How to make different datasets with {peacesciencer}

First, open the {tidyverse} and {peacesciencer}.

```
library(tidyverse)
library(peacesciencer)
```

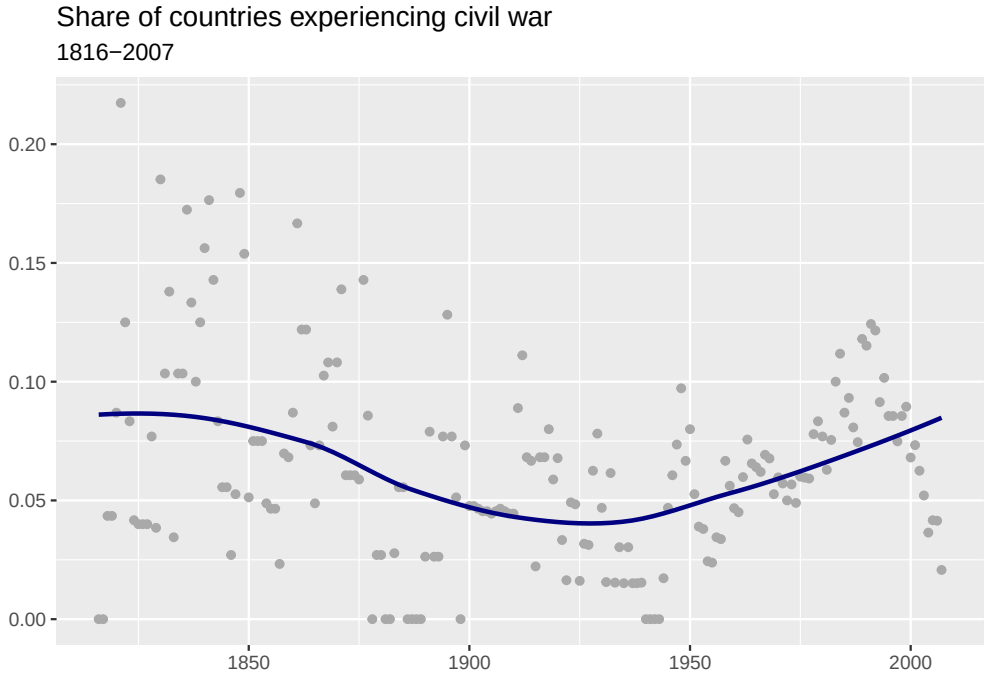
Country-year datasets

Depending on what data you want to look at, making a country-year dataset is pretty straightforward. A common example I work with in class is a country-year dataset of civil wars, plus some economic and quality of democracy data. Here's how I'd make it:

```
create_stateyears(subset_years = 1816:2007) |>
  add_cow_wars(type = "intra") |>
  add_sdp_gdp() |>
  add_democracy() -> Data
```

You can then use it to talk about average trends, say for civil war:

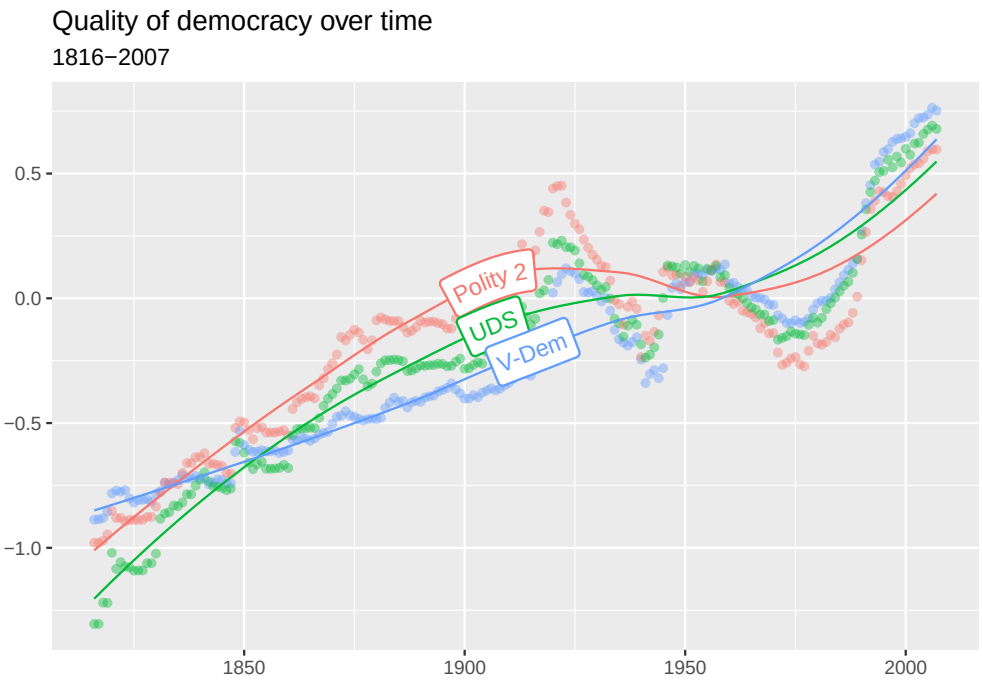
```
Data |>
  group_by(year) |>
  summarize(
    conflict_rate = mean(cowintraongoing, na.rm = T)
  ) |>
  ggplot() +
  aes(x = year, y = conflict_rate) +
  geom_point(color = "darkgray") +
  geom_smooth(color = "navy", se = F) +
  labs(
    x = NULL,
    y = NULL,
    title = "Share of countries experiencing civil war",
    subtitle = "1816-2007"
  )
```



