Package ‘peacesciencer’

October 13, 2021

Type Package

Title Various Tools and Data for Quantitative Peace Science

Version 0.6.0

Depends R (>= 3.5.0)

Maintainer Steve Miller <steven.v.miller@gmail.com>

Description These are useful tools and data sets for the study of quantitative peace science. The goal for this package is to include tools and data sets for doing original research that mimics well what a user would have to previously get from a software package that may not be well-sourced or well-supported. Those software bundles were useful the extent to which they encourage replications of long-standing analyses by starting the data-generating process from scratch. However, a lot of the functionality can be done relatively quickly and more transparently in the R programming language.

License GPL-2

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.1.2

URL https://github.com/svmiller/peacesciencer/

BugReports https://github.com/svmiller/peacesciencer/issues/

Imports bib2df, magrittr, dplyr, geosphere, tidyr, stringr, rlang, stevemisc (>= 1.1.0)

Suggests countrycode, tibble, testthat, knitr, rmarkdown

NeedsCompilation no

Author Steve Miller [aut, cre] (<https://orcid.org/0000-0003-4072-6263>)

Repository CRAN

Date/Publication 2021-10-13 18:32:04 UTC
R topics documented:

add_archigos ............................................. 3
add_atop_alliance ........................................ 4
add_capital_distance ..................................... 5
add_ccode_to_gw .......................................... 6
add_contiguity ............................................. 8
add_cow_alliance .......................................... 9
add_cow_majors ........................................... 10
add_cow_mids .............................................. 11
add_cow_trade ............................................. 12
add_cow_wars .............................................. 13
add_creg_fractionalization ............................... 15
add_democracy ............................................. 16
add_gml_mids .............................................. 18
add_gwcode_to_cow ........................................ 19
add_igos .................................................... 20
add_minimum_distance ...................................... 21
add_nmc .................................................... 23
add_peace_years .......................................... 24
add_rugged_terrain ........................................ 26
add_sdp_gdp ................................................ 27
add_strategic_rivalries .................................... 29
add_ucdp_acd .............................................. 30
add_ucdp_onsets .......................................... 31
archigos .................................................... 33
atop_alliance ............................................. 34
ccode_democracy .......................................... 35
cow_alliance .............................................. 36
cow_capitals .............................................. 37
cow_contdir .............................................. 38
cow_ddy .................................................... 39
cow_gw_years .............................................. 39
cow_igo_ndy ............................................... 40
cow_igo_sy ............................................... 41
cow_majors ............................................... 42
cow_mid_ddydisps ......................................... 43
cow_mid_dirdisps ......................................... 44
cow_mid_disps ............................................ 45
cow_middist .............................................. 46
cow_nmc .................................................... 47
cow_sdp_gdp .............................................. 48
cow_states ................................................ 49
cow_trade_sy ............................................. 50
cow_war_inter ............................................ 51
cow_war_intra ............................................ 52
create_dyadyears .......................................... 53
create_leaderyears ........................................ 55
Description

`add_archigos()` allows you to add some information about leaders to dyad-year or state-year data. The function leans on an abbreviated version of the data, which also comes in this package.

Usage

```r
add_archigos(data)
```
add_atop_alliance

Arguments

data: a dyad-year data frame (either "directed" or "non-directed") or state-year data frame

Details

The function leans on attributes of the data that are provided by the create_dyadyear() or create_stateyear() function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe.

Value

add_archigos() takes a dyad-year or state-year data frame and adds a few summary variables based off the leader-level data. These include whether there was a leader transition in the state-year (or first/second state in the dyad-year), whether there was an "irregular" leader transition, the number of leaders in the state-year, the unique leader ID for Jan. 1 of the year, and the unique leader ID for Dec. 31 of the year.

Author(s)

Steven V. Miller

References


Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)

cow_ddy %>% add_archigos()
create_stateyears() %>% add_archigos()
add_capital_distance

Usage

add_atop_alliance(data)

Arguments

data a dyad-year data frame (either "directed" or "non-directed")

Details

Data are from version 5.0 of the data.

Value

add_atop_alliance() takes a dyad-year data frame and adds information about the alliance pledge in that given dyad-year from the ATOP data. These include whether there was an alliance with a defense pledge, an offense pledge, neutrality pledge, non-aggression pledge, or pledge for consultation in time of crisis.

Author(s)

Steven V. Miller

References


Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)
cow_ddy %>% add_atop_alliance()

---

add_capital_distance  
Add capital-to-capital distance to a dyad-year or state-year data frame

Description

add_capital_distance() allows you to add capital-to-capital distance to a dyad-year or state-year data frame. The capitals are coded in the cow_capitals and gw_capitals data frames, along with their latitudes and longitudes. The distance variable that emerges capdist is calculated using the "Vincenty" method (i.e. "as the crow flies") and is expressed in kilometers.
add_capital_distance(data)

Arguments

data: a dyad-year data frame (either "directed" or "non-directed") or state-year data frame

Details

The function leans on attributes of the data that are provided by the create_dyadyear() or create_stateyear() function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe.

Value

add_capital_distance() takes a dyad-year or state-year data frame and adds the capital-to-capital distance between the first state and the second state (in dyad-year data) or the minimum capital-to-capital distance for a given state in a given year. A minor note about this function: cases of capital transition are recorded in the underlying data, but, in the conversion to capital-years (and eventual merging into a dyad-year data frame), the Jan. 1 capital is used for calculating distances.

Author(s)

Steven V. Miller

Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)
cow_ddy %>% add_capital_distance()
create_stateyears() %>% add_capital_distance()

add_ccode_to_gw

Add Correlates of War state system codes to dyad-year or state-year data with Gleditsch-Ward state codes.

Description

add_ccode_to_gw() allows you to match, as well as one can, Correlates of War system membership data with Gleditsch-Ward system data.
add_ccode_to_gw

Usage

add_ccode_to_gw(data)

Arguments

data: a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame.

Details

The data-raw directory on the project’s Github contains more information about the underlying data that assists in merging in these codes.

The user will invariably need to be careful and ask why they want these data included. The issue here is that both have a different composition and the merging process will not (and cannot) be perfect. We can note that a case like Gran Colombia is not too difficult to handle (i.e. CoW does not have this entity and none of the splinter states conflict with CoW’s coding). However, there is greater weirdness with a case like the unification of West Germany and East Germany. Herein, Correlates of War treats the unification as the reappearance of the original Germany whereas Gleditsch-Ward treat the unification as an incorporation of East Germany into West Germany. The script will *not* create state-year or dyad-year duplicates for the Gleditsch-Ward codes. The size of the original data remain unchanged. However, there will be some year duplicates for various Correlates of War codes (prominently Serbia and Yugoslavia in 2006). Use with care. You can also use the countrycode package. Whether you use this function or the countrycode package, do *not* do this kind of merging without assessing the output.

Value

add_ccode_to_gw() takes a dyad-year data frame or state-year data frame that already has Gleditsch-Ward state system codes and adds their corollary Correlates of War codes.

Author(s)

Steven V. Miller

Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)

create_dyadyears(system = "gw") %>% add_ccode_to_gw()

create_stateyears(system = 'gw') %>% add_ccode_to_gw()
add_contiguity

Add Correlates of War direct contiguity information to a dyad-year or state-year data frame

**Description**

`add_contiguity()` allows you to add Correlates of War contiguity data to a dyad-year or state-year data frame.

**Usage**

```r
add_contiguity(data)
```

**Arguments**

- `data` a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame

**Details**

The contiguity codes in the dyad-year data range from 0 to 5. 1 = direct land contiguity. 2 = separated by 12 miles of water or fewer (a la Stannis Baratheon). 3 = separated by 24 miles of water or fewer (but more than 12 miles). 4 = separated by 150 miles of water or fewer (but more than 24 miles). 5 = separated by 400 miles of water or fewer (but more than 150 miles).

Importantly, 0 are the dyads that are not contiguous at all in the CoW contiguity data. This is a conscious decision on my part as I do not think of the CoW’s contiguity data as exactly ordinal. Cross-reference CoW’s contiguity data with the minimum distance data in this exact package to see how some dyads that CoW codes as not contiguous are in fact very close to each other, sometimes even land-contiguous. For example, Zimbabwe and Namibia are separated by only about a few hundred feet of water at that peculiar intersection of the Zambezi River where the borders of Zambia, Botswana, Namibia, and Zimbabwe meet. There is no contiguity record for this in the CoW data. There are other cases where contiguity records are situationally missing (e.g. India-Bangladesh, and Bangladesh-Myanmar in 1971) or other cases where states are much closer than CoW’s contiguity data imply (e.g. Pakistan and the Soviet Union were separated by under 30 kilometers of Afghani territory). The researcher is free to recode these 0s to be, say, 6s, but this is why `peacescience` does not do this.

For additional clarity, the "master records" produce duplicates for cases when the contiguity relationship changed in a given year. This function returns the *minimum* contiguity relationship observed in that given year. There should be no duplicates in the returned output.

**Value**

`add_contiguity()` takes a dyad-year data frame and adds information about the contiguity relationship based on the "master records" for the Correlates of War direct contiguity data (v. 3.2). If the data are dyad-year, the function returns the lowest contiguity type observed in the dyad-year (if contiguity is observed at all). If the data are state-year, the data return the total number of land and sea borders calculated from these master records.
add_cow_alliance

Author(s)
Steven V. Miller

References

Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)
cow_ddy %>% add_contiguity()
create_stateyears() %>% add_contiguity()

add_cow_alliance  Add Correlates of War alliance data to a dyad-year data frame

Description
add_cow_alliance() allows you to add Correlates of War alliance data to a dyad-year data frame

Usage
add_cow_alliance(data)

Arguments
data a dyad-year data frame (either "directed" or "non-directed")

Details
Duplicates in the original directed dyad-year alliance data were pre-processed. Check cow_alliance for more information.

Value
add_cow_alliance() takes a dyad-year data frame and adds information about the alliance pledge in that given dyad-year. These include whether there was an alliance with a defense pledge, neutrality pledge, non-aggression pledge, or pledge for consultation in time of crisis (entente).
**add_cow_majors**

**Author(s)**
Steven V. Miller

**References**

**Examples**

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)

cow_ddy %>% add_cow_alliance()
```

---

**add_cow_majors**

*Add Correlates of War major power information to a dyad-year or state-year data frame*

**Description**

`add_cow_majors()` allows you to add Correlates of War major power variables to a dyad-year or state-year data frame.

**Usage**

```r
add_cow_majors(data)
```

**Arguments**

- `data`: a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame.

**Details**

The function leans on attributes of the data that are provided by the `create_dyadyear()` or `create_stateyear()` function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe.

**Value**

`add_cow_majors()` takes a dyad-year data frame or state-year data frame and adds information about major power status for the given state or dyad in that year. If the data are dyad-year, the function returns two columns for whether the first state (i.e. `ccode1`) or the second state (i.e. `ccode2`) are major powers in the given year, according to the Correlates of War. 1 = is a major power. 0 = is not a major power. If the data are state-year, the functions returns just one column (`cowmaj`) for whether the state was a major power in a given dyad-year.
**add_cow_mids**

**Author(s)**
Steven V. Miller

**References**

**Examples**

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)

cow_ddy %>% add_cow_majors()
```

---

**add_cow_mids**

*Add Correlates of War (CoW) Militarized Interstate Dispute (MID) data to dyad-year data frame*

---

**Description**

`add_cow_mids()` merges in CoW’s MID data to a dyad-year data frame. The current version of the CoW-MID data is version 5.0.

**Usage**

```r
add_cow_mids(data, keep)
```

**Arguments**

- `data` a dyad-year data frame (either "directed" or "non-directed")
- `keep` an optional parameter, specified as a character vector, passed to the function in a `select(one_of(.))` wrapper. This allows the user to discard unwanted columns from the directed dispute data so that the output does not consume too much space in memory. Note: the Correlates of War system codes (`ccode1, ccode2`), the observation year (`year`), the presence or absence of an ongoing MID (`cowmidongoing`), and the presence or absence of a unique MID onset (`cowmidonset`) are *always* returned. It would be foolish and self-defeating to eliminate those observations. The user is free to keep or discard anything else they see fit. If `keep` is not specified in the function, the ensuing output returns everything.
Details

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID#0112, MID#0113, MID#0306), as illustrative of the problem. This merging process employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame.

Value

`add_cow_mids()` takes a dyad-year data frame and adds dyad-year dispute information from the CoW-MID data.

Author(s)

Steven V. Miller

References


Examples

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
cow_ddy %>% add_cow_mids()

# keep just the dispute number and Side A/B identifiers
cow_ddy %>% add_cow_mids(keep=c("dispnum","sidea1", "sidea2"))
```

---

**add_cow_trade**

*Add Correlates of War trade data to a dyad-year or state-year data frame*

Description

`add_cow_trade()` allows you to add Correlates of War alliance data to a dyad-year data frame

Usage

`add_cow_trade(data)`
Arguments

data a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame

Details

For the dyad-year data, there must be some kind of information loss in order to work within the limited space available to this package. This package loads a truncated version of the data in which the trade values are rounded to integers in order to greatly reduce the disk space for this package. I do not think this to be terribly problematic, though I admit I do not like it. If this is a problem for your research question, you may want to consider not using this function for dyad-year data.

Value

add_cow_trade() takes a dyad-year data frame or state-year data frame and adds information about the volume of trade in that given dyad-year or state-year. For the state-year data, these are minimally the sum of all imports and the sum of all exports. For dyad-year data, this function returns the value of imports in current million USD in the first country from the second country (and vice-versa) along with their “smooth” equivalents.

Author(s)

Steven V. Miller

References


Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)
# The function below works, but depends on running `download_extdata()` beforehand.
# cow_ddy %>% add_cow_trade()
create_stateyears() %>% add_cow_trade()
Arguments

- **data**: state-year data frame
- **type**: the type of war you want to add. Options include "inter" or "intra".
- **intratype**: the types of armed conflicts the user wants to consider, specified as a character vector. Options include "local issues" and "central control". Applicable only if type is "intra".

Details

Intra-state war data are coerced into true state-year data by first selecting the duplicate state-years on unique onsets, then whichever war was the deadliest. The inter-state war data work functionally the same way.

On intra-state wars: the primary_state is used to identify the government principally fighting the domestic non-state actor over central control over local issues. Internationalized civil wars are included in the data, but not for outside actors that intervene on behalf of the government or rebel group.

Extra-state war functionality is not available right now as I try to figure out the demand for its use.

Value

`add_cow_wars()` takes a dyad-year or state-year data frame and returns information about wars from either the inter-state or intra-state war data set from the Correlates of War. The function works for state-year data when the user wants information about extra-state wars or intra-state wars. The function works for dyad-year data when the user wants information about inter-state wars.

Author(s)

Steven V. Miller

References


Examples

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)

create_stateyears(system = "cow") %>%
  add_cow_wars(type = "intra", intratype = "central control")

create_stateyears(system = "cow") %>%
  add_cow_wars(type = "intra", intratype = "local issues")
```
add_creg_fractionalization

Add fractionalization/polarization estimates from CREG to your dyad-year or state-year data

Description

add_creg_fractionalization() allows you to add information about the fractionalization/polarization of a state’s ethnic and religious groups to your dyad-year or state-year data.

Usage

add_creg_fractionalization(data)

Arguments

data a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame

Details

Please see the information for the underlying data creg, and the associated R script in the data-raw directory, to see how these data are generated.

