23 December 2021

Dear Editor and Reviewers,

I thank the editor and the two reviewers to their reception of my manuscript "`{peacesciencer}`: An R Package for Quantitative Peace Science Research" for inclusion in *Conflict Management and Peace Science* (*CMPS*). A conditional acceptance is always a welcome gift from the profession during the holidays. In what follows, I address the editor about the minor changes to the manuscript the editor suggested in order to transition the manuscript to an outright acceptance.

First, I included the citations to *all* the data referenced in the manuscript (i.e. all the data included in `{peacesciencer}`), even if the manuscript itself did not make explicit use of the data. R2 (rightly) noted that a project that is concerned with proper attribution should have a manuscript that is comparably as concerned with proper attribution. R2's application here was the ATOP data. In this case, the manuscript notes coverage of both the ATOP and Correlates of War alliance data, uses (and cites) the Correlates of War alliance data in one application, and incidentally cited neither source outright. It was an oversight to not reference Gibler (2009), though R2 felt Leeds et al. (2002) should be cited as well. This version of the manuscript now cites all data sets included in `{peacesciencer}`.

R2 also advised me of some data updates that should be included in the package, encouraging me to keep everything as up-to-date as possible. R2's case here was version 6.0 of the Correlates of War National Material Capabilities data, suggesting I include these updates in the package. The version of `{peacesciencer}` slated for release alongside the manuscript (v. 1.0.0) now has these updates. I did want to use this particular case as a plea to the editor about why I structured the manuscript the way I did. Notice the manuscript is light on listing version numbers for the data included in the manuscript. In the case of the capabilities data referenced by R2, the data were released on July 22, 2021 and the manuscript was initially submitted earlier that same month. Thus, the peer review process almost concluded by missing this update entirely. I mention this to note that it stands to reason that something like this will almost invariably happen again at the copy-editing stage. Indeed, the manuscript notes that the `create_dyadyears()` and `create_stateyears()` functions will generate data to the most recently concluded calendar year. Right now, that is 2020. In a week, it will be 2021.

I mention this to note that what I offer to the editorial team, the reviewers, and the peace science community is effectively a sincere "scout's honor" that the package will remain as up-to-date as possible, even if interested users may have to bring to my attention updates that I may miss. Features will invariably fall out of date and need to be updated. If anything, I think this is a unique advantage, of a kind, to `{peacesciencer}`. There is nothing in this package that will break upon the release of some new data set, since the data are effectively stored in-house. It just means there is a more current version of the data that needs to be properly cleaned and included in a future update. CRAN—the central repository for R packages—is generally very quick to update packages. All CRAN requires is about a month between updates (so they are not spammed with busy work done on a volunteer basis). The timeline from submission to publication of an R package update is almost always less than week (holidays notwithstanding), and routinely (from my experience) as quick as a day.

In other words, these updates (and more) are going to be part of a new version (1.0.0) slated for release alongside the manuscript. In most cases, I avoid deliberate reference to version numbers of data included in the package

in case they fall out of date prior to the appearance of the manuscript in print. However, I do believe updating `{peacesciencer}` will be much more streamlined, and much quicker, than alternatives like NewGene and EUGene.

Finally, I thank the editor for encouraging me to at least acknowledge, if not outright incorporate, R1's suggestion to show how much time is saved in `{peacesciencer}` relative to doing things manually. I plan to make this another vignette around the time the paper comes to print. The application here will be the `create_dyadyears()` function. If R1 is interested, I encourage them to time `create_dyadyears()` in `{peacesciencer}` with a procedure similar to what I outline here at the link in this footnote.[1] This procedure uses multi-core processing to identify every single day overlap of Correlates of War dyads for all years from 1816 to 2016. I use this mainly to identify what I term in the memo and appendix to be "false" dyads like Suriname and South Vietnam (i.e. states that exist in the same year, but never at the same time in the year). A user can see how much time is required for this procedure that ultimately ensures that these "false" dyads do not appear in their dyad-year data, and how much time is ultimately spared them by `{peacesciencer}` and the `create_dyadyears()` function.

All told, I think the manuscript is ready for outright acceptance and I humbly thank again the editor and the reviewers for their interest. This has been one of the more enjoyable review experiences in my over 10 years in the profession. Enclosed are the revised manuscript and appendix.

