Package 'pvsR'

February 20, 2015

Type Package

Title An R package to interact with the Project Vote Smart API for scientific research

Depends R (>= 3.0)

Imports XML, nnet, httr

Version 0.3

Date 2014-08-13

Author Ulrich Matter

Maintainer Ulrich Matter <ulrich.matter@unibas.ch>

Description The pvsR package facilitates data retrieval from Project Vote Smart's rich online data base on US politics via the Project Vote Smart application programming interface (PVS API). The functions in this package cover most PVS API classes and methods and return the requested data in a data frame.

License GPL-2

Note In order to use this package you need your personal Project Vote Smart API key. To get an API-key you need to sign up for access to PVS' API (see http://votesmart.org/share/api for details). Whenever you are using pvsR make sure that your personal PVS API key is saved as a character string in the variable pvs.key (pvs.key <- ``yourkey''). If there is no valid PVS API key saved in the variable pvs.key the functions in this package won't work. This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-23 09:53:40

R topics documented:

`CandidateBio.getAddlBio` *Get a candidate's additional biographical information*

Description

This function is a wrapper for the CandidateBio.getAddlBio() method of the PVS API CandidateBio class which grabs the extended biographical information for each candidate that has one. The function sends a request with this method to the PVS API for all candidate-IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
CandidateBio.getAddlBio(candidateId)
```

Arguments

`candidateId`	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A data frame with a row for each candidate and columns with the following variables describing the candidate:
addlBio.candidate.shortTitle,
addlBio.candidate.firstName,
addlBio.candidate.nickName,
addlBio.candidate.middleName,
addlBio.candidate.lastName,
addlBio.candidate.suffix,
addlBio.additional.item*.name,
addlBio.additional.item*.data

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/CandidateBio.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get additional biographical data on Barack Obama
## Not run: obama <- CandidateBio.getAddlBio(9490)
## Not run: obama
# get additional biographical data on Barack Obama and Mitt Romney
## Not run: onr <- CandidateBio.getAddlBio(list(9490,21942))
## Not run: onr
```

CandidateBio.getBio *Get a candidate's main biographical information*

Description

This function is a wrapper for the CandidateBio.getBio() method of the PVS API CandidateBio class which grabs the main biographical information for each candidate. The function sends a request with this method to the PVS API for all candidate IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
CandidateBio.getBio(candidateId)
```

Arguments

candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A data frame with a row for each candidate and columns with the following variables describing the candidate:
bio.candidate.crpId (OpenSecrets ID),
bio.candidate.firstName,
bio.candidate.nickName,
bio.candidate.middleName,
bio.candidate.lastName,
bio.candidate.suffix,
bio.candidate.birthDate,
bio.candidate.birthPlace,
bio.candidate.pronunciation,
bio.candidate.gender,
bio.candidate.family,
bio.candidate.photo,
bio.candidate.homeCity,
bio.candidate.homeState,
bio.candidate.education,
bio.candidate.profession,
bio.candidate.political,
bio.candidate.religion,
bio.candidate.congMembership,
bio.candidate.orgMembership,
bio.candidate.specialMsg,
bio.office.parties,
bio.office.title,
bio.office.shortTitle,
bio.office.name,
bio.office.type,
bio.office.status,
bio.office.firstElect,
bio.office.lastElect,
bio.office.nextElect,
bio.office.termStart,
bio.office.termEnd,
bio.office.district,
bio.office.districtId,
bio.office.stateId,
bio.office.committee*.committeeId,
bio.office.committee*.committeeName,
bio.election*.office,
bio.election*.officeId,
bio.election*.officeType,

bio.election*.parties,
bio.election*.district,
bio.election*.districtId,
bio.election*.status,
bio.election*.ballotName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/CandidateBio.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get main biographical data on Barack Obama and Mitt Romney
## Not run: bio <- CandidateBio.getBio(list(9490,21942))
## Not run: bio
```

CandidateBio.getDetailedBio

Get a candidate's detailed biographical information

Description

This function is a wrapper for the CandidateBio.getDetailedBio() method of the PVS API CandidateBio class which grabs the detailed biographical information for each candidate. The function sends a request with this method to the PVS API for all candidate IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
CandidateBio.getDetailedBio(candidateId)
```

Arguments

candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A list with several data frames containing the elements of CandidateBio.getBio(), and expands upon:
bio.candidate.education.degree,
bio.candidate.education.field,
bio.candidate.education.school,
bio.candidate.education.span,
bio.candidate.education.gpa,
bio.candidate.education.fullText,
bio.candidate.profession.title,
bio.candidate.profession.organization,
bio.candidate.profession.span,
bio.candidate.profession.special,
bio.candidate.profession.district,
bio.candidate.profession.fullText,
bio.candidate.political.title,
bio.candidate.political.organization,
bio.candidate.political.span,
bio.candidate.political.special,
bio.candidate.political.district,
bio.candidate.political.fullText,
bio.candidate.congMembership.title,
bio.candidate.congMembership.organization,
bio.candidate.congMembership.span,
bio.candidate.congMembership.special,
bio.candidate.congMembership.district,
bio.candidate.congMembership.fullText,
bio.candidate.orgMembership.title,
bio.candidate.orgMembership.organization,
bio.candidate.orgMembership.span,
bio.candidate.orgMembership.special,
bio.candidate.orgMembership.district,
bio.candidate.orgMembership.fullText.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/CandidateBio.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
```

```
# get main biographical data on Barack Obama and Mitt Romney
## Not run: bio <- CandidateBio.getDetailedBio(list(9490,21942))
## Not run: head(bio$profession)
## Not run: head(bio$orgMembership)
```

`Candidates.getByDistrict`

Get a list of candidates according to the district they represent

---

Description

This function is a wrapper for the Candidates.getByDistrict() method of the PVS API Candidates class which grabs a list of candidates according to the district they represent for each district and election year. The function sends a request with this method to the PVS API for all district IDs and election years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Candidates.getByDistrict(districtId, electionYear=NULL)
```

Arguments

`districtId`	a character string or list of character strings with the district ID(s) (see references for details)
`electionYear`	(optional) a character string or list of character strings with the election year(s) (Default: >= current year)

Value

A data frame with a row for each candidate and year and columns with the following variables describing the candidate:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.preferredName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.ballotName,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionStatus,
candidateList.candidate*.electionStage,
candidateList.candidate*.electionDistrictId,
candidateList.candidate*.electionDistrictName,
candidateList.candidate*.electionOffice,

candidateList.candidate*.electionOfficeId,
candidateList.candidate*.electionStateId,
candidateList.candidate*.electionOfficeTypeId,
candidateList.candidate*.electionYear,
candidateList.candidate*.electionSpecial,
candidateList.candidate*.electionDate,
candidateList.candidate*.officeParties,
candidateList.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeStateId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.runningMateId,
candidateList.candidate*.runningMateName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Candidates.html
Use District.getByOfficeState() or District.getByZip() to get a list of district ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of candidates of certain districts
## Not run: district <- Candidates.getByDistrict(list(25157,25155),2012)
## Not run: district
```

Candidates.getByElection

Get a list of candidates according to the election they are running for

---

Description

This function is a wrapper for the Candidates.getByElection() method of the PVS API Candidates class which grabs a list of candidates according to the election they are running for. The function sends a request with this method to the PVS API for all electionIDs and stageIDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Candidates.getByElection(electionId, stageId=NULL)
```

Arguments

electionId	a character string or list of character strings with the election ID(s) (see references for details)
stageId	(optional) a character string or list of character strings with the stage ID(s) (default: all)

Value

A data frame with a row for each candidate and columns with the following variables describing the candidate:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.preferredName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.ballotName,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionStatus,
candidateList.candidate*.electionStage,
candidateList.candidate*.electionDistrictId,
candidateList.candidate*.electionDistrictName,
candidateList.candidate*.electionOffice,
candidateList.candidate*.electionOfficeId,
candidateList.candidate*.electionStateId,
candidateList.candidate*.electionOfficeTypeId,
candidateList.candidate*.electionYear,
candidateList.candidate*.electionSpecial,
candidateList.candidate*.electionDate,
candidateList.candidate*.officeParties,
candidateList.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeStateId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.runningMateId,
candidateList.candidate*.runningMateName

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Candidates.html
Use Election.getElectionByYearState() or Election.getElectionByZip() to get election ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of candidates for certain election IDs
## Not run: candidates <- Candidates.getByElection(list(2582,2646))
## Not run: candidates
```

Candidates.getByLastname

Get a list of candidates according to their lastname

Description

This function is a wrapper for the Candidates.getByLastname() method of the PVS API Candidates class which grabs a list of candidates according to a lastname match. The function sends a request with this method to the PVS API for all last names and election years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Candidates.getByLastname(lastName, electionYear=NULL)
```

Arguments

lastName	a character string or list of character strings with the last name(s) (see references for details)
electionYear	(optional) a character string or list of character strings with the election year(s) (default: >= current year)

Value

A data frame with a row for each candidate and year and columns with the following variables describing the candidate:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.preferredName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,

candidateList.candidate*.ballotName,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionStatus,
candidateList.candidate*.electionStage,
candidateList.candidate*.electionDistrictId,
candidateList.candidate*.electionDistrictName,
candidateList.candidate*.electionOffice,
candidateList.candidate*.electionOfficeId,
candidateList.candidate*.electionStateId,
candidateList.candidate*.electionOfficeTypeId,
candidateList.candidate*.electionYear,
candidateList.candidate*.electionSpecial,
candidateList.candidate*.electionDate,
candidateList.candidate*.officeParties,
candidateList.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeStateId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.runningMateId,
candidateList.candidate*.runningMateName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Candidates.html
Use CandidateBio.getBio(), Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByElection(), Candidates.getByDistrict(), Candidates.getByZip(), Committee.getCommitteeMembers(), Election.getStageCandidates(), Leadership.getOfficials(), Local.getOfficials(), Officials.getStatewide(), Officials.getByOfficeState(), Officials.getByOfficeTypeState(), Officials.getByDistrict() or Officials.getByZip() to get last name(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get candidates by last names
## Not run: names <- Candidates.getByLastname(list("Obama","Romney"),2012)
## Not run: names
```