The creg data have a few duplicates. When standardizing to true CoW codes, the duplicates concern Serbia/Yugoslavia in 1991 and 1992 as well as Russia/the Soviet Union in 1991. When standardizing to true Gleditsch-Ward codes, the duplicates concern Serbia/Yugoslavia in 1991 and Russia/Soviet Union in 1991. In those cases, the function does a group-by arrange for the more fractionalized/polarized estimate under the (reasonable, I think) assumption that these are estimates prior to the dissolution of those states. If this is problematic, feel free to consult the underlying data and merge those in manually.

The underlying data have both Gleditsch-Ward codes and Correlates of War codes. The merge it makes depends on what you declare as the "master" system at the top of the pipe (i.e. in create_dyadyears() or create_stateyears()). If, for example, you run create_stateyears(system="cow") and follow it with add_gwcode_to_cow(), the merge will be on the Correlates of War codes and not the Gleditsch-Ward codes. You can see the script mechanics to see how this is achieved.
add_democracy

Description

add_democracy() allows you to add estimates of democracy to either dyad-year or state-year data.

Usage

add_democracy(data)
add_democracy

Arguments

data a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame.

Details

The function leans on attributes of the data that are provided by the create_dyadyear() or create_stateyear() function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe.

Value

add_democracy() takes a dyad-year data frame or state-year data frame and adds information about the level of democracy for the state or two states in the dyad in a given year. If the data are dyad-year, the function adds six total columns for the first state (i.e. ccode1 or gwcode1) and the second state (i.e. ccode2 or gwcode2) about the level of democracy measured by the Varieties of Democracy project (v2x_polyarchy), the Polity project (polity2), and Xavier Marquez’ QuickUDS extensions/estimates. If the data are state-year, the function returns three additional columns to the original data that contain that same information for a given state in a given year.

Author(s)

Steven V. Miller

References


Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)

cow_ddy %>% add_democracy()

create_stateyears(system="gw") %>% add_democracy()

create_stateyears(system="cow") %>% add_democracy()
add_gml_mids

Add Gibler-Miller-Little (GML) Militarized Interstate Dispute (MID) data to dyad-year data frame

Description

add_gml_mids() merges in GML’s MID data to a dyad-year data frame. The current version of the GML MID data is 2.2.1.

Usage

add_gml_mids(data, keep)

Arguments

data: a dyad-year data frame (either "directed" or "non-directed")
keep: an optional parameter, specified as a character vector, passed to the function in a select(one_of(.)) wrapper. This allows the user to discard unwanted columns from the directed dispute data so that the output does not consume too much space in memory. Note: the Correlates of War system codes (ccode1, ccode2), the observation year (year), the presence or absence of an ongoing MID (gmlmidongoing), and the presence or absence of a unique MID onset (gmlmidonset) are *always* returned. It would be foolish and self-defeating to eliminate those observations. The user is free to keep or discard anything else they see fit.
If keep is not specified in the function, the ensuing output returns everything.

Details

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID #0112, MID #0113, MID #0306), as illustrative of the problem. This merging process employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame.

Value

add_gml_mids() takes a dyad-year data frame and adds dyad-year dispute information from the GML MID data.

Author(s)

Steven V. Miller
add_gwcode_to_cow

References


Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)
cow_ddy %>% add_gml_mids()

# keep just the dispute number and Side A/B identifiers
cow_ddy %>% add_gml_mids(keep=c("dispnum", "sidea1", "sidea2"))

add_gwcode_to_cow Add Gleditsch-Ward state system codes to dyad-year or state-year data
with Correlates of War state codes.

Description

add_gwcode_to_cow() allows you to match, as well as one can, Gleditsch-Ward system mem-
bership data with Correlates of War state system membership data.

Usage

add_gwcode_to_cow(data)

Arguments

data a dyad-year data frame (either "directed" or "non-directed") or a state-year data
frame.

Details

The data_raw directory on the project’s Github contains more information about the underlying
data that assists in merging in these codes.

The user will invariably need to be careful and ask why they want these data included. The issue
here is that both have a different composition and the merging process will not (and cannot) be
perfect. We can note that a case like Serbia/Yugoslavia is not too difficult to handle (since "Serbia"
ever overlaps with "Yugoslavia" in the Gleditsch-Ward data and Correlates of War understands
Serbia as the predecessor state, dominant state, and successor state to Yugoslavia). However, there
is greater weirdness with a case like Yemen/Yemen Arab Republic. The script will *not* create
state-year or dyad-year duplicates for the Correlates of War codes. The size of the original data
remain unchanged. However, there will be some year duplicates for various Gleditsch-Ward codes
(e.g. Yemen, again). Use with care. You can also use the countrycode package. Whether you use this function or the countrycode package, do *not* do this kind of merging without assessing the output.

Value

add_gwcode_to_cow() takes a dyad-year data frame or state-year data frame that already has Correlates of War state system codes and adds their corollary Gleditsch-Ward codes.

Author(s)

Steven V. Miller

Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)

cow_ddy %>% add_gwcode_to_cow()
create_stateyears() %>% add_gwcode_to_cow()

add_igos

Add Correlates of War international governmental organizations (IGOs) data to dyad-year or state-year data.

Description

add_igos() allows you to add information from the Correlates of War International Governmental Organizations data to dyad-year or state-year data, matching on Correlates of War system codes.

Usage

add_igos(data)

Arguments

data a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame.

Details

The function leans on attributes of the data that are provided by the create_dyadyear() or create_stateyear() function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe.
add_minimum_distance

Value

add_igos() takes a dyad-year data frame or state-year data frame and adds information available from the Correlates of War International Governmental Organizations data. If the data are dyad-year, the function returns the original data with just one additional column for the total number of mutual IGOs for which both members of the dyad are full members. If the data are state-year, the function returns the original data with four additional columns. These are the number of IGOs for which the state is a full member, the number of IGOs for which the state is an associate member, the number of IGOs for which the state is an observer, and the number of IGOs for which the state is involved in any way (i.e. the sum of the other three columns).

Author(s)

 Steven V. Miller

References


Examples

# just call library(tidyverse) at the top of the pipe
library(magrittr)

cow_ddy %>% add_igos()

create_stateyears() %>% add_igos()

add_minimum_distance

Add minimum distance data to a dyad-year or state-year data frame

Description

add_minimum_distance() allows you to add the minimum distance (in kilometers) to a dyad-year or state-year data frame. These estimates are recorded in the cow_mindist and gw_mindist data that come with this package. The data are current as of the end of 2019.

Usage

add_minimum_distance(data)
**add_minimum_distance**

**Arguments**

- **data**: a dyad-year data frame (either "directed" or "non-directed") or state-year data frame

**Details**

The function leans on attributes of the data that are provided by the `create_dyadyear()` or `create_stateyear()` function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe.

**Value**

`add_minimum_distance()` takes a dyad-year or state-year data frame and adds the minimum distance between the first state and the second state (in dyad-year data) or the minimum minimum (sic) distance for a given state in a given year.

**Author(s)**

Steven V. Miller

**References**


**Examples**

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
cow_ddy %>% add_minimum_distance()
create_dyadyears(system = "gw") %>% add_minimum_distance()
create_stateyears(system = "gw") %>% add_minimum_distance()
```
add_nmc

Add Correlates of War National Military Capabilities Data

Description

add_nmc() allows you to add the Correlates of War National Material Capabilities data to dyad-year or state-year data.

Usage

add_nmc(data)

Arguments

data  a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame.

Details

The function leans on attributes of the data that are provided by the create_dyadyear() or create_stateyear() function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe.

Value

add_nmc() takes a dyad-year data frame or state-year data frame and adds information about the national material capabilities for the state or two states in the dyad in a given year. If the data are dyad-year, the function adds 12 total columns for the first state (i.e. ccode1) and the second state (i.e. ccode2) for all estimates of national military capabilities provided by the Correlates of War project. If the data are state-year, the function returns six additional columns to the original data that contain that same information for a given state in a given year.

Author(s)

Steven V. Miller

References


add_peace_years

Description

add_peace_years() calculates peace years for your ongoing conflicts. The function works for both dyad-year and state-year data generated in peacescience.

Usage

add_peace_years(data, pad = FALSE)

Arguments

data: a dyad-year data frame (either "directed" or "non-directed") or state-year data frame

pad: an optional parameter, defaults to FALSE. If TRUE, the peace-year calculations fill in cases where panels are unbalanced/have gaps. Think of a state like Germany disappearing for 45 years as illustrative of this.

Details

The function internally uses sbtscs() from stevemisc. In the interest of full disclosure, sbtscs() leans heavily on btscs() from DAMisc. I optimized some code for performance.

Importantly, the underlying function (sbtscs() in stevemisc, by way of btscs() in DAMisc) has important performance issues if you’re trying to run it when your event data are sandwiched by observations without any event data. Here’s what I mean. Assume you got the full Gleditsch-Ward state-year data from 1816 to 2020 and then added the UCDP armed conflict data to it. If you want the peace-years for this, the function will fail because every year from 1816 to 1945 (along with 2020, as of writing) have no event data. You can force the function to "not fail" by setting pad = TRUE as an argument, but it’s not clear this is advisable for this reason. Assume you wanted event data in UCDP for just the extrasystemic onsets. The data start in 1946 and, in 1946, the United Kingdom, Netherlands, and France had extrasystemic conflicts. For *all* years before 1946, the events are imputed as 1 for those countries that had 1s in the first year of observation and everyone else is NA and implicitly assumed to be a zero. For those NAs, the function runs a sequence
resulting in some wonky spells in 1946 that are not implied by (the absence of) the data. In fact, none of those are implied by the absence of data before 1946.

The function works just fine if you truncate your temporal domain to reflect the nature of your event data. Basically, if you want to use this function more generally, filter your dyad-year or state-year data to make sure there are no years without any event data recorded (e.g. why would you have a CoW-MID analyses of dyad-years with observations before 1816?). This is less a problem when years with all-NAs succeed (and do not precede) the event data. For example, the UCDP conflict data run from 1946 to 2019 (as of writing). Having 2020 observations in there won’t compromise the function output when pad = TRUE is included as an argument.

Finally, add_peace_years() will only calculate the peace years 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 and Signorino (2010) approach as \( \ldots + \text{gmlmidspell} + I(\text{gmlmidspell}^2) + I(\text{gmlmidspell}^3) \) (assuming you ran add_peace_years() on a dyad-year data frame including the Gibler-Miller-Little conflict data). The Beck et al. cubic splines approach is \( \ldots + \text{splines::bs(gmlmidspell,4)} \). 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.

**Value**

add_peace_years() takes a dyad-year or state-year data frame and adds peace years for ongoing conflicts. Dyadic conflict data supported include the Correlates of War (CoW) Militarized Interstate Dispute (MID) data set and the Gibler-Miller-Little (GML) corrections to CoW-MID. State-level conflict data supported in this function include the UCDP armed conflict data and the CoW intra-state war data.

**Author(s)**

Steven V. Miller

**References**


**Examples**

```r
# just call `library(tidyverse)` at the top of your script
```
library(magrittr)
cow_ddy %>%
add_gml_mids(keep = NULL) %>%
add_cow_mids(keep = NULL) %>%
add_contiguity() %>%
add_cow_majors() %>%
filter_prd() %>%
add_peace_years()

---

**add_rugged_terrain**

*Add rugged terrain information to a dyad-year or state-year data frame*

**Description**

`add_rugged_terrain()` allows you to add information, however crude, about the "ruggedness" of a state's terrain to your dyad-year or state-year data.

**Usage**

```r
add_rugged_terrain(data)
```

**Arguments**

- `data`: a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame

**Details**

Please see the information for the underlying data rugged, and the associated R script in the `data-raw` directory, to see how these data are generated. Importantly, these data are time-agnostic and move *slowly*. We're talking about geography here. Both data sets benchmark around 1999-2000 and it's a leap of faith to use these data for comparisons across the entirety of the Correlates of War or Gleditsch-Ward system membership. Every use of data of these types have been either cross-sectional snapshots or for making state-to-state comparisons after World War II (think of your prominent civil war studies here). Be mindful about what you expect to get from these data.

The underlying data have both Gleditsch-Ward codes and Correlates of War codes. The merge it makes depends on what you declare as the "master" system at the top of the pipe (i.e. in `create_dyadyears()` or `create_stateyears()`). If, for example, you run `create_stateyears(system="cow")` and follow it with `add_gwcode_to_cow()`, the merge will be on the Correlates of War codes and not the Gleditsch-Ward codes. You can see the script mechanics to see how this is achieved.
add_sdp_gdp

Value

add_rugged_terrain() takes a dyad-year data frame or state-year data frame, whether the primary state identifiers are from the Correlates of War system or the Gleditsch-Ward system, and returns information about the "ruggedness" of the state’s terrain. The two indicators returned are the "terrain ruggedness index" calculated by Nunn and Puga (2012) and a logarithmic transformation of how mountainous the state is (as calculated by Gibler and Miller [2014]). The dyad-year data get four additional columns (i.e. both indicators for both states in the dyad) whereas the state-year data get just the two additional columns.

Author(s)

Steven V. Miller

References


Examples

# just call `library(tidyverse)` at the top of your script
library(magrittr)

cow_ddy %>% add_rugged_terrain()

create_stateyears() %>% add_rugged_terrain()

create_stateyears(system = "gw") %>% add_rugged_terrain()

---

add_sdp_gdp  Add (Surplus and Gross) Domestic Product Data

Description

add_sdp_gdp() allows you to add estimated GDP and "surplus" domestic product data from a 2020 analysis published in International Studies Quarterly by Anders, Fariss, and Markowitz.
**add_sdp_gdp**

**Usage**

add_sdp_gdp(data)

**Arguments**

data a dyad-year data frame (either "directed" or "non-directed") or a state-year data frame.

**Details**

The function leans on attributes of the data that are provided by the `create_dyadyear()` or `create_stateyear()` function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe. Users will also want to note that the underlying function access two different data sets. It appears that the results published in the International Studies Quarterly used Correlates of War classification, but a follow-up repository on Github uses Gleditsch-Ward classification. The extent to which these estimates are generated by simulation, it does mean the estimates will be slightly different across both data sets even for common observations (e.g. the United States in 1816). Because these are large nominal numbers, the estimates have been log-transformed. Users can always exponentiate these if they choose. Researchers can use these data to construct reasonable estimates of surplus GDP per capita, but must exponentiate the underlying variables before doing this.

**Value**

add_sdp_gdp() takes a dyad-year data frame or state-year data frame and adds information about the estimated gross domestic product (in 2011 USD) for that year, the estimated population in that year, the GDP per capita in that year, and what Anders, Fariss and Markowitz term the "surplus domestic product" in that year. If the data are dyad-year, the function adds eight total columns for the first state (i.e. ccode1) and the second state (i.e. ccode2) for all these estimates. If the data are state-year, the function returns four additional columns to the original data that contain that same information for a given state in a given year.

**Author(s)**

Steven V. Miller

**References**


**Examples**

# just call `library(tidyverse)` at the top of your script
library(magrittr)

cow_ddy %>% add_sdp_gdp()
add_strategic_rivalries

Add Thompson and Dreyer’s (2012) strategic rivalry data to dyad-year data frame

Description

add_strategic_rivalries() merges in Thompson and Dreyer’s (2012) strategic rivalry data to a dyad-year data frame. The right-bound, as of right now, are bound at 2010.

Usage

add_strategic_rivalries(data, across_types = 1)

Arguments

data a dyad-year data frame (either "directed" or "non-directed")
across_types optional, relevant for state-year, takes a value of 1, 2, or 3 to look for whether one of three types fits criteria for ideological, interventionary, positional, spatial rivalry. Defaults to 1.

Details

add_strategic_rivalries() will include some other information derived from the rivalry data that the user may not want (e.g. start year of the rivalry). Feel free to select those out after the fact. Function includes an on-the-fly adjustment for Austria for rivalry #79. In this case, the Austria-Serbia rivalry continues for two years after Austria-Hungary (ccode: 300) became Austria (ccode: 305).

The across_types argument is optional and observed for only state-year calls. It defaults to 1. At the default, the function looks into the rivalry data (in td_rivalries) and focuses on the ’type1’ column. If, say, a state has an ongoing rivalry and it is primarily spatial, it codes that as a spatial rivalry. Assume you input ’across_types = 2’, the function then looks across both the ’type1’ and ’type2’ columns to see if there is a spatial component to the rivalry as either its primary or secondary dimension. If so, it codes that as a 1. across_types must be 1, 2, or 3.

Value

add_strategic_rivalries() takes a dyad-year data frame and adds information about ongoing strategic rivalries. It will also include a simple dummy variable for whether there was an ongoing rivalry in the year or not. For state-year data, it returns the count of ongoing strategic rivalries for the state in the year meeting a certain criteria (i.e. whether the state has an interventionary, ideological, positional, or spatial rivalry in an ongoing year, and how many).
Author(s)
Steven V. Miller

References

Examples

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
cow_ddy %>% add_strategic_rivalries()

# across_types defaults to 1
create_stateyears() %>% add_strategic_rivalries()
```

add_ucdp_acd

---

**Add UCDP Armed Conflict Data to state-year data frame**

Description
add_ucdp_acd() allows you to add UCDP Armed Conflict data to a state-year data frame

