Package 'njschooldata'

January 7, 2026

Type Package

Title A Simple Interface for Accessing NJ DOE School Data in R and Python

Description R functions (with Python bindings) to import NJ school data, including assessment, enrollment, and accountability data from the New Jersey Department of Education.

Version 0.9.0

Author Andrew Martin [aut, cre]

Maintainer Andrew Martin <almartin@gmail.com>

License GPL-3 | file LICENSE

URL https://github.com/almartin82/njschooldata

BugReports https://github.com/almartin82/njschooldata/issues

Depends R (>= 4.1.0)

Imports DescTools, digest, dplyr (>= 1.0.0), downloader, foreign, gtools, httr, janitor, magrittr, purrr, readr, readxl, reshape2, rlang (>= 0.4.0), snakecase, stringr, tibble, tidyr (>= 1.0.0)

Suggests covr, geojsonio, knitr, placement, rmarkdown, sf, sp, testthat (>= 3.0.0)

Remotes DerekYves/placement

LazyData true

VignetteBuilder knitr

RoxygenNote 7.3.3

Encoding UTF-8

Config/testthat/edition 3

Contents

`njschooldata-package` *njschooldata: A Simple Interface for Accessing NJ DOE School Data in R*

Description

R functions to import NJ school data, including assessment, enrollment, and accountability data from the New Jersey Department of Education.

Author(s)

Maintainer: Andrew Martin <almartin@gmail.com>

See Also

Useful links:

- https://github.com/almartin82/njschooldata
- Report bugs at https://github.com/almartin82/njschooldata/issues

---

`add_dfg` *Add DFG classification to district data*

Description

Joins District Factor Group classification to any district-level dataframe. Handles both full CDS format (county+district, e.g., "133570") and district-only format (e.g., "3570").

Usage

```
add_dfg(df, revision = 2000)
```

Arguments

df	Dataframe with district_id column
revision	Numeric. DFG revision year (2000 or 1990). Default 2000.

Value

df with 'dfg' column added

---

add_percentile_rank *Add percentile rank columns for any metric*

Description

The fundamental building block for percentile rank calculations. Given a grouped dataframe and a metric column, calculates the percentile rank within each group. Percentile rank is defined as the percent of comparison entities with lesser or equal performance.

This function respects existing grouping on the dataframe. If you want to calculate percentile rank within specific peer groups (e.g., DFG, county), group your data appropriately before calling this function, or use `define_peer_group()` to set up the grouping.

Note: This differs from the simpler `percentile_rank(x, xo)` in util.R which calculates percentile rank of a single value within a vector. This function operates on dataframe columns and adds rank/percentile columns.

Usage

```
add_percentile_rank(df, metric_col, prefix = NULL)
```

Arguments

df	A dataframe, optionally grouped. Grouping defines the comparison set.
metric_col	Character. The column name to rank on.
prefix	Character. Optional prefix for output column names. If NULL, uses metric_col as prefix. Default NULL.

Value

df with added columns:

- `{prefix}_rank`: The rank within the group (1 = lowest)
- `{prefix}_n`: Number of valid observations in the group
- `{prefix}_percentile`: Percentile rank (0-100)

Examples

```
## Not run:
# Simple statewide percentile
grate %>%
  group_by(end_year, subgroup) %>%
  add_percentile_rank("grad_rate")

# DFG peer percentile
grate %>%
  group_by(end_year, dfg, subgroup) %>%
  add_percentile_rank("grad_rate", prefix = "dfg")

## End(Not run)
```

---

agg_enr_column_order *Helper function to return aggregate enrollment columns in correct order*

Description

Helper function to return aggregate enrollment columns in correct order

Usage

```
agg_enr_column_order(df)
```

Arguments

df	aggregate enrollment dataframe

Value

data.frame

---

agg_enr_pct_total *Helper function to support calculating pct_total on aggregate enr dataframes*

Description

Helper function to support calculating pct_total on aggregate enr dataframes

Usage

```
agg_enr_pct_total(df)
```

Arguments

df	aggregate enrollment dataframe

Value

data.frame

`agg_spec_pop_column_order` *Helper function to return aggregate special populations columns in correct order*

Description

Helper function to return aggregate special populations columns in correct order

Usage

```
agg_spec_pop_column_order(df)
```

Arguments

`df`	aggregate special populations df

Value

data_frame

`agg_sped_column_order` *Helper function to return aggregate sped columns in correct order*

Description

Helper function to return aggregate sped columns in correct order

Usage

```
agg_sped_column_order(df)
```

Arguments

`df`	aggregate sped dataframe

Value

data.frame

allpublic_enr_aggs *Calculate All Public Enrollment Aggregates*

Description

Calculate All Public Enrollment Aggregates

Usage

```
allpublic_enr_aggs(df)
```

Arguments

df	a tidied enrollment dataframe, eg output of 'fetch_enr'

Value

dataframe with all public school option aggregates for any charter host city

allpublic_gcount_aggs *Calculate Charter Sector Grad Count aggregates*

Description

Calculate Charter Sector Grad Count aggregates

Usage

```
allpublic_gcount_aggs(df)
```

Arguments

df	tidied grad count dataframe, eg output of fetch_grad_count

Value

df containing charter sector aggregate grad count

allpublic_grate_aggs *Calculate All Public Grad Rate aggregates*

Description

Calculate All Public Grad Rate aggregates

Usage

```
allpublic_grate_aggs(df)
```

Arguments

df	tidied grate dataframe, eg output of fetch_grate

Value

df containing allpublic aggregate grad rate

allpublic_matric_aggs *Calculate All Public matriculation aggregates*

Description

Calculate All Public matriculation aggregates

Usage

```
allpublic_matric_aggs(df)
```

Arguments

df	tidied postsecondary matriculation dataframe, eg output of enrich_matric_counts()

Value

df containing all-public sector postsecondary matriculation rates

allpublic_parcc_aggs *Calculate All Public Options PARCC Aggregates*

Description

Calculate All Public Options PARCC Aggregates

Usage

```
allpublic_parcc_aggs(df)
```

Arguments

df	tidied PARCC dataframe, eg output of fetch_parcc

Value

df containing all public option aggregates by district

allpublic_spec_pop_aggs *Calculate All City Special Populations aggregates*

Description

Calculate All City Special Populations aggregates

Usage

```
allpublic_spec_pop_aggs(df)
```

Arguments

df	tidied grad count dataframe, eg output of ‘fetch_reportcard_special_pop“

Value

df with all city aggregates per host city

allpublic_sped_aggs *Calculate Charter Sector SPED aggregates*

Description

Calculate Charter Sector SPED aggregates

Usage

```
allpublic_sped_aggs(df)
```

Arguments

df	tidied grad count dataframe, eg output of fetch_sped

Value

df containing charter sector aggregate sped

ALLPUBLIC_SUFFIX *Suffix for all public aggregations*

Description

Suffix for all public aggregations

Usage

```
ALLPUBLIC_SUFFIX
```

Format

An object of class `character` of length 1.

ALT_SCHOOL_ID *Alternative school code (used in some graduation files)*

Description

Alternative school code (used in some graduation files)

Usage

```
ALT_SCHOOL_ID
```

Format

An object of class `character` of length 1.

`analyze_retention_patterns` *Analyze Staff Retention Patterns*

Description

Analyzes staff retention patterns across multiple years and demographic subgroups. Calculates retention rates, turnover rates, and stability indices with trend detection.

Usage

```
analyze_retention_patterns(df_list, by_subgroup = TRUE)
```

Arguments

`df_list`	A named list of data frames from different years. Each element should be named by its end_year (e.g., list("2022" = df_2022, "2024" = df_2024)). Data frames should be from `fetch_staff_retention` or similar containing staff retention data.
`by_subgroup`	Logical; if TRUE (default), analyze retention patterns by demographic subgroups. If FALSE, aggregate across all staff.

Details

The stability index is calculated as:

$$stability = (retention_rate + (100 - turnover_rate))/2$$

Higher values indicate greater staff stability (retained staff + low turnover).

Trend classification uses linear regression on multi-year stability scores:

- improving - Positive slope (stability increasing over time)
- stable - Near-zero slope or insufficient data
- declining - Negative slope (stability decreasing over time)

Value

Data frame with:

- year - Year identifier
- location_id - Combined location identifier
- subgroup - Demographic subgroup (if by_subgroup = TRUE)
- retention_rate - Percentage of staff who returned from previous year
- turnover_rate - Percentage of new staff in current year
- stability_index - Composite score combining retention and turnover (0-100)
- trend - Trend classification: "improving", "stable", "declining", or "insufficient_data"

Examples

```
## Not run:
# Fetch retention data for multiple years
retain_2022 <- fetch_staff_retention(2022)
retain_2023 <- fetch_staff_retention(2023)
retain_2024 <- fetch_staff_retention(2024)

# Combine into named list
df_list <- list(
  "2022" = retain_2022,
  "2023" = retain_2023,
  "2024" = retain_2024
)

# Analyze overall retention patterns
overall_patterns <- analyze_retention_patterns(df_list, by_subgroup = FALSE)

# Analyze by demographic subgroup
subgroup_patterns <- analyze_retention_patterns(df_list, by_subgroup = TRUE)

# View schools with declining retention
subgroup_patterns %>%
  dplyr::filter(trend == "declining") %>%
  dplyr::select(school_name, year, subgroup, retention_rate, trend)

## End(Not run)
```

---

arrange_enr *Arrange enrollment file columns*

Description

Put an enrollment file in the correct column order.

Usage

```
arrange_enr(df)
```

Arguments

df	Cleaned enrollment file

Value

Data frame with columns in standard order

assessment_peer_percentile *Assessment Peer Percentile*

Description

calculates the percentile rank of a school, defined as the percent of comparison schools with lesser or equal performance, for both scale score, percent proficiency, and a composite average of the two. USE CAUTION when invoking this function. This function accepts WHATEVER grouping variables are present in the input data. If your data is not grouped in an intelligible or meaningful way, you may get nonsense percentile ranks (eg, across grade levels, years, subgroups, etc). Please start with the convenience wrappers ‘statewide_peer_percentile()‘ and ‘dfg_peer_percentile()‘ to examine percentile rank using comparison groups that are sensible.

Usage

```
assessment_peer_percentile(df)
```

Arguments

df	tidy PARCC df

Value

PARCC df with percentile ranks

build_sped_url *Build SPED data URL*

Description

Build SPED data URL

Usage

```
build_sped_url(end_year)
```

Arguments

end_year	ending school year

Value

URL string for the SPED data file

cache_exists *Check if a key exists in cache*

Description

Check if a key exists in cache

Usage

```
cache_exists(key)
```

Arguments

key	Cache key

Value

Logical

cache_get *Get data from cache*

Description

Get data from cache

Usage

```
cache_get(key)
```

Arguments

key	Cache key

Value

Cached data or NULL if not found

cache_set *Store data in cache*

Description

Store data in cache

Usage

```
cache_set(key, value)
```

Arguments

key	Cache key
value	Data to cache

Value

The value (invisibly)

calculate_access_rate *Calculate equity access rate*

Description

Calculates the share of students (overall or by subgroup) attending schools that meet or exceed a performance threshold. This replicates the MarGrady metric: "the share of Black students in Newark attending a school that beat the state proficiency average."

Usage

```
calculate_access_rate(
  df_enrollment,
  df_performance,
  performance_metric = "proficient_above",
  threshold_type = c("state_avg", "absolute", "percentile"),
  threshold_value = NULL,
  enrollment_col = "n_students",
  subgroup = NULL,
  join_cols = c("end_year", "county_id", "district_id", "school_id")
)
```

Arguments

df_enrollment	School-level enrollment data with demographic counts
df_performance	School-level performance data
performance_metric	Character. Column in df_performance to use as the performance measure. Default "proficient_above".

threshold_type	Character. How to define "high-performing": • "state_avg": Above the state average for that year/grade/subject • "absolute": Above a fixed numeric threshold • "percentile": Above a percentile threshold within peers
threshold_value	Numeric. The threshold value. Required for "absolute" and "percentile" types. For "state_avg", this is ignored.
enrollment_col	Character. Column in df_enrollment containing student counts. Default "n_students".
subgroup	Character. Which subgroup to calculate access for. If NULL, calculates for total enrollment. Default NULL.
join_cols	Character vector. Columns to join enrollment and performance data on. Default c("end_year", "county_id", "district_id", "school_id").

Value

Dataframe with columns:

- Year and entity identifiers
- `n_students_total`: Total students in subgroup
- `n_students_above`: Students in above-threshold schools
- `pct_access`: Percent with access to high-performing schools

`calculate_agg_parcc_prof` *Aggregate PARCC results across multiple grade levels*

Description

a wrapper around fetch_parcc and parcc_aggregate_calcs that simplifies the calculation of multi-grade PARCC aggregations

Usage

```
calculate_agg_parcc_prof(end_year, subj, gradespan = "3-11")
```

Arguments

end_year	school year / testing year
subj	one of 'ela' or 'math'
gradespan	one of c('3-11', '3-8', '9-11'). default is '3-11'

Value

dataframe

`calculate_subgroup_gap` *Calculate achievement gap between two subgroups*

Description

Calculates the difference in a metric between two subgroups within each entity (district/school) and year. The gap is calculated as subgroup_a - subgroup_b, so positive values mean subgroup_a outperforms.

Usage

```
calculate_subgroup_gap(
  df,
  metric_col,
  subgroup_a,
  subgroup_b,
  year_col = "end_year",
  entity_cols = "district_id"
)
```

Arguments

`df`	Dataframe with a 'subgroup' column and the metric of interest
`metric_col`	Character. The column containing the metric to compare.
`subgroup_a`	Character. The reference subgroup (typically majority/advantaged).
`subgroup_b`	Character. The comparison subgroup (typically minority/disadvantaged).
`year_col`	Character. Year column name. Default "end_year".
`entity_cols`	Character vector. Columns identifying the entity (e.g., c("district_id") or c("county_id", "district_id", "school_id")).

Value

Dataframe with one row per entity-year containing:

- Original entity and year columns
- `{metric}_a`: Value for subgroup_a
- `{metric}_b`: Value for subgroup_b
- `{metric}_gap`: Absolute gap (a - b)
- `{metric}_gap_pct`: Relative gap as percent of subgroup_a

Examples

```
## Not run:
# Calculate Black-White graduation rate gap
grate %>%
  filter(subgroup %in% c("white", "black")) %>%
  calculate_subgroup_gap(
    metric_col = "grad_rate",
```

```
    subgroup_a = "white",
    subgroup_b = "black"
  )

## End(Not run)
```

`calc_discipline_rates_by_subgroup` *Calculate Discipline Rates by Subgroup*

Description

Calculates discipline rates by subgroup for disproportionality analysis. Computes rates per specified base (default 1000 students) and calculates risk ratios compared to total population.

Usage

```
calc_discipline_rates_by_subgroup(df, rate_per = 1000)
```

Arguments

`df`	A data frame from `fetch_disciplinary_removals`, `fetch_violence_vandalism_hib`, or similar discipline data. Must contain subgroup column and a count/number column.
`rate_per`	Base for rate calculation (default: 1000). For example, rate_per = 1000 calculates incidents per 1000 students.

Value

Data frame with discipline rates including:

- All original columns from input data
- discipline_rate - Incidents per rate_per students
- percent_by_subgroup - Percentage of total incidents by subgroup
- risk_ratio - Ratio of subgroup rate to total population rate (values > 1 indicate higher risk than total population)

Examples

```
## Not run:
# Get disciplinary removals data
discipline <- fetch_disciplinary_removals(2024)

# Calculate rates per 1000 students
rates <- calc_discipline_rates_by_subgroup(discipline, rate_per = 1000)

# View disproportionality for racial subgroups
rates %>%
  dplyr::filter(subgroup %in% c("black", "hispanic", "white")) %>%
  dplyr::select(school_name, subgroup, discipline_rate, risk_ratio)

## End(Not run)
```

`calc_staff_diversity_metrics` *Calculate Staff Diversity Metrics*

Description

Calculates diversity indices for staff demographics using Simpson's Diversity Index. Computes racial and gender diversity scores with percentile rankings.

Usage

```
calc_staff_diversity_metrics(df, metrics = c("racial"))
```

Arguments

`df`	A data frame from `fetch_staff_demographics`. Should contain staff demographic breakdowns with counts.
`metrics`	Character vector specifying which diversity metrics to calculate. Options: "racial" (default), "gender", or both c("racial", "gender").

Details

Simpson's Diversity Index is calculated as:

$$D = 1 - \sum(p_i^2)$$

where p_i is the proportion of staff in category i.

Values range from 0 (no diversity - all staff in one category) to 1 (maximum diversity - staff evenly distributed across all categories).

Value

Data frame with:

- location_id - Combined location identifier
- diversity_index - Overall Simpson's Diversity Index (0-1 scale, higher = more diverse)
- racial_diversity_score - Racial diversity score (0-1 scale)
- gender_diversity_score - Gender diversity score (0-1 scale)
- diversity_percentile_rank - Percentile rank vs all schools (0-100)
- diversity_quintile - Quintile rank (1-5, 5 = most diverse)

Examples

```
## Not run:
# Get staff demographics data
demographics <- fetch_staff_demographics(2024)

# Calculate racial diversity
racial_div <- calc_staff_diversity_metrics(demographics, metrics = "racial")
```

```
# Calculate both racial and gender diversity
all_div <- calc_staff_diversity_metrics(demographics,
                                        metrics = c("racial", "gender"))

# View most diverse schools
all_div %>%
  dplyr::arrange(dplyr::desc(diversity_index)) %>%
  dplyr::select(school_name, diversity_index, diversity_quintile)

## End(Not run)
```

`calc_student_staff_ratio` *Calculate and Analyze Student-Staff Ratios*

Description

Calculates student-to-staff ratios with categorization and state comparisons. Analyzes ratios for overall staff, teachers, administrators, and support staff.

