Package: modeldatatoo (via r-universe)

June 28, 2024

Title More Data Sets Useful for Modeling Examples

Version 0.3.0.9000

Description More data sets used for demonstrating or testing model-related packages are contained in this package. The data sets are downloaded and cached, allowing for more and bigger data sets.

License MIT + file LICENSE

URL https://github.com/tidymodels/modeldatatoo, https://modeldatatoo.tidymodels.org/

BugReports https://github.com/tidymodels/modeldatatoo/issues

Depends R (>= 3.6)

Imports pins

Suggests covr, rstudioapi, testthat (>= 3.0.0)

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

LazyData true

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

RemoteUrl https://github.com/tidymodels/modeldatatoo

RemoteRef HEAD

RemoteSha 9bdbe3c06d747299cf9691c5d6f302273fa5f870

Contents

`building_complaints` *NYC Building Complaints*

Description

A subset of the complaints received by the Department of Buildings (DOB) in New York City, USA.

Usage

```
building_complaints
```

Format

`building_complaints`**:**

A data frame with 4,234 rows and 11 columns:

days_to_disposition Days to disposition of the complaint

status Status of the complaint

year_entered Year the complaint was entered

latitude, longitude Geographic coordinates

borough Borough

special_district Special district

unit Unit dispositioning the complaint

community_board Community board. 3-digit identifier: Borough code = first position, last 2 = community board

complaint_category Complaint category

complaint_priority Complaint priority

Source

https://data.cityofnewyork.us/Housing-Development/DOB-Complaints-Received/eabe-havv/about_data

`data_animals` *animals data set*

Description

Data set with characteristics of many animals, including the field `text` which is a long-form description of the animal.

Usage

```
data_animals(...)
```

Arguments

`...`	Arguments passed to `pins::pin_read()`.

Details

This data set contains quite a bit of missing data and malformed fields.

Value

tibble

tibble print

```
data_animals()
#> # A tibble: 610 x 48
#>   text   colour lifespan weight kingdom class phylum diet  conservation_status
#>    <chr>   <chr>  <chr>    <chr>  <chr>   <chr> <chr>  <chr> <chr>
#>  1 "Aardv~ Brown~ 23 years 60kg ~ Animal~ Mamm~ Chord~ Omni~ Least Concern
#>  2 "Abyss~ Fawn,~ <NA>     <NA>   <NA>    <NA>  <NA>   <NA>  <NA>
#>  3 "Adeli~ Black~ 10 - 20~ 3kg -~ Animal~ Aves  Chord~ Carn~ Least Concern
#>  4 "Affen~ Black~ <NA>     <NA>   <NA>    <NA>  <NA>   <NA>  <NA>
#>  5 "Afgha~ Black~ <NA>     <NA>   <NA>    <NA>  <NA>   <NA>  <NA>
#>  6 "Afric~ Grey,~ 60 - 70~ 3,600~ Animal~ Mamm~ Chord~ Herb~ Threatened
#>  7 "Afric~ Black~ 15 - 20~ 1.4kg~ Animal~ Mamm~ Chord~ Omni~ Least Concern
#>  8 "Afric~ Brown~ 8 - 15 ~ 25g -~ Animal~ Amph~ Chord~ Carn~ Least Concern
#>  9 "Afric~ Grey,~ 60 - 70~ 900kg~ Animal~ Mamm~ Chord~ Herb~ Endangered
#> 10 "Afric~ Black~ 15 - 20~ 1.4kg~ Animal~ Mamm~ Chord~ Omni~ Least Concern
#> # i 600 more rows
#> # i 39 more variables: order <chr>, scientific_name <chr>, skin_type <chr>,
#> #   habitat <chr>, predators <chr>, family <chr>, lifestyle <chr>,
#> #   average_litter_size <chr>, genus <chr>, top_speed <chr>,
#> #   favourite_food <chr>, main_prey <chr>, type <chr>, common_name <chr>,
#> #   group <chr>, size <chr>, distinctive_features <chr>, size_l <chr>,
#> #   origin <chr>, special_features <chr>, location <chr>, ...
```

