Package: moderndive (via r-universe)

October 30, 2026

Type Package

Title Tidyverse-Friendly Introductory Linear Regression

Version 0.8.1

Maintainer Albert Y. Kim <albert.ys.kim@gmail.com>

Description Datasets and wrapper functions for tidyverse-friendly introductory linear regression, used in ``Statistical Inference via Data Science: A ModernDive into R and the Tidyverse" available at <https://moderndive.com/>.

Depends R (>= 4.1.0)

License GPL-3

Encoding UTF-8

LazyData true

URL https://moderndive.github.io/moderndive/, https://github.com/moderndive/moderndive/

BugReports https://github.com/moderndive/moderndive/issues

Imports magrittr, dplyr, purrr, tidyr, ggplot2, tibble, janitor, broom (>= 0.4.3), formula.tools, stringr, knitr, infer, rlang (>= 0.2.0), glue, DT

Suggests testthat, covr, rmarkdown, vdiffr, openintro, patchwork, viridis, readr, nycflights13, nycflights23, plotly

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Albert Y. Kim [aut, cre] (ORCID: <https://orcid.org/0000-0001-7824-306X>), Chester Ismay [aut] (ORCID: <https://orcid.org/0000-0003-2820-2547>), Andrew Bray [ctb] (ORCID: <https://orcid.org/0000-0002-4037-7414>), Delaney Moran [ctb], Evgeni Chasnovski [ctb] (ORCID: <https://orcid.org/0000-0002-1617-4019>), Will Hopper [ctb] (ORCID: <https://orcid.org/0000-0002-7848-1946>), Benjamin S.

Baumer [ctb] (ORCID: <https://orcid.org/0000-0002-3279-0516>), Marium Tapal [ctb] (ORCID: <https://orcid.org/0000-0001-5093-6462>), Wayne Ndlovu [ctb], Catherine Peppers [ctb], Annah Mutaya [ctb], Anushree Goswami [ctb], Ziyue Yang [ctb] (ORCID: <https://orcid.org/0000-0002-9299-8327>), Clara Li [ctb] (ORCID: <https://orcid.org/0000-0003-2456-0849>), Caroline McKenna [ctb], Catherine Park [ctb] (ORCID: <https://orcid.org/0000-0002-8273-9620>), Abbie Benfield [ctb], Georgia Gans [ctb], Kacey Jean-Jacques [ctb], Swaha Bhattacharya [ctb], Vivian Almaraz [ctb], Elle Jo Whalen [ctb], Jacqueline Chen [ctb], Michelle Flesaker [ctb], Irene Foster [ctb], Aushanae Haller [ctb], Benjamin Bruncati [ctb] (ORCID: <https://orcid.org/0000-0001-8545-5984>), Quinn White [ctb] (ORCID: <https://orcid.org/0000-0001-5399-0237>), Tianshu Zhang [ctb] (ORCID: <https://orcid.org/0000-0002-3004-4472>), Katelyn Diaz [ctb] (ORCID: <https://orcid.org/0000-0001-6108-1682>), Rose Porta [ctb], Renee Wu [ctb], Arris Moise [ctb], Kate Phan [ctb], Grace Hartley [ctb], Silas Weden [ctb], Emma Vejcik [ctb], Nikki Schuldt [ctb], Tess Goldmann [ctb], Hongtong Lin [ctb], Alejandra Munoz [ctb], Elina Gordon-Halpern [ctb], Haley Schmidt [ctb] (ORCID: <https://orcid.org/0000-0002-6184-2266>)

Repository https://cran.r-universe.dev

Date/Publication 2026-10-30 14:26:34 UTC

RemoteUrl https://github.com/cran/moderndive

RemoteRef HEAD

RemoteSha 7687b9660425be282be29ea5825e8bd8f423aab0

Contents

alaska_flights . . . 3
almonds_bowl . . . 4
almonds_sample . . . 5
almonds_sample_100 . . . 5
amazon_books . . . 6
avocados . . . 6
babies . . . 7
bowl . . . 8
bowl_sample_1 . . . 9
bowl_samples . . . 9
coffee_quality . . . 10
coffee_ratings . . . 11
DD_vs_SB . . . 13
early_january_2023_weather . . . 14
early_january_weather . . . 15
envoy_flights . . . 16

ev_charging . . . 17
evals . . . 18
geom_categorical_model . . . 19
geom_parallel_slopes . . . 21
get_correlation . . . 24
get_regression_points . . . 26
get_regression_summaries . . . 27
get_regression_table . . . 28
gg_parallel_slopes . . . 29
house_prices . . . 31
ipf_lifts . . . 32
MA_schools . . . 33
ma_traffic_2020_vs_2019 . . . 33
mario_kart_auction . . . 34
mass_traffic_2020 . . . 35
moderndive . . . 36
movies_sample . . . 38
mythbusters_yawn . . . 38
old_faithful_2024 . . . 39
orig_pennies_sample . . . 40
pennies . . . 40
pennies_resamples . . . 41
pennies_sample . . . 41
plot_3d_regression . . . 42
pop_sd . . . 43
promotions . . . 43
promotions_shuffled . . . 44
saratoga_houses . . . 45
spotify_52_original . . . 45
spotify_52_shuffled . . . 46
spotify_by_genre . . . 47
tactile_prop_red . . . 48
tidy_summary . . . 49
un_member_states_2024 . . . 50
View . . . 52

Index **54**

`alaska_flights` *Alaska flights data*

Description

On-time data for all Alaska Airlines flights that departed NYC (i.e. JFK, LGA or EWR) in 2013. This is a subset of the `flights` data frame from `nycflights13`.

Usage

```
alaska_flights
```

Format

A data frame of 714 rows representing Alaska Airlines flights and 19 variables

year, month, day Date of departure.

dep_time, arr_time Actual departure and arrival times (format HHMM or HMM), local tz.

sched_dep_time, sched_arr_time Scheduled departure and arrival times (format HHMM or HMM), local tz.

dep_delay, arr_delay Departure and arrival delays, in minutes. Negative times represent early departures/arrivals.

carrier Two letter carrier abbreviation. See `nycflights13::airlines` to get name.

flight Flight number.

tailnum Plane tail number. See `nycflights13::planes` for additional metadata.

origin, dest Origin and destination. See `nycflights13::airports` for additional metadata.

air_time Amount of time spent in the air, in minutes.

distance Distance between airports, in miles.

hour, minute Time of scheduled departure broken into hour and minutes.

time_hour Scheduled date and hour of the flight as a `POSIXct` date. Along with `origin`, can be used to join flights data to `nycflights13::weather` data.

Source

RITA, Bureau of transportation statistics

See Also

`nycflights13::flights`.

`almonds_bowl` *Chocolate-covered almonds data*

Description

5000 chocolate-covered almonds selected from a large batch, weighed in grams.

Usage

```
almonds_bowl
```

Format

A data frame with 5000 observations on the following 2 variables

ID Identification value for a given chocolate-covered almond

weight Weight of the chocolate-covered almond in grams (to the nearest tenth)

---

almonds_sample *Chocolate-covered almonds data sample of size 25*

Description

A sample of 25 chocolate-covered almonds, weighed in grams.

Usage

```
almonds_sample
```

Format

A data frame with 25 observations on the following 2 variables

replicate Replicate number set to 1 since there is only one sample

ID Identification value for a given chocolate-covered almond

weight Weight of the chocolate-covered almond in grams (to the nearest tenth)

---

almonds_sample_100 *Chocolate-covered almonds data sample of size 100*

Description

A sample of 100 chocolate-covered almonds, weighed in grams.

Usage

```
almonds_sample_100
```

Format

A data frame with 100 observations on the following 2 variables

replicate Replicate number set to 1 since there is only one sample

ID Identification value for a given chocolate-covered almond

weight Weight of the chocolate-covered almond in grams (to the nearest tenth)

amazon_books *Sample of Amazon books*

Description

A random sample of 325 books from Amazon.com.