Usage

```
calc_student_staff_ratio(df, ratio_type = "overall")
```

Arguments

`df`	A data frame from `fetch_staff_ratios` or similar. Should contain staff count and student enrollment columns.
`ratio_type`	One of "overall" (default), "teachers", "administrators", or "support". Specifies which staff category to analyze.

Value

Data frame with original columns plus:

- ratio_type - The type of ratio calculated
- student_staff_ratio - Students per staff member (higher = more students per staff)
- ratio_category - "low" (< 10), "medium" (10-20), or "high" (> 20) based on benchmarks
- percent_change_vs_state - Percent difference from state average (if state data available)

Examples

```
## Not run:
# Get staff ratio data
ratios <- fetch_staff_ratios(2024)

# Calculate overall student-staff ratios
overall_ratios <- calc_student_staff_ratio(ratios, ratio_type = "overall")

# Calculate teacher-specific ratios
teacher_ratios <- calc_student_staff_ratio(ratios, ratio_type = "teachers")
```

```
# View schools with highest student-teacher ratios
teacher_ratios %>%
  dplyr::arrange(dplyr::desc(student_staff_ratio)) %>%
  dplyr::select(school_name, student_staff_ratio, ratio_category)

## End(Not run)
```

`charter_city` *Charter School Host District Mapping*

Description

A dataset containing charter schools and their host districts/cities in New Jersey. Charter schools are public schools that operate independently but are accountable to their authorizing districts (host districts).

Usage

```
charter_city
```

Format

A data frame with 137 rows and 6 columns:

district_id Charter school district identifier

district_name Charter school district name

host_county_id County code of the host district

host_county_name County name of the host district

host_district_id Host district identifier

host_district_name Host district name

Source

NJ Department of Education

`CHARTER_COUNTY_ID` *Charter county code (all charters are assigned to county 80)*

Description

Charter county code (all charters are assigned to county 80)

Usage

```
CHARTER_COUNTY_ID
```

Format

An object of class `character` of length 1.

charter_market_share *Calculate charter market share for host cities*

Description

Calculates the percentage of public school students enrolled in charter schools for each host city over time.

Usage

```
charter_market_share(df_enrollment, year_col = "end_year")
```

Arguments

df_enrollment	Enrollment data with charter sector and all-public aggregates (combine fetch_enr() output with charter_sector_enr_aggs() and allpublic_enr_aggs())
year_col	Character. Year column. Default "end_year".

Value

Dataframe with charter market share by city and year

charter_sector_enr_aggs *Calculate Charter Sector Enrollment Aggregates*

Description

Calculate Charter Sector Enrollment Aggregates

Usage

```
charter_sector_enr_aggs(df)
```

Arguments

df	a tidied enrollment dataframe, eg output of ‘fetch_enr‘

Value

dataframe with charter sector aggregates per host city

charter_sector_gcount_aggs *Calculate Charter Sector Grad Count aggregates*

Description

Calculate Charter Sector Grad Count aggregates

Usage

```
charter_sector_gcount_aggs(df)
```

Arguments

df	tidied grad count dataframe, eg output of fetch_grad_count

Value

df containing charter sector aggregate grad count

charter_sector_grate_aggs *Calculate Charter Sector Grad Rate aggregates*

Description

Calculate Charter Sector Grad Rate aggregates

Usage

```
charter_sector_grate_aggs(df)
```

Arguments

df	tidied grate dataframe, eg output of fetch_grate

Value

df containing charter sector aggregate grad rate

charter_sector_matric_aggs

Calculate Charter Sector postsecondary matriculation aggregates

Description

Calculate Charter Sector postsecondary matriculation aggregates

Usage

```
charter_sector_matric_aggs(df)
```

Arguments

df	tidied matriculation dataframe, eg output of enrich_matric_counts

Value

df containing charter sector matriculation aggregates

charter_sector_parcc_aggs

Calculate Charter Sector PARCC aggregates

Description

Calculate Charter Sector PARCC aggregates

Usage

```
charter_sector_parcc_aggs(df)
```

Arguments

df	tidied PARCC dataframe, eg output of fetch_parcc

Value

df containing charter sector aggregate PARCC performance

`charter_sector_spec_pop_aggs`
Calculate Charter Sector Special Populations Aggregates

Description

Calculate Charter Sector Special Populations Aggregates

Usage

```
charter_sector_spec_pop_aggs(df)
```

Arguments

`df`	a tidied enrollment dataframe, eg output of 'fetch_reportcard_special_pop'

Value

dataframe with charter sector aggregates per host city

`charter_sector_sped_aggs`
Calculate Charter Sector SPED Aggregates

Description

Calculate Charter Sector SPED Aggregates

Usage

```
charter_sector_sped_aggs(df)
```

Arguments

`df`	a tidied district special education dataframe, e.g. output of 'fetch_sped'

Value

dataframe with charter sector aggregates per host city

CHARTER_SECTOR_SUFFIX *Suffix for charter sector aggregations*

Description

Suffix for charter sector aggregations

Usage

```
CHARTER_SECTOR_SUFFIX
```

Format

An object of class `character` of length 1.

check_url_accessible *Check if a URL is accessible*

Description

Performs a HEAD request to verify the URL exists without downloading.

Usage

```
check_url_accessible(url, timeout = 10)
```

Arguments

`url`	URL to check
`timeout`	Timeout in seconds (default 10)

Value

Logical indicating if URL is accessible

Examples

```
## Not run:
check_url_accessible("https://www.nj.gov/education/")

## End(Not run)
```

`city_ecosystem_summary` *Summarize sector performance within a city*

Description

Creates a summary showing charter sector, traditional district, and all-public performance for a specific host city, including peer percentile ranks for each.

Usage

```
city_ecosystem_summary(
  df,
  metric_col,
  host_district_id,
  peer_type = "statewide",
  year_col = "end_year"
)
```

Arguments

`df`	Combined data including sector aggregates (output of `*_aggs()` functions combined with base data)
`metric_col`	Character. Metric to summarize.
`host_district_id`	Character. The host district ID (e.g., "3570" for Newark).
`peer_type`	Character. Peer group for percentile calculation.
`year_col`	Character. Year column. Default "end_year".

Value

Summary dataframe with sectors as rows, metrics and percentiles as columns

Examples

```
## Not run:
# Newark ecosystem summary
city_ecosystem_summary(
  df = grate_with_aggs,
  metric_col = "grad_rate",
  host_district_id = "3570"
)

## End(Not run)
```

clean_6yr_grad_subgroups *Clean 6-year graduation subgroup names*

Description

Standardizes subgroup names from the SPR 6-year graduation data to match the naming conventions used elsewhere in the package.

Usage

```
clean_6yr_grad_subgroups(group)
```

Arguments

group	Vector of subgroup names

Value

Vector of cleaned subgroup names

clean_cds_fields *Clean up CDS field names*

Description

Standardizes county, district, and school column names to consistent naming conventions (county_code, district_code, school_code, etc.).

Usage

```
clean_cds_fields(df, tges = FALSE)
```

Arguments

df	data frame with county, district and school variables
tges	if run in the taxpayers guide to ed spending (tges) mode, 'district' resolves to district code. defaults to FALSE.

Value

df, with consistent county_code, district_code, school_code fields

clean_enr_data *Clean enrollment data types*

Description

All columns come back as character; coerce some back to numeric.

Usage

```
clean_enr_data(df)
```

Arguments

df	An enrollment data frame with standardized column names

Value

Data frame with correct data types

clean_enr_grade *Tidy up the grade level field on enrollment data*

Description

Tidy up the grade level field on enrollment data

Usage

```
clean_enr_grade(df)
```

Arguments

df	An enrollment data file. clean_enr_grade is part of a set of chained cleaning functions that live inside process_enr.

Value

Data frame with cleaner grade_level column

clean_enr_names *Clean enrollment column names*

Description

Give consistent names to the enrollment files across all years.

Usage

```
clean_enr_names(df)
```

Arguments

df	An enrollment data frame (eg output of 'get_raw_enr')

Value

Data frame with standardized column names

clean_grate_names *Clean graduation rate names*

Description

Standardizes subgroup names in graduation data.

Usage

```
clean_grate_names(name_vector)
```

Arguments

name_vector	Vector of subgroup names

Value

Vector of cleaned subgroup names

clean_name *Internal helper to map a column name*

Description

Internal helper to map a column name

Usage

```
clean_name(df_names, clean_list)
```

Arguments

df_names	Column name to clean
clean_list	Named list mapping old names to new names

Value

Cleaned column name

clean_name_vector *Clean name vector to snake_case*

Description

Internal function to standardize column names using snake_case convention.

Usage

```
clean_name_vector(x)
```

Arguments

x	character vector of names

Value

character vector of cleaned names

clean_rc_enrollment *Clean Report Card Enrollment*

Description

Clean Report Card Enrollment

Usage

```
clean_rc_enrollment(list_of_dfs)
```

Arguments

list_of_dfs	output of get_one_rc_database (ie, a list of data.frames)

Value

data frame with school enrollment

clean_sped_df *Clean SPED data*

Description

Cleans and standardizes SPED data from NJ DOE.

Usage

```
clean_sped_df(df, end_year)
```

Arguments

df	raw data frame with cleaned names, output of get_raw_sped with clean_sped_names applied.
end_year	academic year, ending year - eg 2023-2024 is 2024.

Value

cleaned data frame

clean_spr_subgroups *Clean SPR subgroup names*

Description

Standardizes subgroup names from SPR database to match package naming conventions.

Usage

```
clean_spr_subgroups(group)
```

Arguments

group	Vector of subgroup names

Value

Vector of cleaned subgroup names

common_fwf_req *common_fwf_req*

Description

common fwf logic across various assessment types. DRY.

Usage

```
common_fwf_req(url, layout)
```

Arguments

url	file location
layout	data frame containing fixed-width file column specifications

Value

layout layout to use

compare_discipline_across_years
Compare Discipline Across Years

Description

Compares discipline metrics across multiple years, calculating year-over-year changes and identifying long-term trends.

Usage

```
compare_discipline_across_years(df_list, metrics = NULL)
```

Arguments

df_list	A named list of data frames from different years. Each element should be named by its end_year (e.g., list("2022" = df_2022, "2024" = df_2024)). Data frames should be from fetch_disciplinary_removals, fetch_violence_vandalism_hib, or similar.
metrics	Character vector of metrics to compare. If NULL (default), attempts to auto-detect numeric metric columns. Metrics should match column names in the data (e.g., "discipline_rate" if calculated).

Value

Data frame with year-over-year comparisons including:

- year - Year identifier
- location_id - Combined location identifier
- metric_name - Name of the metric
- metric_value - Value of the metric in this year
- year_over_year_change - Change from previous year
- year_over_year_pct_change - Percentage change from previous year
- multi_year_trend - Trend classification: "increasing", "decreasing", "stable", or "insufficient_data"

Examples

```
## Not run:
# Fetch data for multiple years
disc_2022 <- fetch_disciplinary_removals(2022)
disc_2023 <- fetch_disciplinary_removals(2023)
disc_2024 <- fetch_disciplinary_removals(2024)

# Combine into named list
df_list <- list(
  "2022" = disc_2022,
  "2023" = disc_2023,
  "2024" = disc_2024
)

# Compare trends
```

```
trends <- compare_discipline_across_years(df_list)

# View schools with increasing discipline rates
trends %>%
  dplyr::filter(metric_name == "discipline_rate", multi_year_trend == "increasing")

## End(Not run)
```

`define_peer_group` *Define a peer comparison group*

Description

Creates grouping on a dataframe that defines which entities are compared to each other for percentile rank calculations. The returned df can be piped directly to `percentile_rank()`.

Usage

```
define_peer_group(
  df,
  peer_type = c("statewide", "dfg", "county", "custom"),
  custom_ids = NULL,
  level = c("district", "school"),
  year_col = "end_year",
  additional_groups = NULL
)
```

Arguments

`df`	Dataframe with entity identifiers
`peer_type`	Character. One of: • "statewide": Compare to all districts/schools in state • "dfg": Compare within District Factor Group • "county": Compare within county • "custom": Compare to custom list of district_ids
`custom_ids`	Character vector of district_ids for custom peer groups. Only used when peer_type = "custom".
`level`	Character. One of "district" or "school" - what level to compare.
`year_col`	Character. Name of the year column. Default "end_year".
`additional_groups`	Character vector. Additional columns to group by (e.g., "subgroup", "grade", "test_name").

Value

Grouped dataframe ready for `add_percentile_rank()`

Examples

```
## Not run:
# DFG peer group for districts
grate %>%
  define_peer_group("dfg", level = "district") %>%
  add_percentile_rank("grad_rate")

# Custom peer group (DFG A districts)
grate %>%
  define_peer_group("custom", custom_ids = dfg_a_districts, level = "district") %>%
  add_percentile_rank("grad_rate")

## End(Not run)
```

---

dfg_peer_percentile *Calculate DFG peer percentile by grade*

Description

calculates DFG percentile by grade/test

Usage

```
dfg_peer_percentile(df)
```

Arguments

df	data.frame with PARCC/assessment data containing DFG and required columns

Value

data.frame with percent proficient and scale score percentile rank

---

dfg_percentile_rank *Calculate DFG peer percentile for any metric*

Description

Calculates percentile rank within District Factor Group for any metric column. This is a convenience wrapper that combines `define_peer_group()` and `add_percentile_rank()`.

Usage

```
dfg_percentile_rank(
  df,
  metric_col,
  year_col = "end_year",
  additional_groups = NULL
)
```

Arguments

df	Dataframe with district data including 'dfg' column
metric_col	Character. The column to rank on.
year_col	Character. The year column. Default "end_year".
additional_groups	Character vector. Additional grouping columns.

Value

df with percentile rank columns

---

district_matric_aggs *Aggregates matriculation data by district*

Description

Only school-level matriculation data reported before 2017. This function approximates district level results. If schools within the district do not report for certain subgroups, the approximation will be further off.

Usage

```
district_matric_aggs(df)
```

Arguments

df	output of enrich_matric_counts

Value

A data frame of district aggregations

---

district_name_to_id *Given friendly district names, returns ids*

Description

Given friendly district names, returns ids

Usage

```
district_name_to_id(district_names, df)
```

Arguments

district_names	vector or list of friendly district names
df	df used to generate the friendly names

Value

vector of district_ids

DISTRICT_TOTAL_SCHOOL_ID *District total school code*

Description

District total school code

Usage

```
DISTRICT_TOTAL_SCHOOL_ID
```

Format

An object of class `character` of length 1.

download_and_clean_pr *Download and clean performance report data*

Description

Download and clean performance report data

Usage

```
download_and_clean_pr(tmp_pr, url, end_year)
```

Arguments

`tmp_pr`	path to tempfile
`url`	url to download
`end_year`	report year

Value

list of dataframes

`ecosystem_trend` *Track sector ecosystem dynamics over time*

Description

Tracks charter market share, sector performance gap, and all-public percentile over time for a host city. Answers questions like "As Newark's charter share grew, how did overall city performance change?"

Usage

```
ecosystem_trend(
  df_enrollment,
  df_performance,
  host_district_id,
  metric_col,
  peer_type = "statewide",
  year_col = "end_year"
)
```

Arguments

`df_enrollment`	Enrollment data (output of `fetch_enr()`) with sector aggregates
`df_performance`	Performance data (e.g., graduation rates) with sector aggregates
`host_district_id`	Character. Host district ID (e.g., "3570").
`metric_col`	Character. Performance metric to track.
`peer_type`	Character. Peer group for percentile. Default "statewide".
`year_col`	Character. Year column. Default "end_year".

Value

Dataframe with yearly ecosystem metrics:

- `charter_enrollment`: Students in charter sector
- `total_enrollment`: All public students in city
- `charter_share`: Percent of students in charters
- `sector_gap`: Charter - district performance difference
- `allpublic_percentile`: City's overall percentile rank

ELEMENTARY_GRADES *Elementary grades (K-5)*

Description

Elementary grades (K-5)

Usage

```
ELEMENTARY_GRADES
```

Format

An object of class `character` of length 6.

enrich_grad_count *Enrich report card matriculation percentages with grad counts*

Description

Joins graduation count data to a data frame containing subgroup percentages.

Usage

```
enrich_grad_count(df, end_year)
```

Arguments

`df`	Data frame including subgroup percentages
`end_year`	Numeric end year of grad counts to join

Value

data_frame with graduated_count and cohort_count columns added

enrich_matric_counts *Enrich matriculation rates with counts from grad counts*

Description

Enrich matriculation rates with counts from grad counts

Usage

```
enrich_matric_counts(df, type = "16 month")
```

Arguments

df	data frame of including matriculation percentages
type	character string specifying matriculation type. One of "16 month" or "12 month"

Value

data_frame

enrich_rc_enrollment *Enrich report card subgroup percentages with best guesses at subgroup numbers*

Description

Enrich report card subgroup percentages with best guesses at subgroup numbers

Usage

```
enrich_rc_enrollment(df)
```

Arguments

df	data frame of including subgroup percentages

Value

data_frame

enrich_school_city_ward *Enrich School Data with City Ward*

Description

Enrich School Data with City Ward

Usage

```
enrich_school_city_ward(df)
```

Arguments

df	any dataframe with a district_id

Value

df enriched with ward, if geographic data is 'registered' for a given district

enrich_school_latlong *Enrich School Data with Lat / Long*

Description

Enrich School Data with Lat / Long

Usage

```
enrich_school_latlong(df, use_cache = TRUE, api_key = "")
```

Arguments

df	dataframe to be enriched
use_cache	if TRUE, will read from cache of school info / lat lng stored on TODO
api_key	optional, personal google maps API key

Value

dataframe enriched with lat lng

`enr_aggs` *Calculate enrollment aggregates*

Description

Aggregates gender columns into racial subgroups (e.g., white_m + white_f = white)

Usage

```
enr_aggs(df)
```

Arguments

`df`	Cleaned enrollment dataframe, eg output of ‘clean_enr_data‘

Value

Data frame with aggregated columns

`enr_grade_aggs` *Custom Enrollment Grade Level Aggregates*

Description

Creates aggregations for common grade groupings: PK (Any), K (Any), K-12, K-12UG, K-8, and HS.

