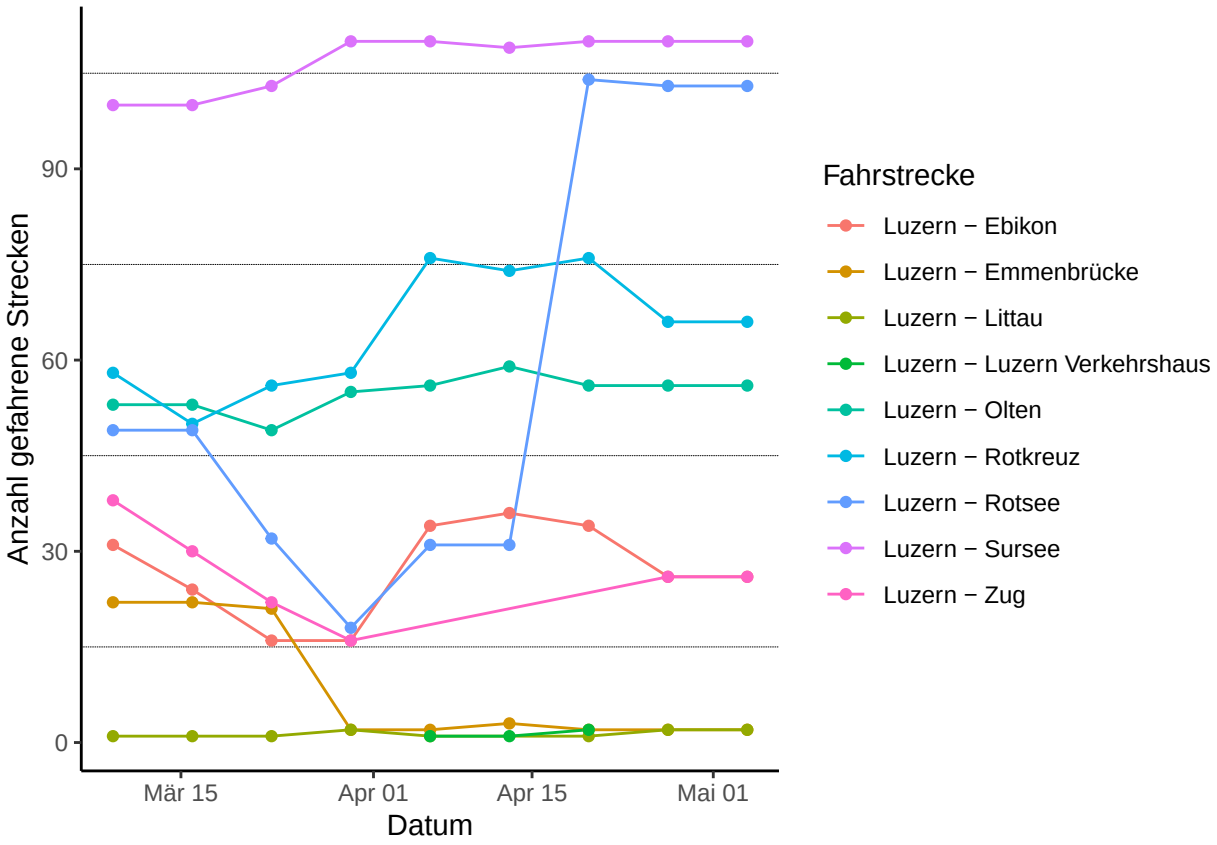
2.2 Erstellung eines GGplot-Diagramms mit einem bestimmten Startbahnhof

Um die GGplot-Visualisierung zu optimieren wurde Luzern als Startbahnhof ausgewählt.

```
test <- sbb_complete %>% filter(from == "Luzern")

plot <- ggplot(test, aes(test$date, test$n, color = specialid)) +
    geom_line() + geom_point() + theme_eldorado() + labs(color = "Fahrstrecke") +
    xlab("Datum") + ylab("Anzahl gefahrene Strecken")

plot
```