Usage

```
amazon_books
```

Format

A data frame of 325 rows representing books listed on Amazon and 13 variables.

title Book title

author Author who wrote book

list_price recommended retail price of book

amazon_price lowest price of book shown on Amazon

hard_paper book is either hardcover or paperback

num_pages number of pages in book

publisher Company that issues the book for sale

pub_year Year the book was published

isbn_10 10-character ISBN number

height, width, thick, weight_oz height, width, weight and thickness of the book

Source

The Data and Story Library (DASL) https://dasl.datadescription.com/datafile/amazon-books

avocados *Avocado Prices by US Region*

Description

Gathered from https://docs.google.com/spreadsheets/d/1cNuj9V-9Xe8fqV3DQRhvsXJhER3zTkO1dSsQ1Q0j96g/edit#gid=1419070688

Usage

```
avocados
```

Format

A data frame of 54 regions over 3 years of weekly results

date Week of Data Recording

average_price Average Price of Avocado

total_volume Total Amount of Avocados

small_hass_sold Amount of Small Haas Avocados Sold

large_hass_sold Amount of Large Haas Avocados Sold

xlarge_hass_sold Amount of Extra Large Haas Avocados Sold

total_bags Total Amount of Bags of Avocados

small_bags Total Amount of Bags of Small Haas Avocados

large_bags Total Amount of Bags of Large Haas Avocados

x_large_bags Total Amount of Bags of Extra Large Haas Avocados

type Type of Sale

year Year of Sale

region Region Where Sale Took Place

`babies` *Data on maternal smoking and infant health*

Description

Data on maternal smoking and infant health

Usage

```
babies
```

Format

A data frame of 1236 rows of individual mothers.

id Identification number

pluralty Marked 5 for single fetus, otherwise number of fetuses

outcome Marked 1 for live birth that survived at least 28 days

date Birth date where 1096 is January 1st, 1961

birthday Birth date in mm-dd-yyyy format

gestation Length of gestation in days, marked 999 if unknown

sex Infant's sex, where 1 is male, 2 is female, and 9 is unknown

wt Birth weight in ounces, marked 999 if unknown

parity Total number of previous pregnancies including fetal deaths and stillbirths, marked 99 if unknown

race Mother's race where 0-5 is white, 6 is Mexican, 7 is Black, 8 is Asian, 9 is mixed, and 99 is unknown

age Mother's age in years at termination of pregnancy, 99=unknown

ed Mother's education 0= less than 8th grade, 1 = 8th -12th grade - did not graduate, 2= HS graduate–no other schooling , 3= HS+trade, 4=HS+some college 5= College graduate, 6&7 Trade school HS unclear, 9=unknown

ht Mother's height in inches to the last completed inch, 99=unknown

wt_1 Mother prepregnancy wt in pounds, 999=unknown

drace Father's race, coding same as mother's race

dage Father's age, coding same as mother's age

ded Father's education, coding same as mother's education

dht Father's height, coding same as for mother's height

dwt Father's weight coding same as for mother's weight

marital 0= legally separated, 1=married, 2= divorced, 3=widowed, 5=never married

inc Family yearly income in $2500 increments 0 = under 2500, 1=2500-4999, ..., 8= 12,500-14,999, 9=15000+, 98=unknown, 99=not asked

smoke Does mother smoke? 0=never, 1= smokes now, 2=until current pregnancy, 3=once did, not now, 9=unknown

time If mother quit, how long ago? 0=never smoked, 1=still smokes, 2=during current preg, 3=within 1 yr, 4= 1 to 2 years ago, 5= 2 to 3 yr ago, 6= 3 to 4 yrs ago, 7=5 to 9yrs ago, 8=10+yrs ago, 9=quit and don't know, 98=unknown, 99=not asked

number Number of cigs smoked per day for past and current smokers 0=never, 1=1-4, 2=5-9, 3=10-14, 4=15-19, 5=20-29, 6=30-39, 7=40-60, 8=60+, 9=smoke but don't know, 98=unknown, 99=not asked

Source

Data on maternal smoking and infant health from https://www.stat.berkeley.edu/~statlabs/labs.html

---

`bowl` *A sampling bowl of red and white balls*

Description

A sampling bowl used as the population in a simulated sampling exercise. Also known as the urn sampling framework https://en.wikipedia.org/wiki/Urn_problem.

Usage

```
bowl
```

Format

A data frame 2400 rows representing different balls in the bowl, of which 900 are red and 1500 are white.

ball_ID ID variable used to denote all balls. Note this value is not marked on the balls themselves

color color of ball: red or white

---

bowl_sample_1 *Tactile sample of size 50 from a bowl of balls*

Description

A single tactile sample of size n = 50 balls from https://github.com/moderndive/moderndive/blob/master/data-raw/sampling_bowl.jpeg

Usage

```
bowl_sample_1
```

Format

A data frame of 50 rows representing different balls and 1 variable.

color Color of ball sampled

See Also

bowl()

---

bowl_samples *Sampling from a bowl of balls*

Description

Counting the number of red balls in 10 samples of size n = 50 balls from https://github.com/moderndive/moderndive/blob/master/data-raw/sampling_bowl.jpeg

Usage

```
bowl_samples
```

Format

A data frame 10 rows representing different groups of students' samples of size n = 50 and 5 variables

group Group name

red Number of red balls sampled

white Number of white balls sampled

green Number of green balls sampled

n Total number of balls samples

See Also

`bowl()`

`coffee_quality` *Coffee Quality Dataset*

Description

This dataset contains detailed information about coffee quality evaluations from various origins. It includes data on the country and continent of origin, farm name, lot number, and various quality metrics. The dataset also includes attributes related to coffee processing, grading, and specific sensory attributes.

Usage

```
coffee_quality
```

Format

A data frame with 207 rows and 30 variables:

country_of_origin `character`. The country where the coffee originated.

continent_of_origin `character`. The continent where the coffee originated.

farm_name `character`. The name of the farm where the coffee was grown.

lot_number `character`. The lot number assigned to the batch of coffee.

mill `character`. The name of the mill where the coffee was processed.

company `character`. The company associated with the coffee batch.

altitude `character`. The altitude range (in meters) where the coffee was grown.

region `character`. The specific region within the country where the coffee was grown.

producer `character`. The name of the coffee producer.

in_country_partner `character`. The in-country partner organization associated with the coffee batch.

harvest_year `character`. The year or range of years during which the coffee was harvested.

grading_date `date`. The date when the coffee was graded.

owner `character`. The owner of the coffee batch.

variety `character`. The variety of the coffee plant.

processing_method `character`. The method used to process the coffee beans.

aroma `numeric`. The aroma score of the coffee, on a scale from 0 to 10.

flavor `numeric`. The flavor score of the coffee, on a scale from 0 to 10.

aftertaste `numeric`. The aftertaste score of the coffee, on a scale from 0 to 10.

acidity `numeric`. The acidity score of the coffee, on a scale from 0 to 10.

body `numeric`. The body score of the coffee, on a scale from 0 to 10.

balance `numeric`. The balance score of the coffee, on a scale from 0 to 10.

uniformity `numeric`. The uniformity score of the coffee, on a scale from 0 to 10.

clean_cup `numeric`. The clean cup score of the coffee, on a scale from 0 to 10.

sweetness `numeric`. The sweetness score of the coffee, on a scale from 0 to 10.

overall `numeric`. The overall score of the coffee, on a scale from 0 to 10.

total_cup_points `numeric`. The total cup points awarded to the coffee, representing the sum of various quality metrics.

moisture_percentage `numeric`. The moisture percentage of the coffee beans.

color `character`. The color description of the coffee beans.

expiration `character`. The expiration date of the coffee batch.

certification_body `character`. The body that certified the coffee batch.

Source

Coffee Quality Institute

`coffee_ratings` *Data from the Coffee Quality Institute's review pages in January 2018*

Description

1,340 digitized reviews on coffee samples from https://database.coffeeinstitute.org/.

Usage

```
coffee_ratings
```

Format

A data frame of 1,340 rows representing each sample of coffee.

total_cup_points Number of points in final rating (scale of 0-100)

species Species of coffee bean plant (Arabica or Robusta)

owner Owner of coffee plant farm

country_of_origin Coffee bean's country of origin

farm_name Name of coffee plant farm

lot_number Lot number for tested coffee beans

mill Name of coffee bean's processing facility

ico_number International Coffee Organization number

company Name of coffee bean's company

altitude Altitude at which coffee plants were grown

region Region where coffee plants were grown

producer Name of coffee bean roaster

number_of_bags Number of tested bags

bag_weight Tested bag weight

in_country_partner Partner for the country

harvest_year Year the coffee beans were harvested

grading_date Day the coffee beans were graded

owner_1 Owner of the coffee beans

variety Variety of the coffee beans

processing_method Method used for processing the coffee beans

aroma Coffee aroma rating

flavor Coffee flavor rating

aftertaste Coffee aftertaste rating

acidity Coffee acidity rating

body Coffee body rating

balance Coffee balance rating

uniformity Coffee uniformity rating

clean_cup Cup cleanliness rating

sweetness Coffee sweetness rating

cupper_points Cupper Points, an overall rating for the coffee

moisture Coffee moisture content

category_one_defects Number of category one defects for the coffee beans

quakers Number of coffee beans that don't dark brown when roasted

color Color of the coffee beans

category_two_defects Number of category two defects for the coffee beans

expiration Expiration date of the coffee beans

certification_body Entity/Institute that certified the coffee beans

certification_address Body address of certification for coffee beans

certification_contact Certification contact for coffee beans

unit_of_measurement Unit of measurement for altitude

altitude_low_meters Lower altitude level coffee beans grow at

altitude_high_meters Higher altitude level coffee beans grow at

altitude_mean_meters Average altitude level coffee beans grow at

Source

Coffee Quality Institute. Access cleaned data available at https://github.com/jldbc/coffee-quality-database

---

`DD_vs_SB` *Dunkin Donuts vs Starbucks*

Description

Number of Dunkin Donuts & Starbucks, median income, and population in 1024 census tracts in eastern Massachusetts in 2016.