Usage
add_ucdp_acd(data, type, issue, only_wars = FALSE)

Arguments
data state-year data frame
type the types of armed conflicts the user wants to consider, specified as a character vector. Options include "extrasystemic", " interstate", "intrastate", and "II". "II" is convenience shorthand for "internationalized intrastate". If you want just one (say: "intrastate"), then the type you want in quotes is sufficient. If you want multiple, wrap it in a vector with c().
issue do you want to subset the data to just different armed conflicts over different types of issues? If so, specify those here as you would with the type argument. Options include "territory", "government", and "both".
only_wars subsets the conflict data to just those with intensity levels of "war" (i.e. >1,000 deaths). Defaults to FALSE.
add_ucdp_onsets

Details
Right now, only state-year data are supported. Function is in true pilot mode.

Value

add_ucdp_acd() takes a state-year data frame and returns state-year information from the UCDP Armed Conflict data set (v. 20.1). The variables returned are whether there is an ongoing armed conflict in that year, whether there was an armed conflict episode onset that year, what was the maximum intensity observed that year (if an armed conflict was observed), and a character vector of the associated conflict IDs that year.

Author(s)

Steven V. Miller

References


Examples

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
library(dplyr)

create_stateyears(system = "gw") %>%
  filter(between(year, 1946, 2019)) %>%
  add_ucdp_acd()