Best regards,

The Author

[1] https://github.com/svmiller/peacesciencer/blob/master/data-raw/false_cow_dyads.R

Previous Memo to the Editorial Team

This was the memo to the editorial team and two reviewers for the first round of revisions. I include it here for posterity, though some text was slightly changed for presentation (i.e. I saw my own typos).

I thank you all for the opportunity to submit revisions of my manuscript "`{peacesciencer}`: An R Package for Quantitative Peace Science Research" for possible inclusion in *Conflict Management and Peace Science* (*CMPS*). I thank the two reviewers and the editorial team for the helpful comments as well. In what follows, I address the editor about the changes to the manuscript that came in light of reviewer feedback and continued development of the package. Thereafter, I address the reviewers' comments individually.

Comments to the Editorial Team

The package and paper evolved considerably since the first version I submitted for review at *CMPS*. In addition to the slight name change, I want to drive home the following major changes.

First, the manuscript is much longer now. An email inquiry to the editorial team informed me that there is no different word limit for data features relative to other article types that *CMPS* would accept. The manuscript I submitted for review was under 6,000 words. This version is just under 9,500 words.

Second, `{peacesciencer}` is *much* more feature-rich. It has robust support for leader-level analyses. The manuscript even shows how you could use built-in functions and data to approximate the kind of analysis that Horowitz and Stam (2014) did.

Third, there are more vignettes for the package that discuss some important topics about this package, some raised during the peer-review process. These are also included in the appendix submitted alongside the memo and the manuscript.

All told, I think the manuscript and the package are much stronger now for these changes, and I think (and certainly hope) the editorial team and reviewers would agree. Enclosed are the revised manuscript and appendix. I thank the editorial team and the reviewers for their helpful comments and for further consideration of my revisions.

Reviewer #1's Comments

Reviewer #1 (R1) had the following comments on the original version of the manuscript. First, I want to thank R1 for this feedback, and for the editorial team for selecting R1 to review this manuscript. R1 had exactly the kind of expertise I wanted to get from this process: someone who has good insight into writing software.

I have identified the following major issues raised by R1 and write here how I have revised the manuscript and the package in light of these issues.

The documentation lacks detail about how these merges work

R1 mentioned that the documentation lacks some detail about how the merges work and, in particular, draws attention to how dyadic dispute-year data are coerced into true dyad-year data. I want to mention three things in response.

First, this has always been communicated to the user if they looked at the underlying code, but I understand the reviewer's point that this should not be necessary. Toward that end, `add_cow_mids()` and `add_gml_mids()` will now communicate what are the case exclusion rules it employs in coercing dyadic dispute-year data to true dyad-year data. This communication appears as a message when the respective function finishes.

Second, this has long been a vignette on the package's website. While more care could be given to point the user in this direction, there is a dedicated vignette that belabors these case exclusion rules. I include this in the appendix as well.

Third, the release of v. 0.6.0 of `{peacesciencer}` came with a class of what I term "whittle" functions. These are specialty functions that coerce dyadic dispute-year data into dyad-year in a given sequence that the researcher deems best fits their particular needs. Table 1 lists these functions, partly abbreviated from Table 2 of the manuscript. For example, `wc_onsets()`—itself a shortcut for the more verbose `whittle_conflicts_onsets()`—will take dyadic dispute-year data (either Correlates of War [CoW] or Gibler-Miller-Little [GML]) and, where duplicates exist in a dyad-year, keep only those observations that are unique dispute onsets. `wc_fatality()`—itself a shortcut for the more verbose `whittle_conflicts_fatality()`—will take dyadic dispute-year data (either CoW or GML) and select those with the highest observed dispute-level fatality.