Usage

```
enr_grade_aggs(df)
```

Arguments

`df`	A tidy enrollment df

Value

df of aggregated enrollment data

ENR_VALID_YEARS *Enrollment data valid years Note: 1999 data was removed from NJ DOE website*

Description

Enrollment data valid years Note: 1999 data was removed from NJ DOE website

Usage

```
ENR_VALID_YEARS
```

Format

An object of class `integer` of length 26.

extract_rc_AP *Extract Report Card Advanced Placement Data*

Description

Extract Report Card Advanced Placement Data

Usage

```
extract_rc_AP(list_of_prs, school_only = TRUE, cds_identifiers = TRUE)
```

Arguments

`list_of_prs`	output of get_rc_databases (ie, a list where each element is) a list of data.frames
`school_only`	some years have district average results, not just school-level. if school_only, return only school data. default is TRUE
`cds_identifiers`	add the county, district and school name? default is TRUE

Value

data frame with all the years of AP participation and AP achievement present in in the input

extract_rc_cds *Extract Report Card CDS*

Description

Extract Report Card CDS

Usage

```
extract_rc_cds(list_of_prs)
```

Arguments

list_of_prs	output of get_rc_databases (ie, a list where each element is) a list of data.frames

Value

data frame with county, district and school ids and identifiers for all years present in the input

extract_rc_college_matric *Extract Report Card Matriculation Rates*

Description

Extract Report Card Matriculation Rates

Usage

```
extract_rc_college_matric(
  list_of_prs,
  type = "16 month",
  school_only = TRUE,
  cds_identifiers = TRUE
)
```

Arguments

list_of_prs	output of get_rc_databases (ie, a list where each element is) a list of data.frames
type	character string specifying matriculation type. One of "16 month" or "12 month"
school_only	some years have district average results, not just school-level. if school_only, return only school data. default is TRUE
cds_identifiers	add the county, district and school name? default is TRUE

Value

data frame with all the years of 4 year and 2 year college matriculation data present in in the input

extract_rc_enrollment *Extract Report Card Enrollment*

Description

Extract Report Card Enrollment

Usage

```
extract_rc_enrollment(list_of_prs, cds_identifiers = TRUE)
```

Arguments

list_of_prs	output of get_rc_databases (ie, a list where each element is a list of data.frames)
cds_identifiers	add the county, district and school name? default is TRUE

Value

data frame with school enrollment

extract_rc_SAT *Extract Report Card SAT School Averages*

Description

Extract Report Card SAT School Averages

Usage

```
extract_rc_SAT(list_of_prs, school_only = TRUE, cds_identifiers = TRUE)
```

Arguments

list_of_prs	output of get_rc_databases (ie, a list where each element is) a list of data.frames
school_only	some years have district average results, not just school-level. if school_only, return only school data. default is TRUE
cds_identifiers	add the county, district and school name? default is TRUE

Value

data frame with all years of SAT School Averages present in the input

`fetch_6yr_grad_rate` *Fetch 6-Year Graduation Rate data*

Description

Downloads and processes 6-year graduation rate data from the NJ School Performance Reports database. This data shows the percentage of students who graduated within six years of entering high school.

Usage

```
fetch_6yr_grad_rate(end_year, level = "school")
```

Arguments

`end_year`	A school year. Year is the end of the academic year - eg 2020-21 school year is end_year '2021'. Valid values are 2021-2024.
`level`	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Details

The 6-year graduation data is from a different source than the 4-year and 5-year data (SPR database vs ACGR files), which is why it has its own fetch function rather than being an option in `fetch_grad_rate()`.

Value

dataframe with 6-year graduation rates including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- subgroup - student group (total population, racial/ethnic groups, etc.)
- grad_rate_6yr - 6-year graduation rate (0-100 scale)
- continuing_rate - percentage of students still enrolled after 6 years
- non_continuing_rate - percentage who dropped out or left
- persistence_rate - graduates + continuing students (high school persistence)
- Aggregation flags (is_state, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 school-level 6-year graduation rates
grad6_2024 <- fetch_6yr_grad_rate(2024)

# Get district-level 6-year graduation rates
grad6_dist <- fetch_6yr_grad_rate(2024, level = "district")

## End(Not run)
```

fetch_absenteeism_by_grade
Fetch Absenteeism by Grade

Description

Downloads and extracts chronic absenteeism data broken down by grade level from the SPR database.

Usage

```
fetch_absenteeism_by_grade(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
level	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with chronic absenteeism by grade including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- subgroup - Student group (total population, racial/ethnic groups, etc.)
- grade_level - Grade level (PK, KG, 01-12)
- chronically_absent_rate - Percentage chronically absent (0-100)
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get school-level chronic absenteeism by grade
ca_grade <- fetch_absenteeism_by_grade(2024)

# Analyze kindergarten absenteeism
k_absent <- ca_grade %>%
  filter(grade_level == "KF", subgroup == "total population")

## End(Not run)
```

`fetch_access` *Fetch ACCESS for ELLs data*

Description

Downloads and processes ACCESS for ELLs (English Language Learner) assessment results. ACCESS measures English language proficiency for ELL students across grades K-12.

Usage

```
fetch_access(end_year, grade = "all")
```

Arguments

`end_year`	A school year. Valid values are 2022-2024.
`grade`	Grade level: "K" or 0 for Kindergarten, 1-12 for other grades, or "all" (default) to get all grades combined.

Value

Processed ACCESS dataframe with columns including:

- testing_year, assess_name, test_name, grade
- county_id, county_name, district_id, district_name
- school_id, school_name, valid_scores
- pct_l1 through pct_l6 (proficiency level percentages)
- proficient_above (L5 + L6 percentage)

Examples

```
## Not run:
# Get 2024 ACCESS results for all grades
access_2024 <- fetch_access(2024)

# Get 2024 ACCESS results for Grade 3 only
access_g3 <- fetch_access(2024, grade = 3)

# Get Kindergarten ACCESS results
access_k <- fetch_access(2024, grade = "K")

## End(Not run)
```

fetch_all_6yr_grad_rate *Fetch all 6-Year Graduation Rate data*

Description

Convenience function to download and combine all available 6-year graduation rate data into a single data frame.

Usage

```
fetch_all_6yr_grad_rate(level = "school")
```

Arguments

level	One of "school", "district", or "both". "both" combines school and district data. Default is "school".

Value

A data frame with all 6-year graduation rate results (2021-2024)

Examples

```
## Not run:
# Get all school-level 6-year graduation data
all_grad6 <- fetch_all_6yr_grad_rate()

# Get both school and district data
all_grad6_both <- fetch_all_6yr_grad_rate(level = "both")

## End(Not run)
```

fetch_all_access *Fetch all ACCESS for ELLs results*

Description

Convenience function to download and combine all available ACCESS for ELLs results into a single data frame.

Usage

```
fetch_all_access()
```

Value

A data frame with all ACCESS results (2022-2024, all grades)

Examples

```
## Not run:
# Get all ACCESS results (takes a while)
all_access <- fetch_all_access()

## End(Not run)
```

`fetch_all_chronic_absenteeism` *Fetch all Chronic Absenteeism data*

Description

Convenience function to download and combine all available chronic absenteeism data into a single data frame.

Usage

```
fetch_all_chronic_absenteeism()
```

Value

A data frame with all chronic absenteeism results (2017-2019, 2022-2024)

Examples

```
## Not run:
# Get all chronic absenteeism data
all_ca <- fetch_all_chronic_absenteeism()

## End(Not run)
```

`fetch_all_njgpa` *Fetch all NJGPA results*

Description

Convenience function to download and combine all NJGPA (graduation proficiency) results into a single data frame.

Usage

```
fetch_all_njgpa()
```

Value

A data frame with all NJGPA results (ELA and Math)

Examples

```
## Not run:
# Get all NJGPA results
all_njgpa <- fetch_all_njgpa()

## End(Not run)
```

---

fetch_all_parcc *Fetch all PARCC results*

Description

Convenience function to download and combine all PARCC/NJSLA results into single data frame, including ELA, Math, and Science assessments.

Usage

```
fetch_all_parcc(include_science = TRUE)
```

Arguments

include_science	Include science assessments (2019+)? Default is TRUE.

Value

A data frame with all PARCC/NJSLA results

Examples

```
## Not run:
# Get all PARCC/NJSLA results (takes a while)
all_parcc <- fetch_all_parcc()

# Exclude science assessments
all_parcc_no_sci <- fetch_all_parcc(include_science = FALSE)

## End(Not run)
```

---

fetch_all_parcc_with_progress *Fetch all PARCC/NJSLA results with progress*

Description

Downloads all available PARCC and NJSLA assessment results with progress indicators showing download status and ETA.

Usage

```
fetch_all_parcc_with_progress()
```

Value

A data frame with all PARCC/NJSLA results

Examples

```
## Not run:
all_results <- fetch_all_parcc_with_progress()

## End(Not run)
```

`fetch_apprenticeship_data` *Fetch Apprenticeship Data*

Description

Downloads and extracts apprenticeship participation data from the SPR database. Contains counts by year (2016-2023).

Usage

```
fetch_apprenticeship_data(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
`level`	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with apprenticeship participation including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- year_2016 through year_2023 - Number of apprentices by year
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 apprenticeship data
app <- fetch_apprenticeship_data(2024)

# Reshape to long format for analysis
app_long <- app %>%
  tidyr::pivot_longer(
    cols = starts_with("year_"),
    names_to = "apprenticeship_year",
    values_to = "apprenticeship_count",
    names_prefix = "year_"
```

```
  ) %>%
  filter(!is.na(apprenticeship_count))

## End(Not run)
```

`fetch_ap_participation`

Fetch AP/IB Participation and Performance Data

Description

Downloads and extracts Advanced Placement (AP) and International Baccalaureate (IB) coursework participation and exam performance from the SPR database.

Usage

```
fetch_ap_participation(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
`level`	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with AP/IB participation and performance including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- apib_coursework_school - Percentage students in AP/IB coursework
- apib_coursework_state - State percentage in AP/IB coursework
- apib_exam_school - Percentage taking AP/IB exams
- apib_exam_state - State percentage taking AP/IB exams
- ap3_ib4_school - Percentage scoring AP 3+ or IB 4+
- ap3_ib4_state - State percentage scoring AP 3+ or IB 4+
- dual_enrollment_school - Dual enrollment participation
- dual_enrollment_state - State dual enrollment percentage
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 AP/IB data
apib <- fetch_ap_participation(2024)

# Compare exam participation vs performance
apib %>%
  select(school_name, apib_exam_school, ap3_ib4_school)

## End(Not run)
```

fetch_ap_performance *Fetch AP/IB Performance Data (Alias)*

Description

Alias for fetch_ap_participation. Returns both participation and performance data for AP/IB coursework.

Usage

```
fetch_ap_performance(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024)
level	One of "school" or "district"

Value

Data frame with AP/IB participation and performance metrics

Examples

```
## Not run:
ap_perf <- fetch_ap_performance(2024)

## End(Not run)
```

fetch_biliteracy_seal *Fetch Seal of Biliteracy Data*

Description

Downloads and extracts Seal of Biliteracy data from the SPR database. The Seal of Biliteracy recognizes students who attain proficiency in English and one or more world languages.

Usage

```
fetch_biliteracy_seal(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
level	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with Seal of Biliteracy data including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- language - Language (e.g., "Spanish", "French", "Chinese")
- seals_earned - Number of seals earned in this language
- pct_12th_graders - Percentage of 12th graders earning seals
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 biliteracy seal data
biliteracy <- fetch_biliteracy_seal(2024)

# Top languages by seal count
biliteracy %>%
  group_by(language) %>%
  summarize(total_seals = sum(seals_earned, na.rm = TRUE)) %>%
  dplyr::arrange(desc(total_seals))

# Schools with most diverse language offerings
biliteracy %>%
  group_by(school_name) %>%
  summarize(num_languages = sum(seals_earned > 0, na.rm = TRUE)) %>%
  dplyr::arrange(desc(num_languages))

## End(Not run)
```

---

fetch_chronic_absenteeism

Fetch Chronic Absenteeism data

Description

Downloads and processes chronic absenteeism data from ESSA Accountability Workbooks. Data shows attendance rates by student subgroup; chronic absenteeism rate = 100 - attendance rate.

Usage

```
fetch_chronic_absenteeism(end_year)
```

Arguments

end_year	A school year. Valid values are 2017-2019 and 2022-2024.

Details

Note: This data is from ESSA accountability workbooks and covers schools included in ESSA accountability calculations (approximately 2,300+ schools). Data for 2020-2021 is not available due to COVID-19 pandemic disruptions.

Value

Processed chronic absenteeism dataframe with columns including:

- county_id, district_id, school_id, configuration
- Attendance rates by student subgroup (asian, black, hispanic, etc.)
- total_attendance_rate, total_chronic_absenteeism_rate

Examples

```
## Not run:
# Get 2024 chronic absenteeism data
ca_2024 <- fetch_chronic_absenteeism(2024)

# Calculate chronic absenteeism rates
ca_2024$chronic_absent_black <- 100 - ca_2024$attendance_black

## End(Not run)
```

`fetch_cte_participation`

Fetch CTE Participation Data

Description

Downloads and extracts Career and Technical Education (CTE) participation from the SPR database. Includes CTE participants and concentrators by subgroup.

Usage

```
fetch_cte_participation(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
`level`	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with CTE participation including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- subgroup - Student group (total population, racial/ethnic groups, etc.)
- cte_participants - Number or percentage of CTE participants
- cte_concentrators - Number or percentage of CTE concentrators
- state_cte_participants - State CTE participants (comparison)
- state_cte_concentrators - State CTE concentrators (comparison)
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 CTE participation
cte <- fetch_cte_participation(2024)

# Compare CTE participation across subgroups
cte %>%
  filter(school_id == "030") %>%
  select(subgroup, cte_participants)

## End(Not run)
```

`fetch_days_absent` *Fetch Days Absent Data*

Description

Downloads and extracts days absent statistics from the SPR database. This includes percentage distributions of students by absence ranges.

Usage

```
fetch_days_absent(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
`level`	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with days absent statistics including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- Percentage distribution columns: '0 '7 '20
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get school-level days absent distribution
days <- fetch_days_absent(2024)

# View absence distribution for a specific school
days %>%
  filter(school_id == "030") %>%
  select(school_name, `0% Absences`, `20% or higher`)

## End(Not run)
```

fetch_dfg *Fetch NJ District Factor Group (DFG) data*

Description

Downloads DFG classification data from NJ DOE. DFGs group districts by socioeconomic status for comparison purposes. DFG A represents the highest-need communities; DFG J represents the lowest-need.

Note: DFGs were last updated using 2000 Census data and are no longer maintained by NJ DOE, but remain useful for peer comparisons.

Usage

```
fetch_dfg(revision = 2000)
```

Arguments

revision	c(2000, 1990) Which census revision to use. Default 2000.

Value

data.frame with columns: county_code, county_name, district_code, district_name, dfg

References

https://www.nj.gov/education/stateaid/dfg.shtml

fetch_disciplinary_removals *Fetch Disciplinary Removals Data*

Description

Downloads discipline data (suspensions/expulsions) from SPR database.

Usage

```
fetch_disciplinary_removals(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024)
level	One of "school" or "district"

Value

Data frame with disciplinary actions

Examples

```
## Not run:
discipline <- fetch_disciplinary_removals(2024)

## End(Not run)
```

`fetch_dropout_rates` *Fetch Dropout Rate Data*

Description

Downloads dropout rate trends from SPR database.

Usage

```
fetch_dropout_rates(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024)
`level`	One of "school" or "district"

Value

Data frame with dropout rates

Examples

```
## Not run:
dropout <- fetch_dropout_rates(2024)

## End(Not run)
```

`fetch_enr` *Gets and processes a NJ enrollment file*

Description

‘fetch_enr‘ is a wrapper around ‘get_raw_enr‘ and ‘process_enr‘ that downloads and cleans enrollment data for a given year.

Usage

```
fetch_enr(end_year, tidy = FALSE)
```

Arguments

end_year	A school year. Year is the end of the academic year - eg 2006-07 school year is year '2007'. Valid values are 2000-2025.
tidy	If TRUE, takes the unwieldy wide data and normalizes into a long, tidy data frame with limited headers - constants (school/district name and code), subgroup (all the enrollment file subgroups), program/grade and measure (row_total, free lunch, etc).

Value

Data frame with processed enrollment data

Examples

```
## Not run:
# Get 2023 enrollment data
enr_2023 <- fetch_enr(2023)

# Get tidy (long format) enrollment data
enr_tidy <- fetch_enr(2023, tidy = TRUE)

## End(Not run)
```

---

fetch_enr_cached *Fetch enrollment data with caching*

Description

This is a cached wrapper around fetch_enr. On the first call with a given set of parameters, data is downloaded from NJ DOE. Subsequent calls with the same parameters return cached data instantly.

