Introduction

This data feature tackles a recurring problem for researchers in the peace science community. True research reproducibility is best achieved creating data from scratch, though no published guide exists that informs researchers how to do this on their own. Instead, researchers may end up reusing old code generated in past studies, leaving them to spend time and energy adjusting the sample of states, the temporal domain, and doing whatever additional troubleshooting may arise from this practice. Researchers may additionally spend too much time reproducing old code for standard information that goes into any dyadic or monadic analysis—like contiguity relationships and democracy—and have to do additional troubleshooting for how these various data sources treat missing data or treat state codes in a manner inconsistent with the more accessible Correlates of War or Gleditsch-Ward state codes. This is all compounded by changes in technology that treat the creation of the data and the analysis of data as a continuous process in which the contemporary quantitative political scientist is increasingly becoming a computer programmer as well. Graduate students, in particular, face unique challenges associated with these developments. Graduate students beginning their careers in peace science must learn how scholarship informs data in peace science and how data inform scholarship at the same time they are needing to learn quantitative methods in a chosen software package.

`{peacesciencer}` is designed to address these problems. Built around the free and open source R programming language, `{peacesciencer}` contains a suite of data and functions for creating data of interest to researchers. Researchers can use `{peacesciencer}` to create dyad-year, leader-year, leader-dyad-year, and state-year data (among some others) from scratch. Afterward, they can add a variety of standard information (e.g. contiguity, alliances, major power status, GDP per capita estimates, capability estimates, and more) to these data with a simple command. This is a considerable time-saver since, in the absence of it, researchers would have to more meticulously code and transform the raw data to conform to the kind of data they want. `{peacesciencer}` comes with some data innovations as well, including a comprehensive data set on democracy by year, an original data set on capitals and capital transitions, and a function to create peace years between ongoing conflicts. All are done with the maximum possible transparency. The project is available for public view of Github ([LINK REDACTED FOR PEER REVIEW]). The `data-raw` directory on the project's Github contains information and comments about how every data set was created and the `R` directory informs the user how each function works. The function manuals ([LINK REDACTED FOR PEER REVIEW]) contain additional comments about what each function returns and, in appropriate cases, why it is doing what it is doing. Thus,

{peacesciencer} not only assists a peace scientist with their research, but it does so in a manner that best conforms to the Data Access and Research Transparency Initiative (DA-RT) initiative across all political science.

This data feature proceeds in the following fashion. The next section expands what need this package fills for peace scientists in the present. Afterward, it provides an overview of what is included in {peacesciencer} to help researchers more quickly conduct the kind of quantitative research they want. Thereafter, it provides a tutorial on how to install and best use {peacesciencer} in the R programming language. A more comprehensive tutorial follows, showing how {peacesciencer} already has a suite of data and functions that can allow for effective replications of a "dangerous dyads" type analysis (Bremer, 1992) and standard state-year analyses of civil conflict onset (e.g. Fearon & Laitin, 2003). This feature concludes with a discussion of what else {peacesciencer} is capable of doing in future updates and how this R package can inform more reasoned design decisions for researchers in peace science.

Why {peacesciencer}?

{peacesciencer} is not the only software available to peace science researchers who want to reduce the time and energy required to faithfully recreate data from scratch. NewGene, for example, is a stand-alone software program for Microsoft Windows and Mac that can create various types of data of interest to international relations scholars (Bennett, Poast & Stam, 2019). NewGene is itself the evolution of EUGene, which served conflict researchers well for over a decade (Bennett & Stam, 2000). No matter, {peacesciencer} is motivated by the following ideals that led to its creation.

For one, researchers invest too much time in the construction of a data set that faithfully captures the unit of analysis. Assume a researcher wants an original data set on all directed dyad-years for Correlates of War states for an analysis of inter-state conflict. How might one do that? The answer has never been immediately obvious. No published guide exists that shows a researcher how to create these data themselves from scratch, which is one reason why software bundles like EUGene and NewGene are attractive to researchers who primarily care about the substance of their research question. After all, EUGene's main value—if not its primary impetus—was allowing researchers to create data sets for replication of previous studies, producing a host of data types (e.g. dyad-year, state-year, dispute-year) along the way that users can amend as they saw fit. {peacesciencer} is primarily born from this question about how to create these data from scratch. The underlying code that produces these data types is available online and {peacesciencer} converts these lines of code into simple

functions for the ease of the researcher.

Second, researchers also invest too much time in retracing steps for peace science analyses for new projects. Assume a researcher finished a state-year analysis on the correlates of civil conflict onset a few years ago and wants to start a new project that analyzes the same outcome from a different angle (or perhaps using newer data). Under these conditions, a researcher will have to find where they stored that replication code and copy-paste it into a new directory for the new project. They may then have to change the name of some files, change some code to account for potentially new column names in the newer data, and troubleshoot instances where their old code does not perform as it once did. At its worst, this process may lead to some errors by the researcher. At best, this is tedium that spends the researcher's time they would rather invest in analyzing the data. The lion's share of `{peacesciencer}`'s functionality is both creating the units of analysis for the researcher and merging in different forms of data in wide use in the peace science community so that the researcher can spend less of their time on tedium.

Third, the creation of the data and the analysis of the data are increasingly becoming one continuous process. Not too long ago, it used to be the case that researchers had to download a data set, or create one from scratch (possibly in a spreadsheet or through a program like EUGene). After downloading or constructing the data, the researcher then opened a specialty program for statistical analysis (e.g. SAS, SPSS, Stata) to recode raw data into a form suitable for analysis before running a statistical model that regresses some outcome on a set of covariates. Current research practices still resemble this process, but the steps between them are no longer as large as they were in the past. Software options exist that allow the researcher to load data, create data, clean data, analyze data, and present the results of the analysis all within one program. `{peacesciencer}`, by itself, does not do all these things, but it seamlessly connects the beginning of the research process to the end of the research process without needing to leave the increasingly popular R programming language and Rstudio (its free-for-use integrated development environment [IDE]).