Usage

```
DD_vs_SB
```

Format

A data frame of 1024 rows representing census tracts and 6 variables

county County where census tract is located. Either Bristol, Essex, Middlesex, Norfolk, Plymouth, or Suffolk county

FIPS Federal Information Processing Standards code identifying census tract

median_income Median income of census tract

population Population of census tract

shop_type Coffee shop type: Dunkin Donuts or Starbucks

shops Number of shops

Source

US Census Bureau. Code used to scrape data available at https://github.com/DelaneyMoran/FinalProject

`early_january_2023_weather` *Early January hourly weather data for 2023*

Description

Hourly meteorological data for LGA, JFK and EWR for the month of January 2023. This is a subset of the `weather` data frame from `nycflights23`.

Usage

```
early_january_2023_weather
```

Format

A data frame of 360 rows representing hourly measurements and 15 variables

origin Weather station. Named `origin` to facilitate merging with `nycflights23::flights` data.

year, month, day, hour Time of recording.

temp, dewp Temperature and dewpoint in F.

humid Relative humidity.

wind_dir, wind_speed, wind_gust Wind direction (in degrees), speed and gust speed (in mph).

precip Precipitation, in inches.

pressure Sea level pressure in millibars.

visib Visibility in miles.

time_hour Date and hour of the recording as a `POSIXct` date.

Source

ASOS download from Iowa Environmental Mesonet, archived copy at https://web.archive.org/web/20221024033531/https://mesonet.agron.iastate.edu/request/download.phtml.

See Also

`nycflights23::weather`.

`early_january_weather` *Early January hourly weather data*

Description

Hourly meteorological data for LGA, JFK and EWR for the month of January 2013. This is a subset of the `weather` data frame from `nycflights13`.

Usage

```
early_january_weather
```

Format

A data frame of 358 rows representing hourly measurements and 15 variables

origin Weather station. Named `origin` to facilitate merging with `nycflights13::flights` data.

year, month, day, hour Time of recording.

temp, dewp Temperature and dewpoint in F.

humid Relative humidity.

wind_dir, wind_speed, wind_gust Wind direction (in degrees), speed and gust speed (in mph).

precip Precipitation, in inches.

pressure Sea level pressure in millibars.

visib Visibility in miles.

time_hour Date and hour of the recording as a `POSIXct` date.

Source

ASOS download from Iowa Environmental Mesonet, archived copy at https://web.archive.org/web/20221024033531/https://mesonet.agron.iastate.edu/request/download.phtml.

See Also

`nycflights13::weather`.

envoy_flights *Envoy Air flights data for 2023*

Description

On-time data for all Envoy Air flights that departed NYC (i.e. JFK, LGA or EWR) in 2023. This is a subset of the `flights` data frame from `nycflights23`.

Usage

```
envoy_flights
```

Format

A data frame of 357 rows representing Alaska Airlines flights and 19 variables

year, month, day Date of departure.

dep_time, arr_time Actual departure and arrival times (format HHMM or HMM), local tz.

sched_dep_time, sched_arr_time Scheduled departure and arrival times (format HHMM or HMM), local tz.

dep_delay, arr_delay Departure and arrival delays, in minutes. Negative times represent early departures/arrivals.

carrier Two letter carrier abbreviation. See `nycflights23::airlines` to get name.

flight Flight number.

tailnum Plane tail number. See `nycflights23::planes` for additional metadata.

origin, dest Origin and destination. See `nycflights23::airports` for additional metadata.

air_time Amount of time spent in the air, in minutes.

distance Distance between airports, in miles.

hour, minute Time of scheduled departure broken into hour and minutes.

time_hour Scheduled date and hour of the flight as a `POSIXct` date. Along with `origin`, can be used to join flights data to `nycflights23::weather` data.

Source

RITA, Bureau of transportation statistics

See Also

`nycflights23::flights`.

`ev_charging` *Electric vehicle charging sessions for a workplace charging program*

Description

This dataset consists of information on 3,395 electric vehicle charging sessions across locations for a workplace charging program. The data contains information on multiple charging sessions from 85 electric vehicle drivers across 25 workplace locations, which are located at facilities of various types.

Usage

```
ev_charging
```

Format

A data frame of 3,395 rows on 24 variables, where each row is an electric vehicle charging session.

session_id Unique identifier specifying the electric vehicle charging session

kwh_total Total energy used at the charging session, in kilowatt hours (kWh)

dollars Quantity of money paid for the charging session in U.S. dollars

created Date and time recorded at the beginning of the charging session

ended Date and time recorded at the end of the charging session

start_time Hour of the day when the charging session began (1 through 24)

end_time Hour of the day when the charging session ended (1 through 24)

charge_time_hrs Length of the charging session in hours

weekday First three characters of the name of the weekday when the charging session occurred

platform Digital platform the driver used to record the session (android, ios, web)

distance Distance from the charging location to the driver's home, expressed in miles NA if the driver did not report their address

user_id Unique identifier for each driver

station_id Unique identifier for each charging station

location_id Unique identifier for each location owned by the company where charging stations were located

manager_vehicle Binary variable that is 1 when the vehicle is a type commonly used by managers of the firm and 0 otherwise

facility_type Categorical variable that represents the facility type:

- 1 = manufacturing
- 2 = office
- 3 = research and development
- 4 = other

mon, tues, wed, thurs, fri, sat, sun Binary variables; 1 if the charging session took place on that day, 0 otherwise

reported_zip Binary variable; 1 if the driver did report their zip code, 0 if they did not

Source

Harvard Dataverse doi:10.7910/DVN/NFPQLW. Note data is released under a CC0: Public Domain license.

`evals` *Teaching evaluations at the UT Austin*

Description

The data are gathered from end of semester student evaluations for a sample of 463 courses taught by 94 professors from the University of Texas at Austin. In addition, six students rate the professors' physical appearance. The result is a data frame where each row contains a different course and each column has information on either the course or the professor `https://www.openintro.org/data/index.php?data=evals`

Usage

```
evals
```

Format

A data frame with 463 observations corresponding to courses on the following 13 variables.

ID Identification variable for course.

prof_ID Identification variable for professor. Many professors are included more than once in this dataset.

score Average professor evaluation score: (1) very unsatisfactory - (5) excellent.

age Age of professor.

bty_avg Average beauty rating of professor.

gender Gender of professor (collected as a binary variable at the time of the study): female, male.

ethnicity Ethnicity of professor: not minority, minority.

language Language of school where professor received education: English or non-English.

rank Rank of professor: teaching, tenure track, tenured.

pic_outfit Outfit of professor in picture: not formal, formal.

pic_color Color of professor's picture: color, black & white.

cls_did_eval Number of students in class who completed evaluation.

cls_students Total number of students in class.

cls_level Class level: lower, upper.

Source

Çetinkaya-Rundel M, Morgan KL, Stangl D. 2013. Looking Good on Course Evaluations. CHANCE 26(2).

See Also

The data in `evals` is a slight modification of `openintro::evals()`.

`geom_categorical_model` *Regression model with one categorical explanatory/predictor variable*

Description

`geom_categorical_model()` fits a regression model using the categorical x axis as the explanatory variable, and visualizes the model's fitted values as piece-wise horizontal line segments. Confidence interval bands can be included in the visualization of the model. Like `geom_parallel_slopes()`, this function has the same nature as `geom_smooth()` from the `ggplot2` package, but provides functionality that `geom_smooth()` currently doesn't have. When using a categorical predictor variable, the intercept corresponds to the mean for the baseline group, while coefficients for the non-baseline groups are offsets from this baseline. Thus in the visualization the baseline for comparison group's median is marked with a solid line, whereas all offset groups' medians are marked with dashed lines.

Usage

```
geom_categorical_model(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  se = TRUE,
  level = 0.95,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept.
- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

se — Display confidence interval around model lines? TRUE by default.

level — Level of confidence interval to use (0.95 by default).

na.rm — If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

See Also

geom_parallel_slopes()

Examples

```
library(dplyr)
library(ggplot2)
library(moderndive)

p <- ggplot(evals, aes(x = rank, y = score)) +
  geom_point() +
  geom_categorical_model()
p

# In the above visualization, the solid line corresponds to the mean score
# for the baseline group "teaching", whereas the dashed lines correspond to
# the means for the non-baseline groups "tenure track" and "tenured".
# In the corresponding regression table the coefficients for "tenure track"
# and "tenured" are presented as offsets from the mean for "teaching":
model <- lm(score ~ rank, data = evals)
get_regression_table(model)

# You can use different colors for each categorical level
p + aes(color = rank)

# But mapping the color aesthetic doesn't change the model that is fit
p + aes(color = ethnicity)
```

---

geom_parallel_slopes *Parallel slopes regression model*

Description

geom_parallel_slopes() fits parallel slopes model and adds its line output(s) to a ggplot object. Basically, it fits a unified model with intercepts varying between groups (which should be supplied as standard {ggplot2} grouping aesthetics: group, color, fill, etc.). This function has the same nature as geom_smooth() from {ggplot2} package, but provides functionality that geom_smooth() currently doesn't have.

Usage

```
geom_parallel_slopes(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  se = TRUE,
  formula = y ~ x,
```

```
  n = 100,
  fullrange = FALSE,
  level = 0.95,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept.

- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`se` Display confidence interval around model lines? `TRUE` by default.

`formula` Formula to use per group in parallel slopes model. Basic linear `y ~ x` by default.

`n` Number of points per group at which to evaluate model.

`fullrange` If `TRUE`, the smoothing line gets expanded to the range of the plot, potentially beyond the data. This does not extend the line into any additional padding created by `expansion`.

`level` Level of confidence band to use (0.95 by default).

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

See Also

`geom_categorical_model()`

Examples

```
library(dplyr)
library(ggplot2)

ggplot(evals, aes(x = age, y = score, color = ethnicity)) +
  geom_point() +
  geom_parallel_slopes(se = FALSE)

# Basic usage
ggplot(evals, aes(x = age, y = score, color = ethnicity)) +
  geom_point() +
  geom_parallel_slopes()
ggplot(evals, aes(x = age, y = score, color = ethnicity)) +
  geom_point() +
  geom_parallel_slopes(se = FALSE)

# Supply custom aesthetics
ggplot(evals, aes(x = age, y = score, color = ethnicity)) +
```

```
  geom_point() +
  geom_parallel_slopes(se = FALSE, size = 4)

# Fit non-linear model
example_df <- house_prices %>%
  slice(1:1000) %>%
  mutate(
    log10_price = log10(price),
    log10_size = log10(sqft_living)
  )
ggplot(example_df, aes(x = log10_size, y = log10_price, color = condition)) +
  geom_point(alpha = 0.1) +
  geom_parallel_slopes(formula = y ~ poly(x, 2))

# Different grouping
ggplot(example_df, aes(x = log10_size, y = log10_price)) +
  geom_point(alpha = 0.1) +
  geom_parallel_slopes(aes(fill = condition))
```

`get_correlation` *Get correlation value in a tidy way*

Description

Determine the Pearson correlation coefficient between an outcome variable on the left-hand side of `formula` and one or more explanatory variables on the right-hand side, using pipeable and formula-friendly syntax.

Usage

```
get_correlation(data, formula, na.rm = FALSE, wide = FALSE, quiet = FALSE, ...)
```

Arguments

`data`	a data frame object
`formula`	a formula with the outcome variable on the left and one or more explanatory variables on the right (e.g. `y ~ x` or `y ~ x1 + x2 + x3`).
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.
`wide`	if `TRUE` and the formula has more than one right-hand-side variable, pivot the result wider so that each predictor becomes a column. Has no effect on single-predictor formulas. Default `FALSE`.
`quiet`	if `TRUE`, suppress the informational message that points to `corrr::correlate()` for full pairwise correlation matrices. Default `FALSE`.
`...`	further arguments passed to `stats::cor()`

Details

For a single right-hand side variable, the result is a 1-column tibble (or one row per group if `data` is grouped) named `cor`. For multiple right-hand side variables, the result is a long tibble with one row per (predictor, group) combination by default; pass `wide = TRUE` to pivot it to one column per predictor.

Value

A tibble. For a single right-hand side variable, a 1×1 tibble (or 1 row per group) with column `cor`. For multiple right-hand side variables: long format (default) with columns `predictor` and `cor` (plus any grouping variables), or wide format (`wide = TRUE`) with one column per predictor.

Examples

```
library(moderndive)
library(dplyr)

# Single explanatory variable:
un_member_states_2024 %>%
  get_correlation(formula = life_expectancy_2022 ~ gdp_per_capita, na.rm = TRUE)

# Multiple explanatory variables — long format:
un_member_states_2024 %>%
  get_correlation(
    formula = life_expectancy_2022 ~ gdp_per_capita + fertility_rate_2022 + hdi_2022,
    na.rm   = TRUE,
    quiet   = TRUE
  )

# Multiple explanatory variables — wide format:
un_member_states_2024 %>%
  get_correlation(
    formula = life_expectancy_2022 ~ gdp_per_capita + fertility_rate_2022 + hdi_2022,
    wide    = TRUE,
    na.rm   = TRUE,
    quiet   = TRUE
  )

# Group by one variable:
un_member_states_2024 %>%
  group_by(continent) %>%
  get_correlation(formula = life_expectancy_2022 ~ gdp_per_capita, na.rm = TRUE)

# Group by two variables:
un_member_states_2024 %>%
  group_by(continent, income_group_2024) %>%
  get_correlation(formula = life_expectancy_2022 ~ gdp_per_capita, na.rm = TRUE)
```

`get_regression_points` *Get regression points*

Description

Output information on each point/observation used in an `lm()` regression in "tidy" format. This function is a wrapper function for `broom::augment()` and renames the variables to have more intuitive names.

Usage

```
get_regression_points(
  model,
  digits = 3,
  print = FALSE,
  newdata = NULL,
  ID = NULL
)
```

Arguments

`model`	an `lm()` or `glm()` model object
`digits`	number of digits precision in output table
`print`	If TRUE, return in print format suitable for R Markdown
`newdata`	A new data frame of points/observations to apply `model` to obtain new fitted values and/or predicted values y-hat. Note the format of `newdata` must match the format of the original `data` used to fit `model`.
`ID`	A string indicating which variable in either the original data used to fit `model` or `newdata` should be used as an identification variable to distinguish the observational units in each row. This variable will be the left-most variable in the output data frame. If `ID` is unspecified, a column `ID` with values 1 through the number of rows is returned as the identification variable.

Value

A tibble-formatted regression table of outcome/response variable, all explanatory/predictor variables, the fitted/predicted value, and residual.

See Also

`augment()`, `get_regression_table()`, `get_regression_summaries()`

Examples

```
library(dplyr)
library(moderndive)

# Fit lm() regression:
life_exp_model <- lm(
  life_expectancy_2022 ~ gdp_per_capita,
  data = un_member_states_2024
)

# Get information on all points in regression:
get_regression_points(life_exp_model, ID = "country")

# Create training and test set based on un_member_states_2024:
training_set <- un_member_states_2024 %>%
  sample_frac(0.5)
test_set <- un_member_states_2024 %>%
  anti_join(training_set, by = "country")

# Fit model to training set:
life_exp_model_train <- lm(
  life_expectancy_2022 ~ gdp_per_capita,
  data = training_set
)

# Make predictions on test set:
get_regression_points(life_exp_model_train, newdata = test_set, ID = "country")
```

---

get_regression_summaries

Get regression summary values

Description

Output scalar summary statistics for an `lm()` regression in "tidy" format. This function is a wrapper function for `broom::glance()`.

Usage

```
get_regression_summaries(model, digits = 3, print = FALSE)
```

Arguments

`model`	an `lm()` or `glm()` model object
`digits`	number of digits precision in output table
`print`	If TRUE, return in print format suitable for R Markdown

Value

A single-row tibble with regression summaries. Ex: `r_squared` and `mse`.

See Also

`glance()`, `get_regression_table()`, `get_regression_points()`

Examples

```
library(moderndive)

# Fit lm() regression:
life_exp_model <- lm(
  life_expectancy_2022 ~ gdp_per_capita,
  data = un_member_states_2024
)

# Get regression summaries:
get_regression_summaries(life_exp_model)
```

---

get_regression_table *Get regression table*

Description

Output regression table for an `lm()` or `glm()` model in "tidy" format. This function is a wrapper function for `broom::tidy()` and includes confidence intervals in the output table by default.

Usage

```
get_regression_table(
  model,
  conf.level = 0.95,
  digits = 3,
  print = FALSE,
  default_categorical_levels = FALSE,
  exponentiate = FALSE
)
```

Arguments

`model`	an `lm()` or `glm()` model object
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`digits`	number of digits precision in output table
`print`	If TRUE, return in print format suitable for R Markdown

default_categorical_levels
: If TRUE, do not change the non-baseline categorical variables in the term column. Otherwise non-baseline categorical variables will be displayed in the format "categorical_variable_name-level_name"

exponentiate
: If TRUE, exponentiate the coefficient estimates and confidence intervals. Useful for `glm()` models with log or logit links (returns rate or odds ratios respectively). Default `FALSE`.

Value

A tibble-formatted regression table along with lower and upper end points of all confidence intervals for all parameters `lower_ci` and `upper_ci`; the confidence levels default to 95\

See Also

`tidy()`, `get_regression_points()`, `get_regression_summaries()`

Examples

```
library(moderndive)

# Fit lm() regression:
life_exp_model <- lm(
  life_expectancy_2022 ~ gdp_per_capita,
  data = un_member_states_2024
)

# Get regression table:
get_regression_table(life_exp_model)

# Vary confidence level of confidence intervals
get_regression_table(life_exp_model, conf.level = 0.99)
```

---

gg_parallel_slopes *Plot parallel slopes model*

Description

NOTE: This function is deprecated; please use `geom_parallel_slopes()` instead. Output a visualization of linear regression when you have one numerical and one categorical explanatory/predictor variable: a separate colored regression line for each level of the categorical variable

Usage

```
gg_parallel_slopes(y, num_x, cat_x, data, alpha = 1)
```

Arguments

y	Character string of outcome variable in data
num_x	Character string of numerical explanatory/predictor variable in data
cat_x	Character string of categorical explanatory/predictor variable in data
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
alpha	Transparency of points

Value

A ggplot2::ggplot() object.