Table 1: The 'whittle' Class of Funtions

Function	**Description**
`wc_fatality()`	Whittle Duplicate Conflict-Years by Highest Fatality
`wc_hostility()`	Whittle Duplicate Conflict-Years by Highest Hostility
`wc_jds()`	Whittle Duplicate Conflict-Years by Just Dropping Something
`wc_onsets()`	Whittle Uniqute Conflict Onset-Years from Conflict-Year Data
`wc_recip()`	Whittle Duplicate Conflict-Years by Conflict Reciprocation
`wc_stmon()`	Whittle Duplicate Conflict-Years by Lowest Start Month

The vignette on the package's website, combined with the message from, for example, the `add_gml_mids()` function, means these two functions will produce the equivalent dyad-year data for an analysis of dispute onsets

(even if the column names will differ).

```
# Approach 1
cow_ddy %>% add_gml_mids()

# Approach 2
gml_dirdisp %>%
  whittle_conflicts_onsets() %>%
  whittle_conflicts_fatality() %>%
  whittle_conflicts_hostility() %>%
  whittle_conflicts_duration(durtype = "mindur") %>%
  whittle_conflicts_duration(durtype = "maxdur") %>%
  whittle_conflicts_reciprocation() %>%
  whittle_conflicts_startmonth() %>%
  whittle_conflicts_jds() %>%
  left_join(cow_ddy, .)
```

I thank R1 for encouraging me to do this—it really was no chore since I had already effectively written these functions—and I think the functionality of the package improved for it.

Better explain the left join

R1 also deduced from the underlying code that the left (outer) join is everywhere in this package and is the primary workhorse for the functions that populate a base data set created in this package. R1 encouraged me to acknowledge this and explain it in some detail.

In light of some other feedback, I felt an entire section on the left join would have been a detour from the flow of the manuscript. Instead, I do acknowledge this type of join as the primary workhorse of the functions included in the package and reference a vignette on the website, included as an appendix in this submission, about what this type of join is doing, why I use it the way I do, and what issues may arise from them. I summarize this as follows.

In short, the left join is a type of "mutating" join that adds information from one data set into another based on some common matching keys. Assuming two data objects, `x` and `y`, then `left_join()` returns all rows from `x` (i.e. a "left" object) with matching information in `y` (i.e. a "right" object) based on those matching keys across `x` and `y`. Its use in this package has the following assumptions. First, the universe of cases of interest to the user—as generated by one of the "create" functions in the package—is in the "left" object. A user may, for example, have a state-year analysis of European countries and want to merge in some, say, GDP information that has observations as well for African countries. Information from the African countries will *not* be merged into the left object with this type of join. Second, there are no unwanted duplicate observations in the "right" object, which would create unwanted duplicate observations in the "left" object. If, for example, the left object has just one observation for a unified Germany in 1990, but the right object has two observations for Germany in 1990 (e.g.. pre- and post-unification), this type of join will duplicate those observations in the left object. Thus, the package invests considerable care in making sure the functions contained in this package do not create additional rows in the output beyond the dimensions of the data supplied to it.

I think this discussion makes more sense in the appendix or as a vignette on the package's website, but I thank

R1 for encouraging me to acknowledge this.

Communicate version numbers of data

R1 liked that I included a `ps_cite()` function for appropriately citing the material I use, but felt that the package needed a corollary for citing the version of the data included in the package. Thus, I have included a function—`ps_version()`—that produces a data set of version numbers associated with the data included in this package. Here is a preview of what this function will do.

```
# What are all the version numbers?
ps_version()
#> # A tibble: 31 x 4
#>    category   data                                    version bibtexkey
#>    <chr>      <chr>                                   <chr>   <chr>
#>  1 states     Correlates of War State System Membership 2016    cowstates2016
#>  2 leaders    LEAD                                    2015    ellisetal2015le~
#>  3 leaders    Archigos                                4.1     goemansetal2009~
#>  4 alliance   ATOP                                    5       leedsetal2002at~
#>  5 democracy  Polity                                  2017    marshalletal201~
#>  6 democracy  {QuickUDS}                              0.2.3   marquez2016qme
#>  7 democracy  V-Dem                                   10      coppedgeetal202~
#>  8 capitals   {peacesciencer}                         2020    peacesciencer-p~
#>  9 contiguity Correlates of War Direct Contiguity     3.2     stinnettetal200~
#> 10 igo        Correlates of War IGOs                  3       pevehouseetal20~
#> # ... with 21 more rows

# What are the version numbers of data where category == "democracy"?
ps_version("democracy")
#> # A tibble: 3 x 4
#>   category  data       version bibtexkey
#>   <chr>     <chr>      <chr>   <chr>
#> 1 democracy Polity     2017    marshalletal2017p
#> 2 democracy {QuickUDS} 0.2.3   marquez2016qme
#> 3 democracy V-Dem      10      coppedgeetal2020vdem
```

A vignette on the package website, also included in the appendix, explains how to interpret some of these version numbers. For example, a version number that is a year includes data that are not formally "versioned", per se, (c.f. Anders, Fariss and Markowitz, 2020; Carter and Smith, 2020) and that the data included reflect the last update or date of publication.