Fourth, it is increasingly the case that as the steps between creating data and analyzing data decrease in size, the lines between them blur as well. In other words, to create data is to code data and the contemporary quantitative political scientist is increasingly becoming a computer programmer (c.f. Bowers & Voors, 2016). This is happening concurrent to innovations in programming languages for statistical analysis, especially the R programming language that `{peacesciencer}` uses. There have been significant advances in add-on packages that allow users to do things like get World Bank data from the internet (Arel-Bundock, 2021a), scrape *any* type of data from the internet (Boehmke, 2016), and even format results from a statistical model for pre-

sentation in a way the reduces the probability of transcription errors to almost zero (Arel-Bundock, 2021b). {peacesciencer} embraces this. This R package reduces the time required to create peace science data for analysis and also informs the user about the code required to create the kind of data the user wants.

Finally, the creation and presentation of data in peace science should be 100% robust and transparent, which {peacesciencer} takes seriously in the following ways. The website for {peacesciencer} has several vignettes that describe its processes in some detail. These include how it provides reasonable estimates of democracy that may not be available in the Polity data or the Varieties of Democracy data and how a researcher can whittle dyadic dispute-years into true dyad-years through reasonable case exclusions. {peacesciencer} subjects itself to a battery of tests before publishing updates, making sure new features do not create duplicate entries in the original data (which is the surest sign of a botched merge). The project's Github contains a publicly available `data-raw` directory that shows how every data set included (and processed) in {peacesciencer} was created. The function manuals included {peacesciencer} contain ample documentation that clarifies what the function is doing, what it returns to the user, and why it is doing it this way. Researchers can also use the project's Github to point out bugs, ask for further clarification, and propose additions. {peacesciencer} takes seriously the Data Access and Research Transparency Initiative (DA-RT) initiative across all political science and endeavors for maximum transparency, leveraging open source and version control software to inform users of what data it uses and how it uses the data.

What is Included in {peacesciencer}

{peacesciencer} comes with a fully developed suite of built-in functions for generating some of the most widespread forms of peace science data and populating the data with important variables that recur in many quantitative analyses. The core functionality of {peacesciencer} reduces to two broad categories of functions. These categories are functions that create the base data of interest to a researcher and functions, called after the base data are created, that add variables of interest to the data frame or subset the base data to a handful of rows that the researcher deems appropriate for analysis. Table 1 and Table 2 list these core functions as of version 0.7, the version of this package slated for release alongside the manuscript.

Table 1 are functions that crease base data frames for a researcher starting an original project. They serve as functions that communicate the units of analysis supported in this package and that the package is capable of generating for an interested user. For example, `create_stateyears()` will generate the full universe of

state-years from the Correlates of War (Correlates of War, 2011: v. 2016) or Gleditsch-Ward (Gleditsch & Ward, 1999: v. 2017) system, encompassing all state-years to the most recently concluded calendar year, depending on the arguments supplied to the user in the function. `create_leaderyears()` will generate the full universe of leader-years from the Archigos leader data (Goemans, Gleditsch & Chiozza, 2009: v. 4.1), optionally standardizing leader-years to the Gleditsch-Ward or Correlates of War state system data. As of version 0.7, `{peacesciencer}` is capable of creating full dyad-year data, leader-day data, leader-dyad-year data, leader-year data, state-day data, and state-year data.[1] A vignette on the package's website shows how users can create other forms of data from these functions as well (e.g. leader-months, state-quarters).

****Table 1 about here****

Table 2 lists the main functions in `{peacesciencer}` that add information or subset the number of rows of the data to just those of interest to the user, describing these functions and listing whether they are applicable to dyad-year (`D`), leader-year (`L`), leader-dyad-year (`LD`), state-year (`S`) data or specialty functions applicable to just the dyadic conflict data (`C`).[2] All these functions use raw or pre-processed data included in the package. For example, `add_gml_mids()` uses a dyadic dispute-year version of the MID data offered by Gibler et al. (2016: v. 2.2.1) and merges in information about whether there was an ongoing MID or MID onset in a dyad-year, leader-year, leader-dyad-year, or state-year. `{peacesciencer}` also has some data innovations included in these functions. For example, `add_capital_distance()` calculates distance between state capitals in kilometers using the Vincenty method (i.e. "as the crow flies") based on an original data set of state capitals that accounts for instances when capitals moved (e.g. Brazil in 1960, Burundi in 2018). `add_democracy()` does more than just add Polity data to a data set. The data underpinning the function feature an innovation in democracy data, providing reasonable estimates of democracy using the Marquez (2016) method of extending the Unified Democracy Scores (UDS) data (Pemstein, Meserve & Melton, 2010) in addition to Polity estimates (Marshall, Gurr & Jaggers, 2017: v. 2017) and V-Dem estimates (Coppedge et al., 2020: v. 10).[3]

****Table 2 about here****

`{peacesciencer}`'s coverage focuses mostly on data that are released as standalone data sets for download,

[1] Leader-dyad-year data are too time-consuming to calculate each time and too size-prohibitive to include in the package. These data can be downloaded through `download_extdata()` for much easier use.

[2] The "whittle" (`wc_`) class of functions are a suite of functions that whittle dyadic dispute-years to true dyad-year data. They are effectively used in `add_cow_mids()` and `add_gml_mids()` for dyad-year data, but are offered here to allow users to employ their own case exclusion rules to create their own dyad-year conflict data. A vignette on `{peacesciencer}`'s website explains these case exclusion rules in some detail.

[3] `{peacesciencer}` comes with a vignette that explains just how much unnecessary missingness pervades data on democracy in the absence of an innovation like this, and how this missingness might adversely affect inferences of inter-state or intra-state conflict.

especially those in the Correlates of War or Gleditsch-Ward ecosystem of data. Data that can be obtained from a stable advanced programming interface—like the World Bank, for example—can be obtained through those other means (e.g. Arel-Bundock, 2021a). Its coverage will assuredly expand with new additions of interest to the peace science community, though the package already offers a lot to meet researcher needs.