See Also

geom_parallel_slopes()

Examples

```
## Not run:
library(ggplot2)
library(dplyr)
library(moderndive)

# log10() transformations
house_prices <- house_prices %>%
  mutate(
    log10_price = log10(price),
    log10_size = log10(sqft_living)
  )

# Output parallel slopes model plot:
gg_parallel_slopes(
  y = "log10_price", num_x = "log10_size", cat_x = "condition",
  data = house_prices, alpha = 0.1
) +
  labs(
    x = "log10 square feet living space", y = "log10 price in USD",
    title = "House prices in Seattle: Parallel slopes model"
  )

# Compare with interaction model plot:
ggplot(house_prices, aes(x = log10_size, y = log10_price, col = condition)) +
  geom_point(alpha = 0.1) +
  geom_smooth(method = "lm", se = FALSE, size = 1) +
  labs(
    x = "log10 square feet living space", y = "log10 price in USD",
    title = "House prices in Seattle: Interaction model"
  )
```

```
## End(Not run)
```

`house_prices` *House Sales in King County, USA*

Description

This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015. This dataset was obtained from Kaggle.com https://www.kaggle.com/datasets/harlfoxem/housesalesprediction/data

Usage

```
house_prices
```

Format

A data frame with 21613 observations on the following 21 variables.

id a notation for a house

date Date house was sold

price Price is prediction target

bedrooms Number of Bedrooms/House

bathrooms Number of bathrooms/bedrooms

sqft_living square footage of the home

sqft_lot square footage of the lot

floors Total floors (levels) in house

waterfront House which has a view to a waterfront

view Has been viewed

condition How good the condition is (Overall)

grade overall grade given to the housing unit, based on King County grading system

sqft_above square footage of house apart from basement

sqft_basement square footage of the basement

yr_built Built Year

yr_renovated Year when house was renovated

zipcode zip code

lat Latitude coordinate

long Longitude coordinate

sqft_living15 Living room area in 2015 (implies– some renovations) This might or might not have affected the lotsize area

sqft_lot15 lotSize area in 2015 (implies– some renovations)

Source

Kaggle https://www.kaggle.com/datasets/harlfoxem/housesalesprediction. Note data is released under a CC0: Public Domain license.

`ipf_lifts` *International Power Lifting Results A subset of international powerlifting results.*

Description

International Power Lifting Results A subset of international powerlifting results.

Usage

```
ipf_lifts
```

Format

A data frame with 41,152 entries, one entry for individual lifter

name Individual lifter name

sex Binary sex (M/F)

event The type of competition that the lifter entered

equipment The equipment category under which the lifts were performed

age The age of the lifter on the start date of the meet

age_class The age class in which the filter falls

division division of competition

bodyweight_kg The recorded bodyweight of the lifter at the time of competition, to two decimal places

weight_class_kg The weight class in which the lifter competed, to two decimal places

best3squat_kg Maximum of the first three successful attempts for the lift

best3bench_kg Maximum of the first three successful attempts for the lift

best3deadlift_kg Maximum of the first three successful attempts for the lift

place The recorded place of the lifter in the given division at the end of the meet

date Date of the event

federation The federation that hosted the meet

meet_name The name of the meet

Source

This data is a subset of the open dataset Open Powerlifting

MA_schools *Massachusetts Public High Schools Data*

Description

Data on Massachusetts public high schools in 2017

Usage

```
MA_schools
```

Format

A data frame of 332 rows representing Massachusetts high schools and 4 variables

school_name High school name.

average_sat_math Average SAT math score. Note 58 of the original 390 values of this variable were missing; these rows were dropped from consideration.

perc_disadvan Percent of the student body that are considered economically disadvantaged.

size Size of school enrollment; small 13-341 students, medium 342-541 students, large 542-4264 students.

Source

The original source of the data are Massachusetts Department of Education reports https://profiles.doe.mass.edu/state_report/, however the data was downloaded from Kaggle at https://www.kaggle.com/datasets/ndalziel/massachusetts-public-schools-data

ma_traffic_2020_vs_2019 *Massachusetts 2020 vs. 2019 Traffic Data Comparison*

Description

This dataset contains information about changes in speed, volume, and accidents of traffic between 2020 and 2019 by community and class of road in Massachusetts.

Usage

```
ma_traffic_2020_vs_2019
```

Format

A data frame of 264 rows each representing a different community in Massachusetts.

community City or Town

functional_class Class or group the road belongs to

change_in_speed Average estimated Speed (mph)

change_in_volume Average traffic

change_in_accidents Average number of accidents

Source

https://massdot-impact-crashes-vhb.opendata.arcgis.com/datasets/MassDOT::2020-vehicle-level-crash-d
explore and the MassDOT Transportation Data Management System (TDMS).

---

mario_kart_auction *Data from Mario Kart Ebay auctions*

Description

Ebay auction data for the Nintendo Wii game Mario Kart.

Usage

```
mario_kart_auction
```

Format

A data frame of 143 auctions.

id Auction ID assigned by Ebay

duration Auction length in days

n_bids Number of bids

cond Game condition, either `new` or `used`

start_pr Price at the start of the auction

ship_pr Shipping price

total_pr Total price, equal to auction price plus shipping price

ship_sp Shipping speed or method

seller_rate Seller's rating on Ebay, equal to the number of positive ratings minus the number of negative ratings

stock_photo Whether the auction photo was a stock photo or not, pictures used in many options were considered stock photos

wheels Number of Wii wheels included in the auction

title The title of the auctions

Source

This data is from https://www.openintro.org/data/index.php?data=mariokart

---

`mass_traffic_2020` *2020 road traffic volume and crash level date for 13 Massachusetts counties*

Description

2020 road traffic volume and crash level date for 13 Massachusetts counties

Usage

```
mass_traffic_2020
```

Format

A data frame of 874 rows representing traffic data at the 874 sites

site_id Site id

county County in which the site is located

community Community in which the site is located

rural_urban Rural (R) or Urban (U)

dir Direction for traffic movement. Either 1-WAY, 2-WAY, EB (eastbound), RAMP or WB (westbound)

functional_class Classification of road. Either Arterial, Collector, Freeway & Expressway, Interstate or Local Road

avg_speed Average traffic speed

total_volume Number of vehicles recorded at each site in 2020

crashes Number of vehicle crashes at each site

nonfatal_injuries Number of non-fatal injuries for all recorded vehicle crashes

fatal_injuries Number of fatal injuries for all recorded vehicle crashes

moderndive *moderndive - Tidyverse-Friendly Introductory Linear Regression*

Description

Datasets and wrapper functions for tidyverse-friendly introductory linear regression, used in "Statistical Inference via Data Science: A ModernDive into R and the tidyverse" available at https://moderndive.com/.

Author(s)

Maintainer: Albert Y. Kim <albert.ys.kim@gmail.com> (ORCID)

Authors:

- Albert Y. Kim <albert.ys.kim@gmail.com> (ORCID)
- Chester Ismay <chester.ismay@gmail.com> (ORCID)

Other contributors:

- Andrew Bray <abray@reed.edu> (ORCID) [contributor]
- Delaney Moran <delaneyemoran@gmail.com> [contributor]
- Evgeni Chasnovski <evgeni.chasnovski@gmail.com> (ORCID) [contributor]
- Will Hopper <wjhopper510@gmail.com> (ORCID) [contributor]
- Benjamin S. Baumer <ben.baumer@gmail.com> (ORCID) [contributor]
- Marium Tapal <mariumtapal@gmail.com> (ORCID) [contributor]
- Wayne Ndlovu <waynedndlovu5@gmail.com> [contributor]
- Catherine Peppers <cpeppers@smith.edu> [contributor]
- Annah Mutaya <annahmutaya18@gmail.com> [contributor]
- Anushree Goswami <anushreeegoswami@gmail.com> [contributor]
- Ziyue Yang <zyang2k@gmail.com> (ORCID) [contributor]
- Clara Li <clarasepianli@gmail.com> (ORCID) [contributor]
- Caroline McKenna <carolinemckenna101@gmail.com> [contributor]
- Catherine Park <jcathyp@gmail.com> (ORCID) [contributor]
- Abbie Benfield <abbidabbers@gmail.com> [contributor]
- Georgia Gans <georgiagans@live.com> [contributor]
- Kacey Jean-Jacques <kjeanjacques@smith.edu> [contributor]
- Swaha Bhattacharya <itsswahabhattacharya@gmail.com> [contributor]
- Vivian Almaraz <viviansofia101@gmail.com> [contributor]
- Elle Jo Whalen <ewhalen@smith.edu> [contributor]
- Jacqueline Chen <jchen76@smith.edu> [contributor]
- Michelle Flesaker <mflesaker@smith.edu> [contributor]

