analysed.ch apps

analysed.ch apps

- COVID-19 und Todesziffern
- Todeszahlen shiny app
- Todeszahlen streamlit python app
- Automatic sleep stage scoring shiny app
- Mortality shinyMobile app

COVID-19 und Todesziffern

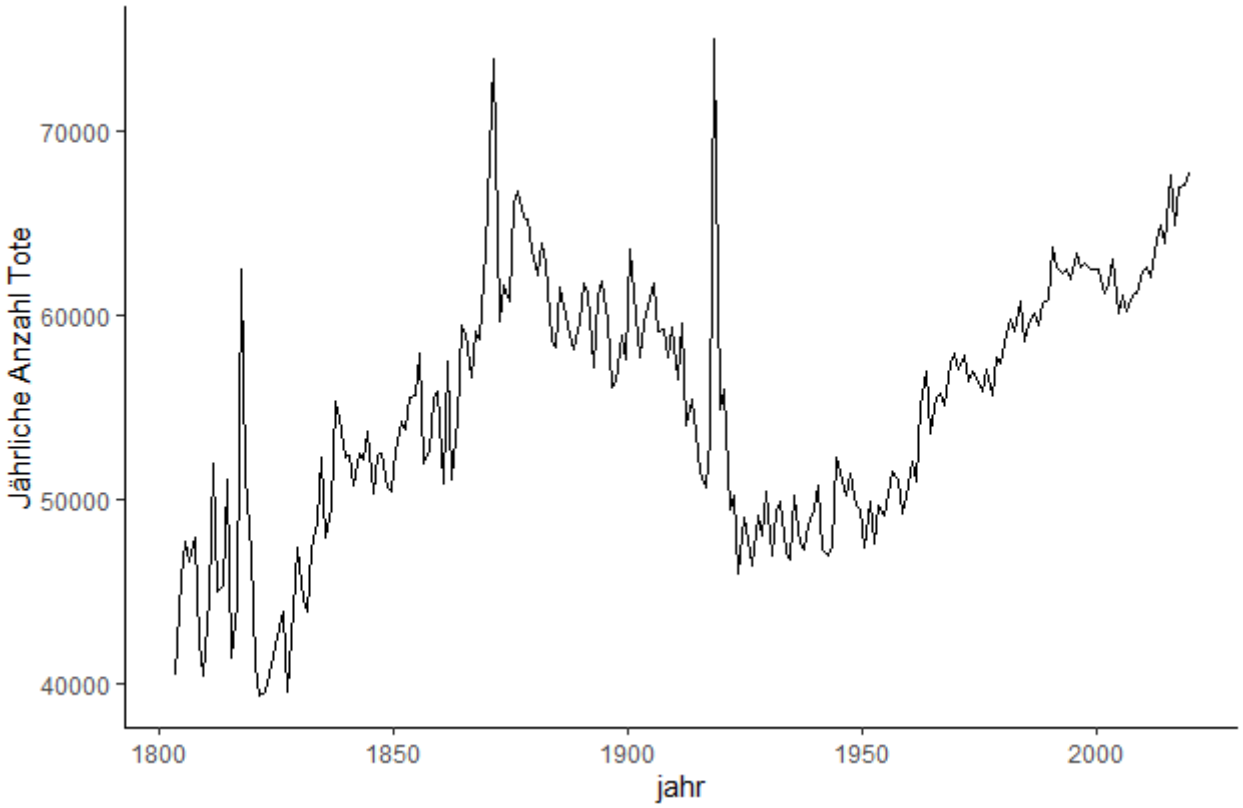
In den Jahren 2020 und 2021 wurde die gesamte Weltbevölkerung regelmässig über die laufende Anzahl Todesfälle und laufende Anzahl Ansteckungen von COVID-19 informiert. Dies in einen breiteren Kontext zu setzen wurde für die Bevölkerung etwas erschwert, da nur selten Todesfälle der vergangenen Jahren gezeigt und noch nie zuvor den Verlauf von viraler RNA/DNA (+ssRNA im Fall von COVID-19) in der Bevölkerung so genau ermittelt wurde.

Schauen wir uns doch in dem Fall mal die Todesfälle seit 1803 an - zum Glück wurden diese Daten vom Schweizerischen Bundesamt für Statistik regelmässig erhoben und können für jedermann einfach runtergeladen werden.

Hierfür benutzen wir hier die open source software R und R studio. Natürlich könnte man die Daten auch mit anderen Programmen anschauen, aber R verfügt über interessante features wie interaktive Darstellungen (z.B. shiny apps) und verschiedene Pakete wie z.B. das vom Bundesamt für Statistik (BFS) für einfaches Runterladen und Bearbeiten von Datenbanken.