`Candidates.getByLevenshtein` *Get a list of candidates according to an approximate lastname match*

Description

This function is a wrapper for the Candidates.getByLevenshtein() method of the PVS API Candidates class which grabs a list of candidates according to an approximate lastname match. The function sends a request with this method to the PVS API for all last names and election years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Candidates.getByLevenshtein(lastName, electionYear=NULL)
```

Arguments

`lastName`	a character string or list of character strings with the last name(s) (see references for details)
`electionYear`	(optional) a character string or list of character strings with the election year(s) (default: >= current year)

Value

A data frame with a row for each candidate and year and columns with the following variables describing the candidate:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.preferredName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.ballotName,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionStatus,
candidateList.candidate*.electionStage,
candidateList.candidate*.electionDistrictId,
candidateList.candidate*.electionDistrictName,
candidateList.candidate*.electionOffice,
candidateList.candidate*.electionOfficeId,
candidateList.candidate*.electionStateId,
candidateList.candidate*.electionOfficeTypeId,
candidateList.candidate*.electionYear,
candidateList.candidate*.electionSpecial,

candidateList.candidate*.electionDate,
candidateList.candidate*.officeParties,
candidateList.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeStateId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.runningMateId,
candidateList.candidate*.runningMateName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Candidates.html
Use CandidateBio.getBio(), Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByElection(), Candidates.getByDistrict(), Candidates.getByZip(), Committee.getCommitteeMembers(), Election.getStageCandidates(), Leadership.getOfficials(), Local.getOfficials(), Officials.getStatewide(), Officials.getByOfficeState(), Officials.getByOfficeTypeState(), Officials.getByDistrict() or Officials.getByZip() to get last name(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get candidates with similar last names
## Not run: names <- Candidates.getByLevenshtein(list("Obama","Romney"),2012)
## Not run: names
```

Candidates.getByOfficeState

Get a list of candidates according to office and state representation

---

Description

This function is a wrapper for the Candidates.getByOfficeState() method of the PVS API Candidate class which grabs a list of candidates according to office and state representation. The function sends a request with this method to the PVS API for all state IDs, office IDs and election years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Candidates.getByOfficeState(stateId="NA", officeId, electionYear=NULL, all=FALSE)
```

Arguments

stateId	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)
officeId	a character string or list of character strings with the office ID(s) (see references for details)
electionYear	(optional) a character string or list of character strings with the election year(s) (default: >= current year)
all	a logical indicator; if TRUE data on all possible combinations of the input variables are returned, if FALSE (default) only the exact combinations of them (see example)

Value

A data frame with a row for each candidate and year and columns with the following variables describing the candidate:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.preferredName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.ballotName,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionStatus,
candidateList.candidate*.electionStage,
candidateList.candidate*.electionDistrictId,
candidateList.candidate*.electionDistrictName,
candidateList.candidate*.electionOffice,
candidateList.candidate*.electionOfficeId,
candidateList.candidate*.electionStateId,
candidateList.candidate*.electionOfficeTypeId,
candidateList.candidate*.electionYear,
candidateList.candidate*.electionSpecial,
candidateList.candidate*.electionDate,
candidateList.candidate*.officeParties,
candidateList.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeStateId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.runningMateId,
candidateList.candidate*.runningMateName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Candidates.html
Use State.getStateIDs() to get a list of state IDs.
See http://api.votesmart.org/docs/semi-static.html for a list of office IDs or use Office.getOfficesByType(), Office.getOfficesByLevel(), Office.getOfficesByTypeLevel() or Office.getOfficesByBranchLevel() to get a list of office ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a data frame of candidates according to all state/office/electionYear combinations
## Not run: candidates <- Candidates.getByOfficeState(stateId=list("NJ","NY"),officeId=list(6,7),
electionYear=list(2012,2008), all=TRUE)
## End(Not run)
## Not run: candidates
# get a data frame of candidates according to the exact state/office/electionYear combinations
# (i.e., "NY"/6/2012, "NJ"/7/2008)
## Not run: candidates <- Candidates.getByOfficeState(stateId=list("NJ","NY"),officeId=list(6,7),
 electionYear=list(2012,2008), all=FALSE)
## End(Not run)
## Not run: candidates
```

Candidates.getByOfficeTypeState

Get a list of candidates according to office type and state representation

---

Description

This function is a wrapper for the Candidates.getByOfficeTypeState() method of the PVS API Candidate class which grabs a list of candidates according to office type and state representation. The function sends a request with this method to the PVS API for all state IDs, office type IDs and election years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Candidates.getByOfficeTypeState(stateId="NA", officeTypeId, electionYear=NULL, all=FALSE)
```

Arguments

stateId	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)
officeTypeId	a character string or list of character strings with the office type ID(s) (see references for details)
electionYear	(optional) a character string or list of character strings with the election year(s) (default: >= current year)
all	a logical indicator; if TRUE data on all possible combinations of the input variables are returned, if FALSE (default) only the exact combinations of them (see example)

Value

A data frame with a row for each candidate and columns with the following variables describing the candidate:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.preferredName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.ballotName,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionStatus,
candidateList.candidate*.electionStage,
candidateList.candidate*.electionDistrictId,
candidateList.candidate*.electionDistrictName,
candidateList.candidate*.electionOffice,
candidateList.candidate*.electionofficeTypeId,
candidateList.candidate*.electionStateId,
candidateList.candidate*.electionOfficeTypeId,
candidateList.candidate*.electionYear,
candidateList.candidate*.electionSpecial,
candidateList.candidate*.electionDate,
candidateList.candidate*.officeParties,
candidateList.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeStateId,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.runningMateId,
candidateList.candidate*.runningMateName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Candidates.html
Use State.getStateIDs() to get a list of state IDs.
See http://api.votesmart.org/docs/semi-static.html or use Office.getTypes() or Office.getOfficesByLevel() to get a list of office types ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a data frame of candidates according to all state/officeType/electionYear combinations
## Not run: candidates <- Candidates.getByOfficeTypeState(stateId=list("NJ","NY"),
officeTypeId=list("C","L"), electionYear=list(2012,2008), all=TRUE)
## End(Not run)
## Not run: candidates
# get a data frame of candidates according to the exact state/officeType/electionYear combinations
# (i.e., "NY"/6/2012, "NJ"/7/2008)
## Not run: candidates <- Candidates.getByOfficeTypeState(stateId=list("NJ","NY"),
officeTypeId=list("C","L"), electionYear=list(2012,2008), all=FALSE)
## End(Not run)
## Not run: candidates
```

Candidates.getByZip *Get a list of candidates according to ZIP code*

Description

This function is a wrapper for the Candidates.getByZip() method of the PVS API Candidates class which grabs a list of candidates according to the ZIP code. The function sends a request with this method to the PVS API for all ZIP codes and election years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Candidates.getByZip(zip5, electionYear=NULL)
```

Arguments

zip5	a character string or list of character strings with the 5-digit ZIP code(s)
electionYear	(optional) a character string or list of character strings with the election year(s) (default: >= current year)

Value

A data frame with a row for each candidate and columns with the following variables describing the candidate:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.preferredName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.ballotName,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionStatus,
candidateList.candidate*.electionStage,
candidateList.candidate*.electionDistrictId,
candidateList.candidate*.electionDistrictName,
candidateList.candidate*.electionOffice,
candidateList.candidate*.electionOfficeId,
candidateList.candidate*.electionStateId,
candidateList.candidate*.electionOfficeTypeId,
candidateList.candidate*.electionYear,
candidateList.candidate*.electionSpecial,
candidateList.candidate*.electionDate,
candidateList.candidate*.officeParties,
candidateList.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeStateId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.runningMateId,
candidateList.candidate*.runningMateName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Candidates.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of candidates according to the ZIP code.
## Not run: candidates <- Candidates.getByZip(list(10001,10002),2012)
```

```
## Not run: candidates
```

`Committee.getCommittee` *Get detailed committee (contact) information*

Description

This function is a wrapper for the Committee.getCommittee() method of the PVS API Committee class which returns detailed committee data. The function sends a request with this method to the PVS API for all committee IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Committee.getCommittee(committeeId)
```

Arguments

`committeeId`	a character string or list of character strings with the committee ID(s) (see references for details)

Value

A data frame with a row for each committee and columns with the following variables describing the committee:
committee.committeeId,
committee.parentId,
committee.stateId,
committee.committeeTypeId,
committee.name,
committee.jurisdiction,
committee.contact.address1,
committee.contact.address2,
committee.contact.city,
committee.contact.state,
committee.contact.zip,
committee.contact.phone,
committee.contact.fax,
committee.contact.email,
committee.contact.url,
committee.contact.staffContact.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Committee.html
Use CandidateBio.getBio(), Committee.getCommitteesByTypeState() or Votes.getBill() to get committee ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get information about a certain committee
## Not run: committee <- Committee.getCommittee(list(1,2))
## Not run: committee
```