Usage

```
fetch_enr_cached(end_year, tidy = FALSE)
```

Arguments

end_year	A school year. Year is the end of the academic year - eg 2006-07 school year is year '2007'. Valid values are 2000-2025.
tidy	If TRUE, takes the unwieldy wide data and normalizes into a long, tidy data frame with limited headers - constants (school/district name and code), subgroup (all the enrollment file subgroups), program/grade and measure (row_total, free lunch, etc).

Value

Enrollment data frame (from cache if available)

See Also

fetch_enr, njsd_cache_info

Examples

```
## Not run:
# First call downloads data
enr <- fetch_enr_cached(2024)

# Second call returns cached data
enr <- fetch_enr_cached(2024)  # Instant!

# Check cache status
njsd_cache_info()

## End(Not run)
```

`fetch_enr_years` *Fetch multiple years of enrollment data with progress*

Description

Fetch multiple years of enrollment data with progress

Usage

```
fetch_enr_years(years, tidy = TRUE)
```

Arguments

`years`	Vector of years to fetch
`tidy`	Return tidy format? (default TRUE)

Value

Data frame with all enrollment data

Examples

```
## Not run:
enr_5yr <- fetch_enr_years(2020:2024)

## End(Not run)
```

fetch_essa_status *Fetch ESSA Accountability Status*

Description

Downloads ESSA accountability ratings from SPR database.

Usage

```
fetch_essa_status(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024)
level	One of "school" or "district"

Value

Data frame with ESSA status ratings

Examples

```
## Not run:
essa <- fetch_essa_status(2024)

## End(Not run)
```

fetch_gepa *gets and processes a GEPA file*

Description

fetch_gepa is a wrapper around get_raw_gepa and process_nj_assess that passes the correct file layout data to each function, given a end_year and grade.

Usage

```
fetch_gepa(end_year)
```

Arguments

end_year	a school end_year. end_year is the end of the academic year - eg 2006-07 school year is end_year '2007'. valid values are 2004-2007.

fetch_grad_count *Fetch Grad Counts*

Description

Downloads and processes graduation count data.

Usage

```
fetch_grad_count(end_year)
```

Arguments

end_year	End of the academic year - eg 2006-07 is 2007. Valid values are 2012-2024.

Value

dataframe with grad counts

Examples

```
## Not run:
# Get 2023 graduation counts
gcount_2023 <- fetch_grad_count(2023)

## End(Not run)
```

fetch_grad_rate *Fetch Grad Rate*

Description

Downloads and processes graduation rate data.

Usage

```
fetch_grad_rate(end_year, methodology = "4 year")
```

Arguments

end_year	End of the academic year - eg 2006-07 is 2007. Valid values are 2011-2024.
methodology	Character string specifying calculation methodology. One of "4 year" or "5 year".

Value

dataframe with grad rate

Examples

```
## Not run:
# Get 2023 graduation rates
grate_2023 <- fetch_grad_rate(2023)

# Get 5-year graduation rates
grate_5yr <- fetch_grad_rate(2019, methodology = "5 year")

## End(Not run)
```

`fetch_hspa` *gets and processes a HSPA file*

Description

`fetch_njask` is a wrapper around `get_raw_hspa` and `process_nj_assess` that passes the correct file layout data to each function, given a end_year and grade.

Usage

```
fetch_hspa(end_year)
```

Arguments

`end_year`	a school end_year. end_year is the end of the academic year - eg 2013-14 school year is end_year '2014'. valid values are 2004-2014.

`fetch_ib_participation`
Fetch IB Participation Data

Description

Downloads and extracts International Baccalaureate participation from the SPR database. Note: Most IB data is included in the AP/IB sheet, use `fetch_ap_participation` for comprehensive data.

Usage

```
fetch_ib_participation(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
`level`	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with IB participation metrics

Examples

```
## Not run:
ib <- fetch_ib_participation(2024)

## End(Not run)
```

`fetch_industry_credentials` *Fetch Industry Valued Credentials Data*

Description

Downloads and extracts industry-valued credentials earned by students from the SPR database. Organized by career cluster.

Usage

```
fetch_industry_credentials(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
`level`	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with industry credentials including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- career_cluster - Career cluster area (e.g., "Health Sciences", "STEM")
- students_enrolled - Number of students enrolled in CTE program
- earned_one_credential - Students earning at least one credential
- credentials_earned - Total industry credentials earned
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 industry credentials
creds <- fetch_industry_credentials(2024)

# Top schools by credentials earned
creds %>%
  group_by(school_name) %>%
  summarize(total_credentials = sum(credentials_earned, na.rm = TRUE)) %>%
  dplyr::arrange(desc(total_credentials))

## End(Not run)
```

---

fetch_many_tges *Fetch Multiple Cleaned Taxpayer's Guides to Educational Spending*

Description

Fetch Multiple Cleaned Taxpayer's Guides to Educational Spending

Usage

```
fetch_many_tges(end_year_vector)
```

Arguments

end_year_vector	vector of years. Current valid values are 2011 to 2017.

Value

list of lists of data frames

---

fetch_math_course_enrollment *Fetch Math Course Enrollment Data*

Description

Downloads math course participation data from SPR database.

Usage

```
fetch_math_course_enrollment(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024)
level	One of "school" or "district"

Value

Data frame with math course enrollment

Examples

```
## Not run:
math <- fetch_math_course_enrollment(2024)

## End(Not run)
```

`fetch_msgp` *Fetch mSGP*

Description

Fetch mSGP

Usage

```
fetch_msgp(end_year)
```

Arguments

`end_year`	ending school year. valid values are currently 2012-2018.

Value

dataframe with mSGP data for the year

`fetch_njask` *gets and processes a NJASK file*

Description

`fetch_njask` is a wrapper around `get_raw_njask` and `process_nj_assess` that passes the correct file layout data to each function, given an end_year and grade.

Usage

```
fetch_njask(end_year, grade)
```

Arguments

`end_year`	a school year. end_year is the end of the academic year - eg 2013-14 school year is end_year '2014'. valid values are 2004-2014.
`grade`	a grade level. valid values are 3,4,5,6,7,8

fetch_njgpa *Fetch NJGPA (NJ Graduation Proficiency Assessment) data*

Description

Downloads and processes NJGPA assessment results. NJGPA is the graduation requirement assessment introduced in 2022, replacing the previous PARCC-based graduation pathway.

Usage

```
fetch_njgpa(end_year, subj, tidy = FALSE)
```

Arguments

end_year	A school year. Valid values are 2022-2024.
subj	Assessment subject: 'ela' or 'math'
tidy	Clean up the data frame? Default is FALSE.

Value

Processed NJGPA dataframe

Examples

```
## Not run:
# Get 2023 NJGPA ELA results
njgpa_ela <- fetch_njgpa(2023, "ela")

# Get 2024 NJGPA Math results
njgpa_math <- fetch_njgpa(2024, "math")

## End(Not run)
```

fetch_old_nj_assess *a simplified interface into NJ assessment data*

Description

this is the workhorse function. given a end_year and a grade (valid years are 2004-present), `fetch_old_nj_assess` will call the appropriate function, process the raw text file, and return a data frame. `fetch_old_nj_assess` is a wrapper around all the individual subject functions (NJASK, HSPA, etc.), abstracting away the complexity of finding the right location/file layout.

Usage

```
fetch_old_nj_assess(end_year, grade, tidy = FALSE)
```

Arguments

end_year	a school year. end_year is the end of the academic year - eg 2013-14 school year is end_year '2014'. valid values are 2004-2014.
grade	a grade level. valid values are 3,4,5,6,7,8,11
tidy	if TRUE, takes the unwieldy, inconsistent wide data and normalizes into a long, tidy data frame with ~20 headers - constants(school/district name and code), subgroup (all the NCLB subgroups) and test_name (LAL, math, etc).

---

fetch_parcc *Gets and cleans up a PARCC data file*

Description

'fetch_parcc' is a wrapper around 'get_raw_parcc' and 'process_parcc' that gets a parcc file and performs any cleanup.

Usage

```
fetch_parcc(end_year, grade_or_subj, subj, tidy = FALSE)
```

Arguments

end_year	A school year. end_year is the end of the academic year - eg 2014-15 school year is end_year 2015. Valid values are 2015-2024.
grade_or_subj	Grade level (eg 8) OR math subject code (eg ALG1, GEO, ALG2). For science, valid grades are 5, 8, and 11.
subj	Assessment subject: 'ela', 'math', or 'science'. Science assessments are only available for 2019+ and grades 5, 8, 11.
tidy	Clean up the data frame to make it more compatible with NJASK naming conventions and do some additional calculations? Default is FALSE.

Value

Processed PARCC/NJSLA dataframe

Examples

```
## Not run:
# Get 2023 grade 4 math results
parcc_2023 <- fetch_parcc(2023, 4, "math")

# Get 2023 Algebra 1 results
alg1_2023 <- fetch_parcc(2023, "ALG1", "math")

# Get 2023 grade 8 science results
science_2023 <- fetch_parcc(2023, 8, "science")

## End(Not run)
```

`fetch_postsecondary` *Fetch Postsecondary Enrollment Rates*

Description

Downloads postsecondary enrollment rate data from the NJ DOE website. Data is sourced from the National Student Clearinghouse and shows the percentage of high school graduates enrolling in postsecondary institutions.

Usage

```
fetch_postsecondary()
```

Details

The data includes both school-level and district-level rates. Rates are reported as ranges because a small percentage of graduates cannot be matched to the National Student Clearinghouse database.

Two measurement types are available:

- `fall`: Enrollment in fall immediately after graduation
- `16_month`: Enrollment within 16 months of graduation

The `lower_bound` represents confirmed enrollments only (conservative). The `upper_bound` assumes non-matched graduates also enrolled (optimistic).

Value

A data frame with postsecondary enrollment rates in long format, containing columns for county, district, school identifiers, cohort_year, measurement_type (fall or 16_month), lower_bound, and upper_bound.

Examples

```
## Not run:
# Get all postsecondary enrollment data
postsec <- fetch_postsecondary()

# Filter for 16-month rates only
postsec_16mo <- postsec[postsec$measurement_type == "16_month", ]

# Filter for district-level data only
district_rates <- postsec[postsec$is_district, ]

## End(Not run)
```

fetch_reportcard_special_pop
Fetch Special Population data

Description

Fetch Special Population data

Usage

```
fetch_reportcard_special_pop(end_year)
```

Arguments

end_year	ending academic year. valid values are 2017, 2018, 2019

Value

data.frame with special population enrollment data

fetch_sat_participation
Fetch SAT/ACT/PSAT Participation Data

Description

Downloads and extracts college entrance exam participation rates from the SPR database. Includes SAT, ACT, and PSAT participation percentages.

Usage

```
fetch_sat_participation(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
level	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with SAT/ACT/PSAT participation rates including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- sat_participation - Percentage of students taking SAT
- act_participation - Percentage of students taking ACT
- psat_participation - Percentage of students taking PSAT

- state_sat - State SAT participation rate (comparison)
- state_act - State ACT participation rate (comparison)
- state_psat - State PSAT participation rate (comparison)
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 SAT/ACT participation
sat <- fetch_sat_participation(2024)

# Analyze SAT participation gaps
sat %>%
  filter(sat_participation < 50) %>%
  select(school_name, sat_participation, state_sat)

## End(Not run)
```

---

fetch_sat_performance *Fetch SAT/ACT/PSAT Performance Data*

Description

Downloads and extracts college entrance exam performance scores from the SPR database. Includes average scores and benchmark achievement rates.

Usage

```
fetch_sat_performance(end_year, level = "school", test_type = "all")
```

Arguments

end_year	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
level	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.
test_type	Filter by test type. Options are "SAT", "ACT", "PSAT", or "all" (default: "all")

Value

Data frame with SAT/ACT/PSAT performance scores including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- test_type - Type of test (SAT, ACT, or PSAT)
- subject - Test subject (e.g., "Math", "Evidence-Based Reading and Writing")
- school_avg - Average score for this school
- state_avg - State average score (comparison)
- benchmark - Whether school meets benchmark (if applicable)
- pct_benchmark - Percentage meeting benchmark (if applicable)
- state_pct_benchmark - State benchmark percentage (comparison)
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 SAT performance
sat_perf <- fetch_sat_performance(2024)

# Filter for SAT Math scores only
sat_math <- sat_perf %>%
  filter(test_type == "SAT", subject == "Math") %>%
  select(school_name, school_avg, state_avg)

# Get only SAT data
sat_only <- fetch_sat_performance(2024, test_type = "SAT")

## End(Not run)
```

`fetch_sped` *Fetch Special Education Classification Data*

Description

Fetches special education classification rate data from NJ DOE. As of 2024, only current data is available. Historical data (2003-2019) is no longer accessible via URL and requires an OPRA request.

Usage

```
fetch_sped(end_year)
```

Arguments

`end_year`	ending school year (e.g., 2024 for 2023-2024 school year). Valid years: 2024+

Value

cleaned sped dataframe with columns: end_year, county_id, county_name, district_id, district_name, gened_num, sped_num, sped_rate

---

`fetch_spr_data` *Fetch SPR Data*

Description

Downloads and extracts data from NJ School Performance Reports database. The SPR database contains 63+ sheets covering various school performance metrics.

Usage

```
fetch_spr_data(sheet_name, end_year, level = "school")
```

Arguments

sheet_name	Exact sheet name from SPR database (case-sensitive). You must know the exact sheet name. See vignette("spr-dictionary") for available sheets.
end_year	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
level	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with standardized columns including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- [Additional columns from requested sheet]
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get chronic absenteeism data
ca <- fetch_spr_data("ChronicAbsenteeism", 2024)

# Get district-level graduation data
grad <- fetch_spr_data("6YrGraduationCohortProfile", 2024, level = "district")

# Get teacher experience data
teachers <- fetch_spr_data("TeachersExperience", 2023)

## End(Not run)
```

---

fetch_staff_demographics

Fetch Staff Demographics Data

Description

Downloads teacher/administrator demographic data from SPR database.

Usage

```
fetch_staff_demographics(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024)
level	One of "school" or "district"

Value

Data frame with staff race/gender breakdowns

Examples

```
## Not run:
staff <- fetch_staff_demographics(2024)

## End(Not run)
```

`fetch_staff_ratios` *Fetch Student-Staff Ratio Data*

Description

Downloads student-to-staff ratio data from SPR database.

Usage

```
fetch_staff_ratios(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024)
`level`	One of "school" or "district"

Value

Data frame with staff ratios

Examples

```
## Not run:
ratios <- fetch_staff_ratios(2024)

## End(Not run)
```

`fetch_teacher_experience` *Fetch Teacher Experience Data*

Description

Downloads teacher experience data from SPR database.

Usage

```
fetch_teacher_experience(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024)
`level`	One of "school" or "district"

Value

Data frame with teacher experience breakdown

Examples

```
## Not run:
teachers <- fetch_teacher_experience(2024)

## End(Not run)
```

`fetch_tges` *Fetch Cleaned Taxpayer's Guide to Educational Spending*

Description

Fetch Cleaned Taxpayer's Guide to Educational Spending

Usage

```
fetch_tges(end_year)
```

Arguments

`end_year`	a school year. end_year is the end of the academic year - eg 2016-17 school year is end_year 2017. valid values are 1999-2017

Value

list of data frames

`fetch_violence_vandalism_hib` *Fetch Violence/Vandalism/HIB Data*

Description

Downloads incident data from SPR database.

Usage

```
fetch_violence_vandalism_hib(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024)
`level`	One of "school" or "district"

Value

Data frame with incident counts

Examples

```
## Not run:
incidents <- fetch_violence_vandalism_hib(2024)

## End(Not run)
```

`fetch_work_based_learning` *Fetch Work-Based Learning Data*

Description

Downloads and extracts work-based learning participation from the SPR database. Organized by career cluster.

Usage

```
fetch_work_based_learning(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
`level`	One of "school" or "district". "school" returns school-level data, "district" returns district and state-level data.

Value

Data frame with work-based learning participation including:

- end_year, county_id, county_name, district_id, district_name
- school_id, school_name (for school-level data)
- career_cluster - Career cluster area
- students_participating - Number of students in work-based learning
- pct_participating - Percentage participating in this career cluster
- Aggregation flags (is_state, is_county, is_district, is_school, is_charter)

Examples

```
## Not run:
# Get 2024 work-based learning data
wbl <- fetch_work_based_learning(2024)

# Schools with highest work-based learning participation
wbl %>%
  group_by(school_name) %>%
  summarize(avg_participation = mean(pct_participating, na.rm = TRUE)) %>%
  dplyr::arrange(desc(avg_participation))

## End(Not run)
```

format_duration *Format duration in human-readable form*

Description

Format duration in human-readable form

Usage

```
format_duration(seconds)
```

Arguments

seconds	Number of seconds

Value

Character string like "2m 30s" or "1h 5m"

format_valid_values *Format valid values for error messages*

Description

Format valid values for error messages

Usage

```
format_valid_values(values, max_show = 10)
```

Arguments

values	Vector of valid values
max_show	Maximum number of values to show

Value

Character string with formatted values

friendly_district_names *Friendly District Names*

Description

Friendly District Names

Usage

```
friendly_district_names(df)
```

Arguments

df	data.frame that contains district_id and district_name

Value

character vector of names, one per district id

friendly_school_names *Friendly School Names*

Description

Friendly School Names

Usage

```
friendly_school_names(df)
```

Arguments

df	data.frame that contains school_id and school_name

Value

character vector of names, one per school id

gap_percentile_rank *Rank entities by achievement gap within peer group*

Description

Calculates percentile rank of achievement gaps. By default, smaller gaps receive higher percentile ranks (better equity = higher rank). This enables questions like "Which DFG A districts have the smallest Black-White achievement gaps?"