Include a license file

R1 drew my attention to the *Journal of Open Source Software* and encouraged me to follow their lead in adequately licensing this software. Thus, the software bundle now includes an actual GPL-2 license file and not just a reference to GPL-2 licensing.

Do a better job engaging with other software packages

R1 also felt the version of the manuscript I submitted for peer review could do a better job engaging with alternatives to what I propose in the paper. Namely, R1 wanted to know what is the current state of this practice and how `{peacesciencer}` is an improvement (or at least different). To accommodate this, I dedicated an entire section of the revised manuscript to how `{peacesciencer}` compares with some alternatives. I will also add to that discussion here.

For one, I want to communicate to R1 and the editorial team that I searched high and low for alternatives to what I propose here and found only NewGene and, before it, EUGene. I will confess up front that the initial impetus for what became `{peacesciencer}` was constantly having to remind myself how to create dyad-year data based on state system membership. I did a Google search for that—"how to create dyad-year data" (without quotes)—and the first search result was to a blog post of mine that itself became a part of the code included in this package. The fifth search result was even to a Stack Overflow post on this exact same question over eight years ago, that I incidentally had asked. Someone on the Stata forum had actually asked this question in 2014 as well.[2] The bulk of the search results on top of that point to this particular package, in addition to references to NewGene and EUGene. I mention this to say that I do not think many alternatives to `{peacesciencer}` exist, certainly as practitioners in our field would think to search for them.

My hunch is that the current state of the practice, at least for the core functionality of creating raw data of interest to IR researchers, is one that relies on data transformations that owe to some functionality built into Structured Query Language (SQL). R1 deduced from the raw code that the left (outer) join is everywhere in this package. The implementation may be in R, but the procedure is very much SQL at heart (e.g. left joins, inner joins). The revised manuscript acknowledges this, and does say that a researcher who wants to create state-year or dyad-year data from state system membership data, or leader-year and leader-dyad-year data from Archigos, could just as well invest time in learning SQL to accomplish this. The process of creating `{peacesciencer}` has made me appreciate SQL more—and I am convinced there is important value for students in learning it—but learning SQL just to create these types of data amounts to a detour in the research process.[3]

EUGene is the clear inspiration for this package. I acknowledge R1's minor point that using EUGene to create raw data from scratch was not the main point of EUGene—it just happened to be how most of us used it over 10 years ago—but the point remains EUGene is effectively retired as a software package. It was last updated in 2017 with some minor bug fixes, though these bug fixes do not come with any change in functionality from a version (3.2) that was last released in 2007. This means `{peacesciencer}` has a major advantage over EUGene in better meeting current research interests and needs. It has support for leader-level data and leader-level analyses where EUGene does not. It has support for the ecosystem of data denominated in Gleditsch-Ward state system data where EUGene does not. Functionally, this means `{peacesciencer}` can support the needs of civil conflict scholars working with UCDP-PRIO data where EUGene cannot. A researcher who may still want to calculate older variables like Tau-b and expected utility values will not find that in `{peacesciencer}` right now, but `{peacesciencer}` is more than capable of meeting current needs in the peace science community.

[2] https://www.statalist.org/forums/forum/general-stata-discussion/general/511471-creating-many-dyads

[3] I am unaware exactly how EUGene and NewGene are programmed. EUGene's documentation for version 3.2 says the application itself was written in the "Borland Delphi language", which I think pertains to the integrated development environment and not the programming language in the software itself. Delphi has long had SQL support, at least from its first release, and I think Delphi itself is derivative of Pascal. Bennett, Poast and Stam (2019) thank what appears to be a C++ programmer in the acknowledgment section of their manuscript and greeting to NewGene announces its C++ basis in the bottom right corner, but some debugging output given by NewGene suggests that important SQL transformations are happening underneath the proverbial hood. I will confess to having no real operational knowledge of C++, Delphi, or Pascal.