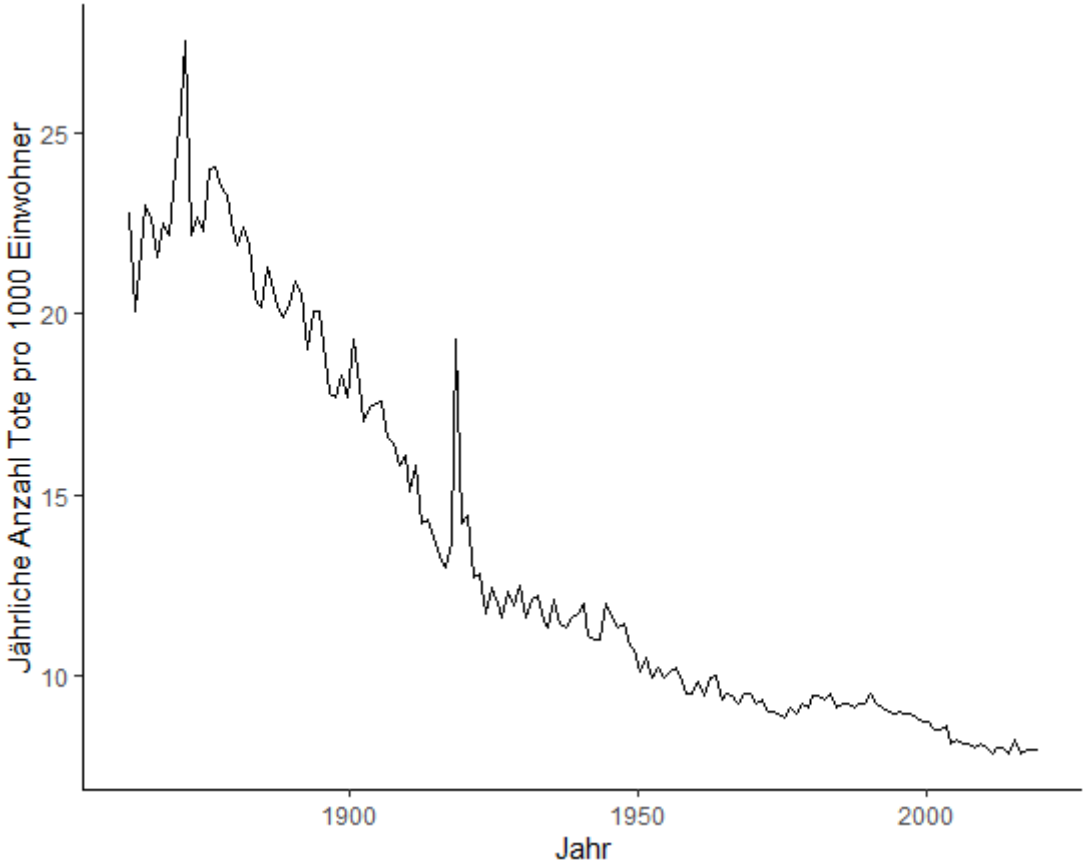
Die untensteheden R code Abschnitte erlauben über einfaches copy-paste das Runterladen der Datensätze, Rausfiltern und Darstellen der Daten die uns interessieren. Also schnell, kompakt und reproduzierbar (Stand 10.6.2021 - an stellen wo nur 1 Datenpunkt pro Jahr/Monat vorhanden ist, wählen wir ein arbiträrer Tag für die Darstellungen)

> library(tidyverse)
>
> BFS::bfs_get_dataset(url_px =
> "https://www.bfs.admin.ch/bfsstatic/dam/assets/16664926/master") %>%
> filter(demografisches_merkmal_und_indikator=='Todesfälle - Total') %>%
> mutate(jahr=lubridate::dmy(glue::glue('1506{jahr}'))) %>%
> filter(!is.na(value), !is.na(jahr)) %>%
> ggplot(aes(x=jahr, y=value)) + geom_line() + theme_classic() +
> labs(y='Jährliche Anzahl Tote')




```
bfs_get_dataset(url_px="https://www.bfs.admin.ch/bfsstatic/dam/assets/16664926/maste
%>% mutate(jahr=lubridate::dmy(glue::glue('1506{jahr}'))) %>%
dplyr::filter(demografisches_merkmal_und_indikator=='Todesfälle je 1000
Einwohner', !is.na(value)) %>%
ggplot(aes(x=jahr, y=value)) + geom_line() + theme_classic() + labs(x="Jahr",
y="Jährliche Anzahl Tote pro 1000 Einwohner")

BFS::bfs_get_dataset(url_px="https://www.bfs.admin.ch/bfsstatic/dam/assets/16664926/r
%>% mutate(Datum=case_when(
demografisches_merkmal_und_indikator=='Todesfälle im Januar' ~
lubridate::dmy(glue::glue('1501{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im Februar' ~
lubridate::dmy(glue::glue('1502{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im März' ~
lubridate::dmy(glue::glue('1503{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im April' ~
lubridate::dmy(glue::glue('1504{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im Mai' ~
lubridate::dmy(glue::glue('1505{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im Juni' ~
lubridate::dmy(glue::glue('1506{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im Juli' ~
lubridate::dmy(glue::glue('1507{jahr}')),
```

```
demografisches_merkmal_und_indikator=='Todesfälle im August' ~
lubridate::dmy(glue::glue('1508{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im September' ~
lubridate::dmy(glue::glue('1509{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im Oktober' ~
lubridate::dmy(glue::glue('1510{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im November' ~
lubridate::dmy(glue::glue('1511{jahr}')),
demografisches_merkmal_und_indikator=='Todesfälle im Dezember' ~
lubridate::dmy(glue::glue('1512{jahr}'))
)) %>% dplyr::filter(!is.na(value), !is.na(Datum)) %>%
ggplot(e3,aes(x=jahr, y=Anzahl_Todesfalle)) +
geom_point(alpha=0.3) + geom_line(alpha=0.4) + geom_smooth() +
theme_classic() + labs(x="Jahr", y='Monatliche Anzahl Todesfälle')
```