Usage

```
gap_percentile_rank(
  df,
  gap_col,
  peer_type = "statewide",
  year_col = "end_year",
  smaller_is_better = TRUE
)
```

Arguments

df	Output of calculate_subgroup_gap() or any dataframe with a gap column
gap_col	Character. The column containing gap values. Default "metric_gap".
peer_type	Character. Peer group type. See define_peer_group().
year_col	Character. Year column name. Default "end_year".
smaller_is_better	Logical. If TRUE (default), smaller gaps get higher percentile ranks. Set to FALSE if larger gaps are preferred.

Value

df with added gap percentile columns

Examples

```
## Not run:
grate %>%
  calculate_subgroup_gap("grad_rate", "white", "black") %>%
  gap_percentile_rank(gap_col = "grad_rate_gap", peer_type = "dfg")

## End(Not run)
```

gap_trajectory *Track achievement gap trends over time*

Description

Combines gap calculation with trend tracking to show how achievement gaps have changed over time for specific entities. Answers questions like "How has Newark's Black-White gap changed from 2015 to 2023?"

Usage

```
gap_trajectory(
  df,
  metric_col,
  subgroup_a,
  subgroup_b,
  year_col = "end_year",
  entity_cols = "district_id"
)
```

Arguments

df	Dataframe with subgroups and metrics over multiple years
metric_col	Character. The metric to track.
subgroup_a	Character. Reference subgroup.
subgroup_b	Character. Comparison subgroup.
year_col	Character. Year column. Default "end_year".
entity_cols	Character vector. Columns identifying entity to track.

Value

df with gap values and trend columns:

- {metric}_gap_yoy_change: Year-over-year change in gap
- {metric}_gap_cumulative_change: Change from baseline year
- {metric}_gap_baseline: Gap value in first year

Examples

```
## Not run:
# Track Newark's Black-White grad rate gap over time
grate %>%
  filter(district_id == "3570") %>%
  gap_trajectory(
    metric_col = "grad_rate",
    subgroup_a = "white",
    subgroup_b = "black"
  )

## End(Not run)
```

gcount_aggregate_calcs

Aggregate multiple grad count rows and produce summary statistics

Description

Aggregate multiple grad count rows and produce summary statistics

Usage

```
gcount_aggregate_calcs(df)
```

Arguments

df	grouped df of gcount data

Value

df with aggregate stats for whatever grouping was provided

gcount_column_order

Grad Count column order

Description

Puts graduation count data frame columns in standard order.

Usage

```
gcount_column_order(df)
```

Arguments

df	Processed grad count df

Value

Data frame with columns in correct order

GCOUNT_VALID_YEARS *Graduation count valid years*

Description

Graduation count valid years

Usage

```
GCOUNT_VALID_YEARS
```

Format

An object of class `integer` of length 27.

geocoded *Geocoded School Addresses*

Description

Cached geocoding results for New Jersey school addresses. Contains latitude, longitude, and formatted addresses for schools, primarily Newark addresses.

Usage

```
geocoded
```

Format

A data frame with 2,549 rows and 8 columns:

lat Latitude coordinate

lng Longitude coordinate

formatted_address Standardized address string

status Geocoding status (e.g., "tidygeocoder cascade")

location_type Location type from geocoding service

error_message Error message if geocoding failed

locations Original location string used for geocoding

input_url Input URL for geocoding request

Source

Geocoded using tidygeocoder package

get_access_url *Get the URL for ACCESS for ELLs data file*

Description

Builds the URL for a given year's ACCESS data file. URL structure changed between years.

Usage

```
get_access_url(end_year)
```

Arguments

end_year	A school year (2022-2024)

Value

URL string

get_and_process_msgp *Get and Process mSGP data*

Description

Get and Process mSGP data

Usage

```
get_and_process_msgp(end_year)
```

Arguments

end_year	ending school year. valid values are currently 2012-2018.

Value

df of msgp data, schoolwide (and district-wide and by grade level, if reported)

get_chronic_absenteeism_url

Get the URL for ESSA Accountability Workbook containing chronic absenteeism data

Description

Builds the URL for a given year's accountability workbook file.

Usage

```
get_chronic_absenteeism_url(end_year)
```

Arguments

end_year	A school year (2017-2024, excluding 2020-2021)

Value

URL string

get_dfg_a_districts

Get DFG A districts (highest-need peer group)

Description

Convenience function to get all DFG A districts. DFG A represents the 37 highest-need communities in New Jersey, including Newark, Camden, Trenton, Paterson, etc.

This is the peer group used in MarGrady Research's Newark analysis: "Newark is in DFG A, which comprises cities or towns with the highest-need populations in New Jersey."

Usage

```
get_dfg_a_districts()
```

Value

Character vector of district_ids in DFG A

References

MarGrady Research. "Moving Up: Progress in Newark's Schools from 2010 to 2017" https://margrady.com/movingup/

get_dfg_districts *Get districts in a specific District Factor Group*

Description

Returns the district IDs for all districts in a specific District Factor Group. DFG A represents the highest-need communities in New Jersey and is commonly used as a peer group for urban districts.

This function fetches the DFG data from NJ DOE and filters to the requested group.

Usage

```
get_dfg_districts(dfg_code, revision = 2000)
```

Arguments

dfg_code	Character. The DFG code to filter to (e.g., "A", "B", "CD").
revision	Numeric. Which DFG revision to use (2000 or 1990). Default 2000.

Value

Character vector of district_ids in the specified DFG

Examples

```
## Not run:
# Get all DFG A districts (highest need)
dfg_a <- get_dfg_districts("A")

# Use as peer group for percentile ranking
grate %>%
  define_peer_group("custom", custom_ids = dfg_a) %>%
  add_percentile_rank("grad_rate")

## End(Not run)
```

get_district_directory *Get NJ District Directory Data*

Description

Get NJ District Directory Data

Usage

```
get_district_directory()
```

Value

dataframe of districts and associated metadata

get_district_directory_url *Get district directory URL*

Description

Get district directory URL

Usage

```
get_district_directory_url()
```

Value

Character string URL

get_enr_column_order *Standard enrollment column order*

Description

Standard enrollment column order

Usage

```
get_enr_column_order()
```

Value

Character vector of column names in standard order

get_enr_types *Enrollment column type specifications*

Description

Enrollment column type specifications

Usage

```
get_enr_types()
```

Value

Named list of column types

get_enr_url *Enrollment file URL configuration*

Description

Returns the URL for enrollment data for a given year.

Usage

```
get_enr_url(end_year)
```

Arguments

end_year	The school year (end year)

Value

Character string URL

get_essa_file *Get an ESSA comprehensive or targeted accountability file*

Description

Get an ESSA comprehensive or targeted accountability file

Usage

```
get_essa_file(end_year, file_type = "comprehensive")
```

Arguments

end_year	a school year. end_year is the end of the academic year - eg 2016-17 school year is end_year 2017. valid values are 2017
file_type	'comprehensive' or 'targeted'?

Value

list of data frames

get_grad_count *Get NJ graduation count data*

Description

Get NJ graduation count data

Usage

```
get_grad_count(end_year)
```

Arguments

end_year	End of the academic year - eg 2006-07 is 2007. Valid values are 2012-2024.

Value

dataframe with the number of graduates per school and district

get_grad_rate *Get NJ graduation rate data*

Description

Get NJ graduation rate data

Usage

```
get_grad_rate(end_year, methodology)
```

Arguments

end_year	End of the academic year - 2011-2012 is 2012. Valid values are 2011-2024.
methodology	Character string specifying calculation methodology. One of "4 year" or "5 year".

Value

dataframe with the number of graduates per school and district

get_grad_url *Get graduation data URL*

Description

Get graduation data URL

Usage

```
get_grad_url(end_year, methodology = "4 year")
```

Arguments

end_year	Graduation cohort year
methodology	Either "4 year" or "5 year"

Value

Character string URL

get_mapped_sheet_name *Get Mapped Sheet Name*

Description

Returns the correct sheet name for a given year, handling historical name variations. If no mapping exists, returns the input name.

Usage

```
get_mapped_sheet_name(canonical_name, end_year)
```

Arguments

canonical_name	Canonical sheet name (e.g., "chronic_absenteeism_by_grade")
end_year	School year end

Value

Actual sheet name to use with fetch_spr_data()

get_merged_rc_database

Combines school and district Performance Reports for 2017-on, when two files were released.

Description

Combines school and district Performance Reports for 2017-on, when two files were released.

Usage

```
get_merged_rc_database(end_year)
```

Arguments

end_year	end of the academic year. Valid values are 2017, 2018, 2019

Value

list of dataframes

get_one_rc_database *Get Raw Report Card Database*

Description

Get Raw Report Card Database

Usage

```
get_one_rc_database(end_year)
```

Arguments

end_year	a school year. end_year is the end of the academic year - eg 2014-15 school year is end_year '2015'. valid values are 2003 to 2019

Value

list of data frames

get_parcc_url *Get PARCC/NJSLA assessment URL*

Description

Get PARCC/NJSLA assessment URL

Usage

```
get_parcc_url(end_year, grade, subj)
```

Arguments

end_year	Assessment year
grade	Grade level or course code
subj	Subject ("ela" or "math")

Value

Character string URL

get_percentile_cols *Get percentile cols*

Description

internal/helper function to facilitate the extraction of relevant ‘assessment_peer_percentile‘ columns when calculating state/dfg-wide percentiles

Usage

```
get_percentile_cols(df)
```

Arguments

df	data.frame, output of ‘assessment_peer_percentile‘

Value

slim df with limited columns

get_raw_access *Reads the raw ACCESS for ELLs Excel file from the state website*

Description

Downloads the ACCESS file and reads a specific grade sheet.

Usage

```
get_raw_access(end_year, grade = "all")
```

Arguments

end_year	A school year (2022-2024)
grade	Grade level: "K" or 0 for Kindergarten, or 1-12 for other grades. Use "all" to get all grades combined.

Value

ACCESS dataframe for the specified grade

get_raw_enr *Read a zipped Excel fall enrollment file from the NJ state website*

Description

Read a zipped Excel fall enrollment file from the NJ state website

Usage

```
get_raw_enr(end_year)
```

Arguments

end_year	A school year. Year is the end of the academic year - eg 2006-07 school year is year '2007'. Valid values are 2000-2025.

Value

Data frame with raw enrollment data

get_raw_gepa *read a fixed width, raw GEPA data file from the NJ state website*

Description

get_raw_gepa builds a url and uses readr's read_fwf to get the fixed width text file into a R data frame

Usage

```
get_raw_gepa(end_year, layout = layout_gepa)
```

Arguments

end_year	a school year. end_year is the end of the academic year - eg 2006-07 school year is end_year '2007'. valid values are 2004-2007.
layout	what layout dataframe to use. default is layout_gepa.

get_raw_grad_file *Get a raw graduation file from the NJ website*

Description

Get a raw graduation file from the NJ website

Usage

```
get_raw_grad_file(end_year, methodology = "4 year")
```

Arguments

end_year	End of the academic year - eg 2006-07 is 2007. Valid values are 1998-2024.
methodology	One of c('4 year', '5 year')

Value

data.frame with raw data from state file

get_raw_hspa *read a fixed width, raw HSPA data file from the NJ state website*

Description

get_raw_hspa builds a url and uses readr's read_fwf to get the fixed width text file into a R data frame

Usage

```
get_raw_hspa(end_year, layout = layout_hspa[c(1:558), ])
```

Arguments

end_year	a school end_year. end_year is the end of the academic year - eg 2013-14 school year is end_year '2014'. valid values are 2004-2014.
layout	what layout dataframe to use. default is layout_hspa.

get_raw_njask *read a fixed width, raw NJASK data file from the NJ state website*

Description

get_raw_njask builds a url and uses readr's read_fwf to get the fixed width text file into a R data frame

Usage

```
get_raw_njask(end_year, grade, layout = layout_njask)
```

Arguments

end_year	a school year. end_year is the end of the academic year - eg 2013-14 school year is end_year '2014'. valid values are 2004-2014.
grade	a grade level. valid values are 3,4,5,6,7,8
layout	what layout dataframe to use. default is layout_njask.

get_raw_njgpa *Reads the raw NJGPA Excel files from the state website*

Description

NJGPA (New Jersey Graduation Proficiency Assessment) is the graduation requirement assessment introduced in 2022.

Usage

```
get_raw_njgpa(end_year, subj)
```

Arguments

end_year	A school year. Valid values are 2022-2024.
subj	NJGPA subject. c('ela' or 'math')

Value

NJGPA dataframe

get_raw_parcc *Reads the raw PARCC Excel files from the state website*

Description

Builds a URL and reads the xlsx file into a dataframe.

Usage

```
get_raw_parcc(end_year, grade_or_subj, subj)
```

Arguments

end_year	A school year. end_year is the end of the academic year - eg 2014-15 school year is end_year 2015. Valid values are 2015-2018.
grade_or_subj	Grade level (eg 8) OR math subject code (eg ALG1, GEO, ALG2)
subj	PARCC subject. c('ela' or 'math')

Value

PARCC dataframe

get_raw_sla *Reads the raw NJSLA Excel files from the state website*

Description

Builds a URL and reads the xlsx file into a dataframe.

Usage

```
get_raw_sla(end_year, grade_or_subj, subj)
```

Arguments

end_year	A school year. end_year is the end of the academic year - eg 2014-15 school year is end_year 2015. Valid values are 2015-2018.
grade_or_subj	Grade level (eg 8) OR math subject code (eg ALG1, GEO, ALG2)
subj	PARCC subject. c('ela' or 'math')

Value

NJSLA dataframe

get_raw_sped *read Special ed excel files from the NJ state website*

Description

read Special ed excel files from the NJ state website

Usage

```
get_raw_sped(end_year)
```

Arguments

end_year	A school year. Year is the end of the academic year - eg 2006-07 school year is year '2007'. Valid values are 2000-2025.

Value

a dataframe with special ed counts, etc.

get_raw_tges *Get Raw Taxpayer's Guide to Educational Spending*

Description

Get Raw Taxpayer's Guide to Educational Spending

Usage

```
get_raw_tges(end_year)
```

Arguments

end_year	a school year. end_year is the end of the academic year - eg 2016-17 school year is end_year 2017. valid values are 1999-2017

Value

list of data frames

get_rc_databases *Get multiple RC databases*

Description

Get multiple RC databases

Usage

```
get_rc_databases(end_year_vector = c(2003:2018))
```

Arguments

end_year_vector	vector of years. Current valid values are 2003 to 2018.

Value

a list of dataframes

get_reportcard_special_pop *Get Special Population Enrollment from Report Card Files*

Description

Get Special Population Enrollment from Report Card Files

Usage

```
get_reportcard_special_pop(end_year)
```

Arguments

end_year	ending academic year. valid values are 2017, 2018, 2019

Value

data.frame with special population enrollment percentages

get_school_directory *Get NJ School Directory Data*

Description

Get NJ School Directory Data

Usage

```
get_school_directory()
```

Value

dataframe of schools and associated metadata

get_school_directory_url *Get school directory URL*

Description

Get school directory URL

Usage

```
get_school_directory_url()
```

Value

Character string URL

get_spr_6yr_grad_url *Get SPR database URL for 6-year graduation rates*

Description

Builds the URL for the School Performance Reports database containing 6-year graduation cohort profile data.

Usage

```
get_spr_6yr_grad_url(end_year, level = "school")
```

Arguments

end_year	A school year (2021-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
level	One of "school" or "district". Determines which database file to download.

Value

URL string

get_spr_url *Get SPR Database URL*

Description

Builds the URL for the School Performance Reports database containing multiple sheets of school performance data.

Usage

```
get_spr_url(end_year, level = "school")
```

Arguments

end_year	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
level	One of "school" or "district". Determines which database file to download.

Value

URL string

get_standalone_rc_database *Get Standalone Raw Report Card Database*

Description

Get Standalone Raw Report Card Database

Usage

```
get_standalone_rc_database(end_year)
```

Arguments

end_year	a school year. end_year is the end of the academic year - eg 2014-15 school year is end_year '2015'. valid values are 2003 to 2016

Value

list of data frames

get_valid_grades *Get valid grades for an assessment type and year*

Description

Returns the valid grade levels for a given assessment type and year.

Usage

```
get_valid_grades(assessment_type, end_year)
```

Arguments

assessment_type	The type of assessment
end_year	The year of the assessment

Value

Vector of valid grades (may include character values like "ALG1")

Examples

```
get_valid_grades("njask", 2010)
get_valid_grades("parcc", 2023)
```

get_valid_sped_years *Valid years for SPED data*

Description

Valid years for SPED data

Usage

```
get_valid_sped_years()
```

Value

vector of valid end years for SPED data

get_valid_years *Get valid year range for a data type*

Description

Get valid year range for a data type

Get valid years for a data type

Usage

```
get_valid_years(data_type)
```

```
get_valid_years(data_type)
```

Arguments

data_type	Character string identifying the data type

Value

Integer vector of valid years

Integer vector of valid years

Examples

```
get_valid_years("enrollment")
get_valid_years("parcc")
```

grad_file_group_cleanup *Grad file group cleanup*

Description

Standardizes subgroup names across years.