- Irene Foster <ifoster25@smith.edu> [contributor]
- Aushanae Haller <aushanaenhaller@gmail.com> [contributor]
- Benjamin Bruncati <kbruncati@smith.edu> (ORCID) [contributor]
- Quinn White <quinnarlise@gmail.com> (ORCID) [contributor]
- Tianshu Zhang <tzhang26@smith.edu> (ORCID) [contributor]
- Katelyn Diaz <katndiaz@gmail.com> (ORCID) [contributor]
- Rose Porta <rporta@smith.edu> [contributor]
- Renee Wu <rwu30@smith.edu> [contributor]
- Arris Moise <amoise@smith.edu> [contributor]
- Kate Phan <kphan@smith.edu> [contributor]
- Grace Hartley <grace.hartley@gmail.com> [contributor]
- Silas Weden <silasweden@gmail.com> [contributor]
- Emma Vejcik <evejcik@gmail.com> [contributor]
- Nikki Schuldt <nikkischuldt@gmail.com> [contributor]
- Tess Goldmann <tessgoldmann@aol.com> [contributor]
- Hongtong Lin <cccynthialht@gmail.com> [contributor]
- Alejandra Munoz <amunozgarcia@smith.edu> [contributor]
- Elina Gordon-Halpern <egordonhalpern@smith.edu> [contributor]
- Haley Schmidt <heschmidt00@gmail.com> (ORCID) [contributor]

See Also

Useful links:

- https://moderndive.github.io/moderndive/
- https://github.com/moderndive/moderndive/
- Report bugs at https://github.com/moderndive/moderndive/issues

Examples

```
library(moderndive)

# Fit regression model:
life_exp_model <- lm(
  life_expectancy_2022 ~ gdp_per_capita,
  data = un_member_states_2024
)

# Regression tables:
get_regression_table(life_exp_model)

# Information on each point in a regression:
get_regression_points(life_exp_model)
```

```
# Regression summaries
get_regression_summaries(life_exp_model)

# Plotting parallel slopes models
library(ggplot2)
ggplot(evals, aes(x = age, y = score, color = ethnicity)) +
  geom_point() +
  geom_parallel_slopes(se = FALSE)
```

`movies_sample` *Random sample of 68 action and romance movies*

Description

A random sample of 32 action movies and 36 romance movies from https://www.imdb.com/ and their ratings.

Usage

```
movies_sample
```

Format

A data frame of 68 rows movies.

title Movie title

year Year released

rating IMDb rating out of 10 stars

genre Action or Romance

See Also

This data was sampled from the `movies` data frame in the `ggplot2movies` package.

`mythbusters_yawn` *Data from Mythbusters' study on contagiousness of yawning*

Description

From a study on whether yawning is contagious https://www.imdb.com/title/tt0768479/. The data here was derived from the final proportions of yawns given in the show.

Usage

```
mythbusters_yawn
```

Format

A data frame of 50 rows representing each of the 50 participants in the study.

subj integer value corresponding to identifier variable of subject ID

group string of either `"seed"`, participant was shown a yawner, or `"control"`, participant was not shown a yawner

yawn string of either `"yes"`, the participant yawned, or `"no"`, the participant did not yawn

---

`old_faithful_2024` *Old Faithful Eruptions Dataset (2024)*

Description

This dataset contains records of eruptions from the Old Faithful geyser in Yellowstone National Park, recorded in 2024. It includes details such as the eruption ID, date and time of eruption, waiting time between eruptions, webcam availability, and the duration of each eruption.

Usage

```
old_faithful_2024
```

Format

A data frame with 114 rows and 6 variables:

eruption_id `numeric`. A unique identifier for each eruption.

date `date`. The date of the eruption.

time `numeric`. The time of the eruption in HHMM format (e.g., 538 corresponds to 5:38 AM, 1541 corresponds to 3:41 PM).

waiting `numeric`. The waiting time in minutes until the next eruption.

webcam `character`. Indicates whether the eruption was captured on webcam ("Yes" or "No").

duration `numeric`. The duration of the eruption in seconds.

Source

Volunteer information from https://geysertimes.org/retrieve.php

orig_pennies_sample *A random sample of 40 pennies sampled from the* pennies *data frame*

Description

A dataset of 40 pennies to be treated as a random sample with pennies() acting as the population. Data on these pennies were recorded in 2011.

Usage

```
orig_pennies_sample
```

Format

A data frame of 40 rows representing 40 randomly sampled pennies from pennies() and 2 variables

year Year of minting
age_in_2011 Age in 2011

Source

StatCrunch https://statcrunch.pearson.com/app/index.html?dataid=301596

See Also

pennies()

pennies *A population of 800 pennies sampled in 2011*

Description

A dataset of 800 pennies to be treated as a sampling population. Data on these pennies were recorded in 2011.

Usage

```
pennies
```

Format

A data frame of 800 rows representing different pennies and 2 variables

year Year of minting
age_in_2011 Age in 2011

Source

StatCrunch https://statcrunch.pearson.com/app/index.html?dataid=301596

pennies_resamples *Bootstrap resamples of a sample of 50 pennies*

Description

35 bootstrap resamples with replacement of sample of 50 pennies contained in a 50 cent roll from Florence Bank on Friday February 1, 2019 in downtown Northampton, Massachusetts, USA https://www.google.com/maps/place/Florence+Bank+-+Downtown+Northampton/@42.3194512,-72.6337622,16z/data=!4m5!3m4!1s0x89e6d7465ea57a19:0xc901966a46d8ceab!8m2!3d42.3194512!4d-72.6293848. The original sample of 50 pennies is available in pennies_sample() .

Usage

```
pennies_resamples
```

Format

A data frame of 1750 rows representing 35 students' bootstrap resamples of size 50 and 3 variables

replicate ID variable of replicate/resample number.

name Name of student

year Year on resampled penny

See Also

pennies_sample()

pennies_sample *A sample of 50 pennies*

Description

A sample of 50 pennies contained in a 50 cent roll from Florence Bank on Friday February 1, 2019 in downtown Northampton, Massachusetts, USA https://www.google.com/maps/place/Florence+Bank+-+Downtown+Northampton/@42.3194512,-72.6337622,16z/data=!4m5!3m4!1s0x89e6d7465ea57a19:0xc901966a46d8ceab!8m2!3d42.3194512!4d-72.6293848.

Usage

```
pennies_sample
```

Format

A data frame of 50 rows representing 50 sampled pennies and 2 variables

ID Variable used to uniquely identify each penny.

year Year of minting.

Note

The original pennies_sample has been renamed orig_pennies_sample() as of moderndive v0.3.0.

---

plot_3d_regression *Interactive 3D scatterplot with a fitted regression plane*

Description

Build an interactive 3D scatterplot of an outcome against two numeric predictors, overlaid with the fitted plane from lm(formula, data). Returns a plotly htmlwidget that you can render in an R Markdown / Quarto document or in the RStudio Viewer.

Usage

```
plot_3d_regression(data, formula, n = 25)
```

Arguments

data	a data frame containing the three variables in formula.
formula	a formula of the form z ~ x + y: a single outcome on the left-hand side and exactly two predictor variables on the right-hand side. In-formula transformations (e.g. log(z) ~ x + y) are not supported in this version — apply transformations to data beforehand.
n	grid resolution per axis used to draw the regression plane. Default 25.

Details

Requires the suggested package plotly. Install it with install.packages("plotly") if you see an error about it being missing.

Value

A plotly htmlwidget object.

See Also

get_regression_table(), get_regression_points()

Examples

```
## Not run:
  library(moderndive)
  plot_3d_regression(
    un_member_states_2024,
    life_expectancy_2022 ~ gdp_per_capita + fertility_rate_2022
  )

## End(Not run)
```

pop_sd *Calculate Population Standard Deviation*

Description

This function calculates the population standard deviation for a numeric vector.

Usage

```
pop_sd(x)
```

Arguments

x	A numeric vector for which the population standard deviation should be calculated.

Value

A numeric value representing the population standard deviation of the vector.

Examples

```
# Example usage:
library(dplyr)
df <- data.frame(weight = c(2, 4, 6, 8, 10))
df |>
  summarize(population_mean = mean(weight),
            population_sd = pop_sd(weight))
```

promotions *Bank manager recommendations based on (binary) gender*

Description

Data from a 1970's study on whether gender influences hiring recommendations. Originally used in OpenIntro.org.

Usage

```
promotions
```

Format

A data frame with 48 observations on the following 3 variables.

id Identification variable used to distinguish rows.

gender gender (collected as a binary variable at the time of the study): a factor with two levels `male` and `female`

decision a factor with two levels: `promoted` and `not`

Source

Rosen B and Jerdee T. 1974. Influence of sex role stereotypes on personnel decisions. Journal of Applied Psychology 59(1):9-14.

See Also

The data in `promotions` is a slight modification of `openintro::gender_discrimination()`.

`promotions_shuffled` *One permutation/shuffle of promotions*

Description

Shuffled/permuted data from a 1970's study on whether gender influences hiring recommendations.