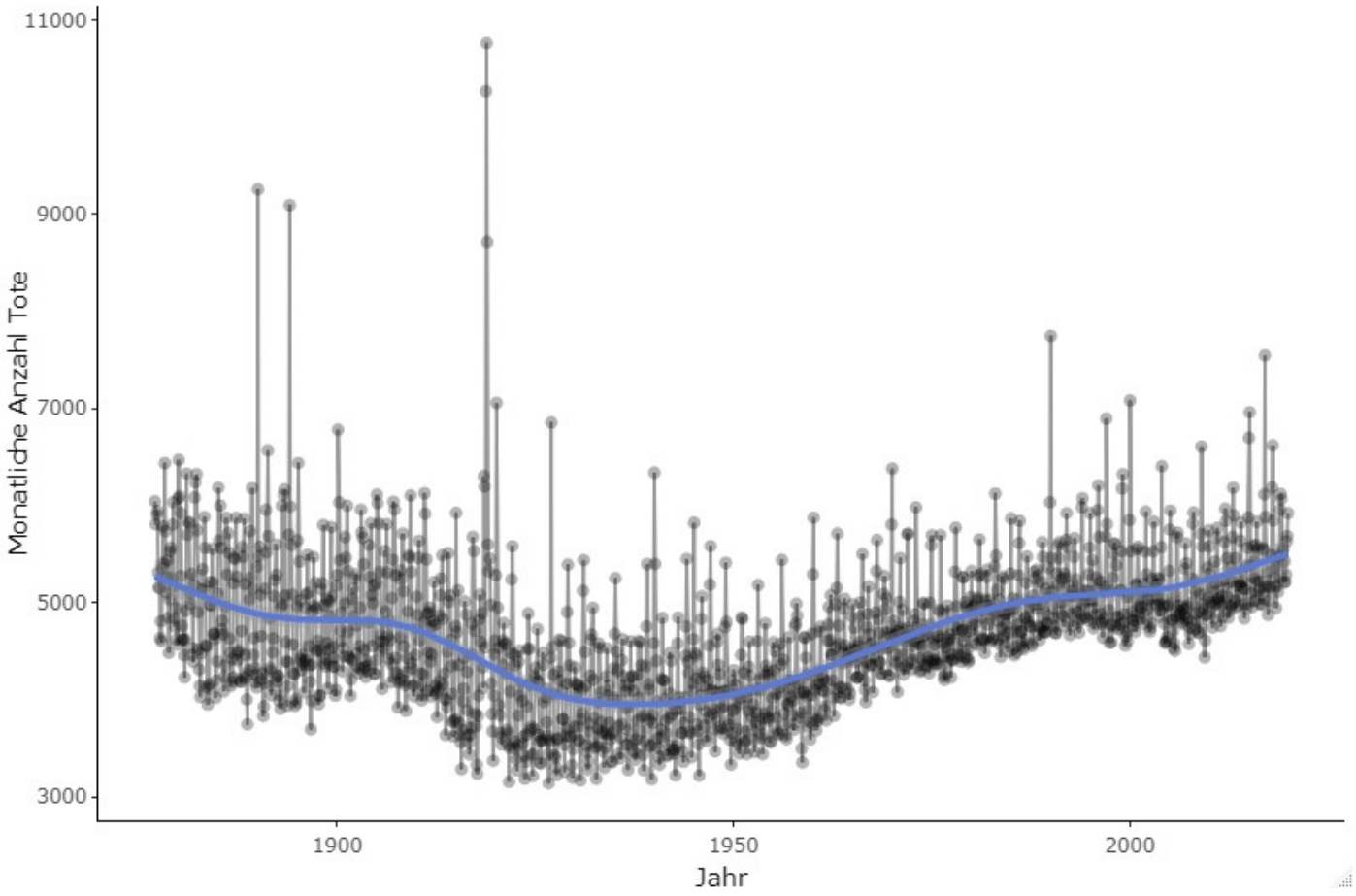


Obige Daten zeigen uns leider nur vergangene Jahr bis und mit 2019. Schauen wir uns nun aber noch die aktuellen Zahlen an. Diese können hier runtergeladen werden.

https://www.bfs.admin.ch/bfs/de/home/statistiken/gesundheit/gesundheitszustand/sterblichkeit-todesursachen.assetdetail.18664451.html

https://www.bfs.admin.ch/bfs/de/home/statistiken/gesundheit/gesundheitszustand/sterblichkeit-todesursachen.assetdetail.12607336.html

> dat = vroom::vroom('<Speicherort 1>') %>% dplyr::select(endend, Alter, AnzTF_HR) %>% dplyr::filter(!is.na(AnzTF_HR)) %>% dplyr::rename("Endend"="endend", "Anzahl_Todesfalle"="AnzTF_HR")
>
> vroom::vroom('<Speicherort 2>') %>% mutate(Anzahl_Todesfalle=as.numeric(Anzahl_Todesfalle)) %>% dplyr::filter(!is.na(Anzahl_Todesfalle)) %>% dplyr::select(Endend, Alter, Anzahl_Todesfalle) %>% rbind.data.frame(dat) %>% dplyr::filter(Anzahl_Todesfalle!='.') %>% mutate(date=as.POSIXct(lubridate::fast_strptime(Endend, format = '%d.%m.%Y')), Alter=as.factor(Alter), Anzahl_Todesfalle=as.numeric(Anzahl_Todesfalle), demografisches_merkmal_und_indikator= 'Todesfälle 2010-2021') %>% ggplot(e4, aes(x=jahr, y=Anzahl_Todesfalle, color=Alter)) + geom_line() + theme_classic() + labs(x="Jahr", y='Todesfälle 2010-2021')