NewGene is the closest thing that approximates what `{peacesciencer}` does, among software options still actively supported. The manuscript mentions where the two differ. NewGene has a current advantage over `{peacesciencer}` in the construction of *k*-adic data, and it right now has some variables that `{peacesciencer}` could easily incorporate in future updates (e.g. ICOW territorial claim data). `{peacesciencer}` has important support for civil conflict data where NewGene's focus is almost entirely aimed for researchers interested in inter-state conflict. I also want to dedicate some additional space here to clarifying some important differences between the two that I do not think the lay reader will understand or appreciate, but I am sure the reviewer will because of their demonstrated insight into how `{peacesciencer}` is working and how to write good software for this community.

For one, the manuscript does make a reference to the fact that `{peacesciencer}` is more "explicit" than NewGene, which is not as "explicit" as EUGene. I want to clarify what I mean here because I think it touches on multiple issues. For one, as I do reference in the manuscript, NewGene is vague on what it understands to be the unit of analysis. NewGene never formally tells the user to declare their unit of analysis in the interface it provides and the closest it does come to this is near the top where it prompts the user to state how many (Correlates of War) country code columns they want. If the user selects 1, NewGene generates monadic data over a temporal domain the user specifies in the next tab. If the user selects 2, NewGene generates dyadic data over a temporal domain the user specifies in the next tab. With three onward, the user is getting some *k*-adic data over a temporal domain the user specifies in the next tab. This would otherwise be no real issue if not for the fact that NewGene offers leader-level data. A researcher can think of leader-level analyses as derivations of state-level, dyadic-level, or *k*-adic-level analyses the extent to which leaders are nested in states, but leader-level units of analysis are still distinct from these other units of analysis. Thus, if I am a naive researcher using NewGene for the first time asking for just one state-year column over a given period of time and checking boxes for Archigos information in return, do I know if I am getting true leader-year data or just state-year data?[4] `{peacesciencer}` forces this upfront by telling the user to state what is their unit of analysis before anything else is generated alongside it. NewGene only gets the user to state their unit of analysis after they indicate the information they want, which, in a sense, conflates the operationalization of the research design with the conceptualization of the research design.

On a related note, NewGene is similarly vague about merging in data between different data ecosystems, or even among the same data ecosystem. For example, assume a researcher is using NewGene to create a state-year data set that has some national material capabilities data (denominated in the Correlates of War state system) along with some IPE data (which, at least in part, is denominated in the Gleditsch-Ward ecosystem). Which system is the master system here? After selecting where the user wants the data saved, they must then pick what is the "top-level variable group" (either the data containing the capabilities or the data containing some IPE variables). I think this suboptimal because it does not force the user to state upfront what is the universe of their cases (i.e. in the `{peacesciencer}` case, the Correlates of War state system over a period of time or the Gleditsch-Ward system over a period of time). This may seem trivial, but it is not. It again encourages the researcher to conflate the conceptualization of the research design with the operationalization of the research design. Stating upfront what is the universe of cases is an important design decision. `{peacesciencer}` does this and NewGene does not appear to do this. Base data approximating the universe of cases should be created *first* before data are appended to them and users should declare what is the population to which they are trying to infer (i.e. the state system data in the quantitative peace science context) before collecting information.

[4] The FAQ on NewGene's website advertises how you would do this when it explains why states are the main decision-making units, though the documentation (`FAQ 14`) describes a NewGene interface that is no longer current as version 2.0. This leaves the user—this particular user—in the dark as to how one could get something like leader-year data with some state-year information (e.g. capabilities) using NewGene.

Think of what {countrycode} does for us as analogous to what is happening here, and why this is important. R users in our field know about this package well since it is the most useful package for matching one state system classification to another, including support for Correlates of War system, Gleditsch-Ward system, Federal Information Processing Standards, UN M49, and much more (Arel-Bundock, Enevoldsen and Yetman, 2018). However, anyone who uses this package for, say, converting Correlates of War state codes to ISO 3166-1 alpha-3 codes for the sake of merging in World Bank data (c.f. Arel-Bundock, 2021), are acknowledging—if implicitly—that their master system is the Correlates of War system. Their unit of analysis is Correlates of War states over a certain time period. It just happens to be the case that Correlates of War state codes can mostly be matched to ISO 3166-1 alpha-3 codes for the cause of merging in information about some economic indicator of interest. However, this matching and merging procedure does not change the fact that there is a master system, which remains the master system, even after merging. That should be declared *before* anything else is added to it, which is what {peacesciencer} does with the create_ family of functions. In terms of current and best practices, I think teaching researchers to acknowledge this is one of those subtly important things that we should be doing better since—as {countrycode}'s development shows and what {peacesciencer}'s raw data and comments emphasize on a more focused scale—this can be quite challenging.[5]