Usage

```
promotions_shuffled
```

Format

A data frame with 48 observations on the following 3 variables.

id Identification variable used to distinguish rows.

gender shuffled/permuted (binary) gender: a factor with two levels `male` and `female`

decision a factor with two levels: `promoted` and `not`

See Also

`promotions()`.

saratoga_houses *House Prices and Properties in Saratoga, New York*

Description

Random sample of 1057 houses taken from full Saratoga Housing Data (De Veaux)

Usage

```
saratoga_houses
```

Format

A data frame with 1057 observations on the following 8 variables

price price (US dollars)

living_area Living Area (square feet)

bathrooms Number of Bathroom (half bathrooms have no shower or tub)

bedrooms Number of Bedrooms

fireplaces Number of Fireplaces

lot_size Size of Lot (acres)

age Age of House (years)

fireplace Whether the house has a Fireplace

Source

Data on 2006 house prices in Saratoga County, New York, USA. Originally gathered from a publicly shared Google Sheet that is no longer publicly accessible.

spotify_52_original *Spotify 52-Track Sample Dataset*

Description

This dataset contains a sample of 52 tracks from Spotify, focusing on two genres: deep-house and metal. It includes metadata about the tracks, the artists, and an indicator of whether each track is considered popular. This dataset is useful for comparative analysis between genres and for studying the characteristics of popular versus non-popular tracks within these genres.

Usage

```
spotify_52_original
```

Format

A data frame with 52 rows and 6 columns:

track_id `character`. Spotify ID for the track. See: https://developer.spotify.com/documentation/web-api

track_genre `character`. Genre of the track, either "deep-house" or "metal".

artists `character`. Names of the artists associated with the track.

track_name `character`. Name of the track.

popularity `numeric`. Popularity score of the track (0-100). See: https://developer.spotify.com/documentation/web-api/reference/get-track

popular_or_not `character`. Indicates whether the track is considered popular ("popular") or not ("not popular"). Popularity is defined as a score of 50 or higher which corresponds to the 75th percentile of the `popularity` column.

Source

https://developer.spotify.com/documentation/web-api

Examples

```
data(spotify_52_original)
head(spotify_52_original)
```

---

spotify_52_shuffled *Spotify 52-Track Sample Dataset with 'popular or not' shuffled*

Description

This dataset contains a sample of 52 tracks from Spotify, focusing on two genres: deep-house and metal. It includes metadata about the tracks, the artists, and a shuffled indicator of whether each track is considered popular.

Usage

```
spotify_52_shuffled
```

Format

A data frame with 52 rows and 6 columns:

track_id `character`. Spotify ID for the track. See: https://developer.spotify.com/documentation/web-api

track_genre `character`. Genre of the track, either "deep-house" or "metal".

artists `character`. Names of the artists associated with the track.

track_name `character`. Name of the track.

popularity numeric. Popularity score of the track (0-100). See: https://developer.spotify.com/documentation/web-api/reference/get-track

popular_or_not character. A shuffled version of the column of the same name in the spotify_52_original data frame.

Source

https://developer.spotify.com/documentation/web-api

Examples

```
data(spotify_52_shuffled)
head(spotify_52_shuffled)
```

---

spotify_by_genre *Spotify by Genre Dataset*

Description

This dataset contains information on 6,000 tracks from Spotify, categorized by one of six genres. It includes various audio features, metadata about the tracks, and an indicator of popularity. The dataset is useful for analysis of music trends, popularity prediction, and genre-specific characteristics.

Usage

```
spotify_by_genre
```

Format

A data frame with 6,000 rows and 21 columns:

track_id character. Spotify ID for the track. See: https://developer.spotify.com/documentation/web-api

artists character. Names of the artists associated with the track.

album_name character. Name of the album on which the track appears.

track_name character. Name of the track.

popularity numeric. Popularity score of the track (0-100). See: https://developer.spotify.com/documentation/web-api/reference/get-track

duration_ms numeric. Duration of the track in milliseconds.

explicit logical. Whether the track has explicit content.

danceability numeric. Danceability score of the track (0-1). See: https://developer.spotify.com/documentation/web-api/reference/get-audio-features

energy numeric. Energy score of the track (0-1).

key numeric. The key the track is in (0-11 where 0 = C, 1 = C#/Db, etc.).

loudness `numeric`. The loudness of the track in decibels (dB).

mode `numeric`. Modality of the track (0 = minor, 1 = major).

speechiness `numeric`. Speechiness score of the track (0-1).

acousticness `numeric`. Acousticness score of the track (0-1).

instrumentalness `numeric`. Instrumentalness score of the track (0-1).

liveness `numeric`. Liveness score of the track (0-1).

valence `numeric`. Valence score of the track (0-1), indicating the musical positiveness.

tempo `numeric`. Tempo of the track in beats per minute (BPM).

time_signature `numeric`. Time signature of the track (typically 3, 4, or 5).

track_genre `character`. Genre of the track (country, deep-house, dubstep, hip-hop, metal, and rock).

popular_or_not `character`. Indicates whether the track is considered popular ("popular") or not ("not popular"). Popularity is defined as a score of 50 or higher which corresponds to the 75th percentile of the `popularity` column.

Source

https://developer.spotify.com/documentation/web-api

Examples

```
data(spotify_by_genre)
head(spotify_by_genre)
```

---

tactile_prop_red *Tactile sampling from a tub of balls*

Description

Counting the number of red balls in 33 tactile samples of size n = 50 balls from https://github.com/moderndive/moderndive/blob/master/data-raw/sampling_bowl.jpeg

Usage

```
tactile_prop_red
```

Format

A data frame of 33 rows representing different groups of students' samples of size n = 50 and 4 variables

group Group members

replicate Replicate number

red_balls Number of red balls sampled out of 50

prop_red Proportion red balls out of 50

See Also

bowl()

---

tidy_summary — *This function calculates the five-number summary (minimum, first quartile, median, third quartile, maximum) for specified numeric columns in a data frame and returns the results in a long format. It also handles categorical, factor, and logical columns by counting the occurrences of each level or value, and includes the results in the summary. The* type *column indicates whether the data is numeric, character, factor, or logical.*

Description

This function calculates the five-number summary (minimum, first quartile, median, third quartile, maximum) for specified numeric columns in a data frame and returns the results in a long format. It also handles categorical, factor, and logical columns by counting the occurrences of each level or value, and includes the results in the summary. The type column indicates whether the data is numeric, character, factor, or logical.

Usage

```
tidy_summary(df, columns = names(df), ...)
```

Arguments

df	A data frame containing the data. The data frame must have at least one row.
columns	Unquoted column names or tidyselect helpers specifying the columns for which to calculate the summary. Defaults to call columns in the inputted data frame.
...	Additional arguments passed to the min, quantile, median, and max functions, such as na.rm.

Value

A tibble in long format with columns:

column The name of the column.

n The number of non-missing values in the column for numeric variables and the number of non-missing values in the group for categorical, factor, and logical columns.

group The group level or value for categorical, factor, and logical columns.

type The type of data in the column (numeric, character, factor, or logical).

min The minimum value (for numeric columns).

Q1 The first quartile (for numeric columns).

mean The mean value (for numeric columns).

median The median value (for numeric columns).

Q3 The third quartile (for numeric columns).

max The maximum value (for numeric columns).

sd The standard deviation (for numeric columns).

Examples

```
# Example usage with a simple data frame
df <- tibble::tibble(
  category = factor(c("A", "B", "A", "C")),
  int_values = c(10, 15, 7, 8),
  num_values = c(8.2, 0.3, -2.1, 5.5),
  one_missing_value = c(NA, 1, 2, 3),
  flag = c(TRUE, FALSE, TRUE, TRUE)
)

# Specify columns
tidy_summary(df, columns = c(category, int_values, num_values, flag))

# Defaults to full data frame (note an error will be given without
# specifying `na.rm = TRUE` since `one_missing_value` has an `NA`)
tidy_summary(df, na.rm = TRUE)

# Example with additional arguments for quantile functions
tidy_summary(df, columns = c(one_missing_value), na.rm = TRUE)
```

---

un_member_states_2024 *UN Member States 2024 Dataset*

Description

This dataset contains information on 193 United Nations member states as of 2024. It includes various attributes such as country names, ISO codes, official state names, geographic and demographic data, economic indicators, and participation in the Olympic Games. The data is designed for use in statistical analysis, data visualization, and educational purposes.