2.3 Erstellung eines GGplot-Diagramms mit einem bestimmten Streckenabschnitt

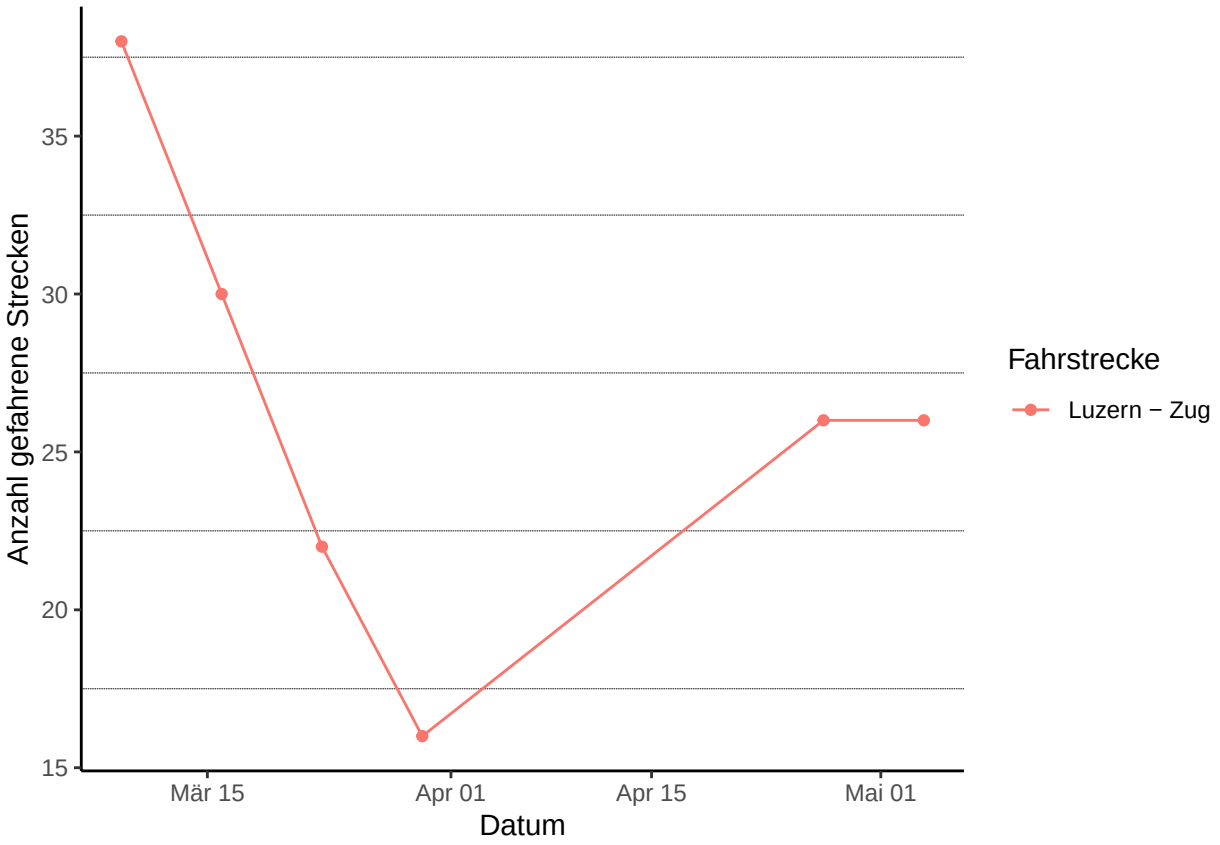
Um die GGplot-Visualisierung zu optimieren wurde Luzern-Zug als Streckenabschnitt ausgewählt.

```
test <- sbb_complete %>% filter(specialid == "Luzern - Zug")



plot <- ggplot(test, aes(test$date, test$n, color = specialid)) +
    geom_line() + geom_point() + theme_eldorado() + labs(color = "Fahrstrecke") +
    xlab("Datum") + ylab("Anzahl gefahrene Strecken")

plot
```



2.4 Benutzeroberfläche programmieren

Die Applikation besteht aus zwei Teilen: der Benutzeroberfläche und des Servers. In der Benutzeroberfläche wurde festgelegt welche Anordnung und Design die verschiedenen Regler und Ausgaben (Karte, Diagramm) haben. Auch wurden Auswahlmöglichkeiten bei den Reglern definiert. Beim Server Teil wird zuerst festgelegt werden Daten verwendet werden, je nach Auswahl bei den Reglern wird der Datensatz entsprechend angepasst.

Für die Diagramm Ausgabe wurde festgelegt, dass wenn bei beiden Reglern die Grundeinstellung vorliegt, wird keine Ausgabe gemacht. Falls nur ein Startbahnhof ausgewählt wurden werden alle Strecken von diesem Bahnhof visualisiert. Sobald aber eine explizite Strecke ausgewählt wird, wird nur noch diese visualisiert.