create_stateyears(system = "gw") %>%
  filter(between(year, 1946, 2019)) %>%
  add_ucdp_acd(type = "intrastate", issue = "government")
```

---

<table>
<thead>
<tr>
<th>add_ucdp_onsets</th>
<th>Add UCDP onsets to state-year data</th>
</tr>
</thead>
</table>

Description

add_ucdp_onsets() allows you to add information about conflict episode onsets from the UCDP data program to state-year data.
Usage

add_ucdp_onsets(data)

Arguments

data a state-year data frame

Details

The function leans on attributes of the data that are provided by the create_dyadyear() or create_stateyear() function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe. The underlying data are version 19.1. Importantly, the UCDP yearly onset data are nominally state-year, but technically state-dyad-episode-year for cases of onsets. For example, there are four France-1946 observations because of four new conflict episodes with Cambodia, Laos, Thailand, and Vietnam. There are two Panama-1989 episodes, one for the invasion by the United States and another for a failed coup attempt. That means the are duplicates in the original data that I process into summaries. The user will probably want to consider some kind of recoding here.

Value

add_ucdp_onsets() takes a state-year data frame and adds a few summary variables based off armed conflict onsets data provided by UCDP. The variables returned are the sum of new conflict dyads (should they exist) in a given state-year, and the sum of new onset episodes (or new conflicts) that are separated by one, two, three, five, or 10 years since the last conflict episode.

Author(s)

Steven V. Miller

References


Examples

# just call `library(tidyverse)` at the top of the your script
library(magrittr)
library(dplyr)

create_stateyears(system="gw") %>% add_ucdp_onsets()

create_stateyears() %>%
  add_gwcode_to_cow() %>% add_ucdp_onsets()

# Recall, these are summaries. You'll need to post-process to what you want.
create_stateyears(system="gw") %>%
  add_ucdp_onsets() %>%
  mutate(onset = ifelse(sumonset1 > 0, 1, 0))

---

**archigos**

*Archigos: A (Subset of a) Dataset on Political Leaders*

**Description**

These are leader-level data drawn from the Archigos data. Space considerations mean I offer here just a few columns based on these data. Data are version 4.1.

**Usage**

archigos

**Format**

A data frame with 3409 observations on the following 10 variables.

- ccode: a numeric vector for the Correlates of War state code
- obsid: a character vector for observation ID
- leadid: the unique leader identifier
- leader: the leader name
- yrborn: the year the leader was born
- gender: a categorical variable for leader gender ("M" for men, "W" for women)
- startdate: a date for the leader start date
- enddate: a date for the leader end date
- entry: a character vector for the leader’s entry type
- exit: a character vector for the leader’s exit type
- exitcode: a character vector for more information about the leader’s exit type

**Details**

Space considerations mean I can only offer a few columns from the overall data. Archigos data are rich with information. Consult the raw data available on Hein Goeman’s website for more.

To best conform with data requirements on CRAN, a few leader names were renamed if they included irregular characters (e.g. umlauts or accents). These leaders are "(Juan Orlando) Hernández" (HON-2014), "(Antonio) Saca González" (SAL-2004), "Julián Trujillo Largacha" (COL-1878), "César Gaviria Trujillo" (COL-1990), "Gabriel García Moreno" (ECU-1869), "Marcos A. Morinigo"
atop_alliance

(PAR-1894-1), "Higinio Morínigo" (PAR-1940), "Sebastián Piñera" (CHL-2010), "Sauli Niinistö" (FIN-2012), "Louis Gerhard De Geer" (SWD-1876), "Stefan Löfven" (SWD-2014), "Lars Løkke Rasmussen" (DEN-2009, DEN-2015), and "Fernando de Araújo" (ETM-2008-1). None of these names contain these special characters in the data here.

References


atop_alliance

Alliance Treaty Obligations and Provisions (ATOP) Project Data (v. 5.0)

Description

These are directed dyad-year-level data for alliance obligations and provisions from the ATOP project

Usage

atop_alliance

Format

A data frame with 272,046 observations on the following eight variables.

ccode1 a numeric vector for the Correlates of War state code for the first state
ccode2 a numeric vector for the Correlates of War state code for the second state
year a numeric vector for the year
atop_defense a numeric vector that equals 1 if there was an alliance observed with a defense pledge
atop_offense a numeric vector that equals 1 if there was an alliance observed with an offense pledge
atop_neutral a numeric vector that equals 1 if there was an alliance observed with a neutrality pledge
atop_nonagg a numeric vector that equals 1 if there was an alliance observed with a non-aggression pledge
atop_consul a numeric vector that equals 1 if there was an alliance observed with a consultation pledge

Details

The data-raw directory on the project’s Github shows how the data were processed.
**References**


---

**Description**

These are democracy data for all Correlates of War state system members.

**Usage**

code_democracy

**Format**

A data frame with 16536 observations on the following 5 variables.

- `ccode`: the Correlates of War system code
- `year`: a numeric vector for the year
- `v2x_polyarchy`: the Varieties of Democracy "polyarchy" estimate
- `polity2`: the Polity project score
- `xm_qudsest`: an extension of the Unified Democracy Scores (UDS) estimates, made possible by the QuickUDS package from Xavier Marquez.

**Details**

Missing data connote data that are unavailable for various reasons. Either there is no democracy data to code or, in the case of the Polity project, the state system member is outright not evaluated for the variable.

The Polity data are from 2017. The Varieties of Democracy data are version 10. Xavier Marquez’ QuickUDS estimates (i.e. extensions of Pemstein et al. (2010)) come from a package Marquez makes available on his Github ([https://github.com/xmarquez/QuickUDS](https://github.com/xmarquez/QuickUDS)).

**References**


**cow_alliance**

Correlates of War directed dyad-year alliance data

**Description**

These are version 4.1 of the Correlates of War directed dyad-year alliance data.

**Usage**

cow_alliance

**Format**

A data frame with 120784 observations on the following 7 variables.

- ccode1: a numeric vector for the Correlates of War state code for the first state
- ccode2: a numeric vector for the Correlates of War state code for the second state
- year: a numeric vector for the year
- cow_defense: a numeric vector that equals 1 if the alliance included a defense pledge
- cow_neutral: a numeric vector that equals 1 if the alliance included a neutrality pledge
- cow_nonagg: a numeric vector that equals 1 if the alliance included a non-aggression pledge
- cow_entente: a numeric vector that equals 1 if the alliance included a pledge to consult if a crisis occurred

**Details**

The directed dyad-year alliance data are for alliance initiations, not straight dyad-years, "per se." This suggests the presence of duplicate directed dyad-years. For computing ease, given the intended use, I take care of these duplicate dyad-years behind the scenes. Consider the case of the U.S. and Canada in 1958. Therein, there were apparently two separate alliance initiations that included defense pledges. My behind-the-scenes cleaning process groups by ccode1, ccode2, and year and summarizes those alliance pledge variables. I then replace any value greater than 1 with 1. This indicates the presence or absence of a defense pledge in a given directed dyad-year.

**References**

Description

This is a complete list of capitals and capital transitions for Correlates of War state system members. I use it internally for calculating capital-to-capital distances in the `add_capital_distances()` function.

Usage

cow_capitals

Format

A data frame with 252 observations on the following 7 variables.

- `ccode` a numeric vector for the Correlates of War state code
- `stname` a character vector for the state
- `capital` a character vector for the name of the capital
- `styear` a character vector for the start year. See details section for more information.
- `endyear` a character vector for the end year. See details section for more information.
- `lat` a numeric vector of the latitude coordinates for the capital
- `lng` a numeric vector of the longitude coordinates for the capital

Details

For convenience, the start year for most states is 1816. Samoa, for example, was not a state in 1816. However, the functions that use the `cow_capitals` data will not create observations for states that did not exist at a given point in time.

The data should be current as of the end of 2020.

Cases where a start year is not 1816 indicate a capital transition. For example, Brazil’s capital moved from Rio de Janeiro to Brasilia (a planned capital) in 1960. Only 25 states in the data experienced a capital transition. The most recent was Burundi in 2018. Indonesia, as of writing, is planning on a capital transition, but this has not been completed yet.

Kazakhstan renamed its capital for the state leader in 2019. These data retain the name of Astana. This will be changed in the event the software I use records this change.

The capitals data are not without some peculiarities. Prominently, Portugal transferred the Portuguese court from Lisbon to Rio de Janeiro from 1808 to 1821. *This is recorded in the data.* A knowledge of the inter-state conflict data will note there was no war or dispute between, say, Portugal and Spain (or Portugal and any other country) at any point during this time, but it does create some weirdness that would suggest a massive distance between two countries, like Portugal and Spain, that are otherwise land-contiguous.
On Spain: the republican government moved the capital at the start of the civil war (in 1936) to Valencia. However, it abandoned this capital by 1937. I elect to not record this capital transition. The data also do some (I think) reasonable back-rating of capitals to coincide with states in transition without necessarily formal capitals by the first appearance in the state system membership data. These concern Lithuania, Kazakhstan, and the Philippines. Kaunas is the initial post-independence capital of Lithuania. Almaty is the initial post-independence capital of Kazakhstan. Quezon City is the initial post-independence capital of the Philippines. This concerns, at the most, one or two years for each of these three countries.

---

**Correlates of War Direct Contiguity Data (v. 3.2)**

**Description**

These contain an abbreviated version of the "master records" for the Correlates of War direct contiguity data. Data contain a few cosmetic changes to assist with some functions downstream from it.

**Usage**

cow_contdir

**Format**

A data frame with 2025840 observations on the following 4 variables.

- ccode1: a numeric vector for the Correlates of War state code for the first state
- ccode2: a numeric vector for the Correlates of War state code for the second state
- conttype: a numeric vector for the contiguity relationship
- begin: the year-month when this contiguity relationship begins (YYYYMM)
- end: the year-month when this contiguity relationship ends (YYYYMM)

**Details**

The "master record" provided by the Correlates of War is "non-directed." I make these data "directed" for convenience.

For clarity, the contiguity codes range from 1 to 5. 1 = direct land contiguity. 2 = separated by 12 miles of water or fewer (a la Stannis Baratheon). 3 = separated by 24 miles of water or fewer (but more than 12 miles). 4 = separated by 150 miles of water or fewer (but more than 24 miles). 5 = separated by 400 miles of water or fewer (but more than 150 miles). Cases of separation by more than 400 miles of water are not included in the master record (but are easily discerned based on complete dyad-year data).

**References**

**cow_ddy**

| cow_ddy | A directed dyad-year data frame of Correlates of War state system members |

**Description**

This is a complete directed dyad-year data frame of Correlates of War state system members. I offer it here as a shortcut for various other functions when I am working on new additions and don't want to invest time in waiting for create_dyadyears() to run.

**Usage**

`cow_ddy`

**Format**

A data frame with 2063670 observations on the following 3 variables.

- `ccode1` a numeric vector for the Correlates of War state code for the first state
- `ccode2` a numeric vector for the Correlates of War state code for the second state
- `year` a numeric vector for the year

**Details**

Data are a quick generation from the `create_dyadyears()` function in this package.

---

**cow_gw_years**

| cow_gw_years | Correlates of War and Gleditsch-Ward states, by year |

**Description**

This is a complete (I believe) data set on Correlates of War states and Gleditsch-Ward states, a byproduct of a `full_join()` between `gw_states` and `cow_states` that leans largely on the state abbreviation variable.

**Usage**

`cow_gw_years`
Format

A data frame with 16936 observations on the following 6 variables.

gwcode  a Gleditsch-Ward state code
stateabb the state abbreviation, which was the greatest source of agreement between both data sets
gw_statename the state name as it appears in the Gleditsch-Ward data
code a Correlates of War state code
cow_statename the state name as it appears in the Correlates of War data
year  a numeric vector for the year

Details

The data-raw directory on the project’s Github contains more information about how these data were created. I’m going to use it for internal stuff. The workflow is going to treat the Correlates of War state system membership codes as more of the “master” codes, for which the user can add Gleditsch-Ward identifiers as they see fit. Data are extended to 2020, assuming no changes to state system membership for either data set.

cow_igo_ndy  Correlates of War Non-Directed Dyad-Year International Governmental Organizations (IGOs) Data

Description

This is a non-directed dyad-year version of the Correlates of War IGOs data. I use it internally for merging IGOs data into dyad-year data.

Usage

cow_igo_ndy

Format

A data frame with 917695 observations on the following 4 variables.

code1  the Correlates of War state system code for the first state
code2  the Correlates of War state system code for the second state
year  the year
dyadigos  the sum of mutual IGOs for which each state appears as a full member in a given year
Details

The data-raw directory on the project’s Github contains additional information about how these data were generated from the otherwise enormous dyad-year IGOs data provided by the Correlates of War project. Given the size of that data, and the size limitations of R packages for CRAN, the data I provide here can only be simpler summaries. If you want specifics, you’ll need to consult the underlying raw data provided on the Correlates of War project.

References


cow_igo_sy

Correlates of War State-Year International Governmental Organizations (IGOs) Data

Description

This is a state-year version of the Correlates of War IGOs data. I use it internally for merging IGOs data into state-year data.

Usage

cow_igo_sy

Format

A data frame with 1557 observations on the following 5 variables.

- ccode: the Correlates of War state system code for the state
- year: the year
- sum_igo_full: the sum of IGOs for which the state is a full member in a given year
- sum_igo_associate: the sum of IGOs for which the state is just an associate member in a given year
- sum_igo_observer: the sum of IGOs for which the state is just an observer in a given year
- sum_igo_type: the sum of IGOs for which the state is a member of any kind in a given year.

Details

The data-raw directory on the project’s Github contains additional information about how these data were generated from the otherwise enormous dyad-year IGOs data provided by the Correlates of War project. Given the size of that data, and the size limitations of R packages for CRAN, the data I provide here can only be simpler summaries. If you want specifics, you’ll need to consult the underlying raw data provided on the Correlates of War project.
References


cow_majors

Correlates of War Major Powers Data (1816-2016)

Description

These are the Correlates of War major powers data.

Usage

cow_majors

Format

A data frame with 14 observations on the following 8 variables.

ccode a numeric vector for the Correlates of War country code
styear the start year as a major power
stmonth the start month as a major power
stday the start day as a major power
endyear the end year as a major power
endmonth the end month as a major power
endday the end day as a major power
version a version identifier

Details

Data are provided "as-is" with no additional re-cleaning before inclusion into this data set (beyond eliminating the state abbreviation).

References

Directed Dyadic Dispute-Year Data with No Duplicate Dyad-Years
(CoW-MID, v. 5.0)

Description

These are directed dyadic dispute year data derived from the Correlates of War (CoW) Militarized Interstate Dispute (MID) project. Data are from version 5.0. These were whittled to where there is no duplicate dyad-years. Its primary aim here is merging into a dyad-year data frame.

Usage

cow_mid_ddydisps

Format

A data frame with 10234 observations on the following 25 variables.

dispnum  a numeric vector for the CoW-MID dispute number
ccode1  a numeric vector for the focal state in the dyad
ccode2  a numeric vector for the target state in the dyad
year  a numeric vector for the dispute-year
cowmidongoing  a numeric vector for whether there was a dispute ongoing in that year
cowmidonset  a numeric vector for whether it was the onset of a new dispute (or new participant-entry into a recurring dispute)
sidea1  is ccode1 on side A of the dispute?
sidea2  is ccode2 on side A of the dispute?
fatality1  a numeric vector for the overall fatality level of ccode1 in the dispute
fatality2  a numeric vector for the overall fatality level of ccode2 in the dispute
fatalpre1  a numeric vector for the known fatalities (with precision) for ccode1 in the dispute
fatalpre2  a numeric vector for the known fatalities (with precision) for ccode2 in the dispute
hiact1  a numeric vector for the highest action of ccode1 in the dispute
hiact2  a numeric vector for the highest action of ccode2 in the dispute
hostlev1  a numeric vector for the hostility level of ccode1 in the dispute
hostlev2  a numeric vector for the hostility level of ccode2 in the dispute
orig1  is ccode1 an originator of the dispute?
orig2  is ccode2 an originator of the dispute?
fatality  a numeric vector for the fatality level of the dispute
hostlev  a numeric vector for the hostility level of the MID
mindur  a numeric vector for the minimum duration of the MID
maxdur  a numeric vector for the maximum duration of the MID
recip  a numeric vector for whether a MID was reciprocated
stmon  a numeric vector for the start month of the MID
Details

The process of creating these is described at one of the references below. Importantly, these data are somewhat "naive." That is: they won’t tell you, for example, that Brazil and Japan never directly fought each other during World War II. Instead, it will tell you that there were two years of overlap for the two on different sides of the conflict and that the highest action for both was a war. The data are thus similar to what the EUGene program would create for users back in the day. Use these data with that limitation in mind.

References


cow_mid_dirdisps Directed Dyadic Dispute-Year Data (CoW-MID, v. 5.0)

Description

These are directed dyadic dispute year data derived from the Correlates of War (CoW) Militarized Interstate Dispute (MID) project. Data are from version 5.0.

Usage

cow_mid_dirdisps

Format

A data frame with 11390 observations on the following 18 variables.

dispnum a numeric vector for the CoW-MID dispute number
ccode1 a numeric vector for the focal state in the dyad
code2 a numeric vector for the target state in the dyad
year a numeric vector for the dispute-year
dispongoing a numeric vector for whether there was a dispute ongoing in that year
disponset a numeric vector for whether it was the onset of a new dispute (or new participant-entry into a recurring dispute)
sidea1 is ccode1 on side A of the dispute?
sidea2 is ccode2 on side A of the dispute?
fatality1 a numeric vector for the overall fatality level of ccode1 in the dispute
fatality2 a numeric vector for the overall fatality level of ccode2 in the dispute
**cow_mid_disps**  
Abbreviated CoW-MID Dispute-level Data (v. 5.0)

**Description**
This is an abbreviated version of the dispute-level CoW-MID data.

**Usage**
cow_mid_disps

**Format**
A data frame with 2436 observations on the following 7 variables.

dispnum  a numeric vector for the CoW-MID dispute number
outcome  a numeric vector for the outcome of the MID
styear   a numeric vector for the start year of the MID

---

fatalpre1  a numeric vector for the known fatalities (with precision) for ccode1 in the dispute
fatalpre2  a numeric vector for the known fatalities (with precision) for ccode2 in the dispute
hiact1    a numeric vector for the highest action of ccode1 in the dispute
hiact2    a numeric vector for the highest action of ccode2 in the dispute
hostlev1   a numeric vector for the hostility level of ccode1 in the dispute
hostlev2   a numeric vector for the hostility level of ccode2 in the dispute
orig1      is ccode1 an originator of the dispute?
orig2      is ccode2 an originator of the dispute?

**Details**
The process of creating these is described at one of the references below. Importantly, these data are somewhat "naive." That is: they won’t tell you, for example, that Brazil and Japan never directly fought each other during World War II. Instead, it will tell you that there were two years of overlap for the two on different sides of the conflict and that the highest action for both was a war. The data are thus similar to what the EUGene program would create for users back in the day. Use these data with that limitation in mind.

**References**
stmon  a numeric vector for the start month of the MID
settle  a numeric vector for the how dispute was settled
fatality  a numeric vector for the fatality level of the dispute
mindur  a numeric vector for the minimum duration of the MID
maxdur  a numeric vector for the maximum duration of the MID
hiact  a numeric vector for the highest action of the MID
hostlev  a numeric vector for the hostility level of the MID
recip  a numeric vector for whether a MID was reciprocated

Details
These data are purposely light on information; they’re not intended to be used for dispute-level
analyses, per se. They’re intended to augment the directed dyadic dispute-year data by adding in
variables that serve as exclusion rules to whittle the data from dyadic dispute-year to just dyad-year
data.

References
Palmer, Glenn, and Roseanne W. McManus and Vito D’Orazio and Michael R. Kenwick and
Mikaela Karstens and Chase Bloch and Nick Dietrich and Kayla Kahn and Kellan Ritter and

---

cow_mindist  The Minimum Distance Between States in the Correlates of War Sys-
tem, 1886-2019

Description
These are non-directed dyad-year data for the minimum distance between states in the Correlates
of War state system from 1886 to 2019. The data are generated from the cshapes package.

Usage
cow_mindist

Format
A data frame with 817053 observations on the following 4 variables.
ccode1  the Correlates of War state system code for the first state
ccode2  the Correlates of War state system code for the second state
year  the year
mindist  the minimum distance between states on Jan. 1 of the year, in kilometers
Details

The data are generated from the `cshapes` package. The package authors purport that the data are generated to be compatible with Correlates of War system codes, but a review I did several years ago for an unrelated project (published in 2017 in *Conflict Management & Peace Science*, which you should cite for all your articles if you're reading this) suggested the output does not seem to perfectly meet that billing. These included oddball cases like Zanzibar, United Arab Republic, Comoros, East Germany, and a few others. Those appear to be fixed in this version.

Data are automatically generated (by default) as directed dyad-years. I elect to make them non-directed for space considerations. Making non-directed dyad-year data into directed dyad-year data isn't too difficult in R. It just looks weird to see the code that does it.

Previous versions of these data were for the minimum distance as of Dec. 31 of the referent year. These are now Jan. 1. Most of the data I prove elsewhere in this package are to be understood as the data as they were at the *start* of the year. This is how I process, for example, the capitals data as they get merged in the `add_capital_distance()` function. However, the script that generates these data are set at Jan. 1 of the year and not Dec. 31. Right now, the `cshapes` does not appear to work on my system and I do not know why. Fortunately, the package authors made these data available.

References


---

**Correlates of War National Military Capabilities Data**

**Description**

These are version 5.0 of the Correlates of War National Military Capabilities data. Data omit the state abbreviation and version identifier for consideration.

**Usage**

cow_nmc

**Format**

A data frame with 15171 observations on the following 9 variables.

ccode  a numeric vector for the Correlates of War country code

year    the year

milex   an estimate of military expenditures (in thousands). See details section for more.
milper  an estimate of the size of military personnel (in thousands) for the state
irst  an estimate of iron and steel production (in thousands of tons)
pec  an estimate of primary energy consumption (thousands of coal-ton equivalents)
tpop  an estimate of the total population size of the state (in thousands)
upop  an estimate of the urban population size of the state (in thousands). See details section for more.
cinc  The Composite Index of National Capability ("CINC") score. See details section for more.

Details

The user will want to be a little careful with how some of these data are used, beyond the typical caveat about how difficult it is to pin-point how many thousands of coal-tons a state like Baden was producing in the 19th century.

First, military expenditures are denominated in British pounds sterling for observations between 1816 and 1913. The observations from 1914 and beyond are denominated in current United States dollars. This is according to the manual.

Second, urban population size is an estimate based on, well, an estimate of the size of the population living in an area with 100,000 or more people.

Third, the Composite Index of National Capability score is calculated as each state’s world share of each of the six composite indicators also included in the data in a given year. It theoretically is bound between 0 and 1. A state with a 1 is 100 in the world, 2) is the only state with a military, 3) does all the iron and steel production, 4) all the world’s primary energy consumption, and 5) is the only state in the world with a population and an urban population. Incidentally, the maximum scores observed in the data belong to the United States in 1945.

References


| cow_sdp_gdp | (Surplus and Gross) Domestic Product for Correlates of War States |

Description

These are state-year level data for surplus and gross domestic product for Correlates of War state system members. Data also include population estimates for per capita standardization.

Usage

cow_sdp_gdp
cow_states

Format

A data frame with 27753 observations on the following five variables.

ccode  a numeric vector for the Correlates of War state code
year  a numeric vector for the year
wbgdp2011est  a numeric vector for the estimated natural log of GDP in 2011 USD (log-transformed)
wbpopest  a numeric vector for the estimated population size (log-transformed)
sdpest  a numeric vector for the estimated surplus domestic product (log-transformed)
wbgdppc2011est  a numeric vector for the estimated GDP per capita (log-transformed)

Details

These were extracted from the actual replication files from *International Studies Quarterly*. Because these data are ultimately being simulated, a user can expect some slight differences between the Correlates of War version of these data (which Anders et al. published) and the Gleditsch-Ward version of these data (which appear to be the one the authors will more vigorously support going forward).

References


cow_states  

Description

These are the Correlates of War state system membership data.

Usage

cow_states

Format

A data frame with 243 observations on the following 10 variables.

stateabb  a character vector for the state abbreviation
ccode  a numeric vector for the Correlates of War country code
statename  a character vector for the state name
styear  the start year in the system
stmonth  the start month in the system
stday  the start day in the system
endyear  the end year in the system
endmonth  the end month in the system
endday    the end day in the system
version   a version identifier

Details

Data are provided "as-is" with no additional re-cleaning before inclusion into this data set.

References


Description

These are state-year-level data for national trade from the Correlates of War project.

Usage

cow_trade_sy

Format

A data frame with 14410 observations on the following four variables.

code  the Correlates of War state system code
year   the year
imports total imports of the state in current million USD
exports total exports of the state in current million USD

Details

The data-raw directory on the project’s Github shows how the data were processed.

References

Barbieri, Katherine and Omar M.G. Keshk. 2016. Correlates of War Project Trade Data Set Codebook, Version 4.0. Online: https://correlatesofwar.org
Description

These are a modified version of the inter-state war data from the Correlates of War project. Data are version 4.0. The temporal domain is 1816-2007. Data are functionally directed dyadic war-year.

Usage

cow_war_inter

Format

A data frame with 1932 observations on the following 15 variables.

- warned: the Correlates of War war number
- ccode1: the Correlates of War state code for side1
- ccode2: the Correlates of War state code for side2
- year: a numeric vector for the year
- cowinteronset: a dummy variable for whether this is an inter-state war onset (i.e. either the year in StartYear1 or StartYear2 in the raw data)
- cowinterongoing: a numeric constant of 1
- sidea1: a numeric vector for the side in the war for ccode1, either 1 or 2
- sidea2: a numeric vector for the side in the war for ccode2, either 1 or 2
- initiator1: a dummy variable that equals 1 if ccode1 initiated the war
- initiator2: a dummy variable that equals 1 if ccode2 initiated the war
- outcome1: the outcome for ccode1 as numeric vector. Outcomes are 1 (winner), 2 (loser), 3 (compromise/tied), 4 (transformed into another type of war), 5 (ongoing at end of 2007, which is not observed in these data), 6 (stalemate), 7 (conflict continues below severity of war), and 8 (changed sides)
- outcome2: the outcome for ccode2 as numeric vector. Outcomes are 1 (winner), 2 (loser), 3 (compromise/tied), 4 (transformed into another type of war), 5 (ongoing at end of 2007, which is not observed in these data), 6 (stalemate), 7 (conflict continues below severity of war), and 8 (changed sides)
- batdeath1: the estimated deaths for ccode1 (-9 = unknown)
- batdeath2: the estimated deaths for ccode2 (-9 = unknown)
- resume: a dummy variable that equals 1 if this is a conflict resumption episode
Details

See data_raw directory for how these data were generated. These data are here if you want it, but I caution against using them as gospel. There are a few problems here. One: -9s proliferate the data for battle deaths on either side, which is unhelpful. There are 10 cases where the sum of battle deaths is exactly 1,000 or 1,001. This is suspicious. The "side" variables are not well-explained—in fact they’re not explained at all in the codebook—and this can lead a user astray if they want to interpret them analogous to the sidea variables in the Correlates of War Militarized Interstate Dispute data. You probably want to use the initiator variables for this. Further, the war data routinely betray the MID data and the two do not speak well to each other. The language Sarkees and Wayman (2010) use in their book talk about how MIDs "precede" a war or are "associated" with a war, which forgets the war data are supposed to be a subset of the MID data. In one case (Gulf War), they get the associated dispute number wrong and, in one prominent case (War of Bosnian Independence), they argue no MID exists at all (it’s actually MID#3557).

References


cow_war_intra

Correlates of War Intra-State War Data (v. 4.1)

Description

These are a modified version of the intra-state war data from the Correlates of War project. Data are version 4.1. The temporal domain is 1816-2007.

Usage

cow_war_intra

Format

A data frame with 1361 observations on the following 17 variables.

- warnum the Correlates of War war number
- warname the Correlates of War war name
- wartype a character vector for the type of war, either "local issues" or "central control"
- year a numeric vector for the year
- cowintraonset a dummy variable for whether this is a civil war onset (i.e. either the year in StartYear1 or StartYear2 in the raw data)
- cowintraongoing a numeric constant of 1
- resume_combat a dummy variable for whether this is a resumption of a conflict (i.e. StartYear2 is not -8)
- primary_state a dummy variable for whether the state is the primary state having the civil war
create_ dyadyears

cco dea the Correlates of War state code for the participant on Side A. -8 = not applicable (participant is not a state)
sidea the name of the participant on Side A. -8 = not applicable (no additional party on this side)
cco de b the Correlates of War state code for the participant on Side B. -8 = not applicable (participant is not a state)
sideb the name of the participant on Side B. -8 = not applicable (no additional party on this side)
in tnl a dummy variable for if this is an internationalized civil war
outcome an unordered-categorical variable for the outcome of the civil war. Values include 1 (Side A wins), 2 (Side B wins), 3 (Compromise), 4 (war transformed into another type of war), 5 (war is ongoing at the end of 2007), 6 (stalemate), 7 (conflict continues below severity of war)
sideadeaths the estimated deaths for the Side A participant (-9 = unknown, -8 = not applicable)
sidebdeaths the estimated deaths for the Side B participant (-9 = unknown, -8 = not applicable)
ongo2007 a dummy variable for if this war is ongoing as of the end of 2007

Details

See data-raw directory for how these data were generated. In the Guinea-Bissau Civil War (1998, 1999), the "Mané Junta" were changed to just "Mane Junta" to coincide with CRAN’s character requirements.

References


create_ dyadyears

Create dyad-years from state system membership data

Description

create_ dyadyears() allows you to dyad-year data from either the Correlates of War (CoW) state system membership data or the Gleditsch-Ward (gw) system membership data. The function leans on internal data provided in the package.

Usage

create_ dyadyears(system = "cow", mry = TRUE, directed = TRUE)
create_dyadyears

Arguments

- **system**: a character specifying whether the user wants Correlates of War state-years ("cow") or Gleditsch-Ward ("gw") state-years. Correlates of War is the default.
- **mry**: optional, defaults to TRUE. If TRUE, the function extends the script beyond the most recent system membership updates to include observation to the most recently concluded calendar year. For example, the Gleditsch-Ward data extend to the end of 2017. When mry == TRUE, the function returns more recent years (e.g. 2018, 2019) under the assumption that states alive at the end of 2017 are still alive today. Use with some care.
- **directed**: optional, defaults to TRUE. If TRUE, the function returns so-called "directed" dyad-year data. In directed dyad-year data, France-Germany (220-255) and Germany-France (255-220) are observationally different. If FALSE, the function returns non-directed data. In non-directed data, France-Germany and Germany-France in the same year are the same observation. The standard here is to drop cases where the country code for the second observation is less than the country code for the first observation.

Value

create_dyadyears() takes state system membership data provided by either Correlates of War or Gleditsch-Ward and returns a dyad-year data frame.

Author(s)

Steven V. Miller

References


Examples