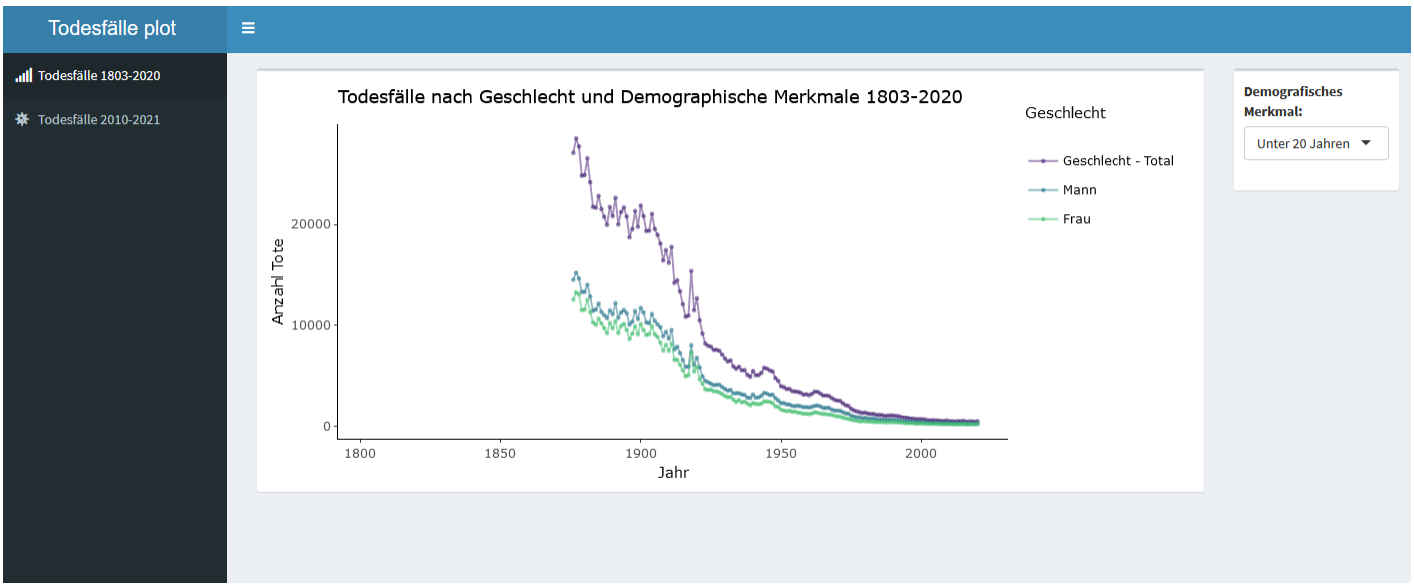
Kommentare oder Fragen sind über discuss.analysed.ch (open source software rocketchat) im Channel #covid19

jederzeit sehr willkommen.

Todeszahlen shiny app

Quellcode der Seite

shiny.analysed.ch/todesfaelle/




```
library(shiny)
library(shinydashboard)
library(plotly)
library(ggplot2)
library(dplyr)

todesfaelle_1803_2020 <- vroom::vroom('todesfaelle_1803_2020.csv') %>%
  mutate(geschlecht = parse_factor(geschlecht), demografisches_merkmal =
parse_factor(demografisches_merkmal))

todesfaelle_woechentlich <- vroom::vroom('tod_woechentlich.csv') %>%
  mutate(Alter = parse_factor(Alter))

ui <- dashboardPage(
  dashboardHeader(title = 'Todesfälle'),
    dashboardSidebar(
        sidebarMenu(
```

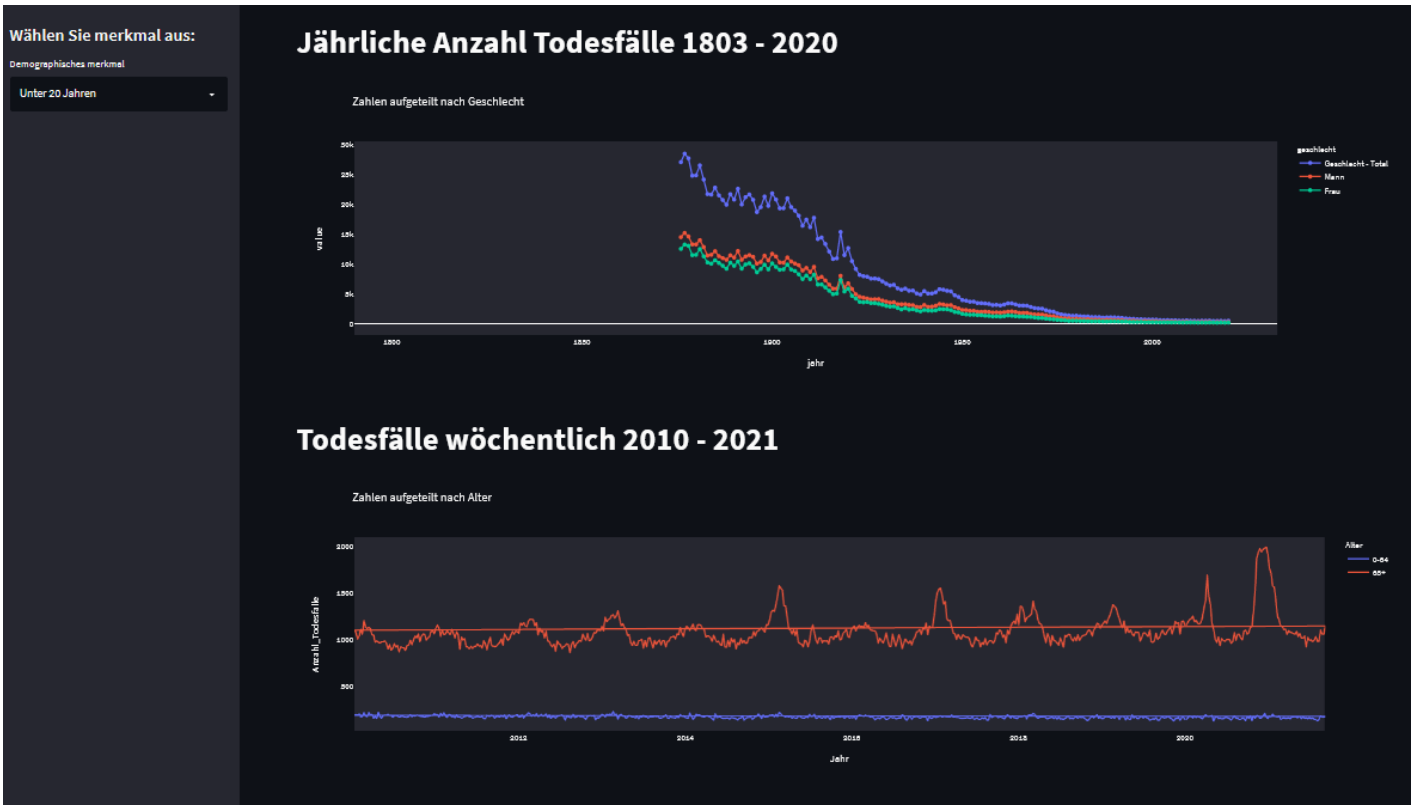
```
menuItem("Todesfälle 1803-2020", tabName = "todesfaelle", icon =
icon("signal")),
                menuItem("Todesfälle 2010-2021", tabName = "tod_woch", icon =
icon("virus"))
                )
                ),
             dashboardBody(
             tabItems(
            tabItem("todesfaelle",
                box(plotlyOutput("plot_1803_2020"), width = 10,
tags$a(href="https://blog.analysed.ch/books/analysedch-apps/page/todeszahlen-shiny-app",
"Quellcode dieser App")),
                    box(selectInput("demo_graph", "Demografisches Merkmal:", c( "Unter 20
Jahren", "20-39 Jahre", "40-64 Jahre", "65-79 Jahre", "80 Jahre und mehr", "Alter unbekannt",
"Schweizerische Staatsangehörigkeit", "Ausländische Staatsangehörigkeit", "Unbekannte