Committee.getCommitteeMembers

Get a list of members of a committee

Description

This function is a wrapper for the Committee.getCommitteeMembers() method of the PVS API Committee class which returns a list of members of a committee. The function sends a request with this method to the PVS API for all committee IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Committee.getCommitteeMembers(committeeId)
```

Arguments

committeeId	a character string or list of character strings with the committee ID(s) (see references for details)

Value

A data frame with a row for each committee member and columns with the following variables describing the committee member:
committeeMembers.committee.committeeId,
committeeMembers.committee.parentId,
committeeMembers.committee.name,
committeeMembers.member*.candidateId,
committeeMembers.member*.title,
committeeMembers.member*.firstName,
committeeMembers.member*.middleName,
committeeMembers.member*.lastName,
committeeMembers.member*.suffix,
committeeMembers.member*.party,
committeeMembers.member*.position.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Committee.html
Use CandidateBio.getBio(), Committee.getCommitteesByTypeState() or Votes.getBill() to get committee ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of members of certain committees
## Not run: comember <- Committee.getCommitteeMembers(1)
## Not run: comember
```

Committee.getCommitteesByTypeState

Get a list of committees according to type and state

Description

This function is a wrapper for the Committee.getCommitteesByTypeState() method of the PVS API Committee class which returns a list of committees for each type in each requested state. The function sends a request with this method to the PVS API for all type and state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Committee.getCommitteesByTypeState(typeId=list("H","S","J"), stateId="NA", all=FALSE)
```

Arguments

typeId	(optional) a character string or list of character strings with the type ID(s) (default: All) (see references for details)
stateId	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)
all	a logical indicator; if TRUE data on all possible combinations of the input variables are returned, if FALSE (default) only the exact combinations of them (see example)

Value

A data frame with a row for each committee and columns with the following variables describing the committee:
committees.committee*.committeeId,
committees.committee*.parentId,
committees.committee*.stateId,
committees.committee*.committeeTypeId,
committees.committee*.name.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Committee.html\cr See http://api.votesmart.org/docs/semi-static.html for a list of committee-type ID(s).
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a data frame of committees according to all type/state combinations
## Not run: committees <- Committee.getCommitteesByTypeState(typeId=list("H","S"),
stateId=list("NY","NJ"), all=TRUE)
## End(Not run)
## Not run: committees
# get a data frame of committees according to the exact type/state combinations
# (i.e., "H"/"NY", "S"/"NJ")
## Not run: committees <- Committee.getCommitteesByTypeState(typeId=list("H","S"),
stateId=list("NY","NJ"), all=FALSE)
## End(Not run)
## Not run: committees
```

Committee.getTypes *Get the committee types (house, senate, joint, etc.)*

Description

This function is a wrapper for the Committee.getTypes() method of the PVS API Committee class which returns the existing committee types.

Usage

```
Committee.getTypes()
```

Value

A data frame with a row for each committee type and columns with the following variables describing the committee type:
committeeTypes.type*.committeeTypeId,
committeeTypes.type*.name.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Committee.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get the committee types
## Not run: comtypes <- Committee.getTypes()
## Not run: comtypes
```

District.getByOfficeState

Get district IDs according to the office and state

Description

This function is a wrapper for the District.getByOfficeState() method of the PVS API District class which grabs district IDs according to the office and state. The function sends a request with this method to the PVS API for all district names and office IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
District.getByOfficeState(officeId, stateId, districtName=NULL, all=FALSE)
```

Arguments

officeId	a character string or list of character strings with the office ID(s) (see references for details)
stateId	a character string or list of character strings with the state ID(s) (see references for details)
districtName	(optional) a character string or list of character strings with the district name (default: All)
all	a logical indicator; if TRUE data on all possible combinations of the input variables are returned, if FALSE (default) only the exact combinations of them (see example)

Value

A data frame with a row for each district and columns with the following variables describing the district:
districtList.district*.districtId,
districtList.district*.name,
districtList.district*.officeId,
districtList.district*.stateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/District.html
See http://api.votesmart.org/docs/semi-static.html for a list of office IDs or use Office.getOfficesByType(), Office.getOfficesByLevel(), Office.getOfficesByTypeLevel() or Office.getOfficesByBranchLevel().
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get districts for certain office and state IDs
## Not run: districts <- District.getByOfficeState(officeId=list(8,9),stateId=list("NY","NJ"))
## Not run: district
# get a data frame of districts according to all state/office/districtName combinations
## Not run: districts <- District.getByOfficeState(officeId=list(8,9),stateId=list("NY","NJ"),
 districtName=list(1,2), all=TRUE)
## End(Not run)
## Not run: districts
# get a data frame of districts according to the exact state/office/districtName combinations
# (i.e., 8/"NY"/1, 9/"NJ"/2)
## Not run: districts <- District.getByOfficeState(officeId=list(8,9),stateId=list("NY","NJ"),
 districtName=list(1,2), all=FALSE)
## End(Not run)
## Not run: districts
```

`District.getByZip` *Get district IDs according to the zip code*

Description

This function is a wrapper for the District.getByZip() method of the PVS API District class which grabs district IDs according to the ZIP code. The function sends a request with this method to the PVS API for all ZIP codes given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
District.getByZip(zip5, zip4=NULL)
```

Arguments

zip5	a character string or list of character strings with the five-digit ZIP code
zip4	(optional) a character string or list of character strings with the expanded ZIP+4 code (default: All)

Value

A data frame with a row for each district and columns with the following variables describing the district:
districtList.district*.districtId,
districtList.district*.name,
districtList.district*.officeId,
districtList.district*.stateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/District.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get district IDs by ZIP Code
## Not run: district <- District.getByZip(list(10001,10002),)
## Not run: district
```

Election.getElection *Get district basic election data*

Description

This function is a wrapper for the Election.getElection() method of the PVS API Election class which grabs district basic election data according to the election ID. The function sends a request with this method to the PVS API for all election IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Election.getElection(electionId)
```

Arguments

electionId	a character string or list of character strings with the election ID(s) (see references for details)

Value

A data frame with a row for each election and columns with the following variables describing the election:
elections.election*.electionId,
elections.election*.name,
elections.election*.stateId,
elections.election*.officeTypeId,
elections.election*.special,
elections.election*.electionYear.
For each stage the following variables are jointly (as one string) in a column:
elections.election*.stage*.stageId,
elections.election*.stage*.name,
elections.election*.stage*.stateId,
elections.election*.stage*.electionDate,
elections.election*.stage*.filingDeadline,
elections.election*.stage*.npatMailed.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Election.html
Use Election.getElectionByYearState() or Election.getElectionByZip() to get election ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get data of a certain election
## Not run: election <- Election.getElection(as.list(2595:2607))
## Not run: election
```

Election.getElectionByYearState

Get district basic election data according to year and state ID

Description

This function is a wrapper for the Election.getElectionByYearState() method of the PVS API Election class which grabs district basic election data. The function sends a request with this method to the PVS API for all state IDs and years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Election.getElectionByYearState(stateId="NA", year)
```

Arguments

stateId	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)
year	a character string or list of character strings with the years (defaults to current year)

Value

A data frame with a row for each election and columns with the following variables describing the election:
elections.election*.electionId,
elections.election*.name,
elections.election*.stateId,
elections.election*.officeTypeId,
elections.election*.special,
elections.election*.electionYear.
For each stage the following variables are jointly (as one string) in a column:
elections.election*.stage*.stageId,
elections.election*.stage*.name,
elections.election*.stage*.stateId,
elections.election*.stage*.electionDate,
elections.election*.stage*.filingDeadline,
elections.election*.stage*.npatMailed.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Election.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get information about an election of a certain year and state
## Not run: election <- Election.getElectionByYearState(list("NY","FL"),2012)
## Not run: election
```

`Election.getElectionByZip` *Get district basic election data according to the ZIP code*

Description

This function is a wrapper for the Election.getElectionByZip() method of the PVS API Election class which grabs district basic election data according to the ZIP code. If another year than the current year is chosen, all election data from that year up to the current year is returned. The function sends a request with this method to the PVS API for all ZIP codes given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Election.getElectionByZip(zip5, zip4=NULL, year=NULL)
```

Arguments

`zip5`	a character string or list of character strings with the five-digit ZIP code
`zip4`	(optional) a character string or list of character strings with the expanded ZIP+4 code (default: NULL)
`year`	a character string or list of character strings with the year (defaults to current year)

Value

A data frame with a row for each election and columns with the following variables describing the election:
elections.election*.electionId,
elections.election*.name,
elections.election*.stateId,
elections.election*.officeTypeId,
elections.election*.special,
elections.election*.electionYear.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Election.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get election data by ZIP code
## Not run: election <- Election.getElectionByZip(zip5=list(10001,10002), year="2012")
## Not run: election
```

`Election.getStageCandidates`

Get district basic election data according to election ID and stage ID

Description

This function is a wrapper for the Election.getStageCandidates() method of the PVS API Election class which grabs district basic election data according to the election and stage ID. The function sends a request with this method to the PVS API for all election and stage IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Election.getStageCandidates(stageId, electionId)
```

Arguments

`electionId`	a character string or list of character strings with the election ID(s) (see references for details)
`stageId`	a character string or list of character strings with the stage ID(s)

Value

A data frame with a row for each combination of stage ID and election and columns with the following variables describing the election:
stagecandidates.election*.electionId,
stagecandidates.election*.name,
stagecandidates.election*.stage,
stagecandidates.election*.stateId,
stagecandidates.candidate*.candidateId,
stagecandidates.candidate*.officeId,
stagecandidates.candidate*.district,
stagecandidates.candidate*.firstName,
stagecandidates.candidate*.middleName,
stagecandidates.candidate*.lastName,
stagecandidates.candidate*.suffix,
stagecandidates.candidate*.party,
stagecandidates.candidate*.status,
stagecandidates.candidate*.voteCount,
stagecandidates.candidate*.votePercent.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Election.html
Use Election.getElectionByYearState() or Election.getElectionByZip() to get election ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get election data for certain election IDs
## Not run: election <- Election.getStageCandidates(stageId=list("P"),electionId=list(2582,2607))
## Not run: election
```

---

getAllBios *Get several candidates' biographical information*

Description

This function is essentially a wrapper around CandidateBio.getBio() specified for large amount of requests.

Usage