Or for democracy:

```
Data |>
  mutate(
    across(v2x_polyarchy:xm_qudsest, scale)
  ) |>
  group_by(year) |>
  summarize(
    across(v2x_polyarchy:xm_qudsest, ~ mean(.x, na.rm = T))
  ) |>
  pivot_longer(-year) |>
  mutate(
    name = case_when(
      name == "v2x_polyarchy" ~ "V-Dem",
      name == "polity2" ~ "Polity 2",
      name == "xm_qudsest" ~ "UDS"
    )
  ) |>
  ggplot() +
  aes(x = year, y = value, color = name) +
  geom_point(alpha = 0.4) +
  geomtextpath::geom_labelsmooth(
    aes(label = name)
  ) +
  labs(
    x = NULL,
    y = NULL,
    title = "Quality of democracy over time",
    subtitle = "1816-2007"
  ) +
  theme(
    legend.position = "none"
```

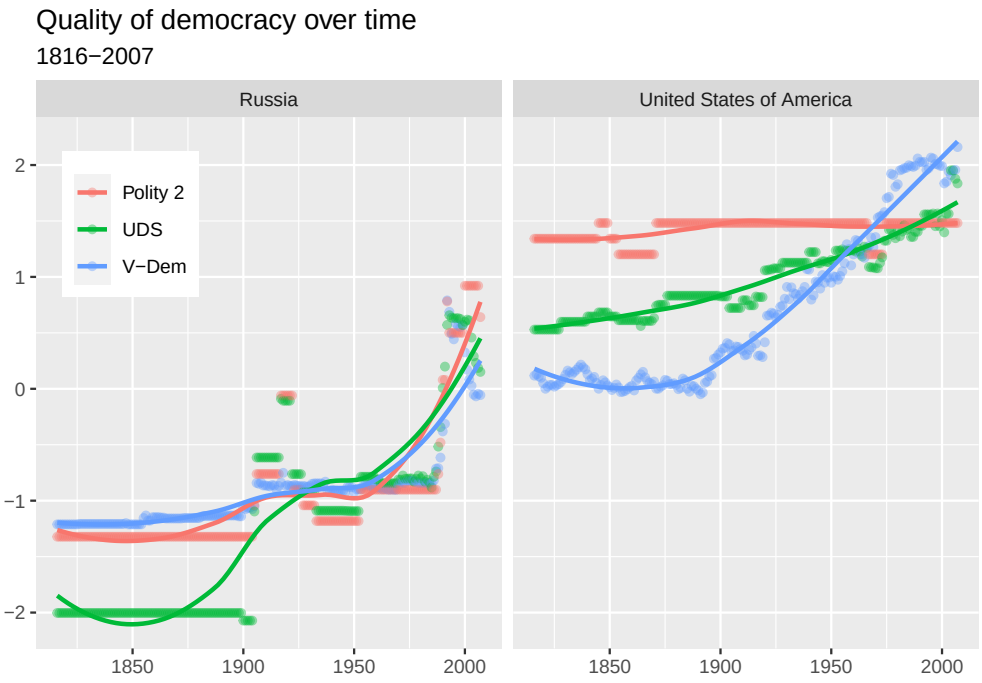
```
)
```



I might also zero in on one or two cases, like the US and Russia:

```
Data |>
  mutate(
    across(v2x_polyarchy:xm_qudsest, scale)
  ) |>
  filter(
    statenme %in% c("United States of America", "Russia")
  ) |>
  pivot_longer(v2x_polyarchy:xm_qudsest) |>
  mutate(
    name = case_when(
      name == "v2x_polyarchy" ~ "V-Dem",
      name == "polity2" ~ "Polity 2",
      name == "xm_qudsest" ~ "UDS"
    )
  ) |>
  ggplot() +
  aes(x = year, y = value, color = name) +
  geom_point(alpha = 0.4) +
  geom_smooth(
    se = F
  ) +
  labs(
    x = NULL,
    y = NULL,
    title = "Quality of democracy over time",
    subtitle = "1816-2007",
    color = NULL
  ) +
```

```
  facet_wrap(~ statenme) +
  theme(
    legend.position = c(0.1, 0.8)
  )
```



From dyad-year to country-year

One annoying thing about {peacesciencer} is that some variables (like international war), require you to use dyadic data. But with a few extra steps, it's easy to aggregate to the state-year level.