Staatsangehörigkeit", "Ledig", "Verheiratet", "Verwitwet", "Geschieden", "Unverheiratet", "In
eingetragener Partnerschaft", "Aufgelöste Partnerschaft", "Zivilstand unbekannt", "Todesfälle
- Total")),
                    width = 2)),
            tabItem("tod_woch",
                box(plotlyOutput("plot_2020_2021"), width = 12,
tags$a(href="https://blog.analysed.ch/books/analysedch-apps/page/todeszahlen-shiny-app",
"Quellcode dieser App"))
                    )
                )
             )
)

server <- function(input, output){
  output$plot_1803_2020 <- renderPlotly({
        ggplotly(ggplot(todesfaelle_1803_2020 %>% dplyr::filter(demografisches_merkmal ==
input$demo_graph), mapping = aes(x = jahr, y = value, color= geschlecht)) +
            geom_point(alpha=0.6, size=0.7)+geom_line(alpha=0.5)+
                labs(x='Jahr', y='Anzahl Tote', color="Geschlecht", title= "Todesfälle nach
Geschlecht und Demographische Merkmale 1803-2020")+
                scale_colour_viridis_d(begin = 0.1, end = 0.7)+ theme_classic())
  })
  output$plot_2020_2021 <- renderPlotly({
        ggplotly(ggplot(todesfaelle_woechentlich, mapping = aes(x = Jahr, y =
Anzahl_Todesfalle, color= Alter)) +
```

```
            geom_ribbon(aes(ymin = untGrenze, ymax = obeGrenze), alpha = 0.15) +
            geom_point(alpha=0.6, size=0.7) + geom_line(alpha=0.6) + ylim(0, 2000)
+
            labs(x='Jahr', y='Anzahl Tote', color = "Alter", title = "Altersgruppen und
Erwartungswerte 2010-2021")+
            scale_colour_viridis_d(begin = 0.1, end = 0.7)+ theme_classic())
            })
}

shinyApp(ui, server)
```