Usage

```
grad_file_group_cleanup(group)
```

Arguments

group	Column of group (subgroup) data from NJ grad file

Value

Column with cleaned up subgroup names

grad_url_config *Graduation data URL configuration table*

Description

This table maps years and methodologies to their corresponding URLs. When NJ DOE changes URL patterns, update this table.

Usage

```
grad_url_config
```

Format

A data frame with columns:

end_year The graduation cohort year

methodology Either "4 year" or "5 year"

file_type File format ("xlsx", "xls", "csv")

skip_rows Number of header rows to skip when reading

GRATE_4YR_VALID_YEARS *4-year graduation rate valid years*

Description

4-year graduation rate valid years

Usage

```
GRATE_4YR_VALID_YEARS
```

Format

An object of class `integer` of length 14.

GRATE_5YR_VALID_YEARS *5-year graduation rate valid years*

Description

5-year graduation rate valid years

Usage

```
GRATE_5YR_VALID_YEARS
```

Format

An object of class `integer` of length 8.

grate_aggregate_calcs *Aggregate multiple grad rate rows and produce summary statistics*

Description

Aggregate multiple grad rate rows and produce summary statistics

Usage

```
grate_aggregate_calcs(df)
```

Arguments

df	grouped df of grate data

Value

df with aggregate stats for whatever grouping was provided

grate_column_order *Grad Rate column order*

Description

Puts graduation rate data frame columns in standard order.

Usage

```
grate_column_order(df)
```

Arguments

df	Processed grad rate df

Value

Data frame with columns in correct order

grate_percentile_rank *Graduation rate percentile rank*

Description

Convenience wrapper for calculating percentile rank of graduation rates within a peer group.

Usage

```
grate_percentile_rank(
  df,
  peer_type = "statewide",
  custom_ids = NULL,
  by_subgroup = TRUE,
  by_methodology = TRUE
)
```

Arguments

df	Output of `fetch_grad_rate()` or similar graduation data
peer_type	Character. Peer group type. See `define_peer_group()`.
custom_ids	Character vector. Custom peer group district IDs.
by_subgroup	Logical. Calculate separate percentiles by subgroup? Default TRUE.
by_methodology	Logical. Calculate separate percentiles by methodology (4-year, 5-year)? Default TRUE.

Value

df with grad_rate_rank, grad_rate_n, grad_rate_percentile columns

grate_validation_summary *Get graduation rate validation summary*

Description

Generates a summary report of graduation rate data quality.

Usage

```
grate_validation_summary(df)
```

Arguments

df	Graduation rate data frame (output of validate_grate_aggregation)

Value

Data frame summarizing validation results by year

HIGH_SCHOOL_GRADES *High school grades (9-12)*

Description

High school grades (9-12)

Usage

```
HIGH_SCHOOL_GRADES
```

Format

An object of class `character` of length 4.

id_charter_hosts *Identify charter host districts*

Description

Identify charter host districts

Usage

```
id_charter_hosts(df)
```

Arguments

df	dataframe of NJ school data containing ‘district_id‘ column

Value

df with host district id and name for every matching record

id_enr_aggs *Identify enrollment aggregation levels*

Description

Adds boolean flags to identify state, county, district, and school level records.

Usage

```
id_enr_aggs(df)
```

Arguments

df	Enrollment dataframe, output of tidy_enr

Value

data.frame with boolean aggregation flags

id_grad_aggs *Identify graduation aggregation levels*

Description

Adds boolean flags to identify state, county, district, and school level records.

Usage

```
id_grad_aggs(df)
```

Arguments

df	Graduation dataframe, output of tidy_grad_rate or tidy_grad_count

Value

data.frame with boolean aggregation flags

id_rc_aggs *Identify reportcard aggregation levels*

Description

Identify reportcard aggregation levels

Usage

```
id_rc_aggs(df)
```

Arguments

df	enrollment dataframe, output of extract_rc_*

Value

data.frame with boolean aggregation flags

id_selected_districtids *District Names to IDs*

Description

District Names to IDs

Usage

```
id_selected_districtids(district_names, lookup_df)
```

Arguments

district_names	vector of names
lookup_df	dataframe with district_id and district_name

Value

list of districtids matching the names

is_charter_district *Check if a district is a charter*

Description

Check if a district is a charter

Usage

```
is_charter_district(county_id)
```

Arguments

county_id	County ID

Value

Logical

is_district_total *Check if a school is a district total*

Description

Check if a school is a district total

Usage

```
is_district_total(school_id)
```

Arguments

school_id	School ID to check

Value

Logical

is_state_aggregate *Check if a record is a state aggregate*

Description

Check if a record is a state aggregate

Usage

```
is_state_aggregate(county_id, district_id)
```

Arguments

county_id	County ID
district_id	District ID

Value

Logical

is_valid_year *Check if year is valid for a data type*

Description

Check if year is valid for a data type

Usage

```
is_valid_year(end_year, data_type)
```

Arguments

end_year	The school year (end year)
data_type	One of "enrollment", "parcc", "grate_4yr", "grate_5yr", "gcount", "legacy_assess"

Value

Logical indicating if year is valid

K12_GRADES *K-12 grades (excluding pre-K)*

Description

K-12 grades (excluding pre-K)

Usage

```
K12_GRADES
```

Format

An object of class `character` of length 13.

kill_padformulas *Kill Excel Formula Padding For Numeric Strings*

Description

Removes Excel formula padding (="01") from strings, leaving just the numeric value.

Usage

```
kill_padformulas(x)
```

Arguments

x	a vector with strings entered as formulas - eg ="01"

Value

a vector with normalized strings

KINDER_PROGRAM_CODES *Kindergarten program codes*

Description

Kindergarten program codes

Usage

```
KINDER_PROGRAM_CODES
```

Format

An object of class `character` of length 4.

layout_gepa *GEPA Fixed-Width File Layout*

Description

Column layout specification for reading Grade Eight Proficiency Assessment (GEPA) fixed-width format data files.

A processed R version of the 'File Layout' for the 2004-2007 GEPA. Original file is in .xls format on NJDOE website

Usage

```
layout_gepa

layout_gepa
```

Format

A data frame with 486 rows and 10 columns:

field_start_position Starting position of field in fixed-width file

field_end_position Ending position of field in fixed-width file

field_length Length of field in characters

data_type Data type of field (e.g., "Text", "Numeric")

description Description of field

comments Additional comments about field

valid_values Valid values or ranges for field

spanner1 First-level column grouping label

spanner2 Second-level column grouping label

final_name Final column name to use in parsed data

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education GEPA file specifications

NJDOE website - eg the 'File Layout' link on http://www.nj.gov/education/schools/achievement/2005/gepa/

`layout_gepa05` *GEPA 2005 Fixed-Width File Layout*

Description

Column layout specification for reading Grade Eight Proficiency Assessment (GEPA) 2005 fixed-width format data files.

Usage

```
layout_gepa05
```

Format

A data frame with 383 rows and 10 columns:

field_start_position Starting position of field in fixed-width file

field_end_position Ending position of field in fixed-width file

field_length Length of field in characters

data_type Data type of field (e.g., "Text", "Numeric")

description Description of field

comments Additional comments about field

valid_values Valid values or ranges for field

spanner1 First-level column grouping label

spanner2 Second-level column grouping label

final_name Final column name to use in parsed data

Source

NJ Department of Education GEPA 2005 file specifications

---

layout_gepa06 *GEPA 2006 Fixed-Width File Layout*

Description

Column layout specification for reading Grade Eight Proficiency Assessment (GEPA) 2006 fixed-width format data files.

Usage

```
layout_gepa06
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

Source

NJ Department of Education GEPA 2006 file specifications

`layout_hspa` *HSPA Fixed-Width File Layout*

Description

Column layout specification for reading High School Proficiency Assessment (HSPA) fixed-width format data files.

A processed R version of the 'File Layout' for the 2011-present HSPA Original file is in .xls format on NJDOE website

Usage

```
layout_hspa

layout_hspa
```

Format

A data frame with 559 rows and 10 columns:

field_start_position Starting position of field in fixed-width file

field_end_position Ending position of field in fixed-width file

field_length Length of field in characters

data_type Data type of field (e.g., "Text", "Numeric")

description Description of field

comments Additional comments about field

valid_values Valid values or ranges for field

spanner1 First-level column grouping label

spanner2 Second-level column grouping label

final_name Final column name to use in parsed data

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education HSPA file specifications

NJDOE website - eg the 'File Layout' link on http://www.nj.gov/education/schools/achievement/14/hspa/

layout_hspa04 *HSPA 2004 Fixed-Width File Layout*

Description

Column layout specification for reading High School Proficiency Assessment (HSPA) 2004 fixed-width format data files.

Usage

```
layout_hspa04
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

Source

NJ Department of Education HSPA 2004 file specifications

layout_hspa05 *HSPA 2005 Fixed-Width File Layout*

Description

Column layout specification for reading High School Proficiency Assessment (HSPA) 2005 fixed-width format data files.

Usage

```
layout_hspa05
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

Source

NJ Department of Education HSPA 2005 file specifications

layout_hspa06 *HSPA 2006 Fixed-Width File Layout*

Description

Column layout specification for reading High School Proficiency Assessment (HSPA) 2006 fixed-width format data files.

Usage

```
layout_hspa06
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

Source

NJ Department of Education HSPA 2006 file specifications

layout_hspa10 *HSPA 2010 Fixed-Width File Layout*

Description

Column layout specification for reading High School Proficiency Assessment (HSPA) 2010 fixed-width format data files.

A processed R version of the 'File Layout' for the 2004-2010 HSPA. Original file is in .xls format on NJDOE website

Usage

```
layout_hspa10

layout_hspa10
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education HSPA 2010 file specifications

NJDOE website - eg the 'File Layout' link on http://www.nj.gov/education/schools/achievement/2005/hspa/

`layout_njask` *NJASK Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) fixed-width format data files.

A processed R version of the 'File Layout' for the most current (2008-present) NJASK. Original file is in .xls format on NJDOE website

Usage

```
layout_njask

layout_njask
```

Format

A data frame with 551 rows and 10 columns:

field_start_position Starting position of field in fixed-width file

field_end_position Ending position of field in fixed-width file

field_length Length of field in characters

data_type Data type of field (e.g., "Text", "Numeric")

description Description of field

comments Additional comments about field

valid_values Valid values or ranges for field

spanner1 First-level column grouping label

spanner2 Second-level column grouping label

final_name Final column name to use in parsed data

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education NJASK file specifications

NJDOE website - eg the 'File Layout' link on http://www.state.nj.us/education/schools/achievement/14/njask5/

`layout_njask04` *NJASK 2004 Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) 2004 fixed-width format data files.

A processed R version of the 'File Layout' for the 2004 NJASK (only offered for grades 3 and 4). Original file is in .xls format on NJDOE website

Usage

```
layout_njask04

layout_njask04
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education NJASK 2004 file specifications

NJDOE website - eg the 'File Layout' link on http://www.state.nj.us/education/schools/achievement/2005/njask3/

`layout_njask05` *NJASK 2005 Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) 2005 fixed-width format data files.

A processed R version of the 'File Layout' for the 2005 NJASK (only offered for grades 3 and 4). Original file is in .xls format on NJDOE website

Usage

```
layout_njask05

layout_njask05
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education NJASK 2005 file specifications

NJDOE website - eg the 'File Layout' link on http://www.state.nj.us/education/schools/achievement/2006/njask3/

`layout_njask06gr3` *NJASK 2006 Grade 3 Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) 2006 Grade 3 fixed-width format data files.

A processed R version of the 'File Layout' for the 2006 NJASK, elementary (gr 3 and 4) version. Original file is in .xls format on NJDOE website

Usage

```
layout_njask06gr3

layout_njask06gr3
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education NJASK 2006 Grade 3 file specifications

NJDOE website - eg the 'File Layout' link on http://www.state.nj.us/education/schools/achievement/2007/njask3/

`layout_njask06gr5` *NJASK 2006 Grade 5 Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) 2006 Grade 5 fixed-width format data files.

A processed R version of the 'File Layout' for the 2006 NJASK, middle school (gr 5, 6, 7) version. Original file is in .xls format on NJDOE website

Usage

```
layout_njask06gr5

layout_njask06gr5
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education NJASK 2006 Grade 5 file specifications

NJDOE website - eg the 'File Layout' link on http://www.state.nj.us/education/schools/achievement/2007/njask57/

`layout_njask07gr3` *NJASK 2007 Grade 3 Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) 2007 Grade 3 fixed-width format data files.

A processed R version of the 'File Layout' for the 2007 and 2008 NJASK, elementary (gr 3 and 4) version. Original file is in .xls format on NJDOE website

Usage

```
layout_njask07gr3

layout_njask07gr3
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education NJASK 2007 Grade 3 file specifications

NJDOE website - eg the 'File Layout' link on http://www.state.nj.us/education/schools/achievement/2008/njask3/

layout_njask07gr5 *NJASK 2007 Grade 5 Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) 2007 Grade 5 fixed-width format data files.

Usage

```
layout_njask07gr5
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

Source

NJ Department of Education NJASK 2007 Grade 5 file specifications

layout_njask09 *NJASK 2009 Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) 2009 fixed-width format data files.

A processed R version of the 'File Layout' for the 2009 NJASK Original file is in .xls format on NJDOE website

Usage

```
layout_njask09

layout_njask09
```

Format

A data frame with column layout specifications including field positions, lengths, data types, descriptions, and final column names.

field_start_position field_start_position

field_end_position field_end_position

field_length field_length

data_type data_type

description description

comments comments

valid_values valid_values

spanner1 spanner1

spanner2 spanner2

final_name final_name #'

Source

NJ Department of Education NJASK 2009 file specifications

NJDOE website - eg the 'File Layout' link on http://www.state.nj.us/education/schools/achievement/2009/njask3/

`layout_njask10` *NJASK 2010 Fixed-Width File Layout*

Description

Column layout specification for reading New Jersey Assessment of Skills and Knowledge (NJASK) 2010 fixed-width format data files.

Usage

```
layout_njask10
```

Format

A data frame with 524 rows and 10 columns:

field_start_position Starting position of field in fixed-width file

field_end_position Ending position of field in fixed-width file

field_length Length of field in characters

data_type Data type of field (e.g., "Text", "Numeric")

description Description of field

comments Additional comments about field

valid_values Valid values or ranges for field

spanner1 First-level column grouping label

spanner2 Second-level column grouping label

final_name Final column name to use in parsed data

Source

NJ Department of Education NJASK 2010 file specifications

LEGACY_ASSESS_VALID_YEARS *Legacy assessment (NJASK/HSPA/GEPA) valid years*

Description

Legacy assessment (NJASK/HSPA/GEPA) valid years

Usage

```
LEGACY_ASSESS_VALID_YEARS
```

Format

An object of class `integer` of length 11.

list_spr_sheets *List Available SPR Sheets*

Description

Returns a vector of all sheet names available in the SPR database for a given year and level. Useful for discovering what data is available.

Usage

```
list_spr_sheets(end_year, level = "school")
```

Arguments

`end_year`	A school year (2017-2024). Year is the end of the academic year - eg 2020-21 school year is end_year '2021'.
`level`	One of "school" or "district". Determines which database file to query.

Value

Character vector of sheet names

Examples

```
## Not run:
# List all school-level sheets for 2024
sheets <- list_spr_sheets(2024)

# List all district-level sheets
district_sheets <- list_spr_sheets(2024, level = "district")

# Search for specific types of sheets
attendance_sheets <- sheets[grepl("Absent|Attendance", sheets, ignore.case = TRUE)]

## End(Not run)
```

lookup_peer_percentile

Looks up caclulated percentile value by searching for closest scale score / proficient above match

Description

Given a peer percentile lookup table with calculated mean scale score and percent proficient distributions by year, grade, subgroup, take the given values (likely of assessment aggregates) and find the closest match to return percentiles

Usage

```
lookup_peer_percentile(assess_agg, assess_percentiles)
```

Arguments

assess_agg	an assessments aggregate such as the output of charter_sector_parcc_aggs
assess_percentiles	calculated assessments peer percentiles such as the output of statewide_peer_percentile

Value

data.frame with percent proficient and scale score percentile ranks

make_cache_key

Generate a cache key for a data request

Description

Generate a cache key for a data request

Usage

```
make_cache_key(fn_name, ...)
```

Arguments

fn_name	Name of the fetch function
...	Arguments passed to the fetch function

Value

Character string cache key

matric_aggregate_calcs

Aggregate multiple postsecondary matriculation rows and produce summary statistics

Description

Aggregate multiple postsecondary matriculation rows and produce summary statistics

Usage

```
matric_aggregate_calcs(df)
```

Arguments

df	grouped df of postsecondary matriculation data

Value

data_frame

matric_column_order

Matriculation column order

Description

Puts matriculation data frame columns in standard order.

Usage

```
matric_column_order(df)
```

Arguments

df	Processed matriculation df

Value

Data frame with columns in correct order

MIDDLE_GRADES *Middle school grades (6-8)*

Description

Middle school grades (6-8)

Usage

```
MIDDLE_GRADES
```

Format

An object of class `character` of length 3.

njdoe_base_urls *NJ DOE base URLs*

Description

NJ DOE base URLs

Usage

```
njdoe_base_urls
```

Format

An object of class `list` of length 4.

njsd_cache_clear *Clear the session cache*

Description

Removes all cached data from memory. Useful when you want to force fresh downloads or free up memory.

Usage

```
njsd_cache_clear(reset_stats = FALSE)
```

Arguments

`reset_stats`	Logical; also reset hit/miss statistics

Value

Number of items cleared (invisibly)

Examples

```
njsd_cache_clear()
```

`njsd_cache_enable` *Enable or disable the session cache*

Description

Controls whether downloaded data is cached in memory for the current session. Caching is enabled by default.

Usage