```
getAllBios(candidateId, batchsize=100, pause=0, backupfile="bios.list.Rdata")
```

Arguments

candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)
batchsize	numerical, indicating how many candidateIds should be processed in one batch (defaults to 100).
pause	numerical, indicating how long (in seconds) the download process should be paused after each batch (defaults to 0)
backupfile	character string for the path/file-name of the Rdata-file where the data should be saved (batch-wise) during the download process (default: "bios.list.Rdata").

Details

This functions splits large requests into several batches. The requests are then processed batch-wise and are saved on the local disc to make sure that not too much RAM is assigned to the pvsR task.

Value

A data frame with a row for each candidate and columns with the following variables describing the candidate:
bio.candidate.crpId (OpenSecrets ID),
bio.candidate.firstName,
bio.candidate.nickName,
bio.candidate.middleName,
bio.candidate.lastName,
bio.candidate.suffix,
bio.candidate.birthDate,
bio.candidate.birthPlace,
bio.candidate.pronunciation,
bio.candidate.gender,
bio.candidate.family,
bio.candidate.photo,
bio.candidate.homeCity,
bio.candidate.homeState,
bio.candidate.education,
bio.candidate.profession,
bio.candidate.political,
bio.candidate.religion,
bio.candidate.congMembership,
bio.candidate.orgMembership,
bio.candidate.specialMsg,
bio.office.parties,
bio.office.title,
bio.office.shortTitle,
bio.office.name,
bio.office.type,
bio.office.status,
bio.office.firstElect,
bio.office.lastElect,
bio.office.nextElect,
bio.office.termStart,
bio.office.termEnd,
bio.office.district,
bio.office.districtId,
bio.office.stateId,
bio.office.committee*.committeeId,
bio.office.committee*.committeeName,
bio.election*.office,
bio.election*.officeId,
bio.election*.officeType,
bio.election*.parties,
bio.election*.district,
bio.election*.districtId,
bio.election*.status,
bio.election*.ballotName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/CandidateBio.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get all officials of a certain state
## Not run: officials <- Officials.getStatewide("FL")
# get all biographical information on those officials
## Not run: bios <- getAllBios(officials$candidateId[1:100], batchsize=20)
## Not run: head(bios)
```

---

getAllCities *Get basic data on all cities*

Description

This function is essentially a wrapper around Local.getCities().

Usage

```
getAllCities()
```

Value

A data frame with a row for each city and columns with the following variables describing the city:
cities.city*.localId,
cities.city*.name,
cities.city*.url.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Local.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of all cities
## Not run: cities <- getAllCities()
## Not run: head(cities)
```

`getAllCounties` *Get basic data on all counties*

Description

This function is essentially a wrapper around Local.getCounties().

Usage

```
getAllCounties()
```

Value

A data frame with a row for each county and columns with the following variables describing the county:
counties.county*.localId,
counties.county*.name,
counties.county*.url.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Local.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of all counties
## Not run: counties <- getAllCounties()
## Not run: head(counties)
```

getAllDistricts *Get basic data on all districts*

Description

This function is essentially a wrapper around District.getByOfficeState().

Usage

```
getAllDistricts()
```

Value

A data frame with a row for each district and columns with the following variables describing the district:
districtList.district*.districtId,
districtList.district*.name,
districtList.district*.officeId,
districtList.district*.stateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/District.html
See http://api.votesmart.org/docs/semi-static.html for a list of office IDs or use Office.getOfficesByType(), Office.getOfficesByLevel(), Office.getOfficesByTypeLevel() or Office.getOfficesByBranchLevel().
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of all districts
## Not run: districts <- getAllDistricts()
## Not run: head(districts)
```

`getAllLocalOfficials` *Fetch data on all local (city- or county-) officials*

Description

This function is essentially a wrapper around Local.getOfficials().

Usage

```
getAllLocalOfficials(locality="counties", batchsize=50,
 pause=0, backupfile="locofs.list.Rdata")
```

Arguments

`locality`	a character string indicating whether data on county-officials ("counties") or city-officials ("cities") should be downloaded.
`batchsize`	numerical, indicating the number of requests that should be processed in one batch (defaults to 50).
`pause`	numerical, indicating how long (in seconds) the download process should be paused after each batch (defaults to 0)
`backupfile`	character string for the path/file-name of the Rdata-file where the data should be saved (batch-wise) during the download process (default: "locofs.list.Rdata").

Details

This functions splits large requests into several batches. The requests are then processed batch-wise and are saved on the local disc to make sure that not too much RAM is assigned to the pvsR task.

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidatelist.candidate*.candidateId,
candidatelist.candidate*.firstName,
candidatelist.candidate*.nickName,
candidatelist.candidate*.middleName,
candidatelist.candidate*.lastName,
candidatelist.candidate*.suffix,
candidatelist.candidate*.title,
candidatelist.candidate*.electionParties,
candidatelist.candidate*.electionDistrictId,
candidatelist.candidate*.electionStateId,
candidatelist.candidate*.officeParties,
candidatelist.candidate*.officeDistrictId,
candidatelist.candidate*.officeDistrictName,
candidatelist.candidate*.officeStateId,

candidatelist.candidate*.officeId,
candidatelist.candidate*.officeName,
candidatelist.candidate*.officeTypeId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Local.html
Use Local.getCounties() or Local.getCities() to get a list of local IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of all local officials
## Not run: all_countyofficials <- getAllLocalOfficials()
## Not run: head(officials)
```

---

getAllVotes *Get several votes*

Description

This function is essentially a wrapper around Votes.getBillActionVotes() specified for large amount of requests.

Usage

```
getAllVotes(actionId, batchsize=100, pause=0, backupfile="votes.list.Rdata")
```

Arguments

actionId	a character string or list of character strings with the action ID(s) (see references for details)
batchsize	numerical, indicating how many actionIds should be processed in one batch (defaults to 100).
pause	numerical, indicating how long (in seconds) the download process should be paused after each batch (defaults to 0)
backupfile	character string for the path/file-name of the Rdata-file where the data should be saved (batch-wise) during the download process (default: "votes.list.Rdata").

Details

This functions splits large requests into several batches. The requests are then processed batch-wise and are saved on the local disc to make sure that not too much RAM is assigned to the pvsR task.

Value

A data frame with a row for each vote and columns with the following variables describing the vote:
votes.vote*.candidateId,
votes.vote*.candidateName,
votes.vote*.officeParties,
votes.vote*.action.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Votes.getBill() or Votes.getByOfficial() to get a list of action IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get all officials of a certain state
## Not run: bill <- Votes.getBill("17623", separate=c("actions", "sponsors"))
## Not run: actionids <- bill$actions$actionId
# get all votes on this acti
## Not run: votes <- getAllVotes(actionids, batchsize=2)
## Not run: head(votes)
```

---

getOffices *Get basic data on all offices*

Description

This function is essentially a wrapper around Office.getOfficesByLevel().

Usage

```
getOffices()
```

Value

A data frame with a row for each office and columns with the following variables describing the office:
offices.office*.officeId,
offices.office*.officeTypeId,
offices.office*.officeLevelId,
offices.office*.officeBranchId,
offices.office*.name,
offices.office*.title,
offices.office*.shortTitle.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Office.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of all offices on all levels
## Not run: offices <- getOffices()
## Not run: head(offices)
```

Leadership.getOfficials

Get officials that hold the leadership role in certain states

---

Description

This function is a wrapper for the Leadership.getOfficials() method of the PVS API Leadership class which grabs a list of officials that hold the leadership role in certain states. The function sends a request with this method to the PVS API for all state and leadership IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Leadership.getOfficials(stateId="NA", leadershipId)
```

Arguments

stateId	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)
leadershipId	a character string or list of character strings with the leadership ID(s) (see references for details)

Value

A data frame with a row for each leadership position and columns with the following variables describing the official:
leaders.leader*.candidateId,
leaders.leader*.firstName,
leaders.leader*.middleName,
leaders.leader*.lastName,
leaders.leader*.suffix,
leaders.leader*.position,
leaders.leader*.officeId,
leaders.leader*.title.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Leadership.html
Use State.getStateIDs() to get a list of state IDs.
Use Leadership.getPositions() to get a list of leadership IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get leaders by state ID and leadership ID
## Not run: officials <- Leadership.getOfficials(list("NY","FL"),list(138,140))
## Not run: officials
```

Leadership.getPositions

Get leadership positions by state and office

Description

This function is a wrapper for the Leadership.getPositions() method of the PVS API Leadership class which returns leadership positions by state and office. The function sends a request with this method to the PVS API for all state and office IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Leadership.getPositions(stateId="NA", officeId=NULL)
```

Arguments

stateId	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)
officeId	(optional) a character string or list of character strings with the office ID(s) (default: All) (see references for details)

Value

A data frame with a row for each leadership position and columns with the following variables describing the position:
leadership.position*.leadershipId,
leadership.position*.name,
leadership.position*.officeId,
leadership.position*.officeName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Leadership.html
Use State.getStateIDs() to get a list of state IDs.
See http://api.votesmart.org/docs/semi-static.html for a list of office IDs or use Office.getOfficesByType(), Office.getOfficesByLevel(), Office.getOfficesByTypeLevel() or Office.getOfficesByBranchLevel() to get a list of office ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get leadership positions by state ID and office ID
## Not run: positions <- Leadership.getPositions(list("AL","FL"),8)
## Not run: positions
```

Local.getCities *Get cities in a state*

Description

This function is a wrapper for the Local.getCities() method of the PVS API Local class which returns a list of cities in a state. The function sends a request with this method to the PVS API for all state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Local.getCities(stateId)
```

Arguments

stateId	a character string or list of character strings with the state ID(s) (see references for details)

Value

A data frame with a row for each city and columns with the following variables describing the city:
cities.city*.localId,
cities.city*.name,
cities.city*.url.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Local.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of cities of a certain state
## Not run: cities <- Local.getCities(list("NY","FL"))
## Not run: head(cities)
```