How to Install `{peacesciencer}`

`{peacesciencer}` is a package for the R programming language. This assumes at least some familiarity with the R programming language. Users should have at least version 3.5 of R, which should not be an issue since the most recent version—as of writing—is 4.1.2. `{peacesciencer}` is designed to be as user-friendly as possible. Those proficient in R, those just learning R, and those with no experience in R should be able to pick up its use fairly quickly.

`{peacesciencer}`'s functions work out of the box, though users should find their experience augmented by two additional downloads. First, RStudio offers an IDE that serves as a user-friendly graphical user interface (GUI) over what is, at its core, a programming language with a command-line interface. Rstudio's design will make it much easier for users to experiment with `{peacesciencer}`'s functionality and read its documentation to assist them with the user of these functions. The second additional download is `{tidyverse}`, itself a suit of packages that share a common form and design (Wickham et al., 2019). `{peacesciencer}` functions make considerable use of the component packages of `{tidyverse}` but installing and loading `{tidyverse}` will allow the researcher to make quicker use of `{peacesciencer}`'s functionality. A user can open an R session by way of Rstudio and install both packages as follows.

```
install.packages(c("tidyverse", "peacesciencer"))
```

R packages once installed need to be loaded with every R session (i.e. every time the user opens RStudio). The user can load both with the `library()` function in R.

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

Thereafter, a researcher can start using `{peacesciencer}` to create the kind of data they need.

A Tutorial on How to Use {peacesciencer}

I encourage users who are using {peacesciencer} for the first time, especially those who are learning R for the first time because of their interest in this package, to approach the package with an idea of the kind of data they want to create for the sake of a project. The core functionality of {peacesciencer} begins with the creation of a data frame, after which can be populated with various indicators of interest. No matter, the suggested use of {peacesciencer} begins with the creation of a data frame. To start, assume a new user without much familiarity with the R programming language installed Rstudio, {tidyverse}, and {peacesciencer} with the idea of using {peacesciencer} to help them start a new research project that seeks to explain civil conflict onset across state-years. Toward that end, they have identified their unit of analysis is state-year and can be created with the `create_stateyears()`. To get started, I encourage entering the following command in the console in Rstudio.

```
?create_stateyears()
```

This will open a documentation file for this function, which is itself quite verbose and informative for the user about what the function is doing. In this case, the documentation file will show there are three arguments in this function, each with built-in defaults. This function will allow the user to choose a state system for which they want state-years (`system`, which accepts either "cow" or "gw" for CoW state system data or Gleditsch-Ward state system data and defaults to CoW), whether they want to extend the state system to the most recently concluded calendar year (`mry`, which defaults to TRUE), and whether they may want to additionally subset the years to a more narrow temporal domain they may already have in mind (`subset_years`, which defaults to no subset of the data, returning all possible state-years). If the user simply ran the function with no overrides, the function would return all Correlates of War state-years from 1816 to the most recent year (2020, as of writing).

```
create_stateyears()
#> # A tibble: 16,731 x 3
#>    ccode statenme                 year
#>    <dbl> <chr>                   <int>
#>  1     2 United States of America  1816
#>  2     2 United States of America  1817
#>  3     2 United States of America  1818
```

```
#>  4     2 United States of America  1819
#>  5     2 United States of America  1820
#>  6     2 United States of America  1821
#>  7     2 United States of America  1822
#>  8     2 United States of America  1823
#>  9     2 United States of America  1824
#> 10     2 United States of America  1825
#> # ... with 16,721 more rows
```

A user who approaches {peacesciencer} with a project in mind will see they can better tailor this function to what they want. Their interest in a state-year analysis of civil conflict will likely gravitate them toward the Gleditsch-Ward state system data, since that is the state system that serves as the basis of the UCDP armed conflict data. They will also know they have no use for pre-1946 observations since civil conflict data, certainly UCDP's suite of data, has coverage only from 1946 forward. Thus, the user can supply some additional arguments to tailor the creation of data to just what they want (here: all Gleditsch-Ward state-years from 1946 to 2019).

```
create_stateyears(system = 'gw', subset_years = c(1946:2019))
#> # A tibble: 10,490 x 3
#>    gwcode statename                year
#>     <dbl> <chr>                   <int>
#>  1      2 United States of America  1946
#>  2      2 United States of America  1947
#>  3      2 United States of America  1948
#>  4      2 United States of America  1949
#>  5      2 United States of America  1950
#>  6      2 United States of America  1951
#>  7      2 United States of America  1952
#>  8      2 United States of America  1953
#>  9      2 United States of America  1954
#> 10      2 United States of America  1955
```

```
#> # ... with 10,480 more rows
```

Another reader may be interested in using {peacesciencer} for a research project using a leader-year unit of analysis, which can be created with create_leaderyears(). Users can read more about what this function is doing by consulting the documentation file in R with the following command.

```
?create_leaderyears()
```

This function has three arguments denoting the leader system to inform the creation of the leader-year data (i.e. Archigos) and two other arguments: standardize and subset_years. The standardize arguments, which when it defaults to "none", returns all leader-years as presented in the raw Archigos data. Archigos' leader data are nominally denominated in the Gleditsch-Ward state system data, if not necessarily Gleditsch-Ward state system dates (e.g. Archigos often has leader entries prior to state system entry). Thus, the user can standardize leader-year data to Gleditsch-Ward state system dates of CoW state system dates. Finally, the user can subset the leader-year data to a more narrow temporal domain that may interest them with the subset_years function. If a user is interested in creating a base data of leader-years for an analysis of inter-state conflict initiation with the GML MID data, they can create the data that interests them with the following function. Notice how create_leaderyears() also returns information about the leader too, like the leader's approximate age that year, their gender, and information about their tenure.

```
create_leaderyears(standardize = "cow", subset_years = c(1870:2010))
#> # A tibble: 15,074 x 7
#>    obsid    ccode leader gender  year yrinoffice leaderage
#>    <chr>    <dbl> <chr>  <chr>  <dbl>      <dbl>     <dbl>
#>  1 USA-1869     2 Grant  M       1870          2        48
#>  2 USA-1869     2 Grant  M       1871          3        49
#>  3 USA-1869     2 Grant  M       1872          4        50
#>  4 USA-1869     2 Grant  M       1873          5        51
#>  5 USA-1869     2 Grant  M       1874          6        52
#>  6 USA-1869     2 Grant  M       1875          7        53
#>  7 USA-1869     2 Grant  M       1876          8        54
#>  8 USA-1869     2 Grant  M       1877          9        55
```

```
#>  9 USA-1877     2 Hayes  M       1877          1        55
#> 10 USA-1877     2 Hayes  M       1878          2        56
#> # ... with 15,064 more rows
```

Researchers using {peacesciencer} to create data for their research project should start with one of these "create" functions. Whether state-year, dyad-year, or a leader-level analysis, these functions will create the full universe of cases of interest to a researcher for the unit of analysis that serves as the core of their project.