R1 raised an issue, addressed above, that {peacesciencer} could better inform the user to how dyadic dispute-year data are coerced into dyad-year data. This information had always been in {peacesciencer} and was a vignette on the package's website, though a recent release (v. 0.6.0) gave users the option to employ their own case exclusion rules to eliminate duplicate dyad-year observations. Toward that same end, NewGene is maximally non-committal about what would be an optimal approach. Indeed, one of NewGene's YouTube videos, published March 6, 2020, tells the user to eliminate duplicate observations through Excel.[6] The video describing the procedure has only music overlaying it (no commentary) and describes a procedure that is brute-force, with no questions asked, and with no explanation given to what it is being eliminated. {peacesciencer}'s transparency and explicitness are major improvements here.

Finally, I am quite confident of the quality control that goes into {peacesciencer} whereas I am less confident in some of the quality control that goes into what NewGene is doing. The following are just two things I noted in comparing v. 0.7.0 of {peacesciencer} with version 2.0 of NewGene. I encourage the reviewer to see these things for themselves.

First, create a simple state-year data set in NewGene that asks for the Correlates of War state code, year, and iron and steel production estimate from the country capabilities tab. Add to it a request to get a Gleditsch-Ward state code and the number of islands from the IPE data resource tab at the bottom. Subset the years to 1985 to 1995 in the next tab and declare the capabilities data to have the top-level variable group. Save the output to some spreadsheet and open that. Find where ccode is 255 and the year is greater than or equal to 1990. Notice that there is no number of islands information for Germany those years because NewGene did not internalize that what is Gleditsch-Ward's 260 in the state system data is Correlates of War's 255 after the unification of Germany. A similar problem occurs for Yemen, whose integration is treated identically. The application here is minimal, and perhaps no one is interested in the number of islands Germany and Yemen have, but this points to a bigger problem in how NewGene tries to integrate Correlates of War data with Gleditsch-Ward data.

Second, create a simple dyad-year data set in NewGene (i.e. ask for two Correlates of War state columns near the top of the interface) and ask for the Correlates of War state code, year, and iron and steel production estimate

[5] A vignette on the package website talks in greater detail how {peacesciencer} handles the integration of Correlates of War state system data with Gleditsch-Ward state system data while also reiterating this same point.

[6] https://www.youtube.com/watch?v=ZBlW77lD4iQ

from the country capabilities tab. In the next tab, ask for a full temporal domain of 1816 to 2020 and save to a spreadsheet. Open the spreadsheet and find where `ccode_1` is 115 and `ccode_2` is 817, with the operative year in question being 1975. This dyad will be there; it should not be. The Republic of Vietnam exits the Correlates of War state system on April 30, 1975 whereas Suriname enters the state system on November 25, 1975. Both appear in the same year, but not at the same time. These are false dyads that happen when expanding the state system data based on unique observations of state-years, but not state-days.[7] `{peacesciencer}` identifies these and removes them in the `create_dyadyears()` function.

All told, I think the reader will agree that `{peacesciencer}` is a useful solution to some important research design problems, albeit one that requires at least some familiarity with the R programming language. It has several innovations that are not available in alternative programs, along with more clarity as to what it is doing. The maximal transparency of `{peacesciencer}` allows for greater confidence in quality control as well.

Include some simpler code snippets for specific use cases

R1 felt that the code snippets I included got too complicated too quickly and asked for simpler code snippets to highlight some more specific cases. I have tried to do this in a few ways.

First, the code snippets I include in the manuscript are amply annotated with comments about what each function is doing. This is a made possible and easy by using R Markdown as front-end for writing the manuscript, allowing me to seamlessly execute code and use comments to explain what each code snippet is doing.