Local.getCounties *Get counties in a state*

Description

This function is a wrapper for the Local.getCounties() method of the PVS API Local class which returns a list of counties in a state. The function sends a request with this method to the PVS API for all state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Local.getCounties(stateId)
```

Arguments

stateId	a character string or list of character strings with the state ID(s) (see references for details)

Value

A data frame with a row for each county and columns with the following variables describing the county:
counties.county*.localId,
counties.county*.name,
counties.county*.url.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Local.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of counties of a certain state
## Not run: counties <- Local.getCounties(list("NY","FL"))
## Not run: head(counties)
```

`Local.getOfficials` *Get officials for a locality*

Description

This function is a wrapper for the Local.getOfficials() method of the PVS API Local class which returns a list of officials in a locality. The function sends a request with this method to the PVS API for all local IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Local.getOfficials(localId)
```

Arguments

`localId`	a character string or list of character strings with the local ID(s) (see references for details)

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidatelist.candidate*.candidateId,
candidatelist.candidate*.firstName,
candidatelist.candidate*.nickName,
candidatelist.candidate*.middleName,
candidatelist.candidate*.lastName,
candidatelist.candidate*.suffix,
candidatelist.candidate*.title,
candidatelist.candidate*.electionParties,
candidatelist.candidate*.electionDistrictId,
candidatelist.candidate*.electionStateId,
candidatelist.candidate*.officeParties,
candidatelist.candidate*.officeDistrictId,
candidatelist.candidate*.officeDistrictName,
candidatelist.candidate*.officeStateId,
candidatelist.candidate*.officeId,
candidatelist.candidate*.officeName,
candidatelist.candidate*.officeTypeId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Local.html
Use Local.getCounties() or Local.getCities() to get a list of local IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of officials according to certain local IDs
## Not run: officials <- Local.getOfficials(list(3200,3203))
## Not run: officials
```

Measure.getMeasure *Get details of a ballot measure*

Description

This function is a wrapper for the Measure.getMeasure() method of the PVS API Measure class which grabs details of a ballot measure. The function sends a request with this method to the PVS API for all measure IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Measure.getMeasure(measureId)
```

Arguments

measureId	a character string or list of character strings with the measure ID(s) (see references for details)

Value

A data frame with a row for each ballot measure and columns with the following variables describing the ballot measure:
measure.measureId,
measure.measureCode,
measure.title,
measure.electionDate,
measure.electionType,
measure.source,
measure.url,
measure.summary,
measure.summaryUrl,

measure.measureText,
measure.textUrl,
measure.proUrl,
measure.conUrl,
measure.yes,
measure.no,
measure.outcome.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Measure.html
Use Measure.getMeasuresByYearState() to get a list of measure IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get details on certain ballot measures
## Not run: measure <- Measure.getMeasure(list(1632,1633))
## Not run: measure
```

Measure.getMeasuresByYearState

Get a list of state ballot measures in a given year

Description

This function is a wrapper for the Measure.getMeasuresByYearState() method of the PVS API Measure class which grabs a list of state ballot measures in a given year. The function sends a request with this method to the PVS API for all years and state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Measure.getMeasuresByYearState(year, stateId)
```

Arguments

year	a character string or list of character strings with the year(s)
stateId	a character string or list of character strings with the state ID(s) (see references for details)

Value

A data frame with a row for each ballot measure and columns with the following variables describing the ballot measure:
measures.measure*.measureId,
measures.measure*.measureCode,
measures.measure*.title,
measures.measure*.outcome.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.v.otesmart.org/docs/Measure.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of ballot measures for a certain state and year
## Not run: measures <- Measure.getMeasuresByYearState(list(2010,2012),"FL")
## Not run: measures
```

Npat.getNpat *Get a candidate's most recently filled out NPAT/PCT (Political Courage Test)*

Description

This function is a wrapper for the Npat.getNpat() method of the PVS API Npat class which returns the candidate's most recently filled out NPAT/PCT. The function sends a request with this method to the PVS API for all candidate IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Npat.getNpat(candidateId)
```

Arguments

candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A data frame with a row for each candidate and columns with the following variables describing the candidate:
bio.candidate.crpId (OpenSecrets ID),
bio.candidate.firstName,
bio.candidate.nickName,
bio.candidate.middleName,
bio.candidate.lastName,
bio.candidate.suffix,
bio.candidate.birthDate,
bio.candidate.birthPlace,
bio.candidate.pronunciation,
bio.candidate.gender,
bio.candidate.family,
bio.candidate.photo,
bio.candidate.homeCity,
bio.candidate.homeState,
bio.candidate.education,
bio.candidate.profession,
bio.candidate.political,
bio.candidate.religion,
bio.candidate.congMembership,
bio.candidate.orgMembership,
bio.candidate.specialMsg,
bio.office.parties,
bio.office.title,
bio.office.shortTitle,
bio.office.name,
bio.office.type,
bio.office.status,
bio.office.firstElect,
bio.office.lastElect,
bio.office.nextElect,
bio.office.termStart,
bio.office.termEnd,
bio.office.district,
bio.office.districtId,
bio.office.stateId,
bio.office.committee*.committeeId,
bio.office.committee*.committeeName,
bio.election*.office,
bio.election*.officeId,
bio.election*.officeType,
bio.election*.parties,
bio.election*.district,
bio.election*.districtId,
bio.election*.status,
bio.election*.ballotName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/CandidateBio.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get political courage tests of Barack Obama and John Sidney McCain III
## Not run: pcts <- Npat.getNpat(list(9490,53270))
## Not run: head(pcts$survey)
## Not run: head(pcts$candidate)
```

Office.getBranches *Get a list of branches of government and their IDs*

Description

This function is a wrapper for the Office.getBranches() method of the PVS API Office class which grabs a list of branches of government and their IDs.

Usage

```
Office.getBranches()
```

Value

A data frame with a row for each branch and columns with the following variables describing the branch:
branches.branch*.officeBranchId,
branches.branch*.name.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Office.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of government branches and their IDs
## Not run: branches <- Office.getBranches()
## Not run: branches
```

Office.getLevels *Get a list of levels of government and their IDs*

Description

This function is a wrapper for the Office.getLevels() method of the PVS API Office class which grabs a list of the levels of government and their IDs.

Usage

```
Office.getLevels()
```

Value

A data frame with a row for each level and columns with the following variables describing the level:
levels.level*.officeLevelId,
levels.level*.name.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Office.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of government levels and their IDs
## Not run: levels <- Office.getLevels()
## Not run: levels
```

`Office.getOfficesByBranchLevel` *Get offices tracked according to branch and level*

Description

This function is a wrapper for the Office.getOfficesByBranchLevel() method of the PVS API Office class which grabs a list of offices Project Vote Smart keeps track of according to government branch and level. The function sends a request with this method to the PVS API for all branch and level IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Office.getOfficesByBranchLevel(branchId, levelId)
```

Arguments

`branchId`	a character string or list of character strings with the branch ID(s) (see references for details)
`levelId`	a character string or list of character strings with the level ID(s) (see references for details)

Value

A data frame with a row for each office and columns with the following variables describing the office:
offices.office*.officeId,
offices.office*.officeTypeId,
offices.office*.officeLevelId,
offices.office*.officeBranchId,
offices.office*.name,
offices.office*.title,
offices.office*.shortTitle.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Office.html
Use Office.getBranches() to get branch ID(s).
Use Office.getLevels() to get level ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get offices tracked for all branch and level IDs
## Not run: offices <- Office.getOfficesByBranchLevel(list("E","L","J"),list("F","S","L"))
## Not run: head(offices)
```

`Office.getOfficesByLevel`

Get offices tracked according to level

Description

This function is a wrapper for the Office.getOfficesByLevel() method of the PVS API Office class which grabs a list of offices Project Vote Smart keeps track of according to their level. The function sends a request with this method to the PVS API for all level IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Office.getOfficesByLevel(levelId)
```

Arguments

`levelId`	a character string or list of character strings with the level ID(s) (see references for details)

Value

A data frame with a row for each office and columns with the following variables describing the office:
offices.office*.officeId,
offices.office*.officeTypeId,
offices.office*.officeLevelId,
offices.office*.officeBranchId,
offices.office*.name,
offices.office*.title,
offices.office*.shortTitle.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Office.html
Use Office.getLevels() to get level ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get offices tracked for all levels
## Not run: offices <- Office.getOfficesByLevel(list("F","S","L"))
## Not run: head(offices)
```

`Office.getOfficesByType`

Get offices tracked according to type

Description

This function is a wrapper for the Office.getOfficesByType() method of the PVS API Office class which grabs a list of offices Project Vote Smart keeps track of according to their type. The function sends a request with this method to the PVS API for all office type IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Office.getOfficesByType(officeTypeId)
```

Arguments

`officeTypeId`	a character string or list of character strings with the office type ID(s) (see references for details)

Value

A data frame with a row for each office and columns with the following variables describing the office:
offices.office*.officeId,
offices.office*.officeTypeId,
offices.office*.officeLevelId,
offices.office*.officeBranchId,
offices.office*.name,
offices.office*.title,
offices.office*.shortTitle.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Office.html
See http://api.votesmart.org/docs/semi-static.html or use Office.getTypes() or Office.getOfficesByLevel() to get a list of office types ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get offices tracked for certain office types
## Not run: offices <- Office.getOfficesByType(list("S","K","L"))
## Not run: head(offices)
```

Office.getTypes *Get all office types tracked*

Description

This function is a wrapper for the Office.getTypes() method of the PVS API Office class which grabs a list of office types Project Vote Smart keeps track of.