Adding Information to Data Created in {peacesciencer}

After creating the base data of interest to their project, researchers can begin to add information they want with the suite of functions outlined in Table 2. For example, researcher interested in a dyad-year analysis of interstate disputes can create a non-directed dyad-year data set from 1816 to 2010 in {peacesciencer} with the create_dyadyears() function, one of the aforementioned "create" function. The following would create the base data of interest to the user (i.e. all non-directed dyad-years from 1816 to 2010).

```
create_dyadyears(directed = FALSE, subset_years = c(1816:2010))
#> # A tibble: 842,655 x 3
#>    ccode1 ccode2  year
#>     <dbl>  <dbl> <int>
#>  1      2     20  1920
#>  2      2     20  1921
#>  3      2     20  1922
#>  4      2     20  1923
#>  5      2     20  1924
#>  6      2     20  1925
#>  7      2     20  1926
#>  8      2     20  1927
#>  9      2     20  1928
#> 10      2     20  1929
#> # ... with 842,645 more rows
```

Adding information to this data frame is a simple matter of joining a series of functions together in a "pipe."

The "pipe"—represented as %>% in the code below—is an operator built into {tidyverse} that allows (R) users to pass forward expressions or functions. These pipes are common in the programming world and, as the code below will show, have the benefit of changing code in a way that is more intuitive and easier to both read and write for the user. While these functions can be modified to work without {tidyverse} installed and loaded into the session, the user should find their experience with {peacesciencer} improved by this important package.

For example, assume the researcher wants just all politically relevant, non-directed dyad-years, where political relevance is traditionally understood as a dyadic relationship involving some form of a contiguity relationship or a major power (Lemke & Reed, 2001). `create_dyadyears(directed = FALSE, subset_years = c(1816:2010))` created the full universe of non-directed dyad-years from 1816 to 2010, though this full universe includes "irrelevant" dyads like Nigeria-Mongolia and Canada-Estonia. Reducing the data to just politically relevand dyads is simple in {peacesciencer} and {tidyverse}. Users first create the data they want (here: `create_dyadyears(directed = FALSE, subset_years = c(1816:2010))`), follow it with the pipe operator (%>%), and then add another function from {peacesciencer} (here: `filter_prd()`, which also quietly executes `add_contiguity()` and `add_cow_majors()`).

```
# create base data, and pipe (%>%) to next function
create_dyadyears(directed = FALSE, subset_years = c(1816:2010)) %>%
  # subset data to politically relevant dyads (PRDs)
  filter_prd()
#> # A tibble: 112,282 x 7
#>    ccode1 ccode2  year conttype cowmaj1 cowmaj2   prd
#>     <dbl>  <dbl> <int>    <dbl>   <dbl>   <dbl> <dbl>
#>  1      2     20  1920        1       1       0     1
#>  2      2     20  1921        1       1       0     1
#>  3      2     20  1922        1       1       0     1
#>  4      2     20  1923        1       1       0     1
#>  5      2     20  1924        1       1       0     1
#>  6      2     20  1925        1       1       0     1
#>  7      2     20  1926        1       1       0     1
```

```
#>  8      2     20  1927        1        1        0     1
#>  9      2     20  1928        1        1        0     1
#> 10      2     20  1929        1        1        0     1
#> # ... with 112,272 more rows
```

Here, the user has created all non-directed dyad-years from 1816 to 2010 and then subset the data to just those with a major power or with some kind of contiguity relationship.

Users will find that the ease of the "pipe" will allow them greater agency in creating the full data set they may want for an analysis. Indeed, the pipe has the effect of forming something analogous to a drop-down menu, in which the user can "select" additional data they may want simply by specifying the function in {peacesciencer} that creates the data for them. For example, a researcher can follow `filter_prd()` with another pipe and communicate they want information about ongoing conflicts and conflict onsets from the Gibler-Miller-Little (GML) dispute data set (Gibler, Miller & Little, 2016). Following `filter_prd()` with `add_gml_mids(keep = NULL)` will add information about ongoing conflicts and onsets in a given dyad-year.[4]

```
# create base data, and pipe (%>%) to next function
create_dyadyears(directed = FALSE, subset_years = c(1816:2010)) %>%
  # subset data to politically relevant dyads (PRDs), pipe to next function
  filter_prd() %>%
  # add conflict information from GML-MID data,
  add_gml_mids(keep = NULL)
#> # A tibble: 112,282 x 9
#>    ccode1 ccode2  year conttype cowmaj1 cowmaj2   prd gmlmidonset gmlmidongoing
#>     <dbl>  <dbl> <dbl>    <dbl>   <dbl>   <dbl> <dbl>       <dbl>         <dbl>
#>  1      2     20  1920        1       1       0     1           0             0
#>  2      2     20  1921        1       1       0     1           0             0
#>  3      2     20  1922        1       1       0     1           0             0
#>  4      2     20  1923        1       1       0     1           0             0
```