Für die Karten Ausgabe wurde zuerst festgelegt, welche Ausschnitt fix angezeigt werden soll. Danach wurden eine Auswahl an Kartematerial (OpenStreetMap und CartoDB.Positron), eine Legende sowie eine Gruppenauswahl (tiefe Verkehrsfrequenz, mittlere Verkehrsfrequenz, hohe Verkehrsfrequenz implementiert. Zusätzlich besteht die Möglichkeit entweder alle Strecken vom Datensatz oder alle Strecken eines bestimmten Startbahnhofs anzeigen zu lassen. Die Strecke mussten mittels einer Schleife innerhalb des Leaflet Pakets visualisiert werden, da sonst die Strecken auf falsche Weise verbunden würden.

```
# Benutzeroberfläche festlegen
ui <- bootstrapPage(tags$style(type = "text/css", "html, body {width:100%;height:100%}"),
    leafletOutput("myMap", width = "100%", height = "65%"), absolutePanel(id = "plot",
        class = "panel panel-default", bottom = 0, left = 10,
        width = 500, fixed = TRUE, draggable = TRUE, height = "auto",
        plotOutput("plot", height = "325px", width = "100%")),
    absolutePanel(id = "control", bottom = 60, right = 120, sliderInput("date",
```

```
        "Datum", min = min(sbb_complete$date), max = max(sbb_complete$date),
        value = min(sbb_complete$date), step = 7), selectInput("start",
        "Startbahnhof", choices = c("- Alle Startbahnhöfe -",
            "Basel SBB", "Zürich HB", "Luzern", "Bern", "Locarno",
            "Davos Dorf", "Genève"), selected = NULL), selectInput("track",
        "Streckenabschnitt", choices = choice <- sort(append(sbb_complete$specialid,
            "- Alle Strecken -")), selected = "- Alle Strecken -")))

# Datengrundlage und reaktive Filter festlegen
server <- function(input, output, session) {

    ## Für die Karte
    sbb_filtered <- reactive({
        data1 <- sbb_complete %>% filter(date == (input$date)) %>%
            filter(from == (input$start))
    })

    ## Für den GGplot

    # Filter Startbahnhof

    sbb_filtered2 <- reactive({
        data2 <- sbb_complete %>% filter(from == (input$start))
    })

    # Filter Streckenabschnitt
    sbb_filtered3 <- reactive({
        data2 <- sbb_complete %>% filter(specialid == (input$track))
    })

    # GGplot Ausgabe
    output$plot <- renderPlot({
        if (input$start == "- Alle Startbahnhöfe -" & input$track ==
            "- Alle Strecken -") {
            data2 <- sbb_complete


        } else if (input$start != "- Alle Startbahnhöfe -" & input$track ==
            "- Alle Strecken -") {
            data2 <- sbb_filtered2()
            ggplot(data2, aes(data2$date, data2$n, color = data2$specialid)) +
                geom_line() + geom_point() + theme_eldorado() +
                labs(color = "Fahrstrecke") + xlab("Datum") +
                ylab("Anzahl gefahrene Strecken")

        } else if (input$start == "- Alle Startbahnhöfe -" & input$track !=
            "- Alle Strecken -") {
            data2 <- sbb_filtered3()
            ggplot(data2, aes(data2$date, data2$n, color = data2$specialid)) +
                geom_line() + geom_point() + theme_eldorado() +
                labs(color = "Fahrstrecke") + xlab("Datum") +
                ylab("Anzahl gefahrene Strecken")
```

```
        } else if (input$start != "- Alle Startbahnhöfe -" & input$track !=
            "- Alle Strecken -") {
            data2 <- sbb_filtered3()
            ggplot(data2, aes(data2$date, data2$n, color = data2$specialid)) +
                geom_line() + geom_point() + theme_eldorado() +
                labs(color = "Fahrstrecke") + xlab("Datum") +
                ylab("Anzahl gefahrene Strecken")


        } else {
            "Platzhalter"
        }
    })

    # Karten Ausgabe
    output$myMap = renderLeaflet({
        myMap <- leaflet(sbb_filtered()) %>% fitBounds(min(sbb_complete$from_lon),
            min(sbb_complete$from_lat), max(sbb_complete$from_lon),
            max(sbb_complete$from_lat)) %>% addProviderTiles(providers$CartoDB.Positron,
            group = "GREY") %>% addProviderTiles(providers$OpenStreetMap,
            group = "OSM") %>% addLegend(title = "Verkehrsabnahme in %",
            position = "bottomright", labels = c("> 50 %", "> 0 % ",
                "Keine Änderung/Neue Strecke", "< 0 %", "< -50 %"),
            colors = c("#a50f15", "#fb6a4a", "black", "#74c476",
                "#006d2c")) %>% addLayersControl(position = c("topright"),
            baseGroups = c("GREY", "OSM"), overlayGroups = c("Tiefe Verkehrsfrequenz (weniger als 10 Fa
                "Mittlere Verkehrsfrequenz (10-100 Fahrten)",
                "Hohe Verkehrsfrequenz (über 100 Fahrten)"))

        if (input$start == "- Alle Startbahnhöfe -") {
            data1 <- sbb_complete %>% filter(date == (input$date))
        } else {
            data1 <- sbb_filtered()
        }

        ## Schleife notwendig weil sonst die Streckenpunkte verbunden
        ## werden
        for (i in 1:nrow(data1)) {
            myMap <- addPolylines(myMap, lat = as.numeric(data1[i,
                c(3, 6)]), lng = as.numeric(data1[i, c(4, 7)]),
                color = as.character(data1[i, c(11)]), weight = as.numeric(data1[i,
                  c(12)]), group = as.character(data1[i, c(13)]),
                popup = paste0("Von: ", as.character(data1[i,
                  c(2)]), "<br>", "Nach: ", as.character(data1[i,
                  c(5)]), "<br>", "Anzahl gefahrene Strecken: ",
                  as.character(data1[i, c(9)])))
        }
        myMap
    })

}

shinyApp(ui, server)
```

Shiny applications not supported in static R Markdown documents