Todeszahlen streamlit python app

Quellcode der app

https://zahlen.analysed.ch/



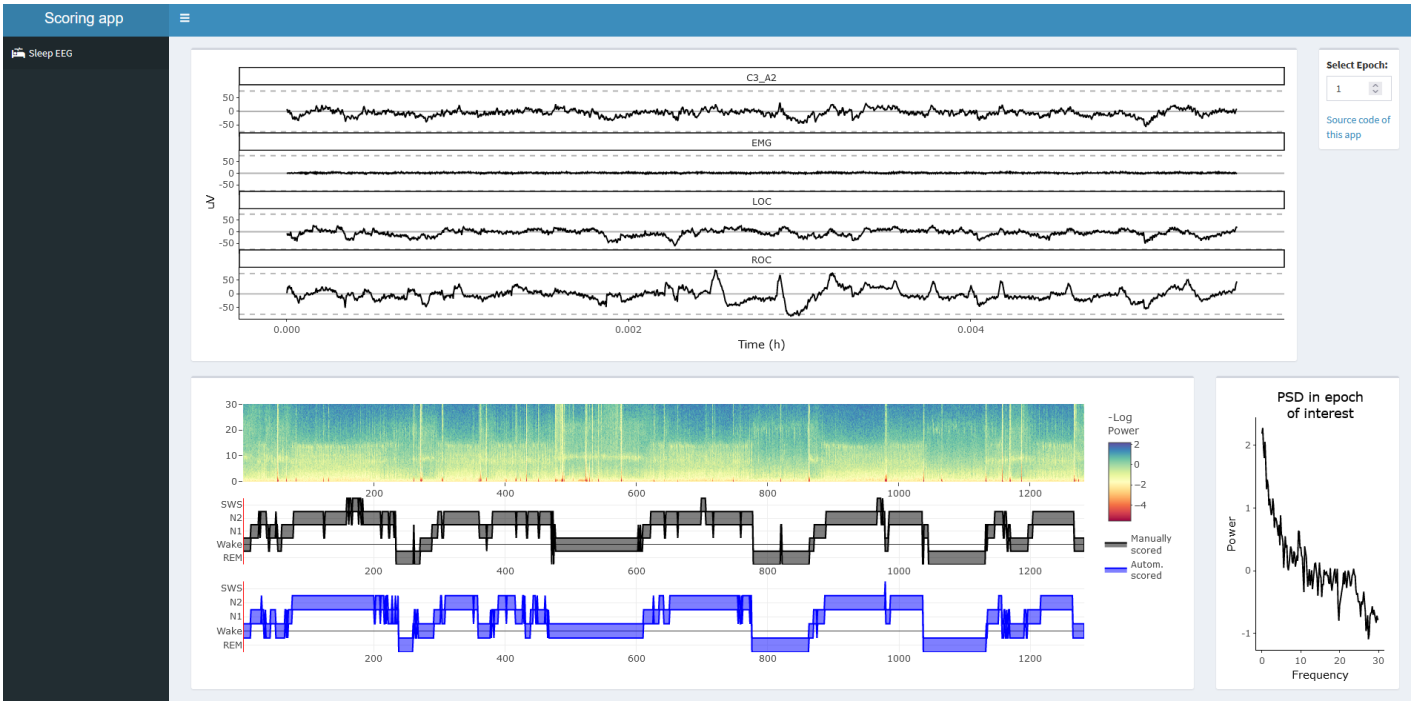

```
import streamlit as st
import pandas as pd
import plotly.express as px

# Breite einstellen
st.set_page_config(layout="wide")

# load data
df_1803_2020 = pd.read_table('tod_1803_2020.csv')
```

```
df_woechentlich = pd.read_table('tod_woechentlich.csv')

# Linke Spalte
dem_selectbox = st.sidebar.selectbox('Demographisches merkmal auswählen', ('Unter 20 Jahren',
'20-39 Jahre', '40-64 Jahre', '65-79 Jahre', '80 Jahre und mehr', 'Alter unbekannt',
'Schweizerische Staatsangehörigkeit',
                        'Ausländische Staatsangehörigkeit', 'Unbekannte
Staatsangehörigkeit', 'Ledig', 'Verheiratet', 'Verwitwet', 'Geschieden', 'Unverheiratet', 'In
eingetragener Partnerschaft', 'Aufgelöste Partnerschaft', 'Zivilstand unbekannt', 'Todesfälle
- Total'))

# Erste Darstellung
fig = px.line(df_1803_2020[df_1803_2020['demografisches_merkmal'] == dem_selectbox],
x='jahr', y='value', color='geschlecht', title='Zahlen aufgeteilt nach Geschlecht',
markers='true')
st.title('Jährliche Anzahl Todesfälle 1803 - 2020')
fig.update_layout(xaxis=dict(showgrid=False), yaxis=dict(showgrid=False))
st.plotly_chart(fig, use_container_width=True)



# Zweite Darstellung
fig2 = px.line(df_woechentlich, x='Jahr', y='Anzahl_Todesfalle', color='Alter', title='Zahlen
aufgeteilt nach Alter')
st.title('Todesfälle wöchentlich 2010 - 2021')
fig2.update_layout(xaxis=dict(showgrid=False), yaxis=dict(showgrid=False))
st.plotly_chart(fig2, use_container_width=True)
```