Usage

```
un_member_states_2024
```

Format

A data frame with 193 rows and 39 columns:

country `character`. Name of the country.

iso `character`. ISO 3166-1 alpha-3 country code. See: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3

official_state_name character. Official name of the country. See: https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_and_their_capitals_in_native_languages

continent factor. Continent where the country is located. See: https://en.wikipedia.org/wiki/Continent

region character. Specific region within the continent.

capital_city character. Name of the capital city. See: https://en.wikipedia.org/wiki/List_of_national_capitals_by_population

capital_population numeric. Population of the capital city.

capital_perc_of_country numeric. Percentage of the country's population living in the capital.

capital_data_year integer. Year the capital population data was collected.

gdp_per_capita numeric. GDP per capita in USD. See: https://data.worldbank.org/indicator/NY.GDP.PCAP.CD

gdp_per_capita_year numeric. Year the GDP per capita data was collected.

summers_competed_in numeric. Number of times the country has competed in the Summer Olympics

summer_golds integer. Number of gold medals won in the Summer Olympics.

summer_silvers integer. Number of silver medals won in the Summer Olympics.

summer_bronzes integer. Number of bronze medals won in the Summer Olympics.

summer_total integer. Total number of medals won in the Summer Olympics.

winters_competed_in integer. Number of times the country has competed in the Winter Olympics

winter_golds integer. Number of gold medals won in the Winter Olympics.

winter_silvers integer. Number of silver medals won in the Winter Olympics.

winter_bronzes integer. Number of bronze medals won in the Winter Olympics.

winter_total integer. Total number of medals won in the Winter Olympics.

combined_competed_ins integer. Total number of times the country has competed in both Summer and Winter Olympics. See: https://en.wikipedia.org/wiki/All-time_Olympic_Games_medal_table

combined_golds integer. Total number of gold medals won in both Summer and Winter Olympics.

combined_silvers integer. Total number of silver medals won in both Summer and Winter Olympics.

combined_bronzes integer. Total number of bronze medals won in both Summer and Winter Olympics.

combined_total integer. Total number of medals won in both Summer and Winter Olympics.

driving_side character. Indicates whether the country drives on the left or right side of the road. See: https://en.wikipedia.org/wiki/Left-_and_right-hand_traffic

obesity_rate_2024 numeric. Percentage of the population classified as obese in 2024. See: https://en.wikipedia.org/wiki/List_of_countries_by_obesity_rate

obesity_rate_2016 numeric. Percentage of the population classified as obese in 2016.

has_nuclear_weapons_2024 logical. Indicates whether the country has nuclear weapons as of 2024. See: https://en.wikipedia.org/wiki/List_of_states_with_nuclear_weapons

population_2024 numeric. Population of the country in 2024. See: https://data.worldbank.org/indicator/SP.POP.TOTL

area_in_square_km numeric. Area of the country in square kilometers. See: https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_area

area_in_square_miles numeric. Area of the country in square miles.

population_density_in_square_km numeric. Population density in square kilometers.

population_density_in_square_miles numeric. Population density in square miles.

income_group_2024 factor. World Bank income group classification in 2024. See: https://data.worldbank.org/indicator/NY.GNP.PCAP.CD

life_expectancy_2022 numeric. Life expectancy at birth in 2022. See: https://en.wikipedia.org/wiki/List_of_countries_by_life_expectancy

fertility_rate_2022 numeric. Fertility rate in 2022 (average number of children per woman). See: https://en.wikipedia.org/wiki/List_of_countries_by_total_fertility_rate

hdi_2022 numeric. Human Development Index in 2022. See: https://en.wikipedia.org/wiki/List_of_countries_by_Human_Development_Index

Examples

```
data(un_member_states_2024)
head(un_member_states_2024)
```

View *View a data object inline when not in an interactive R session*

Description

A drop-in replacement for utils::View(). In an interactive R session it behaves identically to typing View() without moderndive attached, opening the RStudio / R Console data viewer. In a non-interactive context such as an R Markdown or Quarto document, where utils::View() typically errors, it instead renders an interactive DT::datatable() inline so users can still inspect the data without their document failing to render.

Usage

```
View(x, title = NULL, n = 1000L, full = FALSE, seed = NULL, quiet = FALSE)
```

Arguments

x	an R object which can be coerced to a data frame.
title	title for the viewer window. Defaults to the name of x prefixed by Data: , matching the convention used by utils::View().
n	maximum number of rows to show in the inline DT::datatable(). When x has more than n rows and full = FALSE, a random sample of n rows is shown instead of the whole data frame. Default 1000.

full	if TRUE, show every row of x in the inline table instead of sampling (may be slow, and DT may warn, for very large data). Default FALSE.
seed	optional integer seed making the random sample reproducible. The global random-number stream is left unchanged regardless of whether seed is supplied. Default NULL (a fresh sample each call).
quiet	if TRUE, suppress the message announcing that a random sample is being shown. Default FALSE.

Details

For large data frames a client-side DT::datatable() is slow and warns that the data is too big. So in non-interactive contexts, when x has more than n rows and full = FALSE, View() shows a **random sample** of n rows and emits a message saying so. This keeps "just look at the data" fast for beginners while making the size limitation explicit. The interactive utils::View() path is unaffected (RStudio's viewer handles large data natively), so n / full / seed / quiet apply only to the inline DT rendering.

Value

In an interactive session, invisibly returns NULL (called for the side effect of displaying x in a viewer). In a non-interactive session, returns a DT::datatable() htmlwidget – except inside webR (where pandoc is unavailable), where it renders a self-contained static HTML table through the browser viewer and invisibly returns NULL.

See Also

utils::View(), DT::datatable()

Examples

```
## Not run:
  # In an interactive R session, behaves like utils::View():
  View(un_member_states_2024)

  # In an R Markdown or Quarto document, renders an interactive
  # DT::datatable() inline instead of erroring:
  View(un_member_states_2024)

  # For large data a random sample is shown by default; override with:
  View(some_big_data, full = TRUE)   # all rows (may be slow)
  View(some_big_data, n = 500)       # sample 500 rows instead of 1000
  View(some_big_data, seed = 42)     # reproducible random sample
  View(some_big_data, quiet = TRUE)  # no "random sample" message

## End(Not run)
```

Index

∗ **datasets**
alaska_flights, 3
almonds_bowl, 4
almonds_sample, 5
almonds_sample_100, 5
amazon_books, 6
avocados, 6
babies, 7
bowl, 8
bowl_sample_1, 9
bowl_samples, 9
coffee_quality, 10
coffee_ratings, 11
DD_vs_SB, 13
early_january_2023_weather, 14
early_january_weather, 15
envoy_flights, 16
ev_charging, 17
evals, 18
house_prices, 31
ipf_lifts, 32
MA_schools, 33
ma_traffic_2020_vs_2019, 33
mario_kart_auction, 34
mass_traffic_2020, 35
movies_sample, 38
mythbusters_yawn, 38
old_faithful_2024, 39
orig_pennies_sample, 40
pennies, 40
pennies_resamples, 41
pennies_sample, 41
promotions, 43
promotions_shuffled, 44
saratoga_houses, 45
spotify_52_original, 45
spotify_52_shuffled, 46
spotify_by_genre, 47
tactile_prop_red, 48
un_member_states_2024, 50

aes(), *19*, *22*
alaska_flights, 3
almonds_bowl, 4
almonds_sample, 5
almonds_sample_100, 5
amazon_books, 6
annotation_borders(), *20*, *23*
as.data.frame, *30*
augment(), *26*
avocados, 6

babies, 7
bowl, 8
bowl(), *9*, *10*, *49*
bowl_sample_1, 9
bowl_samples, 9

coffee_quality, 10
coffee_ratings, 11

DD_vs_SB, 13
DT::datatable(), *52*, *53*

early_january_2023_weather, 14
early_january_weather, 15
envoy_flights, 16
ev_charging, 17
evals, 18

fortify(), *19*, *22*

geom_categorical_model, 19
geom_categorical_model(), *23*
geom_parallel_slopes, 21
geom_parallel_slopes(), *19*, *21*, *29*, *30*
get_correlation, 24
get_regression_points, 26
get_regression_points(), *28*, *29*, *42*
get_regression_summaries, 27

get_regression_summaries(), *26*, *29*
get_regression_table, 28
get_regression_table(), *26*, *28*, *42*
gg_parallel_slopes, 29
ggplot(), *19*, *22*
ggplot2::ggplot(), *30*
glance(), *28*

house_prices, 31

ipf_lifts, 32

key glyphs, *20*, *23*

layer position, *20*, *22*
layer(), *20*, *22, 23*

MA_schools, 33
ma_traffic_2020_vs_2019, 33
mario_kart_auction, 34
mass_traffic_2020, 35
moderndive, 36
moderndive-package *(*moderndive*)*, 36
movies_sample, 38
mythbusters_yawn, 38

nycflights13::airlines, *4*
nycflights13::airports, *4*
nycflights13::flights, *4*, *15*
nycflights13::planes, *4*
nycflights13::weather, *4*, *15*
nycflights23::airlines, *16*
nycflights23::airports, *16*
nycflights23::flights, *14*, *16*
nycflights23::planes, *16*
nycflights23::weather, *14*, *16*

old_faithful_2024, 39
openintro::evals(), *19*
openintro::gender_discrimination(), *44*
orig_pennies_sample, 40
orig_pennies_sample(), *42*

pennies, 40
pennies(), *40*
pennies_resamples, 41
pennies_sample, 41
pennies_sample(), *41*
plot_3d_regression, 42
plotly, *42*
pop_sd, 43
promotions, 43
promotions(), *44*
promotions_shuffled, 44

saratoga_houses, 45
spotify_52_original, 45
spotify_52_shuffled, 46
spotify_by_genre, 47
stats::cor(), *24*

tactile_prop_red, 48
tidy(), *29*
tidy_summary, 49

un_member_states_2024, 50
utils::View(), *52, 53*

View, 52