```r
# CoW is default, will include years beyond 2016 (most recent CoW update)
create_dyadyears()

# Gleditsch-Ward, include most recent years
create_dyadyears(system="gw")

# Gleditsch-Ward, don't include most recent years
create_dyadyears(system="gw", mry=FALSE)

# Gleditsch-Ward, don't include most recent years, directed = FALSE
create_dyadyears(system="gw", mry=FALSE, directed = FALSE)
```
**Description**

create_leaderyears() allows you to generate leader-year data from leader-level data provided in peacesciencer.

**Usage**

create_leaderyears(system = "archigos")

**Arguments**

- **system**
  
a leader system with which to create leader-years. Right now, only "archigos" is supported.

**Details**

create_leaderyears(), as of writing, only supports the Archigos data set of leaders. Importantly: the absence of much leader-level covariates (of which I am aware) means, for now, the data that are returned are treated as observationally equivalent to state-year data. Users should be careful here, but it does mean the data will work with other functions in peacesciencer that have support for state-year data (e.g. add_nmc(), add_rugged_terrain()). This is declared in the attribute field.

**Value**

create_leaderyears() takes leader-level data available in peacesciencer and returns a leader-year-level data frame.

**Author(s)**

Steven V. Miller

**References**


**Examples**

create_leaderyears()
create_statedays

Create state-days from state system membership data

Description

create_statedays() allows you to create state-day data from either the Correlates of War (CoW) state system membership data or the Gleditsch-Ward (gw) system membership data. The function leans on internal data provided in the package.

Usage

create_statedays(system = "cow", mry = TRUE)

Arguments

- **system**: a character specifying whether the user wants Correlates of War state-years ("cow") or Gleditsch-Ward ("gw") state-years. Correlates of War is the default.
- **mry**: optional, defaults to TRUE. If TRUE, the function extends the script beyond the most recent system membership updates to include observation to the most recently concluded calendar year. For example, the Gleditsch-Ward data extend to the end of 2017. When mry == TRUE, the function returns more recent years (e.g. 2018, 2019) under the assumption that states alive at the end of 2017 are still alive today. Use with some care.

Value

create_statedays() takes state system membership data provided by either Correlates of War or Gleditsch-Ward and returns a simple state-day data frame.

Author(s)

Steven V. Miller

References


Examples

# CoW is default, will include years beyond 2016 (most recent CoW update)
create_statedays()

# Gleditsch-Ward, include most recent years
create_statedays(system="gw")
# Gleditsch-Ward, don't include most recent years
create_statedays(system="gw", mry=FALSE)

---

**create_stateyears**  
*Create state-years from state system membership data*

**Description**

create_stateyears() allows you to generate state-year data from either the Correlates of War (CoW) state system membership data or the Gleditsch-Ward (gw) system membership data. The function leans on internal data provided in the package.

**Usage**

```r
create_stateyears(system = "cow", mry = TRUE)
```

**Arguments**

- `system`  
a character specifying whether the user wants Correlates of War state-years ("cow") or Gleditsch-Ward ("gw") state-years. Correlates of War is the default.

- `mry`  
optional, defaults to TRUE. If TRUE, the function extends the script beyond the most recent system membership updates to include observation to the most recently concluded calendar year. For example, the Gleditsch-Ward data extend to the end of 2017. When mry == TRUE, the function returns more recent years (e.g. 2018, 2019) under the assumption that states alive at the end of 2017 are still alive today. Use with some care.

**Value**

create_stateyears() takes state system membership data provided by either Correlates of War or Gleditsch-Ward and returns a simple state-year data frame.

**Author(s)**

Steven V. Miller

**References**

Miller, Steven V. 2019. “Create Country-Year and (Non)-Directed Dyad-Year Data With Just a Few Lines in R”  
Examples

# CoW is default, will include years beyond 2016 (most recent CoW update)
create_stateyears()

# Gleditsch-Ward, include most recent years
create_stateyears(system="gw")

# Gleditsch-Ward, don't include most recent years
create_stateyears(system="gw", mry=FALSE)

---

**creg**  
*Composition of Religious and Ethnic Groups (CREG) Fractionalization/Polarization Estimates*

Description

This is a data set with state-year estimates for ethnic and religious fractionalization/polarization, by way of the Composition of Religious and Ethnic Groups (CREG) project at the University of Illinois. I-L-L.

Usage

creg

Format

A data frame with 11523 observations on the following 9 variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccode</td>
<td>a Correlates of War state code</td>
</tr>
<tr>
<td>gwcode</td>
<td>a Gleditsch-Ward state code</td>
</tr>
<tr>
<td>creg_ccode</td>
<td>a numeric code for the state, mostly patterned off Correlates of War codes but with important differences. See details section for more.</td>
</tr>
<tr>
<td>year</td>
<td>the year</td>
</tr>
<tr>
<td>ethfrac</td>
<td>an estimate of the ethnic fractionalization index. See details for more.</td>
</tr>
<tr>
<td>ethpol</td>
<td>an estimate of the ethnic polarization index. See details for more.</td>
</tr>
<tr>
<td>relfrac</td>
<td>an estimate of the religious fractionalization index. See details for more.</td>
</tr>
<tr>
<td>relpol</td>
<td>an estimate of the religious polarization index. See details for more.</td>
</tr>
</tbody>
</table>
Details

The data-raw directory on the project’s Github contains more information about how these data were created. Pay careful attention to how I assigned CoW/G-W codes. The underlying data are version 1.02.

The state codes provided by the CREG project are mostly Correlates of War codes, but with some differences. Summarizing these differences: the state code for Serbia from 1992 to 2013 is actually the Gleditsch-Ward code (340). Russia after the dissolution of the Soviet Union (1991-onward) is 393 and not 365. The Soviet Union has the 365 code. Yugoslavia has the 345 code. The code for Yemen (678) is effectively the Gleditsch-Ward code because it spans the entire post-World War II temporal domain. Likewise, the code for post-unification Germany is the Gleditsch-Ward code (260) as well. The codebook actually says it’s 265 (which would be East Germany’s code), but this is assuredly a typo based on the data.

The codebook cautions there are insufficient data for ethnic group estimates for Cameroon, France, India, Kosovo, Montenegro, Mozambique, and Papua New Guinea. The French case is particularly disappointing but the missing data there are a function of both France’s constitution and modelling issues for CREG (per the codebook). There are insufficient data to make religious group estimates for China, North Korea, and the short-lived Republic of Vietnam.

The fractionalization estimates are the familiar Herfindahl-Hirschman concentration index. The polarization formula comes by way of Montalvo and Reynal-Querol (2000), though this book does not appear to be published beyond its placement online. I recommend Montalvo and Reynal-Querol (2005) instead. You can cite Alesina (2003) for the fractionalization measure if you’d like.

In the most literal sense of "1", the group proportions may not sum to exactly 1 because of rounding in the data. There were only two problem cases in these data worth mentioning. First, in both data sets, there would be the occasional duplicates of group names by state-year (for example: Afghanistan in 1951 in the ethnic group data and the United States in 1948 in the religious group data). In those cases, the script I make available in the data-raw directory just select distinct values and that effectively fixes the problem of duplicates, where they do appear. Finally, Costa Rica had a curious problem for most years in the religious group data. All Costa Rica years have group data for Protestants, Roman Catholics, and "others." Up until 1964 or so, the "others" are zero. Afterward, there is some small proportion of "others". However, the sum of Protestants, Roman Catholics, and "others" exceeds 1 (pretty clearly) and the difference between the sum and 1 is entirely the "others." So, I drop the "others" for all years. I don’t think that’s terribly problematic, but it’s worth saying that’s what I did.

References


Description

download_extdata() leverages R’s inst directory flexibility to allow you to download some extra data and store it in the package.

Usage

download_extdata(overwrite = FALSE)

Arguments

overwrite logical, defaults to FALSE. If FALSE, the function checks to see if you’ve already downloaded the data and, if you already have, it does nothing. If TRUE, the function redownsloads the data.

Value

download_extdata() downloads some extra data stored on my website (http://svmiller.com) and sticks them in the extdata directory in the package. Right now, these data are just the directed dyad-year Correlates of War trade data.

A Description of Various Data Sets This Will Download

Running download_extdata() returns the following data that will be stored in the package’s extdata directory.

Correlates of War Dyadic Trade Data Set (v. 4.0):

These are directed dyad-year-level data for national trade from the Correlates of War project.

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccode1</td>
<td>a numeric vector for the Correlates of War state code for the first state</td>
</tr>
<tr>
<td>ccode2</td>
<td>a numeric vector for the Correlates of War state code for the second state</td>
</tr>
<tr>
<td>year</td>
<td>the year</td>
</tr>
<tr>
<td>flow1</td>
<td>imports of ccode1 from ccode2, in current million USD</td>
</tr>
<tr>
<td>flow2</td>
<td>imports of ccode2 from ccode1, in current million USD</td>
</tr>
<tr>
<td>smoothflow1</td>
<td>smoothed flow1 values</td>
</tr>
<tr>
<td>smoothflow2</td>
<td>smoothed flow2 values</td>
</tr>
</tbody>
</table>

Author(s)

Steven V. Miller
References


Examples

```r
## Not run:
download_extdata()

## End(Not run)
```

### filter_prd

_filter dyad-year data to include just politically relevant dyads_

**Description**

filter_prd() filters a dyad-year data frame to just those that are "politically relevant." This is useful for discarding unnecessary (and unwanted) observations that just consume space in memory.

**Usage**

```r
filter_prd(data)
```

**Arguments**

- `data`: a dyad-year data frame (either "directed" or "non-directed")

**Details**

"Political relevance" can be calculated a few ways. Right now, the function considers only "direct" contiguity and Correlates of War major power status. You can employ maximalist definitions of "direct contiguity" to focus on just the land-contiguous. This function is inclusive of any type of contiguity relationship.

As of the slated release of version 0.5, filter_prd() is a shortcut for add_contiguity() and/or add_cow_majors() if the function is executed in the absence of the data needed to create politically relevant dyads. See the example below for what this means.

**Value**

filter_prd() takes a dyad-year data frame, assuming it has columns for major power status and contiguity type, calculates whether the dyad is "politically relevant", and subsets the data frame to just those observations.

**Author(s)**

Steven V. Miller
References


Examples

```r
# just call `library(tidyverse)` at the top of your script
library(magrittr)