[4] `keep = NULL` in this context will focus the information returned to just the information about ongoing conflicts and onsets, discarding potentially unwanted columns about things like Side A, hostility levels, and other information included in the data set. Type `?add_gml_mids()` for more information.

```
#>  5      2     20  1924        1       1       0     1           0           0
#>  6      2     20  1925        1       1       0     1           0           0
#>  7      2     20  1926        1       1       0     1           0           0
#>  8      2     20  1927        1       1       0     1           0           0
#>  9      2     20  1928        1       1       0     1           0           0
#> 10      2     20  1929        1       1       0     1           0           0
#> # ... with 112,272 more rows
```

Users can also calculate peace-years for these conflicts with `add_peace_years()` by using the pipe to pass forward the data set and applying the `add_peace_years()` function to it.[5]

```
# create base data, and pipe (%>%) to next function
create_dyadyears(directed = FALSE, subset_years = c(1816:2010)) %>%
  # subset data to politically relevant dyads (PRDs), pipe to next function
  filter_prd() %>%
  # add conflict information from GML-MID data, pipe to next function
  add_gml_mids(keep = NULL) %>%
  # add peace years
  add_peace_years()
#> # A tibble: 112,282 x 10
#>    ccode1 ccode2  year conttype cowmaj1 cowmaj2   prd gmlmidonset gmlmidongoing
#>     <dbl>  <dbl> <dbl>    <dbl>   <dbl>   <dbl> <dbl>       <dbl>         <dbl>
#>  1      2     20  1920        1       1       0     1           0             0
#>  2      2     20  1921        1       1       0     1           0             0
#>  3      2     20  1922        1       1       0     1           0             0
#>  4      2     20  1923        1       1       0     1           0             0
#>  5      2     20  1924        1       1       0     1           0             0
```

[5] `add_peace_years()` will only calculate the peace years—titled, in this case, `gmlmidspell`—and will leave the temporal dependence adjustment to the taste of the researcher. Importantly, I do not recommend manually creating splines or square/cube terms because it creates more problems in adjusting for temporal dependence in model predictions. In a regression formula in R, you can specify the Carter & Signorino (2010) approach as `... + gmlmidspell + I(gmlmidspell^2) + I(gmlmidspell^3)`. The Beck, Katz & Tucker (1998) cubic splines approach is `... + splines::bs(gmlmidspell, 4)`. This function includes the spell and three splines (hence the 4 in the command). Either approach makes for easier model predictions, given R's functionality.

```
#>  6      2     20  1925        1        1        0     1            0             0
#>  7      2     20  1926        1        1        0     1            0             0
#>  8      2     20  1927        1        1        0     1            0             0
#>  9      2     20  1928        1        1        0     1            0             0
#> 10      2     20  1929        1        1        0     1            0             0
#> # ... with 112,272 more rows, and 1 more variable: gmlmidspell <dbl>
```

Researchers should see that {peacesciencer}'s functionality can scale up nicely from there. For example, the following would round out the kind of information necessary to replicate Bremer's (1992) famous "dangerous dyads" analysis by adding information about national material capabilities for both states in the dyad (add_nmc()), estimates of democracy for both states in the dyad (add_democracy()), information about alliance commitments in the dyad-year (add_cow_alliance()), and finishing with information about estimated population size and (surplus, gross) domestic product based on simulations reported by Anders, Fariss & Markowitz (2020). Whereas add_sdp_gdp() is the last command in the pipe-based workflow, the {peacesciencer} call ends by assigning to an object called Data. This type of assignment is done with the "right hand" assignment operator (i.e. ->).

```
# create base data, and pipe (%>%) to next function
create_dyadyears(directed = FALSE, subset_years = c(1816:2010)) %>%
  # subset data to politically relevant dyads (PRDs), pipe to next function
  filter_prd() %>%
  # add conflict information from GML-MID data, pipe to next function
  add_gml_mids(keep = NULL) %>%
  # add peace years, pipe to next function
  add_peace_years() %>%
  # add capabilities data, pipe to next function
  add_nmc() %>%
  # add some estimates about democracy for each state, pipe to next function
  add_democracy() %>%
  # add information about alliance commitments in dyad-year
  add_cow_alliance() %>%
```

```
  # finish with information about population and GDP/SDP
  # and then assign to object, called, minimally, 'Data'
  add_sdp_gdp() -> Data


Data
#> # A tibble: 112,282 x 42
#>    ccode1 ccode2  year conttype cowmaj1 cowmaj2   prd gmlmidonset gmlmidongoing
#>     <dbl>  <dbl> <dbl>    <dbl>   <dbl>   <dbl> <dbl>       <dbl>         <dbl>
#>  1      2     20  1920        1       1       0     1           0             0
#>  2      2     20  1921        1       1       0     1           0             0
#>  3      2     20  1922        1       1       0     1           0             0
#>  4      2     20  1923        1       1       0     1           0             0
#>  5      2     20  1924        1       1       0     1           0             0
#>  6      2     20  1925        1       1       0     1           0             0
#>  7      2     20  1926        1       1       0     1           0             0
#>  8      2     20  1927        1       1       0     1           0             0
#>  9      2     20  1928        1       1       0     1           0             0
#> 10      2     20  1929        1       1       0     1           0             0
#> # ... with 112,272 more rows, and 33 more variables: gmlmidspell <dbl>,
#> #   milex1 <dbl>, milper1 <dbl>, irst1 <dbl>, pec1 <dbl>, tpop1 <dbl>,
#> #   upop1 <dbl>, cinc1 <dbl>, milex2 <dbl>, milper2 <dbl>, irst2 <dbl>,
#> #   pec2 <dbl>, tpop2 <dbl>, upop2 <dbl>, cinc2 <dbl>, v2x_polyarchy1 <dbl>,
#> #   polity21 <dbl>, xm_qudsest1 <dbl>, v2x_polyarchy2 <dbl>, polity22 <dbl>,
#> #   xm_qudsest2 <dbl>, cow_defense <dbl>, cow_neutral <dbl>, cow_nonagg <dbl>,
#> #   cow_entente <dbl>, wbgdp2011est1 <dbl>, wbpopest1 <dbl>, sdpest1 <dbl>, ...
```