Second, the section on creating and amending a dyad-year data set is done somewhat piecemeal. I start with the creation of the dyad-year data. I then talk about how the pipe operator allows the user to create dyad-year data and and subset the data to just politically relevant dyads. I then talk about how the user can extend this process forward another step by creating dyad-year data, subsetting the data to just politically relevant dyads, and then populating the data with information about ongoing inter-state disputes and dispute onsets from the Gibler-Miller-Little (GML) data set. This entire section shows the reader how to build the data they want, piece by piece, using `{peacesciencer}` functions and basic features of the `{tidyverse}`. The point is well-taken that the code snippets got too complicated without much explanation. I believe this version of the manuscript strikes a balance between both. There are simple use cases along with a guide for more thorough data-building. Everything is amply annotated with an explanation to what is happening.

Third, and this is more a comment about what I cannot do, but I interpreted the comment from R1 here as more about the manuscript and not the documentation files included in the package (which R1 indicates they have seen). Thus, I worked hard to make clearer what the code is doing in the manuscript, but R documentation files do not have a lot of opportunity for highlighting specific use cases because of CRAN testing requirements. CRAN requires every test of a function in the documentation file be quick enough to execute in under 5 seconds on their amalgam of testing computers. Some of these testing computers can be quite slow, and indeed some of the functions in this package can be a little slow as well even on fast computers. I circumvent some of this with the package's website (which does execute every piece of code in the documentation file) as well as a vignette that showcases various parlor tricks (included in the appendix). However, I am unable to be too creative in the documentation files of the R package itself.

[7] After loading `{peacesciencer}` into the R session, the reader can see what these "false" dyads are in the Correlates of War state system with `false_cow_dyads`. There is a Gleditsch-Ward equivalent as well (`false_gw_dyads`).

Reviewer #2's Comments

I thank Reviewer #2 (R2) for their comments. On the balance, I read R2's feedback as supportive of what I had offered. However, they suggested more functionality should be included and the manuscript could be better streamlined for the reader. I address these main points below.

Introduce leader-level functionality

R2 felt leader-level functions and leader-level data would be a major improvement over the version of the package I had submitted for peer review. As of v. 0.7.0, `{peacesciencer}` now has multiple functions and built-in data for researchers interested in doing leader-level analyses. I will summarize these below.

`create_leaderdays()`, `create_leaderyears()`, and `create_leaderdyadyears()` will create leader-day, leader-year, and leader-dyad-year data. Depending on the `standardize` argument specified in the function, the function will optionally standardize these data to the Correlates of War and Gleditsch-Ward state system. Though Archigos (the underlying leader data) is nominally denominated in Gleditsch-Ward state codes, the leader data have observations for leaders outside state system dates. Think of Lynden Pindling (`BHM-1967`) as at least one illustration here. Archigos has his tenure starting in 1967 though statehood for Bahamas begins in 1973. Thus, the user will still need to standardize the leader data to Gleditsch-Ward system dates, if they want this.

`{peacesciencer}` also has some specialty functions designed for leader-level analyses. `add_lead()` and `add_lwuf()` will add information to leader-level analyses based on information from the LEAD data set (Ellis, Horowitz and Stam, 2015) and estimates about leader willingness to use force as generated by Carter and Smith (2020). Both depend on the `LEAD` and `lwuf` data also supplied in the package. `add_gml_mids()` also painstakingly matches leaders to dispute onset for leader-level analyses (both monadic and dyadic) on dispute initiation. `add_spells()` is a more flexible peace-spell function designed for the unbalanced nature of leader panels.

Most other functions primarily written around state-year or dyad-year data will also work with leader-level data, For example, `add_democracy()` will return democracy estimates in the state-year, even for leader-year data. Where appropriate, users are reminded in the documentation files that data like these (e.g. democracy estimates in `add_democracy()`) are fundamentally state-year and that extensions to leader-level data should be understood as approximations for leaders in a given state-year.

In addition, there is a helper data set provided in the package—`leader_codes`—that is used internally for helping sync leader observations across varying leader codes across Archigos 4.1, Archigos 2.9, and the LEAD data set. These data treat version 4.1 of the Archigos data as the gospel leader data (if you will) for which the observation ID (`obsid`) is the master code indicating a leader tenure period. It also builds in an assumption that various observations that duplicate in the LEAD data should not have duplicated. This concerns Francisco Aguilar Barquer (who appears twice), Emile Reuter (who appears twice), and Gunnar Thoroddsen (who appears three times) in the LEAD data despite having uninterrupted tenures in office. None of the covariates associated with these leaders change in the LEAD data, which is why I assume they were duplicates.