First, create some dyadic international conflict data. By default, the dyads are directed, which is what we want. That ensures that each country in the data appears on both the left and right side of a country pairing.

```
create_dyadyears(subset_years = 1816:2007) |>
  add_cow_wars(type = "inter") |>
  add_sdp_gdp() |>
  add_democracy() -> Data
```

Now that we have the dyadic data, we can aggregate to the level of individual countries like so. We'll keep the year and indicators of war onset and whether a war is still ongoing. And we'l also keep columns for side 1.

```
Data |>
  select(ccode1, year, cowinteronset, cowinterongoing, contains("1")) |>
  distinct() -> cy_Data
```

Then, let's get rid of those pesky "1"s in the variable names and also throw in a character string version of country names:

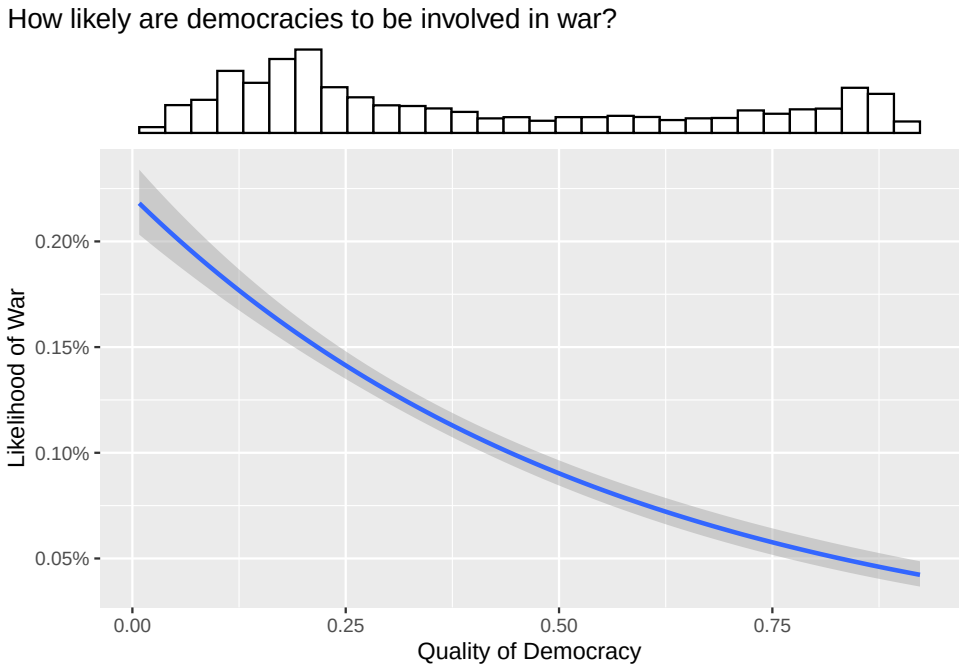
```
cy_Data |>
  rename_all(~ str_remove(.x, "1")) |>
  mutate(
    statenme = countrycode::countrycode(
      ccode, "cown", "country.name"
```

```
    )
  ) |>
  select(
    statenme,
    everything()
  ) -> cy_Data
```

With the data, you can check things like, do democracies go to war more frequently?

```
library(patchwork)
ggplot(cy_Data) +
  aes(x = v2x_polyarchy, y = cowinterongoing) +
  stat_smooth(
    method = "glm",
    method.args = list(family = binomial)
  ) +
  scale_y_continuous(
    labels = scales::percent
  ) +
  labs(
    x = "Quality of Democracy",
    y = "Likelihood of War"
  )-> main_plot
ggplot(cy_Data) +
  aes(x = v2x_polyarchy) +
  geom_histogram(
    color = "black",
    fill = "white"
  ) +
  theme_void() -> dens_plot

dens_plot / main_plot +
  plot_layout(
    heights = c(1, 5)
  ) +
  plot_annotation(
    title = "How likely are democracies to be involved in war?"
  )
```



On weird thing to note, if you decide after you've aggregated the data that you want to add additional variables using one of the `add_*()` functions from `{peacesciencer}`, you'll have to update one of the attributes of the data. When you create a dyad-year dataset, it gets assigned a "ps_data_type" of "dyad_year." That means all the `add_*()` functions expect to see `ccode1` and `ccode2` in the data for joining the additional datasets. Since the data only has `ccode`, the the functions to add different variables won't work unless we change "ps_data_type" to "state_year".