If the user wants to move these data into Stata for analysis, they can save it to their current working directory with a command like `haven::write_dta(Data, "my-data.dta")` and import it into Stata when they are done. No matter, `{peacesciencer}` has pre-processed, cleaned, recoded, and merged the desired data that has greatly reduced the time and energy a researcher might otherwise spend doing something like

hard-coding `-9s` in these data to be `NA` in the National Material Capabilities data. In this particular application, it has already created the main data required for a replication of Bremer (1992). There is only some slight data work to create the desired indicators for a statistical model of conflict onset, like a dummy variable for land-contiguity, the presence of a major power in the dyad, and some "weak-link" indicators of militarization, relative power in the dyad, level of democracy in the dyad (using the Marquez (2016) for extending the Unified Democracy Scores data), and the GDP per capita in the dyad. Table 3 is a formatted version of the results of a logistic regression model of conflict onset using these "dangerous dyads" indicators. Users typically do not end their analysis here—often looking for new predictors of conflict onset with these covariates in mind—but `{peacesciencer}` greatly reduces the time and energy researchers must invest into cleaning and processing data for analysis.

****Table 3 about here****

A Basic Replication of Fearon and Laitin (2003) in `{peacesciencer}`

`{peacesciencer}` is capable of generating data for replications of analyses at multiple levels beyond dyad-year, including leader-year, leader-dyad-year, and state-year. Suppose a researcher wants to create a state-year data frame to conduct an analysis of civil conflict onset analogous to Fearon and Laitin's (2003) well-cited analysis of civil conflict onset, but using UCDP conflict data and the Gleditsch-Ward state system for creating the appropriate universe of state-years. The pipe-based workflow will start with `create_stateyears(system = 'gw', subset_years = c(1946:2019))`, creating the full universe of Gleditsch-Ward state years and subsetting them to just 1946-2019 (because the UCDP data included in `{peacesciencer}` include just the observations in that time frame). Next, we can use the `add_ucdp_acd()` function to return information about ongoing UCDP conflicts and onsets for these states. `add_ucdp_acd()` takes three arguments: `type`, `issue`, and `only_wars`. `type` is an optional argument for the type of armed conflicts for which the researcher wants information. Options include "extrasystemic", "interstate", "intrastate", and "II" (short for "internationalized intrastate"). If no `type` is specified, the function returns information about ongoing disputes and onsets for all states for all tyeps of conflict. If the user wants information about multiple types of conflict–say: intra-state wars and internationalized intra-state wars—they can specify that as a character vector (e.g. `type = c("intrastate", "II")`). `issue` is another optional argument for what issue types of conflicts the user wants beyond the type of armed conflict. Options include "territory", "government", and "both". If no `issue` is specified, the function returns information for all conflicts irregarding the particular issue. `only_wars` is

an argument that subsets the data to just those with the intensity levels of "war" when `only_wars = TRUE`. The argument defaults to `FALSE`, returning information about conflicts with at least 25 deaths in addition to the conflicts with more than 1,000 deaths. In this application, `add_ucdp_acd(type = "intrastate", only_wars = FALSE))` returns state-year information about ongoing intra-state conflicts over any issue and at either of UCDP's severity thresholds.[6]

```
create_stateyears(system = 'gw', subset_years = c(1946:2019)) %>%
  filter(year %in% c(1946:2019)) %>%
  add_ucdp_acd(type="intrastate", only_wars = FALSE) %>%
  add_peace_years() %>%
  add_democracy() %>%
  add_creg_fractionalization() %>%
  add_sdp_gdp() %>%
  add_rugged_terrain() -> Data
```

Finally, we can add some covariates of interest to these data. `add_peace_years()` calculates peace spells between ongoing conflicts in the data generated by `add_ucdp_acd()`. `add_democracy()` adds information about the level of democracy in the year using three prominent data sets on democracy (Polity, V-Dem, and Marquez' (2016) extension of Pemstein et al's (2010) Unified Democracy Scores). `add_creg_fractionalization()` adds information about the fractionalization and polarization of a state's ethnic and religious groups from the Composition of Religious and Ethnic Groups (CREG) Project at the University of Illinois. `add_sdp_gdp()` will add information about a state's estimated GDP, population, and GDP per capita from the Anders, Fariss & Markowitz (2020) simulations. Finally, `add_rugged_terrain()` provide two estimates of the ruggedness of a state's terrain. The first is the terrain ruggedness index calculated by Nunn & Puga (2012) and the second is the Gibler & Miller (2014) extension of the natural logged percentage of the state that is mountainous (originally calculated by Fearon & Laitin (2003)). At the end of the pipe, the data returned by `{peacesciencer}` is assigned to an object minimally called `Data`.

`{peacesciencer}`'s tight integration with the `{tidyverse}` permits wide flexibility for the researcher. For example, assume the researcher wants to discern the estimated effect of the same set of covariates on intra-state conflicts at the threshold of war and those intra-state conflicts at or below the threshold of war.

[6] The function also returns some background information of interest to the researcher, including the maximum intensity observed and the IDs associated with all ongoing conflicts in the state that year.

The first call included all conflicts with at least 25 deaths, per the UCDP's inclusion rules, and the peace years were calculated for those as well. If the researcher wants a new set of conflicts with a new set of peace years, it would be a simple matter of repeating the pipe-based workflow, but altering the argument in `add_ucdp_acd()` to be `only_wars = TRUE`. {peacesciencer} would then calculate the peace years for those (`add_peace_years()`). To avoid confusion with the overlapping column names, the researcher can use some {tidyverse} verbs to rename all those conflict variables to have a distinct prefix of `war_` (i.e. `rename_at(vars(ucdpongoing:ucdpspell), ~paste0("war_", .))`) before finally joining these data into the master data frame (i.e. `left_join(Data, .) -> Data`). Table 4 shows the fruits of the data {peacesciencer} generated after some post-processing for lagging important variables.

```
create_stateyears(system = 'gw', subset_years = c(1946:2019)) %>%
  add_ucdp_acd(type="intrastate", only_wars = TRUE) %>%
  add_peace_years() %>%
  rename_at(vars(ucdpongoing:ucdpspell), ~paste0("war_", .)) %>%
  left_join(Data, .) -> Data
```