A <- cow_ddy %>% add_contiguity() %>% add_cow_majors() %>% filter_prd()
A

# you can also use it as a shortcut for the other functions required
# to calculate politically relevant dyads.
B <- cow_ddy %>% filter_prd()
B

identical(A,B)
```

description

These are directed dispute-year data from the most recent version (2.2.1) of the Gibler-Miller-Little (GML) militarized interstate dispute (MID) data. They are used internally for merging into full dyad-year data frames.

Usage

gml_dirdisp

Format

A data frame with 10330 observations on the following 39 variables.

dispnum  the dispute number
code1    a numeric vector for the Correlates of War state code for the first state
code2    a numeric vector for the Correlates of War state code for the second state
year  a numeric vector for the year
midongoing  a constant of 1 for ongoing disputes
midonset  a numeric vector that equals 1 for the onset year of a given dispute
sidea1  is the first state (in ccode1) on the side that took the first militarized action?
sidea2  is the second state (in ccode2) on the side that took the first militarized action?
revstate1  is the first state (in ccode1) a revisionist state in the dispute?
revstate2  is the second state (in ccode2) a revisionist state in the dispute?
revtype11 what is the revtype1 value for ccode1?
revtype12 what is the revtype1 value for ccode2?
revtype21 what is the revtype2 value for ccode1?
revtype22 what is the revtype2 value for ccode2?
fatality1 what is the fatality value for ccode1?
fatality2 what is the fatality value for ccode2?
fatalpre1 what is the fatalpre value for ccode1?
fatalpre2 what is the fatalpre value for ccode2?
hiact1 what is the hiact value for ccode1?
hiact2 what is the hiact value for ccode2?
hostlev1 what is the hostlev value for ccode1?
hostlev2 what is the hostlev value for ccode2?
orig1 is ccode1 an originator (1) of the dispute or a joiner (0)?
orig2 is ccode2 an originator (1) of the dispute or a joiner (0)?
hiact  the highest level of action observed in the dispute
hostlev  the hostility level of action observed in the dispute
mindur  the minimum length of the dispute (in days)
maxdur  the maximum length of the dispute (in days)
outcome  the dispute-level outcome
settle  the settlement value for the dispute
fatality  the ordinal fatality level for the dispute
fatalpre  the fatalities (with precision, if known) for the dispute
stmon  the start month of the dispute (dispute-level)
endmon  the end month of the dispute (dispute-level)
recip  was the dispute reciprocated (i.e. did Side B also have a militarized action)?
numa  the number of participants on Side A
numb  the number of participants on Side B
ongo2010  was the dispute ongoing as of 2010?
version  a version identifier
Details
Data are the directed dispute-year data made available in version 2.1.1 of the GML MID data. I would caution against using the `revtype` variables. They are not informative. They are however included for legacy reasons.

References

Description
These are directed dyadic dispute year data derived from the Gibler-Miller-Little (GML) Militarized Interstate Dispute (MID) project. Data are from version 2.2.1. These were whittled to where there is no duplicate dyad-years. Its primary aim here is merging into a dyad-year data frame.

Usage
`gml_mid_ddydisps`

Format
A data frame with 9262 observations on the following 25 variables.
- `dispnum` a numeric vector for the dispute number
- `ccode1` a numeric vector for the focal state in the dyad
- `ccode2` a numeric vector for the target state in the dyad
- `year` a numeric vector for the dispute-year
- `gmlmidongoing` a numeric vector for whether there was a dispute ongoing in that year
- `gmlmidonset` a numeric vector for whether it was the onset of a new dispute (or new participant-entry into a recurring dispute)
- `sidea1` is `ccode1` on side A of the dispute?
- `sidea2` is `ccode2` on side A of the dispute?
- `fatality1` a numeric vector for the overall fatality level of `ccode1` in the dispute
- `fatality2` a numeric vector for the overall fatality level of `ccode2` in the dispute
- `fatalpre1` a numeric vector for the known fatalities (with precision) for `ccode1` in the dispute
- `fatalpre2` a numeric vector for the known fatalities (with precision) for `ccode2` in the dispute
- `hiact1` a numeric vector for the highest action of `ccode1` in the dispute
- `hiact2` a numeric vector for the highest action of `ccode2` in the dispute
hostlev1 a numeric vector for the hostility level of ccode1 in the dispute
hostlev2 a numeric vector for the hostility level of ccode2 in the dispute
orig1 is ccode1 an originator of the dispute?
orig2 is ccode2 an originator of the dispute?
fatality a numeric vector for the fatality level of the dispute
hostlev a numeric vector for the hostility level of the MID
mindur a numeric vector for the minimum duration of the MID
maxdur a numeric vector for the maximum duration of the MID
recip a numeric vector for whether a MID was reciprocated
stmon a numeric vector for the start month of the MID

Details

The process of creating these is described at one of the references below. Importantly, these data are somewhat "naive." That is: they won’t tell you, for example, that Brazil and Japan never directly fought each other during World War II. Instead, it will tell you that there were two years of overlap for the two on different sides of the conflict and that the highest action for both was a war. The data are thus similar to what the EUGene program would create for users back in the day. Use these data with that limitation in mind.

References


---

**gml_mid_disps**

*Abbreviated GML MID Dispute-level Data (v. 2.2.1)*

**Description**

This is an abbreviated version of the dispute-level Gibler-Miller-Little (GML) MID data.

**Usage**

`gml_mid_disps`
Format

A data frame with 2436 observations on the following 7 variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dispnum</td>
<td>a numeric vector for the CoW-MID dispute number</td>
</tr>
<tr>
<td>styear</td>
<td>a numeric vector for the start year of the MID</td>
</tr>
<tr>
<td>stmon</td>
<td>a numeric vector for the start month of the MID</td>
</tr>
<tr>
<td>outcome</td>
<td>a numeric vector for the outcome of the MID</td>
</tr>
<tr>
<td>settle</td>
<td>a numeric vector for the how dispute was settled</td>
</tr>
<tr>
<td>fatality</td>
<td>a numeric vector for the fatality level of the dispute</td>
</tr>
<tr>
<td>mindur</td>
<td>a numeric vector for the minimum duration of the MID</td>
</tr>
<tr>
<td>maxdur</td>
<td>a numeric vector for the maximum duration of the MID</td>
</tr>
<tr>
<td>hiact</td>
<td>a numeric vector for the highest action of the MID</td>
</tr>
<tr>
<td>hostlev</td>
<td>a numeric vector for the hostility level of the MID</td>
</tr>
<tr>
<td>recip</td>
<td>a numeric vector for whether a MID was reciprocated</td>
</tr>
</tbody>
</table>

Details

These data are purposely light on information; they’re not intended to be used for dispute-level analyses, per se. They’re intended to augment the directed dyadic dispute-year data by adding in variables that serve as exclusion rules to whittle the data from dyadic dispute-year to just dyad-year data.

References


Description

These are democracy data for all Correlates of War state system members.

Usage

gwcode_democracy
Format

A data frame with 18289 observations on the following 5 variables.

- gwcode: the Gleditsch-Ward system code
- year: a numeric vector for the year
- v2x_polyarchy: the Varieties of Democracy "polyarchy" estimate
- polity2: the the polity2 score from the Polity project
- xm_qudsest: an extension of the Unified Democracy Scores (UDS) estimates, made possibly by the QuickUDS package from Xavier Marquez.

Details

Missing data connote data that are unavailable for various reasons. Either there is no democracy data to code or, in the case of the Polity project, the state system member is outright not evaluated for the variable.

The Polity data are from 2017. The Varieties of Democracy data are version 10. Xavier Marquez’ QuickUDS estimates (i.e. extensions of Pemstein et al. (2010)) come from a package Marquez makes available on his Github (https://github.com/xmarquez/QuickUDS).

References


 gw_capitals

A complete list of capitals and capital transitions for Gleditsch-Ward state system members

Description

This is a complete list of capitals and capital transitions for Gleditsch-Ward state system members. I use it internally for calculating capital-to-capital distances in the add_capital_distances() function.
Usage

`gw_capitals`

Format

A data frame with 248 observations on the following 7 variables.

- `gwcode` a numeric vector for the Gleditsch-Ward state code
- `stname` a character vector for the state
- `capital` a character vector for the name of the capital
- `styear` a character vector for the start year. See details section for more information.
- `endyear` a character vector for the end year. See details section for more information.
- `lat` a numeric vector of the latitude coordinates for the capital
- `lng` a numeric vector of the longitude coordinates for the capital

Details

For convenience, the start year for most states is 1816. Samoa, for example, was not a state in 1816. However, the functions that use the `gw_capitals` data will not create observations for states that did not exist at a given point in time.

The data should be current as of the end of 2020.

Cases where a start year is not 1816 indicate a capital transition. For example, Brazil’s capital moved from Rio de Janeiro to Brasília (a planned capital) in 1960. Only 25 states in the data experienced a capital transition. The most recent was Burundi in 2018. Indonesia, as of writing, is planning on a capital transition, but this has not been completed yet.

Kazakhstan renamed its capital for the state leader in 2019. These data retain the name of Astana. This will be changed in the event the software I use records this change.

The capitals data are not without some peculiarities. Prominently, Portugal transferred the Portuguese court from Lisbon to Rio de Janeiro from 1808 to 1821. *This is recorded in the data.* A knowledge of the inter-state conflict data will note there was no war or dispute between, say, Portugal and Spain (or Portugal and any other country) at any point during this time, but it does create some weirdness that would suggest a massive distance between two countries, like Portugal and Spain, that are otherwise land-contiguous.

On Spain: the republican government moved the capital at the start of the civil war (in 1936) to Valencia. However, it abandoned this capital by 1937. I elect to not record this capital transition.

On Myanmar: the Gleditsch-Ward system stands out as having Myanmar entered for the bulk of the 19th century. The capitals recorded for Myanmar (Burma) coincide with capitals of the Konbaung dynasty.

The data also do some (I think) reasonable back-dating of capitals to coincide with states in transition without necessarily formal capitals by the first appearance in the state system membership data. These concern Lithuania, Kazakhstan, and the Philippines. Kaunas is the initial post-independence capital of Lithuania. Almaty is the initial post-independence capital of Kazakhstan. Quezon City is the initial post-independence capital of the Philippines. This concerns, at the most, one or two years for each of these three countries.
**gw_cow_years**

**Gleditsch-Ward states and Correlates of War, by year**

**Description**

This is a complete (I believe) data set on Gleditsch-Ward states and Correlates of War states, a byproduct of a `full_join()` between `gw_states` and `cow_states` that leans largely on the state abbreviation variable.

**Usage**

`gw_cow_years`

**Format**

A data frame with 18425 observations on the following 6 variables.

- `gwcode` a Gleditsch-Ward state code
- `stateabb` the state abbreviation, which was the greatest source of agreement between both data sets
- `gw_statename` the state name as it appears in the Gleditsch-Ward data
- `ccode` a Correlates of War state code
- `cow_statename` the state name as it appears in the Correlates of War data
- `year` a numeric vector for the year

**Details**

The `data-raw` directory on the project’s Github contains more information about how these data were created. I’m going to use it for internal stuff. The workflow is going to treat the Gleditsch-Ward state system membership codes as more of the “master” codes, for which the user can add Correlates of War identifiers as they see fit. Data are extended to 2020, assuming no changes to state system membership for either data set.

---

**gw_ddy**

**A directed dyad-year data frame of Gleditsch-Ward state system members**

**Description**

This is a complete directed dyad-year data frame of Gleditsch-Ward state system members. I offer it here as a shortcut for various other functions.

**Usage**

`gw_ddy`
Format

A data frame with 1999558 observations on the following 4 variables.

gwcode1 a numeric vector for the Correlates of War state code for the first state
gwcode2 a numeric vector for the Correlates of War state code for the second state
year a numeric vector for the year

Details

Data are a quick generation from the `create_dyadyears(system="gw")` function in this package.

---

gw_mindist  The Minimum Distance Between States in the Gleditsch-Ward System, 1886-2019

Description

These are non-directed dyad-year data for the minimum distance between states in the Gleditsch-Ward state system from 1886 to 2018. The data are generated from the `cshapes` package.

Usage

`gw_mindist`

Format

A data frame with 868813 observations on the following 4 variables.

gwcode1 the Gleditsch-Ward state system code for the first state
gwcode2 the Gleditsch-Ward state system code for the second state
year the year
mindist the minimum distance between states on Jan. 1 of the year, in kilometers

Details

The data are generated from the `cshapes` package. The package authors purport that the data are generated to be compatible with the Gleditsch-Ward system. I trust them on this; indeed, Gleditsch is one of the authors of the `cshapes` package.

Data are automatically generated (by default) as directed dyad-years. I elect to make them non-directed for space considerations. Making non-directed dyad-year data into directed dyad-year data isn’t too difficult in R. It just looks weird to see the code that does it.

Previous versions of these data were for the minimum distance as of Dec. 31 of the referent year. These are now Jan. 1. Most of the data I prove elsewhere in this package are to be understood as the data as they were at the *start* of the year. This is how I process, for example, the capitals data as they get merged in the `add_capital_distance()` function. However, the script that generates these data are set at Jan. 1 of the year and not Dec. 31. Right now, the `cshapes` does not appear to work on my system and I do not know why. Fortunately, the package authors made these data available.
## gw_sdp_gdp

### (Surplus and Gross) Domestic Product for Gleditsch-Ward States

#### Description

These are state-year level data for surplus and gross domestic product for Correlates of War state system members. Data also include population estimates for per capita standardization.

#### Usage