Usage

```
Office.getTypes()
```

Value

A data frame with rows for each office type and columns with the following variables describing the office type:
officeTypes.type*.officeTypeId,
officeTypes.type*.officeLevelId,
officeTypes.type*.officeBranchId,
officeTypes.type*.name.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Office.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get all office types tracked
## Not run: officetypes <- Office.getTypes()
## Not run: officetypes
```

`Officials.getByDistrict` *Get a list of officials according to the district they are running for*

Description

This function is a wrapper for the Officials.getByDistrict() method of the PVS API Officials class which grabs a list of officials according to the district they are running for. The function sends a request with this method to the PVS API for all district IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Officials.getByDistrict(districtId)
```

Arguments

`districtId`	a character string or list of character strings with the district ID(s) (see references for details)

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionstatus,
candidateList.candidate*.officeParties,
candidatelist.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeStateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Officials.html
Use District.getByOfficeState() or District.getByZip() to get a list of district ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get officials by dristrict ID
## Not run: officials <- Officials.getByDistrict(as.list(25157:25163))
## Not run: officials
```

`Officials.getByLastname`

Get a list of officials according to a last name match

Description

This function is a wrapper for the Officials.getByLastname() method of the PVS API Officials class which grabs a list of officials according to a last name match. The function sends a request with this method to the PVS API for all last names given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Officials.getByLastname(lastName)
```

Arguments

`lastName`	a character string or list of character strings with the last name(s) (see references for details)

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.electionParties,
candidateList.candidate*.officeParties,
candidatelist.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,

candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeStateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Officials.html
Use CandidateBio.getBio(), Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByElection(), Candidates.getByDistrict(), Candidates.getByZip()
, Committee.getCommitteeMembers(), Election.getStageCandidates(), Leadership.getOfficials(), Local.getOfficials(), Officials.getStatewide(), Officials.getByOfficeState(), Officials.getByOfficeTypeState(), Officials.getByDistrict() or Officials.getByZip() to get last name(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of officials with the same last name
## Not run: miller <- Officials.getByLastname(list("Miller","Fine"))
## Not run: miller
```

Officials.getByLevenshtein

Get a list of officials according to an approximate last name match

---

Description

This function is a wrapper for the Officials.getByLevenshtein() method of the PVS API Officials class which grabs a list of officials according to an approximate last name match. The function sends a request with this method to the PVS API for all last names given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Officials.getByLevenshtein(lastName)
```

Arguments

lastName	a character string or list of character strings with the last name(s) (see references for details)

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.electionParties,
candidateList.candidate*.officeParties,
candidatelist.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeStateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Officials.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of officials with similar last names
## Not run: names <- Officials.getByLevenshtein(list("Miller","Fine"))
## Not run: head(names)
```

Officials.getByOfficeState

Get a list of officials according to office

Description

This function is a wrapper for the Officials.getByOfficeState() method of the PVS API Officials class which grabs a list of officials according to office. The function sends a request with this method to the PVS API for all office IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Officials.getByOfficeState(stateId="NA", officeId)
```

Arguments

stateId	a character string or list of character strings with the state ID(s) (default is "NA", for national) (see references for details)
officeId	a character string or list of character strings with the office ID(s) (see references for details)

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.electionParties,
candidateList.candidate*.officeParties,
candidatelist.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeStateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Officials.html
Use State.getStateIDs() to get a list of state IDs.
See http://api.votesmart.org/docs/semi-static.html or use Office.getOfficesByType(), Office.getOfficesByLevel(), Office.getOfficesByTypeLevel() or Office.getOfficesByBranchLevel() to get a list of office ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of officials by state and office ID
## Not run: officials <- Officials.getByOfficeState(,as.list(60:69))
## Not run: officials
```

`Officials.getByOfficeTypeState` *Get a list of officials according to office type and state*

Description

This function is a wrapper for the Officials.getByOfficeTypeState() method of the PVS API Officials class which grabs a list of officials according to the office type and state they represent. The function sends a request with this method to the PVS API for all state and office type IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Officials.getByOfficeTypeState(stateId="NA", officeTypeId)
```

Arguments

`stateId`	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)
`officeTypeId`	a character string or list of character strings with the office type ID(s) (see references for details)

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.electionParties,
candidateList.candidate*.officeParties,
candidatelist.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeStateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Officials.html
Use State.getStateIDs() to get a list of state IDs.
See http://api.votesmart.org/docs/semi-static.html or use Office.getTypes or Office.getOfficesByLevel to get a list of office types ID(s).

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
# Note that some officeTypeIds are only available on the state level or national level
# (e.g. "L" for State Legislature only if stateId is specified!)
## Not run: pvs.key <- "yourkey"
# get a list of officials by state and office type
## Not run: CAlegislators <- Officials.getByOfficeTypeState(officeTypeId="L", stateId="CA")
## Not run: head(CAlegislators)
## Not run: suprcourt <- Officials.getByOfficeTypeState(officeTypeId="J")
## Not run: head(suprcourt)
```

Officials.getByZip *Get a list of officials according to the ZIP code*

Description

This function is a wrapper for the Officials.getByZip() method of the PVS API Officials class which grabs a list of officials according to the ZIP code of the area they represent. The function sends a request with this method to the PVS API for all ZIP Codes given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Officials.getByZip(zip5, zip4=NULL)
```

Arguments

zip5	a character string or list of character strings with the five-digit ZIP code
zip4	(optional) a character string or list of character strings with the expanded ZIP+4 code (default: all)

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidateList.zipMessage,
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,

candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.electionParties,
candidateList.candidate*.electionstatus,
candidateList.candidate*.officeParties,
candidatelist.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeStateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Officials.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of officials by ZIP code
## Not run: officials <- Officials.getByZip(list(10001,10002))
## Not run: head(officials)
```

`Officials.getStatewide`

Get a list of officials according to state representation

Description

This function is a wrapper for the Officials.getStatewide() method of the PVS API Officials class which grabs a list of officials according to the state they are representing. The function sends a request with this method to the PVS API for all state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Officials.getStatewide(stateId="NA")
```

Arguments

`stateId`	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)

Value

A data frame with a row for each official and columns with the following variables describing the official:
candidateList.candidate*.candidateId,
candidateList.candidate*.firstName,
candidateList.candidate*.nickName,
candidateList.candidate*.middleName,
candidateList.candidate*.lastName,
candidateList.candidate*.suffix,
candidateList.candidate*.title,
candidateList.candidate*.electionParties,
candidateList.candidate*.officeParties,
candidatelist.candidate*.officeStatus,
candidateList.candidate*.officeDistrictId,
candidateList.candidate*.officeDistrictName,
candidateList.candidate*.officeTypeId,
candidateList.candidate*.officeId,
candidateList.candidate*.officeName,
candidateList.candidate*.officeStateId.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Officials.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get all officials of a certain state
## Not run: officials <- Officials.getStatewide("FL")
## Not run: head(officials)
```

Rating.getCandidateRating

Get a candidate's rating by special interest groups

Description

This function is a wrapper for the Rating.getCandidateRating() method of the PVS API Rating class which grabs a candidate's rating by special interest groups (SIG). The function sends a request with this method to the PVS API for all candidate and SIG IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Rating.getCandidateRating(candidateId, sigId=NULL)
```

Arguments

candidateId	a character string or list of character strings with the candidateId(s) (see references for details)
sigId	(optional) a character string or list of character strings with the special interest group's ID(s) (see references for details)

Value

A data frame with a row for each rating of a candidate and columns with the following variables describing the candidate:
candidateRating.candidate.title,
candidateRating.candidate.firstName,
candidateRating.candidate.middleName,
candidateRating.candidate.lastName,
candidateRating.candidate.suffix,
candidateRating.candidate.office,
candidateRating.rating*.sigId,
candidateRating.rating*.ratingId,
candidateRating.rating*.categories.category*.categoryId,
candidateRating.rating*.categories.category*.name,
candidateRating.rating*.timeSpan,
candidateRating.rating*.rating,
candidateRating.rating*.ratingName,
candidateRating.rating*.ratingText.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Rating.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.
Use Rating.getSigList() to get a list of special interest group's IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get ratings by candidate and special interest group
## Not run: rating <- Rating.getCandidateRating(candidateId="9490")
## Not run: head(rating)
```

`Rating.getCategories` *Get categories that contain released ratings according to state*

Description

This function is a wrapper for the Rating.getCategories() method of the PVS API Rating class which dumps categories that contain released ratings according to state. The function sends a request with this method to the PVS API for all state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Rating.getCategories(stateId="NA")
```

Arguments

`stateId`	(optional) a character string or list of character strings with the stateId(s) (default: "NA", for national) (see references for details)

Value

A data frame with a row for each rating and columns with the following variables describing the rating:
categories.category*.categoryId,
categories.category*.name.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Rating.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get rating categories
## Not run: rating <- Rating.getCategories()
## Not run: rating
```

`Rating.getRating` *Get all candidate ratings from an evaluation by a special interest group*

Description

This function is a wrapper for the Rating.getRating() method of the PVS API Rating class which dumps all candidate ratings from a scorecard by a special interest group (SIG). The function sends a request with this method to the PVS API for all rating IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Rating.getRating(ratingId)
```

Arguments

`ratingId`	a character string or list of character strings with the rating ID(s) (see references for details)

Value

A data frame with a row for each candidate and columns with the following variables describing the candidate:
candidateRating*.candidateId,
candidateRating*.rating.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Rating.html
Use Rating.getSigRatings() or Rating.getCandidateRating() to get a list of rating IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get the candidate rating by a certain special interest group
## Not run: scorecard <- Rating.getRating(77)
## Not run: scorecard
```

`Rating.getSig` *Get detailed information about a special interest group*

Description

This function is a wrapper for the Rating.getSig() method of the PVS API Rating class which dumps detailed information about special interest groups (SIGs). The function sends a request with this method to the PVS API for all SIG IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Rating.getSig(sigId)
```