You can fix this by changing this attribute manually with `data.table::setattr()`.

```
data.table::setattr(cy_Data, "ps_data_type", "state_year")
```

And now, if we want to add some additional variables to the data, we can do so:

```
cy_Data <- cy_Data |>
  ## national military capabilities:
  add_nmc() |>
  ## are countries "major powers"?
  add_cow_majors()
```

Bringing in leader attributes

There are a few functions that you can use to bring in information about political leaders. Some of these will only work if the data is at the leader-year level however. Here's an example of what to do if you want to talk about leader experience, gender, age, education, past military service, physical and mental health, and willingness to use force:

```
create_leaderyears() |>
  add_lead() |>
  add_lwuf() -> lead_data
```

War level data

Things get more complicated if you want to create a dataset at the level of individual wars. Instead of having students make this kind of data, I just spoon feed it to them and make the code I used to make it available for them to check out if they're interested.

For example, here's the code I used to create the data for MA2:

```
## create a dyad-year dataset
Data <- create_dyadyears() |>
  ## add conflict dataset
  left_join(cow_war_inter) |>
  ## add democracy data
  add_democracy() |>
  ## add capabilities
  add_nmc()

## rescale the democracy measures
Data |>
  mutate(
    across(v2x_polyarchy1:xm_qudsest2, scale)
  ) -> Data

## create measures of weakest democracy
Data |>
  mutate(
    min_polyarchy = ifelse(
      v2x_polyarchy1 < v2x_polyarchy2,
      v2x_polyarchy1, v2x_polyarchy2
    ),
    min_polity2 = ifelse(
      polity21 < polity22,
      polity21, polity22
    ),
    min_qudsest = ifelse(
      xm_qudsest1 < xm_qudsest2,
      xm_qudsest1, xm_qudsest2
    )
  ) -> Data

## make function to combine country names into a list
clist <- function(x) {
  x <- countrycode::countrycode(x, "cown", "country.name")
  paste0(unique(x), collapse = ", ")
}

## aggregate the data to individual wars
Data |> filter(cowinterongoing == 1) |>
  group_by(warnum) |>
  summarize(
    start_year = min(year),
    duration = max(year) - min(year) + 1,
    side1 = clist(ccode1[sidea1==1]),
    side2 = clist(ccode1[sidea1==2]),
    side1_initiator = max(
```

```
      initiator1[sidea1==1], na.rm = T
    ),
    side2_initiator = max(
      initiator1[sidea1==2], na.rm = T
    ),
    side1_winner = min(
      outcome1[sidea1==1]==1, na.rm = T
    ),
    side2_winner = min(
      outcome1[sidea1==2]==1, na.rm = T
    ),
    side1_deaths = sum(
      batdeath1[sidea1==1 & batdeath1 > 0], na.rm = T
    ),
    side2_deaths = sum(
      batdeath1[sidea1==2 & batdeath1 > 0], na.rm = T
    ),
    side1_military_expenditures = sum(
      milex1[sidea1==1], na.rm = T
    ),
    side2_military_expenditures = sum(
      milex1[sidea1==2], na.rm = T
    ),
    side1_military_personnel = sum(
      milper1[sidea1==1], na.rm = T
    ),
    side2_military_personnel = sum(
      milper1[sidea1==2], na.rm = T
    ),
    side1_polyarchy = mean(
      v2x_polyarchy1[sidea1==1], na.rm = T
    ),
    side2_polyarchy = mean(
      v2x_polyarchy1[sidea1==2], na.rm = T
    ),
    side1_polity2 = mean(
      polity21[sidea1==1], na.rm = T
    ),
    side2_polity2 = mean(
      polity21[sidea1==2], na.rm = T
    ),
    side1_qudsest = mean(
      xm_qudsest1[sidea1==1], na.rm = T
    ),
    side2_qudsest = mean(
      xm_qudsest1[sidea1==2], na.rm = T
    ),
    min_polyarchy = min(v2x_polyarchy1, na.rm = T),
    min_polity2 = min(polity21, na.rm = T),
    min_qudsest = min(xm_qudsest1, na.rm = T)
  ) -> sm_data

## deal with pesky infinite values
```

```r
apply(sm_data, c(1, 2), function(x) ifelse(is.infinite(x), NA_real_, x)) |>
  as_tibble() -> sm_data
```