```
njsd_cache_enable(enable = TRUE)
```

Arguments

`enable`	Logical; TRUE to enable caching, FALSE to disable

Value

Previous cache state (invisibly)

Examples

```
njsd_cache_enable(FALSE)  # Disable caching
njsd_cache_enable(TRUE)   # Re-enable caching
```

`njsd_cache_enabled` *Check if caching is enabled*

Description

Check if caching is enabled

Usage

```
njsd_cache_enabled()
```

Value

Logical indicating whether caching is currently enabled

njsd_cache_info *Get cache statistics and information*

Description

Returns information about the current cache state including number of items, memory usage, and hit/miss statistics.

Usage

```
njsd_cache_info()
```

Value

A list with cache information

Examples

```
njsd_cache_info()
```

njsd_cache_list *List all cached items*

Description

List all cached items

Usage

```
njsd_cache_list()
```

Value

Character vector of cache keys

njsd_cache_remove *Remove a specific item from cache*

Description

Remove a specific item from cache

Usage

```
njsd_cache_remove(key)
```

Arguments

key	The cache key to remove

Value

TRUE if item was removed, FALSE if not found (invisibly)

---

njsd_progress_enable *Enable or disable progress indicators*

Description

Controls whether progress messages are shown during batch operations.

Usage

```
njsd_progress_enable(enable = TRUE)
```

Arguments

enable	Logical; TRUE to enable progress, FALSE to disable

Value

Previous state (invisibly)

Examples

```
njsd_progress_enable(FALSE)  # Quiet mode
njsd_progress_enable(TRUE)   # Show progress
```

---

nj_coltype_parser *nj_coltype_parser*

Description

turns layout datatypes into compact string required by read_fwf

Usage

```
nj_coltype_parser(datatypes)
```

Arguments

datatypes	vector of datatypes (from a layout df)

Value

a character string of the types, for read_fwf

nwk_address_addendum *Newark Address Addendum*

Description

Additional address data for Newark schools to supplement geocoding. Contains school addresses that were not included in the main school directory or needed corrections for proper geocoding.

Usage

```
nwk_address_addendum
```

Format

A data frame with 98 rows and 5 columns:

district_id District identifier (Newark district is 3570)

school_id School identifier code

school_name Name of the school

address Full street address with city, state

in_geocode Logical indicating if address is in geocode cache

Source

Manual address corrections for Newark schools

pad_cds *Pad CDS fields*

Description

Zero-pads county, district, and school codes to their standard lengths (2, 4, and 3 digits respectively).

Usage

```
pad_cds(df)
```

Arguments

df	containing county_code, district_code, school_code

Value

data frame with zero padded cds columns

pad_grade *Pad grade level*

Description

Ensures grade level is two characters with leading zero if needed.

Usage

```
pad_grade(x)
```

Arguments

x	a grade level argument, length 1

Value

a string, length 2, with appropriate padding for PARCC naming conventions

pad_leading *Pad leading digits*

Description

Ensures a numeric value has exactly the specified number of digits by adding leading zeros.

Usage

```
pad_leading(vector, digits)
```

Arguments

vector	character vector
digits	ensure exactly this many digits by leading zero-padding

Value

character vector

parcc_aggregate_calcs *Aggregate multiple PARCC rows and produce summary statistics*

Description

Aggregate multiple PARCC rows and produce summary statistics

Usage

```
parcc_aggregate_calcs(df)
```

Arguments

df	grouped df of PARCC data

Value

df with aggregate stats for whatever grouping was provided

parcc_column_order *PARCC column order*

Description

Puts PARCC dataframe columns in coherent order.

Usage

```
parcc_column_order(df)
```

Arguments

df	Tidied PARCC dataframe. Called as final step in fetch_parcc when tidy=TRUE.

Value

PARCC df with columns in coherent order

PARCC_LEVELS *PARCC/NJSLA performance levels*

Description

PARCC/NJSLA performance levels

Usage

```
PARCC_LEVELS
```

Format

An object of class `character` of length 5.

parcc_percentile_rank *Assessment proficiency percentile rank*

Description

Convenience wrapper for calculating percentile rank of assessment proficiency within a peer group. This is the generic version of the assessment-specific functions in peer_percentiles.R.

Usage

```
parcc_percentile_rank(
  df,
  peer_type = "statewide",
  custom_ids = NULL,
  metric = c("proficient_above", "scale_score_mean"),
  by_grade = TRUE,
  by_subject = TRUE,
  by_subgroup = TRUE
)
```

Arguments

df	Output of `fetch_parcc()` or similar assessment data
peer_type	Character. Peer group type. See `define_peer_group()`.
custom_ids	Character vector. Custom peer group district IDs.
metric	Character. Which metric to rank on. One of "proficient_above" or "scale_score_mean". Default "proficient_above".
by_grade	Logical. Calculate separate percentiles by grade? Default TRUE.
by_subject	Logical. Calculate separate percentiles by test/subject? Default TRUE.
by_subgroup	Logical. Calculate separate percentiles by subgroup? Default TRUE.

Value

df with percentile rank columns for the specified metric

parcc_perf_level_counts *PARCC counts by performance level*

Description

Calculates the count of students at each performance level based on percentages and total valid scores.

Usage

```
parcc_perf_level_counts(df)
```

Arguments

df	dataframe, output of fetch_parcc

Value

df with counts of students by performance level

PARCC_VALID_YEARS *PARCC/NJSLA assessment valid years (skip 2020 - no testing due to COVID)*

Description

PARCC/NJSLA assessment valid years (skip 2020 - no testing due to COVID)

Usage

```
PARCC_VALID_YEARS
```

Format

An object of class `integer` of length 9.

---

parse_parcc_subj *Parse PARCC subject*

Description

Converts subject name to standardized PARCC code.

Usage

```
parse_parcc_subj(x)
```

Arguments

x	a subject (function parameter subj)

Value

the subject coded as ELA, MAT, or SC.

parse_postsec_range *Parse postsecondary enrollment range string*

Description

Extracts lower and upper bounds from range strings like "58.3-60.1

Usage

```
parse_postsec_range(range_string)
```

Arguments

range_string	Character vector of range strings

Value

Data frame with lower_bound and upper_bound columns

peek *Peek at a data frame*

Description

Displays a random sample of rows from a data frame for quick inspection.

Usage

```
peek(df, nrows = 5)
```

Arguments

df	data.frame
nrows	how many rows to sample

Value

prints random sample of nrows of a dataframe

percentile_rank *Percentile Rank*

Description

Calculates the percentile rank of a target value within a distribution.

Usage

```
percentile_rank(x, xo)
```

Arguments

x	vector of values
xo	target value

Value

numeric percentile rank

percentile_rank_trend *Calculate percentile rank change over time*

Description

Given a dataframe with percentile ranks by year, calculates year-over-year and cumulative change. This enables the "39th to 78th percentile" style analysis from MarGrady Research.

Usage

```
percentile_rank_trend(
  df,
  percentile_col,
  year_col = "end_year",
  entity_cols = c("district_id")
)
```

Arguments

df	Dataframe with a percentile column and year column
percentile_col	Character. Name of the percentile column to track.
year_col	Character. Name of year column. Default "end_year".
entity_cols	Character vector. Columns identifying entities to track over time (e.g., c("district_id", "subgroup")).

Value

df with added columns:

- {percentile_col}_yoy_change: Year-over-year change
- {percentile_col}_cumulative_change: Change from first year
- {percentile_col}_baseline: Value in the first year

Examples

```
## Not run:
# Track Newark's percentile rank over time
grate_ranked %>%
  filter(district_id == "3570") %>%
  percentile_rank_trend(
    percentile_col = "grad_rate_percentile",
    entity_cols = c("district_id", "subgroup")
  )

## End(Not run)
```

PREK_PROGRAM_CODES *Pre-K program codes*

Description

Pre-K program codes

Usage

```
PREK_PROGRAM_CODES
```

Format

An object of class character of length 4.

---

print.njsd_cache_info *Print cache information*

Description

Print cache information

Usage

```
## S3 method for class 'njsd_cache_info'
print(x, ...)
```

Arguments

x	Object from njsd_cache_info()
...	Additional arguments (unused)

process_access *Process raw ACCESS for ELLs data*

Description

Cleans and standardizes ACCESS data.

Usage

```
process_access(access_file, end_year)
```

Arguments

access_file	Output of get_raw_access
end_year	A school year (2022-2024)

Value

Processed ACCESS dataframe

process_chronic_absenteeism_cols *Process chronic absenteeism columns*

Description

Standardizes column names for chronic absenteeism data by detecting and renaming the chronic absenteeism rate column.

Usage

```
process_chronic_absenteeism_cols(df)
```

Arguments

df	Raw data frame from SPR database

Value

Data frame with standardized columns

process_days_absent_cols *Process days absent columns*

Description

Standardizes column names for days absent data.

Usage

```
process_days_absent_cols(df)
```

Arguments

df	Raw data frame from SPR database

Value

Data frame with standardized columns

process_enr *Process a NJ enrollment file*

Description

Does cleanup of dataframes returned by ‘get_raw_enr‘.

Usage

```
process_enr(df)
```

Arguments

df	An enrollment data frame (eg output of ‘get_raw_enr‘)

Value

Cleaned and processed enrollment data frame

process_enr_program *Join program code to program name*

Description

Decode the program name using the prog_codes lookup table.

Usage

```
process_enr_program(df)
```

Arguments

df	Cleaned enrollment file

Value

Data frame with program_name added

process_grad_count *Process Grad Count Data*

Description

Creates composite subgroups like black (black_m + black_f).

Usage

```
process_grad_count(df, end_year)
```

Arguments

df	Output of get_grad_count
end_year	End of the academic year

Value

Data frame with composite subgroups

process_grad_rate *Process Grad Rate*

Description

Custom processing for grad rate data beyond generic process_grate.

Usage

```
process_grad_rate(df, end_year, methodology)
```

Arguments

df	Output of get_grad_rate
end_year	Ending academic year
methodology	One of c('4 year', '5 year')

Value

Data frame with normalized grad rate variables

process_grate *Process graduation rate data*

Description

Does cleanup of the grad rate ('grate') file.

Usage

```
process_grate(df, end_year)
```

Arguments

df	The output of get_raw_grad_file
end_year	A school year. Year is the end of the academic year - eg 2006-07 school year is year '2007'. Valid values are 1998-2024.

Value

Cleaned graduation data frame

process_nj_assess *Process a NJ assessment file*

Description

Does cleanup of the raw assessment file, primarily ensuring that columns tagged as 'one implied' are displayed correctly.

Usage

```
process_nj_assess(df, layout)
```

Arguments

df	A raw NJASK, HSPA, or GEPA data frame (eg output of 'get_raw_njask')
layout	Which layout file to use to determine which columns are one implied decimal.

Value

Processed assessment data frame

process_parcc *Process a raw PARCC data file*

Description

All the logic needed to clean up the raw PARCC files.

Usage

```
process_parcc(parcc_file, end_year, grade, subj)
```

Arguments

parcc_file	Output of get_raw_parcc
end_year	A school year. end_year is the end of the academic year - eg 2014-15 school year is end_year 2015. Valid values are 2015-2024.
grade	Integer or character specifying grade level
subj	PARCC subject. c('ela', 'math', or 'science')

Value

A tbl_df / data frame

process_reportcard_special_pop *Process Report Card Special Population Data*

Description

Process Report Card Special Population Data

Usage

```
process_reportcard_special_pop(df)
```

Arguments

df	raw special pop dataframe - output of get_reportcard_special_pop

Value

tidy dataframe with conforming id columns

PROFICIENT_LEVELS *Proficient levels (L4 and L5)*

Description

Proficient levels (L4 and L5)

Usage

```
PROFICIENT_LEVELS
```

Format

An object of class `character` of length 2.

progress_enabled *Check if progress indicators are enabled*

Description

Check if progress indicators are enabled

Usage

```
progress_enabled()
```

Value

Logical

progress_tracker *Create a simple progress tracker*

Description

Creates a progress tracker for batch operations that displays progress messages to the console.

Usage

```
progress_tracker(total, task_name = "Processing")
```

Arguments

total	Total number of items to process
task_name	Name of the task being performed

Value

A list with update() and done() functions

Examples

```
## Not run:
pb <- progress_tracker(10, "Downloading files")
for (i in 1:10) {
  Sys.sleep(0.1)
  pb$update(i, sprintf("File %d", i))
}
pb$done()

## End(Not run)
```

prog_codes *NJ Program Codes*

Description

Program codes used by the New Jersey Department of Education for student enrollment reporting. Includes grade levels, special education programs, and other educational programs.

A compilation of all the "program codes" used by the state of NJ. example: http://www.nj.gov/education/data/enr/enr15/st

Usage

```
prog_codes

prog_codes
```

Format

A data frame with 682 rows and 3 columns:

end_year School year ending year

program_code Two-digit program code

program_name Program name or description

end_year end_year

program_code program_code

program_name program_name #'

Source

NJ Department of Education

NJDOE website for each year of enrollment data (see link above)

`rc_numeric_cleaner` *Report Card Numeric Data Cleaner*

Description

Cleans numeric data from report cards by removing percent signs, handling suppression codes, and converting N/A representations.

Usage

```
rc_numeric_cleaner(data_vector)
```

Arguments

`data_vector`	vector of data that has percent signs, supression codes, or a variety of representations of N/A

Value

numeric vector

rc_year_matcher *Report card year matcher*

Description

Filters a report card table to only include data for the specified year.

Usage

```
rc_year_matcher(df)
```

Arguments

df	a report card table that includes trailing/longitudinal data

Value

data frame with only data for the year of the report

recover_suppressed_grate *Recover suppressed district graduation rates from school data*

Description

When NJ DOE suppresses district-level graduation rates (showing them as NA), this function attempts to calculate them from school-level data. This is useful when district data is suppressed but school-level data is available for the same subgroup.

Usage

```
recover_suppressed_grate(
  df,
  min_schools = 1,
  min_cohort = 10,
  log_dir = tempdir()
)
```

Arguments

df	Graduation rate data frame with both school and district level data. Must include columns: district_id, school_id, subgroup, grad_rate, cohort_count, end_year, is_district, is_school
min_schools	Minimum number of schools required to calculate district rate. Default is 1.
min_cohort	Minimum total cohort size required. Default is 10.
log_dir	Directory for log files. Default is tempdir().

Details

The function calculates district rates as the weighted average of school rates, weighted by cohort count. This matches NJ DOE's methodology.

Recovery only occurs when: - District-level grad_rate is NA (suppressed) - At least 'min_schools' schools have non-NA rates for that subgroup - Total cohort is at least 'min_cohort'

A log file is written with details of all recoveries.

Value

Data frame with recovered district rates where possible. Adds columns: - 'grad_rate_recovered': TRUE if the rate was recovered from school data - 'grad_rate_original': Original (suppressed) value before recovery - 'recovered_n_schools': Number of schools used in calculation - 'recovered_cohort': Total cohort used in calculation

---

school_name_to_id *Given friendly school names, returns ids*

Description

Given friendly school names, returns ids

Usage

```
school_name_to_id(school_names, df)
```

Arguments

school_names	vector or list of friendly school names
df	df used to generate the friendly names

Value

vector of school_ids

---

sector_gap *Calculate performance gap between charter and district sectors*

Description

Calculates the difference in performance between the charter sector aggregate and traditional district for host cities. Positive values indicate charter sector outperforms district.

Usage

```
sector_gap(df, metric_col, year_col = "end_year")
```

Arguments

df	Dataframe with charter sector and district rows, including is_charter_sector, is_district, and is_allpublic flags
metric_col	Character. Performance metric column.
year_col	Character. Year column. Default "end_year".

Value

Dataframe with one row per city-year containing:

- charter_value: Charter sector metric value
- district_value: Traditional district metric value
- sector_gap: charter_value - district_value
- sector_leader: "charter", "district", or "tie"

Examples

```
## Not run:
# Get sector gaps for all host cities
grate_with_aggs %>%
  sector_gap(metric_col = "grad_rate")

## End(Not run)
```

---

sector_percentile_comparison

Compare percentile ranks across sectors

Description

Calculates and compares percentile ranks for charter sector, district sector, and all-public aggregates. Useful for MarGrady-style sector comparisons.

Usage

```
sector_percentile_comparison(
  df,
  metric_col,
  host_district_id = NULL,
  year_col = "end_year"
)
```

Arguments

df	Dataframe containing both charter and district data with is_charter_sector, is_district, is_allpublic flags
metric_col	Character. The metric to calculate percentile ranks for.
host_district_id	Character. Optional. Filter to specific host district.
year_col	Character. Year column name. Default "end_year".

Value

Dataframe with sector comparison including percentile ranks

`spec_pop_aggregate_calcs` *Aggregate multiple special populations rows and produce summary statistics*

Description

Aggregate multiple special populations rows and produce summary statistics

Usage

```
spec_pop_aggregate_calcs(df)
```

Arguments

`df`	grouped df of special populations data

Value

data_frame

`sped_aggregate_calcs` *Aggregate multiple sped rows and produce summary statistics*

Description

Aggregate multiple sped rows and produce summary statistics

Usage

```
sped_aggregate_calcs(df)
```

Arguments

`df`	grouped df of sped data

Value

df with aggregate stats for whatever grouping was provided

sped_lookup_map *Special Education District Lookup Map*

Description

Mapping of county names to district names and district IDs for New Jersey school districts. Used for matching special education data to districts.

Usage