Arguments

`sigId`	a character string or list of character strings with the special interest group's ID(s) (see references for details)

Value

A data frame with a row for each special interest group and columns with the following variables describing the special interest group:
sig.sigId,
sig.parentId,
sig.stateId,
sig.name,
sig.description,
sig.address,
sig.city,
sig.state,
sig.zip,
sig.phone1,
sig.phone2,
sig.fax,
sig.email,
sig.url,
sig.contactName.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Rating.html
Use Rating.getSigList() to get a list of special interest group's IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get information about certain special interest groups
## Not run: info <- Rating.getSig(list(1016,1120))
## Not run: info
```

Rating.getSigList — *Get a list of special interest groups according to rating category and state.*

Description

This function is a wrapper for the Rating.getSigList() method of the PVS API Rating class which dumps special interest groups (SIGs) according to category and state. The function sends a request with this method to the PVS API for all category and state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Rating.getSigList(stateId="NA", categoryId)
```

Arguments

stateId	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)
categoryId	a character string or list of character strings with the category ID(s) (see references for details)

Value

A data frame with a row for each special interest group and columns with the following variables describing the special interest group:
sigs.sig*.sigId,
sigs.sig*.parentId,
sigs.sig*.name.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Rating.html
Use State.getStateIDs() to get a list of state IDs.
Use Rating.getCategories() or Rating.getCandidateRating() to get category IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of special interest groups for certain categories and all states
## Not run: sig <- Rating.getSigList(categoryId=list(2,4,5,7))
## Not run: sig
```

Rating.getSigRatings *Get all ratings (scorecards) by a special interest group*

Description

This function is a wrapper for the Rating.getSigRatings() method of the PVS API Rating class which dumps all ratings (scorecards) by a special interest group. The function sends a request with this method to the PVS API for all candidate and SIG IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Rating.getSigRatings(sigId)
```

Arguments

sigId	a character string or list of character strings with the special interest group's ID(s) (see references for details)

Value

A data frame with a row for each special interest group and columns with the following variables describing the rating:
sig.sigId,
sig.name,
rating*.ratingId,
rating*.timespan,
rating*.ratingName,
rating*.ratingText.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Rating.html
Use Rating.getSigList() to get a list of special interest group's IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get ratings of certain special interest groups
## Not run: rating <- Rating.getSigRatings(list(568,1120,1704))
## Not run: rating
```

`State.getState` *Get information about a state*

Description

This function is a wrapper for the State.getState() method of the PVS API State class which grabs various data on a state. The function sends a request with this method to the PVS API for all state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
State.getState(stateId)
```

Arguments

`stateId`	a character string or list of character strings with the state ID(s) (see references for details)

Value

A data frame with a row for each state and columns with the following variables describing the state:
state.details.stateId,
state.details.stateType,
state.details.name,
state.details.nickName,
state.details.capital,
state.details.area,
state.details.population,
state.details.statehood,
state.details.motto,
state.details.flower,
state.details.tree,
state.details.bird,
state.details.highPoint,
state.details.lowPoint,
state.details.bicameral,
state.details.upperLegis,
state.details.lowerLegis,

state.details.ltGov,
state.details.senators,
state.details.reps,
state.details.termLimit,
state.details.termLength,
state.details.billUrl,
state.details.voteUrl,
state.details.voterReg,
state.details.primaryDate,
state.details.generalDate,
state.details.absenteeWho,
state.details.absenteeHow,
state.details.absenteeWhen,
state.details.largestCity,
state.details.rollUpper,
state.details.rollLower,
state.details.usCircuit.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/State.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get information about certain states
## Not run: stateinfo <- State.getState(list("FL","NY"))
## Not run: stateinfo
```

State.getStateIDs *Get a list of states and their IDs.*

Description

This function is a wrapper for the State.getStateIDs() method of the PVS API State class which returns a simple state ID and name list for mapping IDs to state names.

Usage

```
State.getStateIDs()
```

Value

A data frame with a row for each state:
statelist.list.state*.stateId,
statelist.list.state*.name

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/State.html

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of states with their IDs
## Not run: stateIDs <- State.getStateIDs()
## Not run: stateIDs
```

Votes.getBill *Get general information on a bill*

Description

This function is a wrapper for the Votes.getBill() method of the PVS API Votes class which grabs the general information on a bill. The function sends a request with this method to the PVS API for all bill IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBill(billId,...)
```

Arguments

billId	a character string or list of character strings with the bill ID(s) (see references for details)
...	further arguments that are passed on to internal functions. Currently the argument separate can be defined: separate is a vector of character strings defining subnodes that should be returned separately (e.g., "sponsors").

Value

If separate is not specified, a data frame with a row for each bill and columns with variables describing the bill. If separate is specified, a list containing several data frames, one for each subnode mentioned in separate and additionally one data frame containing all remaining nodes not mentioned in separate. The returned data frame contains a row for each bill and columns with the following variables describing the bill:
bill.billnumber,
bill.parentbill,
bill.title,
bill.officialtitle,
bill.dateintroduced,
bill.type,
bill.categories.category*.categoryId,
bill.categories.category*.name,
bill.billtextLink,
bill.sponsors.sponsor*.candidateId,
bill.sponsors.sponsor*.name,
bill.sponsors.sponsor*.type,
bill.committeeSponsors.committeeSponsor*.committeeId,
bill.committeeSponsors.committeeSponsor*.name,
bill.actions.action*.actionId,
bill.actions.action*.level,
bill.actions.action*.stage,
bill.actions.action*.outcome,
bill.actions.action*.statusDate,
bill.actions.action*.rollNumber,
bill.actions.action*.yea,
bill.actions.action*.nay,
bill.actions.action*.voiceVote,
bill.amendments.amendment*.billNumber,
bill.amendments.amendment*.actionId,
bill.amendments.amendment*.title,
bill.amendments.amendment*.statusDate.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Votes.getByBillNumber(), Votes.getBillsByCategoryYearState(), Votes.getBillsByYearState(), Votes.getBillsByOfficialYearOffice(), Votes.getBillsByOfficialCategoryOffice(), Votes.getByOfficial(), Votes.getBillsBySponsorYear(), Votes.getBillsBySponsorCategory() or Votes.getBillsByStateRecent() to get a list of bill IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
```

```
## Not run: pvs.key <- "yourkey"
# get information about certain bills
## Not run: billinfo <- Votes.getBill(list(2819,6427))
## Not run: billinfo
# let some variables with subnodes be returned separately (here: "sponsors" and "actions")
## Not run: billinfo2 <- Votes.getBill(billId=list(2819,6427,6590),
separate=c("sponsors","actions"))
## End(Not run)
## Not run: billinfo2
# check the sponsors of the requested bill (argument of separate)...
## Not run: billinfo2$sponsors
# ... and the usual variables describing the bill (nodes not mentioned in separate)
## Not run: billinfo2$main
```

`Votes.getBillAction` *Get detailed action information on a certain stage of the bill*

Description

This function is a wrapper for the Votes.getBillAction() method of the PVS API Votes class which grabs detailed action information on a certain stage of the bill. The function sends a request with this method to the PVS API for all action IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillAction(actionId)
```

Arguments

`actionId`	a character string or list of character strings with the action ID(s) (see references for details)

Value

A data frame with a row for each action and columns with the following variables describing the action:
action.billId,
action.billNumber,
action.actionId,
action.category,
action.categoryId,
action.type,
action.stateId,
action.level,
action.stage,
action.outcome,
action.rollNumber,

action.yea,
action.nay,
action.voiceVote,
action.title,
action.officialTitle,
action.highlight,
action.synopsis,
action.officialSynopsis,
action.note.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Votes.getBill() or Votes.getByOfficial() to get a list of action IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get information about certain actions
## Not run: actioninfo <- Votes.getBillAction(actionId=list(2575,18436,10194))
## Not run: actioninfo
```

Votes.getBillActionVoteByOfficial

Get a single vote according to official and action

Description

This function is a wrapper for the Votes.getBillActionVoteByOfficial() method of the PVS API Votes class which grabs single vote according to official and action. The function sends a request with this method to the PVS API for all action and candidate IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillActionVoteByOfficial(actionId, candidateId)
```

Arguments

actionId	a character string or list of character strings with the action ID(s) (see references for details)
candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A data frame with a row for each vote and columns with the following variables describing the vote:
votes.vote.candidateId,
votes.vote.candidateName,
votes.vote.officeParties,
votes.vote.action.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.
Use Votes.getBill() or Votes.getByOfficial() to get a list of action IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get information about certain votes
## Not run: vote <- Votes.getBillActionVoteByOfficial(list(28686,31712),9490)
## Not run: vote
```

Votes.getBillActionVotes

Get votes listed by candidate on a certain bill action

Description

This function is a wrapper for the Votes.getBillActionVotes() method of the PVS API Votes class which provides votes listed by candidate on a certain bill action. The function sends a request with this method to the PVS API for all action IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillActionVotes(actionId)
```

Arguments

actionId	a character string or list of character strings with the action ID(s) (see references for details)

Value

A data frame with a row for each vote and columns with the following variables describing the vote:
votes.vote*.candidateId,
votes.vote*.candidateName,
votes.vote*.officeParties,
votes.vote*.action.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Votes.getBill() or Votes.getByOfficial() to get a list of action IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get votes of a certain candidate on a certain action
## Not run: actionvote <- Votes.getBillActionVotes(list(31712,28686))
## Not run: actionvote
```