```r
gw_sdp_gdp
```

#### Format

A data frame with 27387 observations on the following five variables.

- **gwcode**: a numeric vector for the Gleditsch-Ward state code
- **year**: a numeric vector for the year
- **wbgdp2011est**: a numeric vector for the estimated natural log of GDP in 2011 USD (log-transformed)
- **wbpopest**: a numeric vector for the estimated population size (log-transformed)
- **sdpest**: a numeric vector for the estimated surplus domestic product (log-transformed)
- **wbgdppc2011est**: a numeric vector for the estimated GDP per capita (log-transformed)

#### Details

These were provided by Anders on a separate Github repository for this project. Because these data are ultimately being simulated, a user can expect some slight differences between the Correlates of War version of these data (which Anders et al. published) and the Gleditsch-Ward version of these data (which appear to be the one the authors will more vigorously support going forward).

#### References


Description
These are the independent states in Gleditsch and Ward’s data.

Usage

gw_states

Format
A data frame with 216 observations on the following 5 variables.

gwcode  a numeric vector for the Gleditsch-Ward country code
stateabb  a character vector for state abbreviation
statename  a character vector for the state name
startdate the start date in the data
enddate the end date in the data

Details
Data originally provided by Gleditsch with no column names. Column names were added before some light re-cleaning in order to generate these data.

References

Description
This is a data set with state-year estimates for ethnic fractionalization.

Usage

hief
maoz_powers

Format
A data frame with 8808 observations on the following 5 variables.

ccode  a Correlates of War state code
gwcode  a Gleditsch-Ward state code
year  the year
efindex  a numeric vector for the estimate of ethnic fractionalization

Details
The data-raw directory on the project’s Github contains more information about how these data were created.

References

maoz_powers  

Zeev Maoz’ Regional/Global Power Data

Description
These are Zeev Maoz’ data for what states are regional or global powers at a given point time. They are extensions of the Correlates of War major power data, which only codes “major” power without consideration of regional or global distinctions. Think of Austria-Hungary as intuitive of the issue here. Austria-Hungary is a major power in the Correlates of War data, but there is good reason to treat Austria-Hungary as a major power only within Europe. That is what Zeev Maoz tries to do here.

Usage
maoz_powers

Format
A data frame with 20 observations on the following 5 variables.

ccode  a numeric vector for the Correlates of War country code
regstdate  the start date for regional power status
regenddate  the end date for regional power status
globstdate  the start date for global power status
globenddate  the end date for global power status
References


Description

This is a BibTeX file, loaded as a data frame, to assist the user in properly citing the source material that is used in this package.

Usage

ps_bib

Format

A data frame with 37 observations on the following 40 variables.

- CATEGORY  the BibTeX entry type
- BIBTEXKEY  the BibTeX unique entry key
- ADDRESS    another BibTeX field
- ANNOTE     another BibTeX field
- AUTHOR     a list of authors for this entry
- BOOKTITLE  another BibTeX field, for book title (if appropriate)
- CHAPTER    another BibTeX field, for chapter (if appropriate)
- CROSSREF   another BibTeX field
- EDITION    another BibTeX field, for edition of book (if appropriate)
- EDITOR     another BibTeX field, for book editor (if appropriate)
- HOWPUBLISHED another BibTeX field
- INSTITUTION another BibTeX field
- JOURNAL    another BibTeX field, for the journal name (if appropriate)
- KEY        another BibTeX field
- MONTH      another BibTeX field
- NOTE       another BibTeX field
- NUMBER     another BibTeX field, for journal volume number (if appropriate)
- ORGANIZATION another BibTeX field
- PAGES      another BibTeX field, for pages of the entry
- PUBLISHER  another BibTeX field, for book publisher (if appropriate)
- SCHOOL     another BibTeX field
SERIES another BibTeX field
TITLE another BibTeX field, for title of the entry
TYPE another BibTeX field
VOLUME another BibTeX field, for journal volume (if appropriate)
YEAR another BibTeX field, for year of publication
KEYWORDS another BibTeX field, used primarily for selective filtering in this package
URL another BibTeX field, for website (if appropriate)
OWNER another BibTeX field
TIMESTAMP another BibTeX field, used occasionally when I started populating my master file (you will see some old entries here)
DOI another BibTeX field, for a digital object identifier (used rarely)
EPRINT another BibTeX field
JOURNALTITLE another BibTeX field, which I think is actually a BibLaTeX field
ISSN another BibTeX field
ABSTRACT another BibTeX field, for entry abstract (if appropriate)
DATE.ADDED another BibTeX field
DATE.MODIFIED another BibTeX field

Details

See data-raw directory for how these data were generated. The data were created by bib2df, which is now a package dependency. I assume the user has some familiarity with BibTeX. Some entries were copy-pasted from my master bibliography file that I started in 2008 or so.

---

ps_cite

Get BibTeX Entries Associated with peacescience Data and Functions

Description

ps_cite() allows the user to get citations to scholarship that they should include in their papers that incorporate the functions and data in this package.

Usage

ps_cite(x)

Arguments

x a character vector
Details

The base functionality here is simple pattern-matching on keywords in ps_bib. This simple pattern-matching is in base R. I assume the user has some familiarity with BibTeX.

Value

ps_cite() takes a character vector and scans the ps_bib data in this package to return a BibTeX citation (or citations) for the researcher to use to properly cite the material they are getting from this package. The citations are returned as a full BibTeX entry (or entries) that they can copy-paste into their own BibTeX file.

Author(s)

Steven V. Miller

Examples

# You can cite the package
ps_cite("peacesciencer")

# You can do partial matching
ps_cite("democracy")

# Or more partial matching
ps_cite("alliance")

# You can also get all citations for a particular function
ps_cite("add_archigos()")

rugged  
Rugged/Mountainous Terrain Data

Description

This is a data set on state-level estimates for the “ruggedness” of a state’s terrain.

Usage

rugged

Format

A data frame with 192 observations on the following 6 variables.

ccode  a Correlates of War state code
gwcode  a Gleditsch-Ward state code
rugged  the terrain ruggedness index
newlmtnest  the (natural log) percentage estimate of the state’s terrain that is mountainous
show_duplicates

Details
The data-raw directory on the project’s Github contains more information about how these data were created. It goes without saying that these data move *slowly* so the data are really only applicable for making state-to-state comparisons and not states-in-time comparisons. The terrain ruggedness index is originally introduced by Riley et al. (1999) but is amended by Nunn and Puga (2012). The mountain terrain data was originally created by Fearon and Laitin (2003) but extended and amended by Gibler and Miller (2014). The data are functionally time-agnostic—use with caution in your state-year analyses—but all data sets seem to benchmark around 1999-2000. I’m not sure it matters *that* much, but it matters a little at the margins, I suppose, if you suspect there are major differences in interpretation of how much more "rugged" the Soviet Union was than Russia, or Yugoslavia than Serbia.

References

show_duplicates() shows which data are duplicated in data generated in peacesciencer. It’s a useful diagnostic tool for users doing some do-it-yourself functions with peacesciencer.

Usage

show_duplicates(data)

Arguments

data a dyad-year data frame or a state-year data frame created in peacesciencer.

Details
The function leans on attributes of the data that are provided by the create_dyadyear() or create_stateyear() function. Make sure that function (or data created by that function) appear at the top of the proverbial pipe.
The data returned will also have a new column called duplicated. Thus, an implicit assumption in this function is the user does not have a column in the data with this name that is of interest to the user. It will be overwritten.
show_duplicates() takes a dyad-year data frame or state-year data frame generated in peace-sciencer and shows what observations are duplicated by unique combination of dyad-year or state-year, contingent on what was supplied to it.

Author(s)
Steven V. Miller

Examples

# just call 'library(tidyverse)' at the top of the your script
library(magrittr)
gml_dirdisp %>% show_duplicates()
cow_mid_dirdisps %>% show_duplicates()

Description
A simple summary of all strategic (inter-state) rivalries from Thompson and Dreyer (2012).

Usage
td_rivalries

Format
A data frame with 197 observations on the following 10 variables.
rivalryno a numeric vector for the rivalry number
rivalryname a character vector for the rivalry name
ccode1 the Correlates of War state code for the state with the lowest Correlates of War state code in the rivalry
ccode2 the Correlates of War state code for the state with the highest Correlates of War state code in the rivalry
styear a numeric vector for the start year of the rivalry
endyear a numeric vector for the end year of the rivalry
region a character vector for the region of the rivalry, per Thompson and Dreyer (2012)
type1 a character vector for the primary type of the rivalry (spatial, positional, ideological, or interventionary)
type2 a character vector for the secondary type of the rivalry, if applicable (spatial, positional, ideological, or interventionary)
type3 a character vector for the tertiary type of the rivalry, if applicable (spatial, positional, ideological, or interventionary)
ucdp_acd

Details

Information gathered from the appendix of Thompson and Dreyer (2012). Ongoing rivalries are right-bound at 2010, the date of publication for Thompson and Dreyer’s handbook. Users are free to change this if they like. Data are effectively identical to strategic_rivalries in stevemisc, but include some behind-the-scenes processing (described in a blog post on http://svmiller.com) that is available to see on the project’s Github repository. The data object is also renamed to avoid a conflict.

References


ucdp_acd  UCDP Armed Conflict Data (ACD) (v. 20.1)

Description

These are (kind of) dyadic, but mostly state-level data, used internally for doing stuff with the UCDP armed conflict data

Usage

ucdp_acd

Format

A data frame with 4164 observations on the following 15 variables.

conflict_id  a conflict identifier, not to be confused with an episode identifier (which I don’t think UCDP offers)
year  a numeric vector for the year
gwno_a  the Gleditsch-Ward state code for the state on side A of the armed conflict
gwno_a_2nd  the Gleditsch-Ward state code for the state that actively supported side A of the armed conflict with the use of troops
gwno_b  the Gleditsch-Ward state code for the actor on side B of the armed conflict
gwno_b_2nd  the Gleditsch-Ward state code for the state that actively supported side B of the armed conflict with the use of troops
incompatibility  a character vector for the main conflict issue ("territory", "government", "both")
intensity_level  a numeric vector for the intensity level in the calendar year (1 = minor (25-999 deaths), 2 = war (>1,000 deaths))
type_of_conflict  a character vector for the type of conflict ("extrasystemic", "interstate", "intrastate", "II"). "II" is a simple abbreviation of "internationalized intrastate"
ucdp_onsets

start_date  a date of the first battle-related death in the conflict, not to be confused with the first battle-related death of the episode

start_prec  the level of precision for start_date

start_date2  a date of the first battle-related death in the episode, not to be confused with the first battle-related death of the conflict

start_prec2  the level of precision for start_date2

ep_end  a dummy variable for whether the conflict episode ended in the calendar year of observation

ep_end_date  the episode end date, if applicable

Details

The data-raw directory on the project’s Github will show how I processed the multiple strings for when there are multiple states on a given side.

References


ucdp_onsets  UCDP Onset Data (v. 19.1)

Description

These are state-year level data for armed conflict onsets provided by the Uppsala Conflict Data Program (UCDP).

Usage

ucdp_onsets

Format

A data frame with 10142 observations on the following eight variables.

- gwcode  a numeric vector for the Gleditsch-Ward state code
- year  a numeric vector for the year
- sumnewconf  a numeric vector for the sum of new conflicts/conflict-dyads
- sumonset1  a numeric vector for the sum of new conflict episodes, whether because this is a new conflict or because there is more than one year since last conflict episode
- sumonset2  a numeric vector for the sum of new conflict episodes, whether because this is a new conflict or because there is more than two years since last conflict episode
sumonset3 a numeric vector for the sum of new conflict episodes, whether because this is a new conflict or because there is more than three years since last conflict episode
sumonset5 a numeric vector for the sum of new conflict episodes, whether because this is a new conflict or because there is more than five years since last conflict episode
sumonset10 a numeric vector for the sum of new conflict episodes, whether because this is a new conflict or because there is more than 10 years since last conflict episode

Details

The user will want to note that the data provided by UCDP are technically not country-year observations. They instead duplicate observations for cases of new conflicts or new conflict episodes. Further, the original data do not provide any information about the conflict-dyad in question to which those duplicates pertain. That means the most these data can do for the package’s mission is provide summary information. The user should probably recode these variables into something else they may want for a particular application

References


Description

whittle_conflicts_duration() is in a class of do-it-yourself functions for coercing (i.e. "whit- tling") conflict-year data with cross-sectional units to unique conflict-year data by cross-sectional unit. The inspiration here is clearly the problem of whittling dyadic dispute-year data into true dyad-year data (like in the Gibler-Miller-Little conflict data). This particular function will keep the observations with the highest estimated duration.

Usage

whittle_conflicts_duration(data, durtype = "mindur")
wc_duration(...)

Arguments

data a data frame with a declared conflict attribute type.
durtype a duration on which to filter/whittle the data. Options include "mindur" or "maxdur". The default is "mindur".
... optional, only to make the shortcut work
Details

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID#0112, MID#0113, MID#0306), as illustrative of the problem. The default process in `peacesciencer` employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame. These are available in `add_cow_mids()` and `add_gml_mids()`.

Some conflicts can be of an unknown length and often come with estimates of a minimum duration and a maximum duration. This will concern the `durtype` parameter in this function. In many/most conflicts, certainly thinking of the inter-state dispute data, dates are known with precision (to the day) and the estimate of minimum conflict duration is equal to the estimate of maximum conflict duration. For some conflicts, the estimates will vary. This does importantly imply that using this particular whittle function with the default (`mindur`) will produce different results than using this particular whittle function and asking to retain the highest maximum duration (`maxdur`). Use the function with that in mind.

`wc_duration()` is a simple, less wordy, shortcut for the same function.

Value

`whittle_conflicts_duration()` takes a dyad-year data frame with a declared conflict attribute type and, grouping by the dyad and year, returns just those observations that have the highest observed dispute-level fatality. This will not eliminate all duplicates, far from it, but it’s a sensible cut later into the procedure (after whitting onsets in `whittle_conflicts_onsets()`), and maybe some other things the extent to which dispute-level duration is a heuristic for dispute-level severity/importance.

Author(s)

Steven V. Miller

References


Examples

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
gml_dirdisp %>% whittle_conflicts_onsets() %>% whittle_conflicts_duration()
cow_mid_dirdisps %>% whittle_conflicts_onsets() %>% whittle_conflicts_duration()
```
whittle_conflicts_fatality

**Whittle Duplicate Conflict-Years by Highest Fatality**

**Description**

`whittle_conflicts_fatality()` is in a class of do-it-yourself functions for coercing (i.e. "whittling") conflict-year data with cross-sectional units to unique conflict-year data by cross-sectional unit. The inspiration here is clearly the problem of whittling dyadic dispute-year data into true dyad-year data (like in the Gibler-Miller-Little conflict data). This particular function will keep the observations with the highest observed fatality.

**Usage**

```r
whittle_conflicts_fatality(data)
wc_fatality(...)```

**Arguments**

- `data` a data frame with a declared conflict attribute type.
- `...` optional, only to make the shortcut work

**Details**

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID#0112, MID#0113, MID#0306), as illustrative of the problem. The default process in `peacesciencer` employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame. These are available in `add_cow_mids()` and `add_gml_mids()`.

As of writing, the Correlates of War and Gibler-Miller-Little conflict data record some -9s for fatalities. In those cases, dispute-level fatality is momentarily recoded to be .5 (i.e. fatal, but without too many fatalities). This is a missing data problem that Gibler and Miller correct in a forthcoming publication in *Journal of Conflict Resolution*. Until then, this function makes that kind of determination about disputes with missing fatalities.

`wc_fatality()` is a simple, less wordy, shortcut for the same function.

**Value**

`whittle_conflicts_fatality()` takes a dyad-year data frame with a declared conflict attribute type and, grouping by the dyad and year, returns just those observations that have the highest observed dispute-level fatality. This will not eliminate all duplicates, far from it, but it's a sensible second cut (after whittling onsets in `whittle_conflicts_onsets()`) the extent to which dispute-level fatality is a good heuristic for dispute-level severity/importance.
**whittle_conflicts_hostility**

**Author(s)**

Steven V. Miller

**References**


**Examples**

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
gml_dirdisp %>% whittle_conflicts_onsets() %>% whittle_conflicts_fatality()
cow_mid_dirdisp %>% whittle_conflicts_onsets() %>% whittle_conflicts_fatality()
```

---

**whittle_conflicts_hostility**

*Whittle Duplicate Conflict-Years by Conflict Hostility*

**Description**

`whittle_conflicts_hostility()` is in a class of do-it-yourself functions for coercing (i.e. "whittling") conflict-year data with cross-sectional units to unique conflict-year data by cross-sectional unit. The inspiration here is clearly the problem of whittling dyadic dispute-year data into true dyad-year data (like in the Gibler-Miller-Little conflict data). This particular function will keep the observations with the highest observed hostility.

**Usage**

```r
whittle_conflicts_hostility(data)
```

**Arguments**

- `data` a data frame with a declared conflict attribute type.
- `...` optional, only to make the shortcut work
Details

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID#0112, MID#0113, MID#0306), as illustrative of the problem. The default process in peacesciencer employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame. These are available in add_cow_mids() and add_gml_mids().

wc_hostility() is a simple, less wordy, shortcut for the same function.

Value

whittle_conflicts_hostility() takes a dyad-year data frame with a declared conflict attribute type and, grouping by the dyad and year, returns just those observations that have the highest observed dispute-level fatality. This will not eliminate all duplicates, far from it, but it’s a sensible second or third cut (after whittling onsets in whittle_conflicts_onsets() the extent to which dispute-level hostility is a good heuristic for dispute-level severity/importance.

Author(s)

Steven V. Miller

References

Miller, Steven V. 2021. "How peacesciencer Coerces Dispute-Year Data into Dyad-Year Data".
URL: http://svmiller.com/peacesciencer/articles/coerce-dispute-year-dyad-year.html

Examples

# just call `library(tidyverse)` at the top of your script
library(magrittr)
gml_dirdisp %>% whittle_conflicts_onsets() %>% whittle_conflicts_hostility()
cow_mid_dirdisps %>% whittle_conflicts_onsets() %>% whittle_conflicts_hostility()
**Description**

*whittle_conflicts_jds()* is in a class of do-it-yourself functions for coercing (i.e. "whittling") conflict-year data with cross-sectional units to unique conflict-year data by cross-sectional unit. The inspiration here is clearly the problem of whittling dyadic dispute-year data into true dyad-year data (like in the Gibler-Miller-Little conflict data). This particular function will just drop something, as a kind of nuclear option.