Automatic sleep stage scoring shiny app

Source code of the shiny app at

https://shiny.analysed.ch/scoring_app/



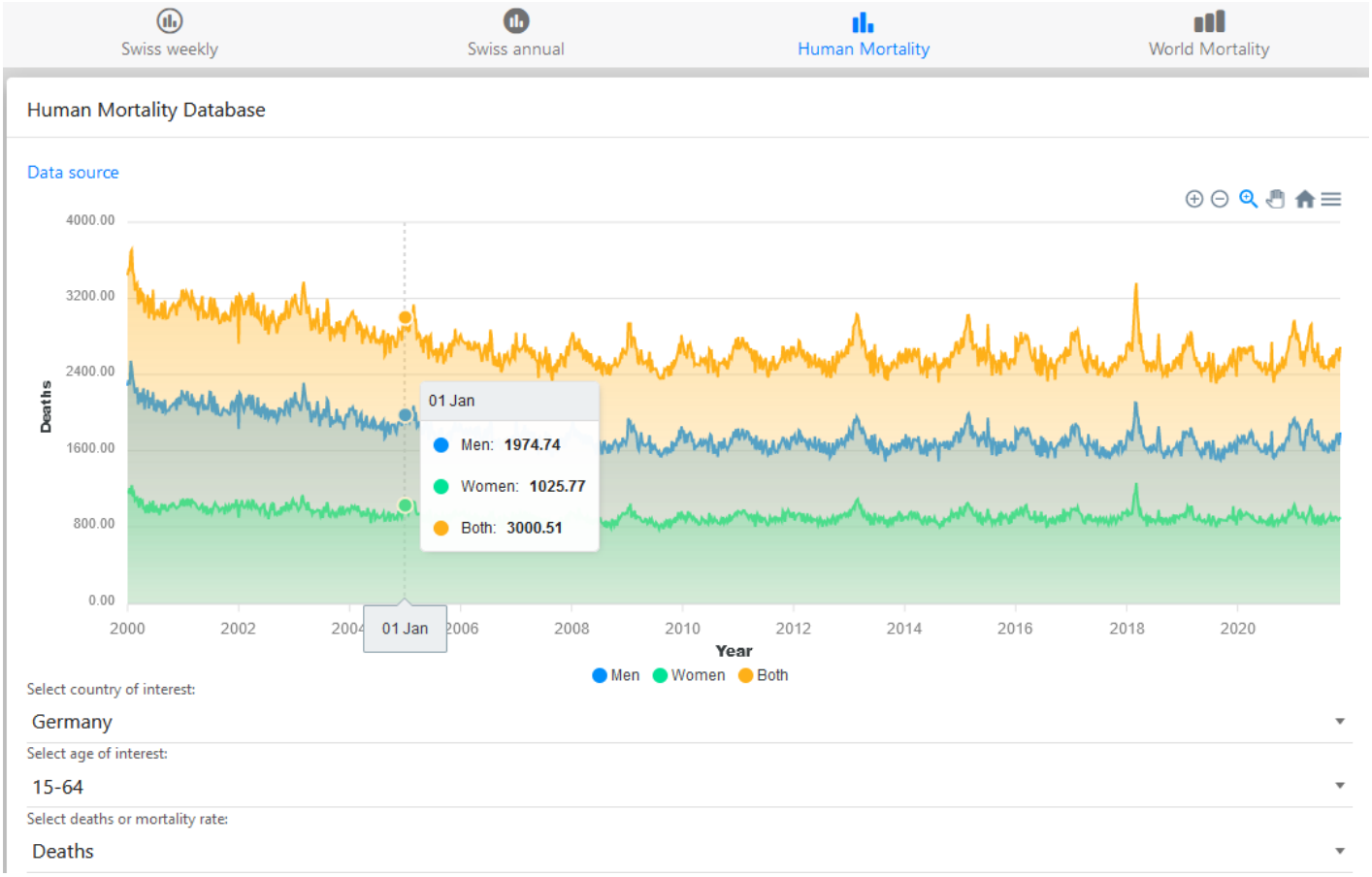

```
library(shiny)
library(shinydashboard)
library(plotly)
library(dplyr)

# Load spectra
spectra <- vroom::vroom('spectra.txt')
# stages
stages <- vroom::vroom('stages.txt')
# signal
signal <- vroom::vroom('signal.txt')
```

```
ui <- dashboardPage(
  dashboardHeader(title = 'Scoring app'),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Sleep EEG", tabName = "spe_tra", icon = icon("procedures"))
    )
  ),
  dashboardBody(
    tabItems(
      tabItem("spe_tra",
              box(plotlyOutput("eeg_plot"), width = 11),
              box(numericInput("Epoch_sl", "Select Epoch:", 1, min = 1, max = 1282), width =
1,
                  tags$a(href="https://blog.analysed.ch/books/analysedch-apps/page/automatic-
sleep-stage-scoring-shiny-app", "Source code of this app")),
              box(plotlyOutput("plot_old"), width = 10),
              box(plotlyOutput("plot_spec"), width = 2)
      )
    )
  )
)

server <- function(input, output){

  output$eeg_plot <- renderPlotly({
    ggplotly(ggplot(signal %>% dplyr::filter(Epoch == input$Epoch_sl), mapping = aes(x =
time, Amplitude)) +
               geom_line()+ facet_wrap(vars(Channel),nrow = 4)+ geom_hline(yintercept = 0,
alpha = 0.3)+
               geom_hline(yintercept = c(-75,75), lty = 7, alpha = 0.3)+
               labs(x = 'Time (h)', y = 'uV') +
               scale_colour_viridis_d(begin = 0.1, end = 0.7) + theme_classic())
  })
  output$plot_old <- renderPlotly({

    fig <- plot_ly(spectra, x = ~Epoch, y = ~Freq, z = ~Power, colors = "Spectral")
%>%
      add_heatmap() %>% colorbar(title = "-Log\nPower")
```

```
    fig1 <- plot_ly(stages, x = ~Epoch, name = 'Manually\nscored') %>%
      add_ribbons(ymin = ~man_scored - 0.5, ymax = ~man_scored + 0.5, color= I("black"))
%>%
      add_segments(x = input$Epoch_sl, xend = input$Epoch_sl, y = -1.5, yend = 3.5,
showlegend = FALSE, color=I('red'))  %>%
      plotly::layout(
        yaxis = list(
          range=c(-1.5,4),
          ticktext = list("REM", "Wake", "N1", "N2", "SWS"),
          tickvals = list(-1, 0, 1, 2, 3),
          tickmode = "array"
        ))

    fig2 <- plot_ly(stages, x = ~Epoch, name = 'Autom.\nscored') %>%
      add_ribbons(ymin = ~Stages - 0.5, ymax = ~Stages + 0.5, color = I("blue")) %>%
      add_segments(x = input$Epoch_sl, xend = input$Epoch_sl, y = -1.5, yend = 3.5,
showlegend = FALSE, color=I('red'))  %>%
      plotly::layout(
        yaxis = list(
          range=c(-1.5,4),
          ticktext = list("REM", "Wake", "N1", "N2", "SWS"),
          tickvals = list(-1, 0, 1, 2, 3),
          tickmode = "array"
        )
      )
    subplot(fig, fig1, fig2 , nrows = 3)
  })

  output$plot_spec <- renderPlotly({
    ggplotly(ggplot(spectra %>% dplyr::filter(Epoch == input$Epoch_sl), mapping = aes(x =
Freq, y=-Power)) +
               geom_line() + labs(x = 'Frequency', y = 'Power', title='PSD in epoch\nof
interest') +
               scale_colour_viridis_d(begin = 0.1, end = 0.7) + theme_classic() +
theme(plot.title = element_text(hjust = 0.5)))
  })
}

shinyApp(ui, server)
```