Votes.getBillsByCategoryYearState

Get a list of bills according to category, year and state

Description

This function is a wrapper for the Votes.getBillsByCategoryYearState() method of the PVS API Votes class which grabs a list of bills according to category, year and state. The function sends a request with this method to the PVS API for all years, state and category IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillsByCategoryYearState(year, stateId, categoryId)
```

Arguments

year	a character string or list of character strings with the year ID(s)
stateId	a character string or list of character strings with the state ID(s) (see references for details)
categoryId	a character string or list of character strings with the category ID(s) (see references for details)

Value

A data frame with a row for each bill and columns with the following variables describing the bill:
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.title,
bills.bill*.type.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use State.getStateIDs() to get a list of state IDs.
Use Votes.getCategories() or Rating.getCandidateRating() to get a list of category IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of bills in a certain category, year and state
## Not run: bills <- Votes.getBillsByCategoryYearState(as.list(2010:2012),"NY",10)
## Not run: bills
```

Votes.getBillsByOfficialCategoryOffice

Get a list of bills according to office, candidate and category

---

Description

This function is a wrapper for the Votes.getBillsByOfficialCategoryOffice() method of the PVS API Votes class which grabs a list of bills that fit the candidate and category. The function sends a request with this method to the PVS API for all category, candidate and office IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillsByOfficialCategoryOffice(categoryId, candidateId, officeId=NULL)
```

Arguments

categoryId	a character string or list of character strings with the category ID(s) (see references for details)
candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)
officeId	(optional) a character string or list of character strings with the office ID(s) (default: all) (see references for details)

Value

A data frame with a row for each bill and columns with the following variables describing the bill:
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.title,
bills.bill*.type.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Votes.getCategories() to get a list of category IDs.
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.
See http://api.votesmart.org/docs/semi-static.html for a list of office IDs or use Office.getOfficesByType(), Office.getOfficesByLevel(), Office.getOfficesByTypeLevel() or Office.getOfficesByBranchLevel().

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of bills of a certain office, candidate and category
## Not run: bills <- Votes.getBillsByOfficialCategoryOffice(list(30,10),9490,6)
## Not run: bills
```

Votes.getBillsByOfficialYearOffice

Get a list of bills according to office (optional), candidate and year

---

Description

This function is a wrapper for the Votes.getBillsByOfficialYearOffice() method of the PVS API Votes class which grabs a list of bills that fit the candidate and year. The function sends a request with this method to the PVS API for all years, candidate and office IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillsByOfficialYearOffice(year, candidateId, officeId=NULL)
```

Arguments

year	a character string or list of character strings with the year ID(s)
candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)
officeId	(optional) a character string or list of character strings with the office ID(s) (default: all) (see references for details)

Value

A data frame with a row for each bill and columns with the following variables describing the bill:
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.title,
bills.bill*.type.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.
See http://api.votesmart.org/docs/semi-static.html for a list of office IDs or use Office.getOfficesByType(), Office.getOfficesByLevel(), Office.getOfficesByTypeLevel() or Office.getOfficesByBranchLevel

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of bills for a certain office, candidate and year
## Not run: bills <- Votes.getBillsByOfficialYearOffice(list(2010,2011,2012),107800,)
## Not run: bills
```

Votes.getBillsBySponsorCategory

Get a list of bills according to sponsor(candidate) and category

---

Description

This function is a wrapper for the Votes.getBillsBySponsorCategory() method of the PVS API Votes class which grabs a list of bills that fit the sponsor's candidateId and category. The function sends a request with this method to the PVS API for all candidate and category IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillsBySponsorCategory(categoryId, candidateId)
```

Arguments

categoryId	a character string or list of character strings with the category ID(s) (see references for details)
candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A data frame with a row for each bill and columns with the following variables describing the bill:
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.title,
bills.bill*.type.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Votes.getCategories() to get a list of category IDs.
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of bills for a certain candidate and category
## Not run: bills <- Votes.getBillsBySponsorCategory(as.list(1:50),107800)
## Not run: bills
```

Votes.getBillsBySponsorYear

Get a list of bills according to sponsor(candidate) and year

Description

This function is a wrapper for the Votes.getBillsBySponsorYear() method of the PVS API Votes class which grabs a list of bills that fit the sponsor's candidateId and year. The function sends a request with this method to the PVS API for all candidate IDs and years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillsBySponsorYear(year, candidateId)
```

Arguments

year	a character string or list of character strings with the year(s)
candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A data frame with a row for each bill and columns with the following variables describing the bill:
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.title,
bills.bill*.type.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get additional biographical data on Barak Obama
## Not run: obama <- CandidateBio.getAddlBio(9490)
## Not run: obama
# get additional biographical data on Barak Obama and Mitt Romney
## Not run: onr <- CandidateBio.getAddlBio(list(9490,21942))
```

Votes.getBillsByStateRecent

Get a list of recent bills according to the state.

Description

This function is a wrapper for the Votes.getBillsByStateRecent() method of the PVS API Votes class which returns a list of recent bills according to the state. The maximum number of bills returned is 100. The function sends a request with this method to the PVS API for all states given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillsByStateRecent(amount, state="NA")
```

Arguments

amount	(optional) the amount of bills returned (default: 100, max: 100)
state	(optional) a character string or list of character strings with the state(s) (default: "NA", for national) (see references for details)

Value

A data frame with a row for each bill and columns with the following variables describing the bill:
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.title,
bills.bill*.type.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of recent bills according to the state
## Not run: recentbills <- Votes.getBillsByStateRecent(40,list("FL","NY"))
## Not run: recentbills
```

Votes.getBillsByYearState

Get a list of bills according to year and state

Description

This function is a wrapper for the Votes.getBillsByYearState() method of the PVS API Votes class which returns a list of bills that fit the year and state input. The function sends a request with this method to the PVS API for all years and state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getBillsByYearState(year, stateId, all=FALSE)
```

Arguments

year	a character string or list of character strings with the year(s)
stateId	a character string or list of character strings with the state ID(s) (see references for details)
all	a logical indicator; if TRUE data on all possible combinations of the stateId and year are returned, if FALSE (default) only the exact combinations (see example)

Value

A data frame with a row for each bill and columns with the following variables describing the bill:
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.title,
bills.bill*.type.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a data frame of bills according to all year and state combinations
## Not run: bills <- Votes.getBillsByYearState(year=list(2011,2012),
stateId=list("NY","NJ"), all=TRUE)
## End(Not run)
## Not run: head(bills)
# get a data frame of bills according to the exact year and state combinations
# (i.e., 2011/"NY", 2012/"NJ")
## Not run: bills <- Votes.getBillsByYearState(year=list(2011,2012),
stateId=list("NY","NJ"), all=FALSE)
## End(Not run)
## Not run: head(bills)
```

Votes.getByOfficial *Get all the bills an official has voted on by year*

Description

This function is a wrapper for the Votes.getByOfficial() method of the PVS API Votes class which dumps all the bills an official has voted on based on the candidateId and year. The function sends a request with this method to the PVS API for all candidate IDs and years given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getByOfficial(year, candidateId)
```

Arguments

year	a character string or list of character strings with the year(s)
candidateId	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A data frame with a row for each bill and columns with the following variables describing the bill:
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.title,
bills.bill.categories.category*.categoryId,
bills.bill.categories.category*.name,
bills.bill*.officeId,
bills.bill*.office,
bills.bill*.vote,
bills.bill*.actionId,
bills.bill*.stage.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get additional biographical data on Barack Obama
## Not run: votes <- Votes.getByOfficial(as.list(2010:2012),9490)
## Not run: votes
```

`Votes.getCategories` *Get a list of categories that contain released bills according to year and state*

Description

This function is a wrapper for the Votes.getCategories() method of the PVS API Votes class which dumps categories that contain released bills according to year and state. The function sends a request with this method to the PVS API for all years and state IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getCategories(year, stateId="NA")
```

Arguments

`year`	a character string or list of character strings with the year(s)
`stateId`	(optional) a character string or list of character strings with the state ID(s) (default: "NA", for national) (see references for details)

Value

A data frame with a row for each category and columns with the following variables describing the category:
categories.category*.categoryId,
categories.category*.name.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use State.getStateIDs() to get a list of state IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get a list of categories
## Not run: categories <- Votes.getCategories(2012)
## Not run: categories
```

`Votes.getVetoes` *Get a list of vetoes according to candidate*

Description

This function is a wrapper for the Votes.getVetoes() method of the PVS API Votes class which returns a list of vetoes according to candidate. The function sends a request with this method to the PVS API for all candidate IDs given as a function input, extracts the XML values from the returned XML file(s) and returns them arranged in one data frame.

Usage

```
Votes.getVetoes(candidateId)
```

Arguments

`candidateId`	a character string or list of character strings with the candidate ID(s) (see references for details)

Value

A data frame with a row for each veto and columns with the following variables describing the veto:
bills.bill*.vetoId,
bills.bill*.statusDate,
bills.bill*.billId,
bills.bill*.billNumber,
bills.bill*.billTitle,
bills.bill*.vetoCode,
bills.bill*.vetoType,
bills.bill*.billSummary,
bills.bill*.billLink,
bills.bill*.vetoLetterLink.

Author(s)

Ulrich Matter <ulrich.matter-at-unibas.ch>

References

http://api.votesmart.org/docs/Votes.html
Use Candidates.getByOfficeState(), Candidates.getByOfficeTypeState(), Candidates.getByLastname(), Candidates.getByLevenshtein(), Candidates.getByElection(), Candidates.getByDistrict() or Candidates.getByZip() to get a list of candidate IDs.

Examples

```
# First, make sure your personal PVS API key is saved as character string in the pvs.key variable:
## Not run: pvs.key <- "yourkey"
# get vetoes by Barack Obama
## Not run: vetoes <- Votes.getVetoes(9490)
## Not run: vetoes
```

Index