```
sped_lookup_map
```

Format

A data frame with 578 rows and 3 columns:

county_name County name

district_name District name

district_id District identifier code

Source

NJ Department of Education

split_enr_cols *Split enrollment columns*

Description

Splits enrollment columns that combine IDs and names (pre 2009-10 format)

Usage

```
split_enr_cols(df)
```

Arguments

df	An enrollment data frame (eg output of 'get_raw_enr')

Value

Data frame with split columns

spr_sheet_mapping *Map Sheet Names Across Years*

Description

This list maps common sheet name variations. Format is: `list(canonical_name = list(year_range = "actual_sheet_name"))`

Usage

```
spr_sheet_mapping
```

Format

An object of class `list` of length 2.

Details

Internal data structure mapping sheet name variations across years. Some SPR sheet names changed over time (e.g., 2018-2019 vs 2020+).

standard_assess *call the correct* `fetch` *function for normal assessment years*

Description

for 2008-2014, this function will grab the NJASK for gr 3-8, and HSPA for grade 11

Usage

```
standard_assess(end_year, grade)
```

Arguments

`end_year`	a school year. end_year is the end of the academic year - eg 2013-14 school year is end_year '2014'. valid values are 2004-2014.
`grade`	a grade level. valid values are 3,4,5,6,7,8

statewide_peer_percentile *Calculate statewide peer percentile by grade*

Description

calculates statewide percentile by grade/test

Usage

```
statewide_peer_percentile(df)
```

Arguments

df	data.frame with PARCC/assessment data containing required columns

Value

data.frame with percent proficient and scale score percentile rank

STATE_COUNTY_ID *State aggregate indicator codes*

Description

State aggregate indicator codes

Usage

```
STATE_COUNTY_ID
```

Format

An object of class `character` of length 1.

STATE_DISTRICT_ID *State aggregate indicator codes*

Description

State aggregate indicator codes

Usage

```
STATE_DISTRICT_ID
```

Format

An object of class `character` of length 1.

SUBGROUP_PAIRS *Standard subgroup pairs for gap analysis*

Description

Pre-defined pairs of subgroups commonly used for achievement gap calculations. Each pair consists of c(reference_group, comparison_group) where gap = reference - comparison.

Usage

```
SUBGROUP_PAIRS
```

Format

An object of class `list` of length 6.

tges_name_cleaner *TGES name cleaner*

Description

internal function for converting cryptic variable codes to full name

Usage

```
tges_name_cleaner(x, indicator_fields)
```

Arguments

`x`	vector of names
`indicator_fields`	list of key/value variables to convert

Value

character vector of names

tidy_administrative_costs *Tidy Administrative Costs*

Description

Tidy Administrative Costs

Usage

```
tidy_administrative_costs(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_admin_salaries *Tidy Administrative Salaries and Benefits*

Description

Tidy Administrative Salaries and Benefits

Usage

```
tidy_admin_salaries(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_budgetary_per_pupil_cost

Tidy Budgetary Per Pupil data frame

Description

Tidy Budgetary Per Pupil data frame

Usage

```
tidy_budgetary_per_pupil_cost(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_budgeted_vs_actual_fund_balance

Tidy Budgeted vs Actual Fund Balance

Description

Tidy Budgeted vs Actual Fund Balance

Usage

```
tidy_budgeted_vs_actual_fund_balance(df, end_year)
```

Arguments

df	general fund vs actual used data frame, eg CSG20 output from get_raw_tges()
end_year	end year that the report was published

Value

data frame

tidy_classroom_general_supplies
Tidy Classroom General Supplies and Textbooks

Description

Tidy Classroom General Supplies and Textbooks

Usage

```
tidy_classroom_general_supplies(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_classroom_purchased_services
Tidy Classroom Purchased Services and Other

Description

Tidy Classroom Purchased Services and Other

Usage

```
tidy_classroom_purchased_services(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_classroom_salaries_benefits *Tidy Classroom Salaries and Benefits*

Description

Tidy Classroom Salaries and Benefits

Usage

```
tidy_classroom_salaries_benefits(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_enr *Tidy enrollment data*

Description

Transforms wide enrollment data to long format with subgroup column.

Usage

```
tidy_enr(df)
```

Arguments

df	A wide data.frame of processed enrollment data - eg output of ‘fetch_enr‘

Value

A long data.frame of tidied enrollment data

tidy_equipment *Tidy Equipment Costs*

Description

Tidy Equipment Costs

Usage

```
tidy_equipment(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_excess_unreserved_general_fund *Tidy Excess Unreserved General Fund*

Description

Tidy Excess Unreserved General Fund

Usage

```
tidy_excess_unreserved_general_fund(df, end_year)
```

Arguments

df	excess unreserved general fund data frame, eg CSG21 output from get_raw_tges()
end_year	end year that the report was published

Value

data frame

tidy_extracurricular *Tidy Extracurricular Costs*

Description

Tidy Extracurricular Costs

Usage

```
tidy_extracurricular(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_food_service *Tidy Food Service Costs*

Description

Tidy Food Service Costs

Usage

```
tidy_food_service(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_generic_budget_indicator

tidy common/generic budget indicator data frame

Description

tidy common/generic budget indicator data frame

Usage

```
tidy_generic_budget_indicator(df, end_year, indicator)
```

Arguments

df	indicator data frame, eg output of get_raw_tges() indicators 1-15
end_year	end year that the report was published
indicator	character, indicator name

Value

long, tidy data frame

tidy_generic_personnel

tidy generic personnel indicator data frame

Description

tidy generic personnel indicator data frame

Usage

```
tidy_generic_personnel(df, end_year, indicator)
```

Arguments

df	personnel data frame, eg output of get_raw_tges() indicators 16-19
end_year	end year that the report was published
indicator	character, indicator name

Value

long, tidy data frame

tidy_grad_count *Tidy Grad Count*

Description

Transforms graduation count data to long format.

Usage

```
tidy_grad_count(df, end_year)
```

Arguments

df	Output of process_grad_count
end_year	End of the academic year - eg 2006-07 is 2007. Valid values are 1998-present.

Value

data.frame with number of graduates

tidy_grad_rate *Tidy grad rate*

Description

Tidies a processed grate data frame, producing a data frame with consistent headers and values, suitable for longitudinal analysis.

Usage

```
tidy_grad_rate(df, end_year, methodology = "4 year")
```

Arguments

df	The output of process_grad_rate
end_year	A school year. Year is the end of the academic year - eg 2006-07 school year is year '2007'. Valid values are 1998-2024.
methodology	One of '4 year' or '5 year'

Value

Tidied graduation rate data frame

tidy_legal_services *Tidy Legal Services*

Description

Tidy Legal Services

Usage

```
tidy_legal_services(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_nj_assess *tidies NJ assessment data*

Description

tidy_nj_assess is a utility/internal function that takes the somewhat messy/inconsistent assessment headers and returns a tidy data frame.

Usage

```
tidy_nj_assess(assess_name, df)
```

Arguments

assess_name	NJASK, GEPA, HSPA
df	a processed data frame (eg, output of process_njask)

tidy_parcc_subgroup *Tidy PARCC subgroup names*

Description

Standardizes subgroup names across years.

Usage

```
tidy_parcc_subgroup(sv)
```

Arguments

sv	Subgroup column from PARCC data file

Value

Character vector with consistent subgroup names

tidy_personal_services_benefits *Tidy Personal Services and Benefits Costs*

Description

Tidy Personal Services and Benefits Costs

Usage

```
tidy_personal_services_benefits(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_plant_operations_maintenance
Tidy Plant Operations and Maintenance

Description

Tidy Plant Operations and Maintenance

Usage

```
tidy_plant_operations_maintenance(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_plant_operations_maintenance_salaries
Tidy Plant Operations and Maintenance - Salaries and Benefits

Description

Tidy Plant Operations and Maintenance - Salaries and Benefits

Usage

```
tidy_plant_operations_maintenance_salaries(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

`tidy_ratio_faculty_to_administrators` *Tidy Ratio of Faculty to Administrators*

Description

Tidy Ratio of Faculty to Administrators

Usage

```
tidy_ratio_faculty_to_administrators(df, end_year)
```

Arguments

`df`	indicator data frame, eg output of get_raw_tges()
`end_year`	end year that the report was published

Value

data.frame

`tidy_ratio_students_to_administrators` *Tidy Ratio of Students to Administrators*

Description

Tidy Ratio of Students to Administrators

Usage

```
tidy_ratio_students_to_administrators(df, end_year)
```

Arguments

`df`	indicator data frame, eg output of get_raw_tges()
`end_year`	end year that the report was published

Value

data.frame

tidy_ratio_students_to_special_service
Tidy Ratio of Students to Special Service

Description

Tidy Ratio of Students to Special Service

Usage

```
tidy_ratio_students_to_special_service(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_ratio_students_to_teachers
Tidy Ratio of Students to Teachers

Description

Tidy Ratio of Students to Teachers

Usage

```
tidy_ratio_students_to_teachers(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_support_services_salaries *Tidy Support Services Salaries*

Description

Tidy Support Services Salaries

Usage

```
tidy_support_services_salaries(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_tges_data *Tidy list of TGES data frames*

Description

Tidy list of TGES data frames

Usage

```
tidy_tges_data(list_of_dfs, end_year)
```

Arguments

list_of_dfs	list of TGES data frames, eg output of get_raw_tges(). Current valid values are 2011 to 2017.
end_year	year that the report was published

Value

list of cleaned (wide to long, tidy) dataframes

tidy_total_classroom_instruction *Tidy Total Classroom Instruction data frame*

Description

Tidy Total Classroom Instruction data frame

Usage

```
tidy_total_classroom_instruction(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_total_spending_per_pupil *tidy total spending per pupil*

Description

tidy total spending per pupil

Usage

```
tidy_total_spending_per_pupil(df, end_year)
```

Arguments

df	total spending data frame, eg CSG1AA_AVGS output from get_raw_tges()
end_year	end year that the report was published

Value

data frame

tidy_total_support_services *Tidy Total Support Services*

Description

Tidy Total Support Services

Usage

```
tidy_total_support_services(df, end_year)
```

Arguments

df	indicator data frame, eg output of get_raw_tges()
end_year	end year that the report was published

Value

data.frame

tidy_vitstat *Tidy Vital Statistics*

Description

Tidy Vital Statistics

Usage

```
tidy_vitstat(df, end_year)
```

Arguments

df	vital statistics data frame, eg VITSTAT_TOTAL output from get_raw_tges()
end_year	end year that the report was published

Value

data frame

TOTAL_PROGRAM_CODE *Program code for school/district totals*

Description

Program code for school/district totals

Usage

```
TOTAL_PROGRAM_CODE
```

Format

An object of class `character` of length 1.

trim_whitespace *Trim whitespace*

Description

Removes leading and trailing whitespace from strings.

Usage

```
trim_whitespace(x)
```

Arguments

x	string or vector of strings

Value

a string or vector of strings, with whitespace removed.

trunc2 *Truncate with configurable precision*

Description

Truncates numeric values to specified decimal precision.

Usage

```
trunc2(x, prec = 0)
```

Arguments

x	numeric vector
prec	desired precision

Value

truncated numeric vector

`unzipper` *Download and unzip a file from URL*

Description

Downloads a zip file from a URL to a temporary location and extracts it.

Usage

```
unzipper(url, file_pattern = "njschooldata")
```

Arguments

`url`	character, target url
`file_pattern`	character, stub to use for temporary file

Value

list of unzipped files in a temp directory

`validate_end_year` *Validate end_year parameter*

Description

Validates that end_year is a valid integer within the allowed range for the specified data type.

Usage

```
validate_end_year(end_year, data_type)
```

Arguments

`end_year`	The year to validate
`data_type`	The type of data being requested (e.g., "enrollment", "parcc")

Value

TRUE invisibly if valid, otherwise throws an error

Examples

```
## Not run:
validate_end_year(2024, "enrollment")  # Valid
validate_end_year(1990, "enrollment")  # Error: year out of range
validate_end_year(2020, "parcc")       # Error: COVID year

## End(Not run)
```

validate_grade *Validate grade parameter*

Description

Validates that grade is valid for the specified assessment type and year.

Usage

```
validate_grade(grade, assessment_type, end_year)
```

Arguments

grade	The grade to validate (numeric or character like "ALG1")
assessment_type	The type of assessment ("njask", "parcc", etc.)
end_year	The year of the assessment

Value

TRUE invisibly if valid, otherwise throws an error

validate_grate_aggregation *Validate graduation rate aggregation*

Description

Compares district-level graduation rates against weighted averages calculated from school-level data. Flags discrepancies that exceed a threshold, which may indicate data quality issues in the source files.

Usage

```
validate_grate_aggregation(df, tolerance = 2, log_dir = tempdir())
```

Arguments

df	Graduation rate data frame with both school and district level data
tolerance	Maximum allowed difference (in percentage points) between reported district rate and calculated rate. Default is 2 (2 percentage points).
log_dir	Directory for log files. Default is tempdir().

Details

This function helps identify potential data quality issues where: - District totals don't match sum of school data - Rates are mathematically inconsistent - Data may have been incorrectly entered or processed

A detailed log file is written documenting all discrepancies found.

Value

Data frame with validation columns added: - ‘calculated_from_schools‘: Rate calculated from school-level data - ‘rate_discrepancy_pp‘: Difference in percentage points - ‘aggregation_flag‘: "OK", "DISCREPANCY", "MISSING_SCHOOL_DATA", or "SUPPRESSED"

---

validate_logical *Validate logical/boolean parameter*

Description

Validate logical/boolean parameter

Usage

```
validate_logical(value, param_name)
```

Arguments

value	The value to validate
param_name	The name of the parameter (for error messages)

Value

TRUE invisibly if valid, otherwise throws an error

---

validate_methodology *Validate methodology parameter for graduation data*

Description

Validates that methodology is either "4 year" or "5 year" and that 5-year rates are available for the specified year.

Usage

```
validate_methodology(methodology, end_year = NULL)
```

Arguments

methodology	The methodology to validate
end_year	Optional year to check availability

Value

TRUE invisibly if valid, otherwise throws an error

validate_parcc_call *Validate PARCC/NJSLA parameters*

Description

Validates all parameters for fetch_parcc() in one call.

Usage

```
validate_parcc_call(end_year, grade_or_subj, subj)
```

Arguments

end_year	The year
grade_or_subj	The grade or subject code
subj	The subject ("ela" or "math")

Value

TRUE invisibly if valid, otherwise throws an error

validate_subject *Validate subject parameter*

Description

Validates that subject is either "ela" or "math".

Usage

```
validate_subject(subj)
```

Arguments

subj	The subject to validate

Value

TRUE invisibly if valid, otherwise throws an error

valid_call *determine if a end_year/grade pairing can be downloaded from the state website*

Description

valid_call returns a boolean value indicating if a given end_year/grade pairing is valid for assessment data

Usage

```
valid_call(end_year, grade)
```

Arguments

end_year	a school year. end_year is the end of the academic year - eg 2013-14 school year is end_year '2014'. valid values are 2004-2014.
grade	a grade level. valid values are 3,4,5,6,7,8

verify_data_urls *Verify all configured URLs for a data type*

Description

Checks that URLs for a range of years are accessible.

Usage

```
verify_data_urls(data_type, years = NULL)
```

Arguments

data_type	One of "enrollment", "graduation", "assessment"
years	Vector of years to check (defaults to recent 3 years)

Value

Data frame with URL, year, and accessibility status

Examples

```
## Not run:
verify_data_urls("enrollment", 2022:2024)

## End(Not run)
```

ward_enr_aggs *Aggregates enrollment data by ward*

Description

Aggregates enrollment data by ward

Usage

```
ward_enr_aggs(df)
```

Arguments

df	data frame containing enrollment data

Value

A data frame of ward aggregations

ward_gcount_aggs *Aggregates grad counts data by ward*

Description

Aggregates grad counts data by ward

Usage

```
ward_gcount_aggs(df)
```

Arguments

df	output of `fetch_grad_count`

Value

A data frame of ward aggregations

ward_grate_aggs *Aggregates grad rate data by ward*

Description

Aggregates grad rate data by ward

Usage

```
ward_grate_aggs(df)
```

Arguments

df	output of fetch_grad_rate

Value

A data frame of ward aggregations

ward_matric_aggs *Aggregates matriculation data by ward*

Description

Aggregates matriculation data by ward

Usage

```
ward_matric_aggs(df)
```

Arguments

df	output of enrich_matric_counts

Value

A data frame of ward aggregations

ward_parcc_aggs *Aggregates assessment data by ward*

Description

Aggregates assessment data by ward

Usage

```
ward_parcc_aggs(list_of_dfs)
```

Arguments

list_of_dfs	output of fetch_all_parcc or fetch_parcc

Value

A data frame of ward aggregations

WARD_SUFFIX *Suffix for ward aggregations*

Description

Suffix for ward aggregations

Usage

```
WARD_SUFFIX
```

Format

An object of class character of length 1.

with_cache *Create a cached version of a fetch function*

Description

Wraps a data fetching function to use the session cache.

Usage

```
with_cache(fn, fn_name)
```

Arguments

fn	The fetch function to wrap
fn_name	Name of the function (for cache keys)

Value

A cached version of the function

`year_ranges` *Valid year ranges for each data type*

Description

Valid year ranges for each data type

Valid year ranges by data type

Usage

```
year_ranges
```

Format

An object of class `list` of length 12.

`year_variable_converter`
year variable converter

Description

for the 1999-2003 tges files, the 'year' of the data was encoded in the variable names

Usage

```
year_variable_converter(df, end_year)
```

Arguments

`df`	a tges indicator data frame published between 1999 and 2003
`end_year`	year published

Value

data frame that conforms to 2004-2009 style

Index