Mortality shinyMobile app

Source code of shinyMobile mortality app




```
library(shiny)
library(shinyMobile)
library(apexcharter)
library(dplyr)

w_path <- './'
deaths_y_swiss <- fst::read_fst(paste0(w_path,'deaths_1803_2020.fst'))
deaths_w_swiss <- fst::read_fst(paste0(w_path, 'deaths_weekly.fst'))
w_mortality <- fst::read_fst(paste0(w_path, 'worldmortality.fst'))
mortality <- fst::read_fst(paste0(w_path,'mortality.fst'))

ui <- f7Page(title = "Mortality over past years",options = list(theme="auto"),
             f7Tabs(animated = TRUE,
```

```
f7Tab(tabName = "Swiss weekly", icon = f7Icon("chart_bar_circle"), active =
TRUE,
                        f7Shadow(intensity = 10,hover =
TRUE,
                                 f7Card(title = "Deaths per week in Switzerland separated
in 2 age groups",
                                        f7Link(label = "Data source", href =
"https://www.bfs.admin.ch/bfs/en/home/statistics/health/state-health/mortality-causes-
death.assetdetail.12607336.html"),

apexchartOutput("weekplot"),
                                        f7Link(label = "Source code of this app", href =
"https://blog.analysed.ch/books/analysedch-apps/page/mortality-shinymobile-
app"))
                        )
                ),
                f7Tab(tabName = "Swiss annual", icon = f7Icon("chart_bar_circle_fill"),
active = FALSE,
                        f7Shadow(intensity = 10, hover =
TRUE,
                                 f7Card(title = "Deaths since 1803 separated by
sex",
                                        f7Link(label = "Data source", href =
"https://www.bfs.admin.ch/bfs/en/home/statistics/catalogues-
databases/data.assetdetail.17664404.html"),

apexchartOutput("yearplot"),
                                        f7Select(inputId = "swiss_y",label = "Demographic
feature:",
                                                 choices = c('Under 20 years', '20-39
years','40-64 years','65-79 years', '80 years or more','Age unknown','Swiss','Foreigner',
'Nationality unknown', 'Single','Married','Widowed', 'Divorced','Unmarried','In a registered
partnership', 'Partnership dissolved', 'Marital status unknown','Deaths - total'), selected =
'20-39 years')
                                 )
                        )
                ),
                f7Tab(tabName = "H. mortality data", icon = f7Icon("chart_bar_alt_fill"),
active = FALSE,
                        f7Shadow(intensity = 10,hover =
TRUE,
```

```
f7Card(title = "Human Mortality Database",
                                          f7Link(label = "Data source", href =
"https://www.mortality.org/"),

apexchartOutput("worldplot"),
                                          f7Select(inputId = "world_country", label = "Select
country of interest:",
                                                   choices = c("Australia", "Austria",
"Belgium", "Bulgaria", "Canada", "Chile", "Croatia", "Czech Republic", "Denmark", "England
and Wales", "Estonia", "Finland", "France", "Germany", "Greece", "Hungary", "Iceland",
"Israel", "Italy", "Latvia", "Lithuania", "Luxembourg", "Netherlands", "New Zealand",
"Northern Ireland", "Norway", "Poland", "Portugal", "Russia", "S. Korea", "Scotland",
"Slovakia", "Slovenia", "Spain", "Sweden", "Switzerland", "Taiwan", "USA"), selected =
"Germany"),
                                          f7Select(inputId = "world_age", label = "Select age
of interest:",
                                                   choices = c('0-14', '15-64', '65-74', '75-
84', '85+', 'Total'), selected = '15-64'),
                                          f7Select(inputId = "world_rate", label = "Select
deaths or mortality rate:",
                                                   choices = c('Deaths', 'Rate'), selected =
'Deaths')
                                    )
                          )
                    ),
                    f7Tab(tabName = "W. mortality data", icon = f7Icon("chart_bar_fill"),
active = FALSE,
                          f7Shadow(intensity = 10, hover =
TRUE,
                                   f7Card(title = "World Mortality
Dataset",
                                          f7Link(label = "Data source", href =
"https://github.com/akarlinsky/world_mortality"),

apexchartOutput("world_mortality"),
                                          f7Select(inputId = "country_n",label = "Select
country:",
                                                   choices = c("Albania", "Andorra", "Antigua
and Barbuda", "Argentina", "Armenia", "Aruba","Australia", "Austria", "Azerbaijan",
"Belarus", "Belgium", "Belize", "Bermuda","Bolivia", "Bosnia", "Brazil", "Bulgaria",
"Canada","Chile","Colombia","Costa Rica","Croatia", "Cuba", "Cyprus", "Czechia", "Denmark",
```

```
"Dominican Republic", "Ecuador","Egypt", "El Salvador", "Estonia","Faroe
Islands","Finland","France","French Guiana","French
Polynesia","Georgia","Germany","Gibraltar","Greece","Greenland","Guadeloupe","Guatemala","Hong
Kong","Hungary","Iceland","Iran","Ireland","Israel","Italy","Jamaica","Japan","Kazakhstan","Kos
Caledonia","New Zealand","Nicaragua","North
Macedonia","Norway","Oman","Panama","Paraguay","Peru","Philippines","Poland","Portugal","Puerto
Rico","Qatar","Réunion","Romania","Russia","San
Marino","Serbia","Seychelles","Singapore","Slovakia","Slovenia","South Africa","South
Korea","Spain","Sweden","Switzerland","Taiwan","Tajikistan","Thailand","Transnistria","Tunisia",
Kingdom","United States","Uruguay","Uzbekistan"), selected =
'Switzerland')
                                    )
                            )
                    )
            )

)

server = function(input, output, session) {

  output$weekplot <- renderApexchart({apex(data =
deaths_w_swiss,
                                         type = "area", mapping = aes(x = Year, y = Deaths,
fill = Age)) %>%
      ax_yaxis(title = list(text = "Deaths")) %>%
      ax_xaxis(title = list(text = "Year"))})

  output$yearplot <- renderApexchart({apex(data = deaths_y_swiss %>% filter(demo_var ==
input$swiss_y),
                                         type = "area", mapping = aes(x = year, y = deaths,
fill = sex)) %>%
      ax_yaxis(title = list(text = "Deaths")) %>%
      ax_xaxis(title = list(text =  "Year"))})

  output$worldplot <- renderApexchart({
    apex(data = mortality %>% filter(CountryCode == input$world_country, Age==
input$world_age, Rate==input$world_rate),
         type = "area", mapping = aes(x = Year, y = Deaths, fill = Sex)) %>%
      ax_yaxis(title = list(text = "Deaths")) %>%
      ax_xaxis(title = list(text = "Year"))})
```

```
  output$world_mortality <- renderApexchart({apex(data = w_mortality %>% filter(country_name
== input$country_n),
                                                type = "area", mapping = aes(x = date, y =
deaths)) %>%
      ax_yaxis(title = list(text = "Deaths")) %>%
      ax_xaxis(title = list(text =  "Year"))})

}



shinyApp(ui, server)
```