****Table 4 about here****

Conclusion

{peacesciencer} is already more than capable of creating the kind of data in high demand in peace science. It can create dyad-year, leader-year, leader-dyad-year, and state-year data (among others). It is also generalizable to the dispute data included in the package, allowing for merging into *dispute-year* data as well. This feature showed how it can effectively approximate two types of analyses in wide use in the peace science community. Surely researchers can and will add more information to these simple analyses after using {peacesciencer}, but the package already does a lot of the tedious work for researchers. It also does this in a maximally transparent way that conforms well to the DA-RT initiative across all political science. This is not to say {peacesciencer} does everything, but {peacesciencer} can only evolve and expand on what it already does well. This package can only evolve to meet new analytical demands for the peace science community. Users are free to request new features as "issues" on the project's Github.

Finally, a skeptical reader should not think that making the process as simple as possible necessarily facilitates poor decision-making by the user. In cases where it is evident what the user wants (e.g. an estimate of the

level of democracy in the state-year), `{peacesciencer}` does the necessary work to provide the user that information. However, the package makes sure to leave important decision-making to the researcher. For example, `add_cow_alliance()` returns information about various types of alliance pledges in the dyad-year—should one exist—but leaves it to the researcher to say whether they want to define the presence of an alliance to be just a defense pledge or any type of alliance pledge. `add_contiguity()` returns information about the type of contiguity relationship in the dyad-year, but leaves it to the researcher whether they want to code a contiguity variable as the presence of a mutual land border or some other type of contiguity threshold. The documentation included in this package, and on the website, is replete with caveats about the underlying data (e.g. the contiguity data are not ordinal and should not be treated as such), how and where data issues arise (e.g. how CoW state system data differ from Gleditsch-Ward data and how one is coerced into the other), and how researchers should consider optimally using its functionality (e.g. `add_ucdp_acd()` probably should not lump all forms of conflict together). `{peacesciencer}` does not endeavor to make researchers lazy or sloppy, and it does not ultimately do this. Instead, `{peacesciencer}` reduces the tedium associated with starting quantitative peace science research, achieving this in a quick, robust, and transparent way.

References

Anders, Therese, Christopher J Fariss & Jonathan N Markowitz (2020) Bread before guns or butter: Introducing surplus domestic product (SDP). *International Studies Quarterly* 64(2): 392–405.

Arel-Bundock, Vincent (2021b) *Modelsummary: Summary Tables and Plots for Statistical Models and Data: Beautiful, Customizable, and Publication-Ready* (https://CRAN.R-project.org/package=modelsummary).

Arel-Bundock, Vincent (2021a) *WDI: World Development Indicators and Other World Bank Data* (https://CRAN.R-project.org/package=WDI).

Beck, Nathaniel, Jonathan N Katz & Richard Tucker (1998) Taking time seriously: Time-series-cross-section analysis with a binary dependent variable. *American Journal of Political Science* 42(4): 1260–1288.

Bennett, DScott, Paul Poast & Allan C Stam (2019) NewGene: An introduction for users. *Journal of Conflict Resolution* 63(6): 1579–1592.

Bennett, DScott & Allan Stam (2000) EUGene: A conceptual manual. *International Interactions* 26(2): 179–204.

Boehmke, Bradley C (2016) *Data Wrangling with R*. Springer.

Bowers, Jake & Maarten Voors (2016) How to improve your relationship with your future self. *Revista de Ciencia Politica* 36(3): 829–848.

Bremer, Stuart A (1992) Dangerous dyads: Conditions affecting the likelihood of interstate war, 1816-1965. *Journal of Conflict Resolution* 36(2): 309–341.

Carter, David B & Curtis S Signorino (2010) Back to the future: Modeling time dependence in binary data. *Political Analysis* 18(3): 271–292.

Coppedge, Michael, John Gerring, Carl Henrik Knutsen, Staffan I Lindberg, Jan Teorell, David Altman, Michael Bernhard, MSteven Fish, Adam Glynn, Allen Hicken, Anna Luhrmann, Kyle L Marquardt, Kelly McMann, Pamela Paxton, Daniel Pemstein, Brigitte Seim, Rachel Sigman, Svend-Erik Skaaning, Jeffrey Staton, Agnes Cornell, Lisa Gastaldi, Haakon Gjerløw, Valeriya Mechkova, Johannes von Römer, Aksel Sundtröm, Eitan Tzelgov, Luca Uberti, Yi-ting Wang, Tore Wig & Daniel Ziblatt (2020) V-dem codebook v10.

Correlates of War (2011) State system membership list, v2016 (http://correlatesofwar.org).

Fearon, James D & David D Laitin (2003) Ethnicity, insurgency, and civil war. *American Political Science Review* 97(1): 75–90.

Gibler, Douglas M & Steven V Miller (2014) External territorial threat, state capacity, and civil war. *Journal of Peace Research* 51(5): 634–646.

Gibler, Douglas M, Steven V Miller & Erin K Little (2016) An analysis of the Militarized Interstate Dispute (MID) dataset, 1816-2001. *International Studies Quarterly* 60(4): 719–730.

Gleditsch, Kristian S & Michael D Ward (1999) A revised list of independent states since the Congress of Vienna. *International Interactions* 25(4): 393–413.

Goemans, Henk E, Kristian Skrede Gleditsch & Giacomo Chiozza (2009) Introducing Archigos: A dataset on political leaders. *Journal of Peace Research* 46(2): 269–83.

Lemke, Douglas & William Reed (2001) The relevance of politically relevant dyads. *Journal of Conflict Resolution* 45(1): 126–144.

Marquez, Xavier (2016) A quick method for extending the Unified Democracy Scores (http://dx.doi.org/10.2139/ssrn.2753830).

Marshall, Monty G, Ted Robert Gurr & Keith Jaggers (2017) Polity IV project: Political regime characteristics and transitions, 1800-2016.