Better explain the pipe operator

R2 felt the discussion around the pipe operator (`%>%`) was a bit distracting and asked to make it easier for the reader to follow along with what it is doing and why it is important. R2 alternatively felt it might be better to minimize the discussion of the pipe operator altogether.

This is challenging, because I do see R2's point. The manuscript is primarily about the R package, its benefits, and why people should use it. It also dovetails with, effectively, an R primer that also elevates the `{tidyverse}`. The extent to which I have primarily written this as almost an advocacy piece for those who still otherwise do their statistical analysis in Stata to use this package (and R) to at least generate their data, R2 is right that I risk losing the reader. Thus, from that perspective, it may make some sense to minimize the discussion of the pipe operator since it is not a core function native to the package.

However, I also strongly believe that the package becomes that much harder to use in the absence of the pipe operator. Compare the two code snippets below for creating Correlates of War state-years from 1816 to 2020 and populating it with information about major power status, democracy, and suprplus/gross domestic product information. The first approach uses the pipe operator as I advocate it. The second approach builds the same data, but without the pipe operator.

```
# Approach 1, with the pipe operator
create_stateyears() %>%
  add_cow_majors() %>%
  add_democracy() %>%
  add_sdp_gdp() → A

# Approach 2, without the pipe operator
add_sdp_gdp(add_democracy(add_cow_majors(create_stateyears()))) → B

# Are these two things identical?
identical(A, B)
#> [1] TRUE
```

The data sets are identical, but that is also less the point. The first approach is much more readable than the second approach. I think the pipe operator is worth mentioning, and worth advocating, because it makes what is still—at its core—a programming language with a command-line interface seem *much* more intuitive. I think this is true even for beginners without much experience in R. I mention in the manuscript that the pipe operator has the effect of forming something analogous to a drop-down menu, in which the user can "select" additional data/commands they may want simply by specifying the function built into the package. The pipe operator is clearly not necessary, as the second approach in the code snippet shows, but it makes `{peacesciencer}` much more user-friendly.

That said, I understand R2's point that I cannot lose the reader here, and I risked doing so in the first draft. I believe this version has much more explicit description of what the pipe operator is and how to use it. This, in particular, concerns the guide on creating dyad-year data for analysis, also addressing R1's point that the manuscript could benefit from simpler code snippets.

The passage about the graduate students adds no value to the manuscript

R2 felt I could drop the passage about the graduate students—not because they disagreed with it, but because they felt it added no real value to the paper. I think this is fair, and I removed that material from this version of the mansucript.

References

Anders, Therese, Christopher J. Fariss and Jonathan N. Markowitz. 2020. "Bread Before Guns or Butter: Introducing Surplus Domestic Product (SDP)." *International Studies Quarterly* 64(2):392–405.

Arel-Bundock, Vincent. 2021. *WDI: World Development Indicators and Other World Bank Data*. R package version 2.7.4.
URL: *https://CRAN.R-project.org/package=WDI*

Arel-Bundock, Vincent, Nils Enevoldsen and CJ Yetman. 2018. "countrycode: An R package to convert country names and country codes." *Journal of Open Source Software* 3(28):848.
URL: *https://doi.org/10.21105/joss.00848*

Bennett, D. Scott, Paul Poast and Allan C. Stam. 2019. "NewGene: An Introduction for Users." *Journal of Conflict Resolution* 63(6):1579–1592.

Carter, Jeff and Charles E. Smith. 2020. "A Framework for Measuring Leaders' Willingness to Use Force." *American Political Science Review* 114(4):1352–1358.

Ellis, Cali Mortenson, Michael C. Horowitz and Allan C. Stam. 2015. "Introducing the LEAD Data Set." *International Interactions* 41(4):718–741.

Gibler, Douglas M. 2009. *International Military Alliances, 1648-2008*. Washington DC: CQ Press.

Horowitz, Michael C. and Allan C. Stam. 2014. "How Prior Military Experience Influences the Future Militarized Behavior of Leaders." *International Organization* 68(3):527–559.

Leeds, Bretty Ashley, Jeffrey M. Ritter, Sara McLaughlin Mitchell and Andrew G. Long. 2002. "Alliance Treaty Obligations and Provisions, 1815-1944." *International Interactions* 28:237–260.