**Usage**

```r
whittle_conflicts_jds(data)

wc_jds(...)```

**Arguments**

data: a data frame with a declared conflict attribute type.

... optional, only to make the shortcut work

**Details**

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID#0112, MID#0113, MID#0306), as illustrative of the problem. The default process in *peacesciencer* employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame. These are available in *add_cow_mids()* and *add_gml_mids()*.

This really should be the absolute last exclusion rules a researcher uses. It’s a "nuclear option", if you will. Assuming you’ve run other case exclusion rules to isolate onsets and severe disputes, what remains at the end should be duplicates that are functionally equivalent observations. Your data cannot have duplicates, and these remaining observations are basically the same. Therefore, just drop something.

*wc_jds()* is a simple, less wordy, shortcut for the same function.

**Value**

*whittle_conflicts_jds()* takes a dyad-year data frame with a declared conflict attribute type and, grouping by the dyad and year, returns just those observations that have the lowest start month.

**Author(s)**

Steven V. Miller

**References**

Examples

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
gml_dirdisp %>% whittle_conflicts_onsets() %>% whittle_conflicts_jds()

cow_mid_dirdisps %>% whittle_conflicts_onsets() %>% whittle_conflicts_jds()
```

### Description

`whittle_conflicts_onsets()` is in a class of do-it-yourself functions for coercing (i.e. "whittling") conflict-year data with cross-sectional units to unique conflict-year data by cross-sectional unit. The inspiration here is clearly the problem of whittling dyadic dispute-year data into true dyad-year data (like in the Gibler-Miller-Little conflict data). This particular function will drop ongoing conflicts in the presence of unique onsets.

### Usage

```r
whittle_conflicts_onsets(data)

wc_onsets(...)
```

### Arguments

- **data**
  - a data frame with a declared conflict attribute type.
- **...**
  - optional, only to make the shortcut work

### Details

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID#0112, MID#0113, MID#0306), as illustrative of the problem. The default process in `peacesciencer` employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame. These are available in `add_cow_mids()` and `add_gml_mids()`.

`wc_onsets()` is a simple, less wordy, shortcut for the same function.
whittle_conflicts_reciprocation

**Value**

`whittle_conflicts_onsets()` takes a dyad-year data frame with a declared conflict attribute type and, grouping by the dyad and year, returns just those observations with unique onsets where duplicates exist. This will not eliminate all duplicates, far from it, but it’s a sensible place to start.

**Author(s)**

Steven V. Miller

**References**


**Examples**

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
gml_dirdisp %>% whittle_conflicts_onsets()
cow_mid_dirdisps %>% whittle_conflicts_onsets()
```

**Description**

`whittle_conflicts_reciprocation()` is in a class of do-it-yourself functions for coercing (i.e. "whittling") conflict-year data with cross-sectional units to unique conflict-year data by cross-sectional unit. The inspiration here is clearly the problem of whittling dyadic dispute-year data into true dyad-year data (like in the Gibler-Miller-Little conflict data). This particular function will keep the observations that are reciprocated (i.e. have militarized actions on both sides of the conflict).

**Usage**

```r
whittle_conflicts_reciprocation(data)
```

wc_recip(...)

whittle_conflicts_reciprocation

*Whittle Duplicate Conflict-Years by Conflict Reciprocation*
Arguments

data a data frame with a declared conflict attribute type.

Details

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID#0112, MID#0113, MID#0306), as illustrative of the problem. The default process in peacesciencer employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame. These are available in add_cow_mids() and add_gml_mids().

Scholars are free to use this as a heuristic for whittling conflict-year data to be coerced into true dyad-year data, but I would be remiss if I did not offer a caveat about the reciprocation variable in inter-state dispute data. Namely, it is noisy and is not doing what scholars often think it’s doing in the inter-state dispute data. Reciprocation is observed only when there is a militarized action on both sides of the conflict. By definition, someone on Side A will have a militarized action. Not every state on Side B does. However, scholars should *not* interpret that as the absence of militarized responses. In a forthcoming article in Journal of Conflict Resolution, Doug Gibler and I make the case that reciprocation isn’t a useful variable to maintain at all because it can only invite errors (as is often the case in the CoW-MID data) and will obscure the fact that states that are attacked by another side routinely fight back. On many occasions, they also successfully repel the attack. Scholars who uncritically use this variable, certainly for hypothesis-testing on audience costs, are borrowing trouble with this measure.

wc_recip() is a simple, less wordy, shortcut for the same function.

Value

whittle_conflicts_reciprocation() takes a dyad-year data frame with a declared conflict attribute type and, grouping by the dyad and year, returns just those observations that have militarized actions on both sides of the conflict. This will not eliminate all duplicates, far from it, but it’s a sensible cut later into the procedure (after whittling onsets in whittle_conflicts_onsets(), and maybe some other things the extent to which dispute-level reciprocation is a heuristic for dispute-level severity/importance (after some other considerations).

Author(s)

Steven V. Miller

References


Examples
whittle_conflicts_startmonth

Whittle Duplicate Conflict-Years by Lowest Start Month

Description

whittle_conflicts_startmonth() is in a class of do-it-yourself functions for coercing (i.e. "whit-tling") conflict-year data with cross-sectional units to unique conflict-year data by cross-sectional unit. The inspiration here is clearly the problem of whittling dyadic dispute-year data into true dyad-year data (like in the Gibler-Miller-Little conflict data). This particular function will keep the observations that have the lowest start month.

Usage

whittle_conflicts_startmonth(data)

wc_stmon(...)  

Arguments

data a data frame with a declared conflict attribute type.

... optional, only to make the shortcut work

Details

Dyads are capable of having multiple disputes in a given year, which can create a problem for merging into a complete dyad-year data frame. Consider the case of France and Italy in 1860, which had three separate dispute onsets that year (MID#0112, MID#0113, MID#0306), as illustrative of the problem. The default process in peacesciencer employs several rules to whittle down these duplicate dyad-years for merging into a dyad-year data frame. These are available in add_cow_mids() and add_gml_mids().

This really should be one of the last exclusion rules a researcher uses. There is no substantive reason to assume the lower start month matters for the cause of isolating "serious" or "severe" disputes in the presence of duplicates. It's really just a way of isolating which duplicated observation happened first where remaining duplicates are otherwise very similar to each other.

wc_stmon() is a simple, less wordy, shortcut for the same function.
whittle_conflicts_startmonth

Value

`whittle_conflicts_startmonth()` takes a dyad-year data frame with a declared conflict attribute type and, grouping by the dyad and year, returns just those observations that have the lowest start month.

Author(s)

Steven V. Miller

References


Examples

```r
# just call `library(tidyverse)` at the top of the your script
library(magrittr)
gml_dirdisp %>% whittle_conflicts_onsets() %>% whittle_conflicts_startmonth()

cow_mid_dirdisps %>% whittle_conflicts_onsets() %>% whittle_conflicts_startmonth()
```
Index

* datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>archigos</td>
<td>33</td>
</tr>
<tr>
<td>atop_alliance</td>
<td>34</td>
</tr>
<tr>
<td>ccode_democracy</td>
<td>35</td>
</tr>
<tr>
<td>cow_alliance</td>
<td>36</td>
</tr>
<tr>
<td>cow_capitals</td>
<td>37</td>
</tr>
<tr>
<td>cow_contdir</td>
<td>38</td>
</tr>
<tr>
<td>cow_ddy</td>
<td>39</td>
</tr>
<tr>
<td>cow_gw_years</td>
<td>39</td>
</tr>
<tr>
<td>cow_igo_ndy</td>
<td>40</td>
</tr>
<tr>
<td>cow_igo_sy</td>
<td>41</td>
</tr>
<tr>
<td>cow_majors</td>
<td>42</td>
</tr>
<tr>
<td>cow_mid_ddydisps</td>
<td>43</td>
</tr>
<tr>
<td>cow_mid_dirdisps</td>
<td>44</td>
</tr>
<tr>
<td>cow_mid_disps</td>
<td>45</td>
</tr>
<tr>
<td>cow_mindist</td>
<td>46</td>
</tr>
<tr>
<td>cow_nmc</td>
<td>47</td>
</tr>
<tr>
<td>cow_sdp_gdp</td>
<td>48</td>
</tr>
<tr>
<td>cow_states</td>
<td>49</td>
</tr>
<tr>
<td>cow_trade_sy</td>
<td>50</td>
</tr>
<tr>
<td>cow_war_inter</td>
<td>51</td>
</tr>
<tr>
<td>cow_war_intra</td>
<td>52</td>
</tr>
<tr>
<td>creg</td>
<td>58</td>
</tr>
<tr>
<td>gml_dirdisp</td>
<td>62</td>
</tr>
<tr>
<td>gml_mid_ddydisps</td>
<td>64</td>
</tr>
<tr>
<td>gml_mid_disps</td>
<td>65</td>
</tr>
<tr>
<td>gw_capitals</td>
<td>67</td>
</tr>
<tr>
<td>gw_cow_years</td>
<td>69</td>
</tr>
<tr>
<td>gw_ddy</td>
<td>69</td>
</tr>
<tr>
<td>gw_mindist</td>
<td>70</td>
</tr>
<tr>
<td>gw_sdp_gdp</td>
<td>71</td>
</tr>
<tr>
<td>gw_states</td>
<td>72</td>
</tr>
<tr>
<td>gwcode_democracy</td>
<td>66</td>
</tr>
<tr>
<td>hief</td>
<td>72</td>
</tr>
<tr>
<td>maoz_powers</td>
<td>73</td>
</tr>
<tr>
<td>ps_bib</td>
<td>74</td>
</tr>
<tr>
<td>rugged</td>
<td>76</td>
</tr>
<tr>
<td>td_rivalries</td>
<td>78</td>
</tr>
<tr>
<td>ucdp_acd</td>
<td>79</td>
</tr>
<tr>
<td>ucdp_onsets</td>
<td>80</td>
</tr>
<tr>
<td>add_archigos</td>
<td>3</td>
</tr>
<tr>
<td>add_atop_alliance</td>
<td>4</td>
</tr>
<tr>
<td>add_capital_distance</td>
<td>5</td>
</tr>
<tr>
<td>add_ccode_to_gw</td>
<td>6</td>
</tr>
<tr>
<td>add_contiguity</td>
<td>8</td>
</tr>
<tr>
<td>add_cow_alliance</td>
<td>9</td>
</tr>
<tr>
<td>add_cow_majors</td>
<td>10</td>
</tr>
<tr>
<td>add_cow_mids</td>
<td>11</td>
</tr>
<tr>
<td>add_cow_trade</td>
<td>12</td>
</tr>
<tr>
<td>add_cow_wars</td>
<td>13</td>
</tr>
<tr>
<td>add_creg_fractionalization</td>
<td>15</td>
</tr>
<tr>
<td>add_democracy</td>
<td>16</td>
</tr>
<tr>
<td>add_gml_mids</td>
<td>18</td>
</tr>
<tr>
<td>add_gwcode_to_cow</td>
<td>19</td>
</tr>
<tr>
<td>add_igos</td>
<td>20</td>
</tr>
<tr>
<td>add_minimum_distance</td>
<td>21</td>
</tr>
<tr>
<td>add_nmc</td>
<td>23</td>
</tr>
<tr>
<td>add_peace_years</td>
<td>24</td>
</tr>
<tr>
<td>add_ruggedTerrain</td>
<td>26</td>
</tr>
<tr>
<td>add_sdp_gdp</td>
<td>27</td>
</tr>
<tr>
<td>add_strategic_rivalries</td>
<td>29</td>
</tr>
<tr>
<td>add_ucdp_acd</td>
<td>30</td>
</tr>
<tr>
<td>add_ucdp_onsets</td>
<td>31</td>
</tr>
<tr>
<td>archigos</td>
<td>33</td>
</tr>
<tr>
<td>atop_alliance</td>
<td>34</td>
</tr>
<tr>
<td>ccode_democracy</td>
<td>35</td>
</tr>
<tr>
<td>cow_alliance</td>
<td>36</td>
</tr>
<tr>
<td>cow_capitals</td>
<td>37</td>
</tr>
<tr>
<td>cow_contdir</td>
<td>38</td>
</tr>
<tr>
<td>cow_ddy</td>
<td>39</td>
</tr>
<tr>
<td>cow_gw_years</td>
<td>39</td>
</tr>
<tr>
<td>cow_igo_ndy</td>
<td>40</td>
</tr>
<tr>
<td>cow_igo_sy</td>
<td>41</td>
</tr>
<tr>
<td>cow_majors</td>
<td>42</td>
</tr>
<tr>
<td>cow_mid_ddydisps</td>
<td>43</td>
</tr>
<tr>
<td>cow_mid_dirdisps</td>
<td>44</td>
</tr>
<tr>
<td>cow_mid_disps</td>
<td>45</td>
</tr>
<tr>
<td>ccode_democracy</td>
<td>35</td>
</tr>
<tr>
<td>cow_alliance</td>
<td>36</td>
</tr>
<tr>
<td>cow_capitals</td>
<td>37</td>
</tr>
<tr>
<td>cow_contdir</td>
<td>38</td>
</tr>
<tr>
<td>cow_ddy</td>
<td>39</td>
</tr>
<tr>
<td>cow_gw_years</td>
<td>39</td>
</tr>
<tr>
<td>cow_igo_ndy</td>
<td>40</td>
</tr>
<tr>
<td>cow_igo_sy</td>
<td>41</td>
</tr>
<tr>
<td>cow_majors</td>
<td>42</td>
</tr>
<tr>
<td>cow_mid_ddydisps</td>
<td>43</td>
</tr>
<tr>
<td>cow_mid_dirdisps</td>
<td>44</td>
</tr>
<tr>
<td>cow_mid_disps</td>
<td>45</td>
</tr>
</tbody>
</table>
INDEX

cow_mindist, 46
cow_nmc, 47
cow_sdp_gdp, 48
cow_states, 49
cow_trade_sy, 50
cow_war_inter, 51
cow_war_intra, 52
create_dyadyears, 53
create_leaderyears, 55
create_statedays, 56
create_stateyears, 57
creg, 58
download_extdata, 60
filter_prd, 61
gml_dirdisp, 62
gml_mid_ddydisps, 64
gml_mid_disps, 65
gw_capitals, 67
gw_cow_years, 69
gw_ddy, 69
gw_mindist, 70
gw_sdp_gdp, 71
gw_states, 72
gwcode_democracy, 66
hief, 72
maoz_powers, 73
ps_bib, 74
ps_cite, 75
rugged, 76
show_duplicates, 77
td_rivalries, 78
ucdp_acd, 79
ucdp_onsets, 80
wc_duration
  (whittle_conflicts_duration), 81
wc_fatality
  (whittle_conflicts_fatality), 83
wc_jds (whittle_conflicts_jds), 85
wc_onsets (whittle_conflicts_onsets), 87
wc_recip
  (whittle_conflicts_reciprocation), 88
wc_stmon
  (whittle_conflicts_startmonth), 90
whittle_conflicts_duration, 81
whittle_conflicts_fatality, 83
whittle_conflicts_hostility, 84
whittle_conflicts_jds, 85
whittle_conflicts_onsets, 87
whittle_conflicts_reciprocation, 88
whittle_conflicts_startmonth, 90