Nunn, Nathan & Diego Puga (2012) Ruggedness: The blessing of bad geography in Africa. *Review of Economics and Statistics* 94(1): 20–36.

Pemstein, Daniel, Stephen A Meserve & James Melton (2010) Democratic compromise: A latent variable analysis of ten measures of regime type. *Political Analysis* 18(4): 426–449.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo & Hiroaki Yutani (2019) Welcome to the tidyverse. *Journal of Open Source Software* 4(43): 1686.

Table 1: Functions that Create Base Data

Function	**Description**
`create_dyadyears()`	Create dyad-years from state system membership data
`create_leaderdays()`	Create leader-day data from Archigos
`create_leaderdyadyears()`	Create leader-dyad-years data from Archigos
`create_leaderyears()`	Create leader-years data from Archigos
`create_statedays()`	Create state-days from state system membership data
`create_stateyears()`	Create state-years from state system membership data

Table 2: Functions that Add Data

Function	Type	Description
add_archigos()	D,S	Add Archigos political leader information to a data frame
add_atop_alliance()	D, LD	Add Alliance Treaty Obligations and Provisions (ATOP) alliance data to data frame
add_capital_distance()	D, L, LD, S	Add capital-to-capital distance to a data frame
add_ccode_to_gw()	D, L, LD, S	Match CoW state codes to G-W state codes
add_contiguity()	D, L, LD, S	Add CoW direct contiguity information to a data frame
add_cow_alliance()	D, LD	Add CoW alliance data to a data frame
add_cow_majors()	D, L, LD, S	Add CoW major power information to a data frame
add_cow_mids()	D	Add CoW-MID data to a dyad-year data frame
add_cow_trade()	D, L, LD, S	Add CoW trade data to a data frame
add_cow_wars()	D, S	Add CoW war data to a dyad-year or state-year data frame
add_creg_fractionalization()	D, L, LD, S	Add fractionalization/polarization estimates from CREG to a data fram
add_democracy()	D, L, LD, S	Add democracy information to a data frame
add_fpsim()	D, LD	Add dyadic foreign policy similarity measures to a data frame
add_gml_mids()	D, L, LD, S	Add GML MID data to a data frame
add_gwcode_to_cow()	D, L, LD, S	Match G-W state codes to CoW state codes
add_igos()	D, S	Add CoW IGO data to a data frame
add_lead()	L, LD	Add leader experience/attributes (LEAD) to data
add_lwuf()	L, LD	Add leader willingness to use force estimates to data
add_minimum_distance()	D, L, LD, S	Add minimum distance data to data frame
add_nmc()	D, L, LD, S	Add minimum distance estimates (in km) to data frame
add_peace_years()	D, S	Add 'peace years' to dyad-year/state-year conflict data
add_rugged_terrain()	D, L, LD, S	Add rugged terrain information to a data frame
add_sdp_gdp()	D, L, LD, S	Add (surplus, gross) domestic product data to a data frame
add_spells()	D, L, LD, S	Add duration 'spells' to a data frame
add_strategic_rivalries()	D, S	Add strategic rivalry information to a data frame
add_ucdp_acd()	S	Add UCDP Armed Conflict Data to state-year data frame
add_ucdp_onsets	S	Add UCDP onsets to state-year data
filter_prd()	D, LD	Filter dyad-year data to just politically relevant dyads
wc_duration()	C	Whittle Duplicate Conflict-Years by (Minimum or Maximum) Conflict Duration
wc_fatality()	C	Whittle Duplicate Conflict-Years by Highest Fatality
wc_hostility()	C	Whittle Duplicate Conflict-Years by Highest Hostility
wc_jds()	C	Whittle Duplicate Conflict-Years by Just Dropping Something
wc_onsets()	C	Whittle Uniqute Conflict Onset-Years from Conflict-Year Data
wc_recip()	C	Whittle Duplicate Conflict-Years by Conflict Reciprocation
wc_stmon()	C	Whittle Duplicate Conflict-Years by Lowest Start Month

Table 3: A "Dangerous Dyads" Analysis of Non-Directed Dyad-Years from {peacesciencer}

	Model 1
Land Contiguity	1.062*
	(0.057)
Dyadic CINC Proportion (Lower/Higher)	0.453*
	(0.036)
CoW Major Power in Dyad	0.144*
	(0.057)
Defense Pact	−0.119*
	(0.058)
Dyadic Democracy (Weak-Link)	−0.493*
	(0.052)
Dyadic GDP per Capita (Weak-Link)	0.293*
	(0.051)
Dyadic Militarization (Minimum)	0.263*
	(0.023)
t	−0.146*
	(0.005)
t^2	0.002*
	(0.000)
t^3	0.000*
	(0.000)
Intercept	−3.045*
	(0.063)
Num.Obs.	103 919

+ $p < 0.1$, * $p < 0.05$

Table 4: A Civil Conflict Analysis of Gleditsch-Ward State-Years in {peacesciencer}

	All UCDP Conflicts	**Wars Only**
GDP per Capita (Lagged)	−0.285*	−0.343*
	(0.110)	(0.172)
Population Size (Lagged)	0.229*	0.272*
	(0.067)	(0.106)
Extended UDS Democracy Score (Lagged)	0.257	−0.085
	(0.181)	(0.270)
Extended UDS Democracy Score^2 (Lagged)	−0.726*	−0.761*
	(0.211)	(0.352)
% Mountainous Terrain (Logged)	0.055	0.342*
	(0.067)	(0.112)
Ethnic Fractionalization	0.442	0.333
	(0.358)	(0.554)
Religious Fractionalization	−0.389	−0.281
	(0.402)	(0.593)
t	−0.074+	−0.111*
	(0.039)	(0.056)
t^2	0.004*	0.005+
	(0.002)	(0.003)
t^3	0.000*	0.000+
	(0.000)	(0.000)
Intercept	−5.098*	−6.591*
	(1.351)	(2.084)
Num.Obs.	8192	8192

+ $p < 0.1$, * $p < 0.05$