glimpse()

```
tibble::glimpse(data_animals())
#> Rows: 610
#> Columns: 48
#> $ text                    <chr> "Aardvark Classification and Evolution\nAard~
#> $ colour                  <chr> "Brown, grey, yellow", "Fawn, Red, Blue, Gre~
#> $ lifespan                <chr> "23 years", NA, "10 - 20 years", NA, NA, "60~
#> $ weight                  <chr> "60kg - 80kg (130lbs - 180lbs)", NA, "3kg - ~
#> $ kingdom                 <chr> "Animalia", NA, "Animalia", NA, NA, "Animali~
#> $ class                   <chr> "Mammalia", NA, "Aves", NA, NA, "Mammalia", ~
#> $ phylum                  <chr> "Chordata", NA, "Chordata", NA, NA, "Chordat~
#> $ diet                    <chr> "Omnivore", NA, "Carnivore", NA, NA, "Herbiv~
#> $ conservation_status     <chr> "Least Concern", NA, "Least Concern", NA, NA~
#> $ order                   <chr> "Tubulidentata", NA, "Sphenisciformes", NA, ~
#> $ scientific_name         <chr> "Orycteropus afer", NA, "Pygoscelis adeliae"~
#> $ skin_type               <chr> "Hair", NA, "Feathers", NA, NA, "Leather", "~
#> $ habitat                 <chr> "Sandy and clay soil", NA, "Antarctic land a~
#> $ predators               <chr> "Lions, Leopards, Hyenas", NA, "Leopard Seal~
#> $ family                  <chr> "Orycteropodidae", NA, "Spheniscidae", NA, N~
#> $ lifestyle               <chr> "Nocturnal", NA, "Diurnal", NA, NA, "Diurnal~
#> $ average_litter_size     <chr> "1", "6", NA, "3", "7", "1", "3", NA, "1", "~
#> $ genus                   <chr> "Orycteropus", NA, "Pygoscelis", NA, NA, "Lo~
#> $ top_speed               <chr> "40kph (25mph)", NA, "72kph (45mph)", NA, NA~
#> $ favourite_food          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ main_prey               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ type                    <chr> NA, "Shorthair", NA, "Terrier", "Hound", NA,~
#> $ common_name             <chr> "Aardvark", "Abyssinian", "Adelie Penguin", ~
#> $ group                   <chr> "Mammal", "Cat", "Bird", "Dog", "Dog", "Mamm~
#> $ size                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ distinctive_features    <chr> NA, "Silky fur and almond shaped eyes", NA, ~
#> $ size_l                  <chr> "1.05m - 2.20m (3.4ft - 7.3ft)", NA, NA, NA,~
#> $ origin                  <chr> NA, "Egypt", NA, "Germany", "Afghanistan", N~
#> $ special_features        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ location                <chr> "Sub-Saharan Africa", NA, "Coastal Antarctic~
#> $ number_of_species       <chr> "18", NA, "1", NA, NA, "1", "1", "1", "1", "~
#> $ average_clutch_size     <chr> NA, NA, "2", NA, NA, NA, NA, NA, NA, NA, "2"~
#> $ size_h                  <chr> NA, NA, "40cm - 75cm (16in - 30in)", NA, NA,~
#> $ group_behaviour         <chr> "Solitary", NA, "Colony", NA, NA, "Herd", "S~
#> $ fun_fact                <chr> "Can move up to 2ft of soil in just 15 secon~
#> $ age_of_sexual_maturity  <chr> "2 years", NA, "2 - 3 years", NA, NA, "11 - ~
#> $ name_of_young           <chr> "Cub", NA, "Chicks", NA, NA, "Calf", "Pup", ~
#> $ prey                    <chr> "Termites, Ants", NA, "Krill, Fish, Squid", ~
#> $ estimated_population_size <chr> "Unknown", NA, "5 million", NA, NA, "300,000~
#> $ biggest_threat          <chr> "Habitat loss", NA, "Rapid ice melt", NA, NA~
#> $ average_lifespan        <chr> NA, "15 years", NA, "12 years", "14 years", ~
#> $ most_distinctive_feature <chr> "Long, sticky tongue and rabbit-like ears", ~
#> $ other_name_s            <chr> "Antbear, Earth Pig", NA, NA, NA, NA, "Afric~
#> $ gestation_period        <chr> "7 months", NA, NA, NA, NA, "20 - 24 months"~
```

```
#> $ age_of_weaning          <chr> "3 months", NA, NA, NA, NA, "6 - 18 months",~
#> $ average_weight          <chr> NA, "4.5kg (10lbs)", NA, "3.6kg (8lbs)", "27~
#> $ temperament             <chr> NA, "Intelligent and curious", NA, "Alert an~
#> $ wingspan                <chr> NA, NA, "35cm - 70cm (14in - 27.5in)", NA, N~
```

Source

https://github.com/emilhvitfeldt/animals

Examples

```
data_animals()
```

`data_building_complaints` *NYC Building Complaints*

Description

A subset of the complaints received by the Department of Buildings (DOB) in New York City, USA.

Usage

```
data_building_complaints(...)
```

Arguments

`...`	Arguments passed to `pins::pin_read()`.

Details

A data frame with 4,234 rows and 11 columns:

days_to_disposition Days to disposition of the complaint

status Status of the complaint

year_entered Year the complaint was entered

latitude, longitude Geographic coordinates

borough Borough

special_district Special district

unit Unit dispositioning the complaint

community_board Community board. 3-digit identifier: Borough code = first position, last 2 = community board

complaint_category Complaint category

complaint_priority Complaint priority

Value

tibble

tibble print

```
data_building_complaints()
#> # A tibble: 4,234 x 11
#>    days_to_disposition status year_entered latitude longitude borough
#>                  <dbl> <chr>  <fct>           <dbl>     <dbl> <fct>
#>  1                  72 ACTIVE 2023             40.7     -74.0 Brooklyn
#>  2                   1 ACTIVE 2023             40.6     -74.0 Brooklyn
#>  3                  41 ACTIVE 2023             40.7     -73.9 Queens
#>  4                  45 ACTIVE 2023             40.7     -73.8 Queens
#>  5                  16 ACTIVE 2023             40.6     -74.0 Brooklyn
#>  6                  62 ACTIVE 2023             40.7     -73.8 Queens
#>  7                  56 ACTIVE 2023             40.7     -74.0 Brooklyn
#>  8                  11 ACTIVE 2023             40.7     -74.0 Brooklyn
#>  9                  35 ACTIVE 2023             40.7     -73.8 Queens
#> 10                  38 ACTIVE 2023             40.7     -73.9 Queens
#> # i 4,224 more rows
#> # i 5 more variables: special_district <fct>, unit <fct>,
#> #   community_board <fct>, complaint_category <fct>, complaint_priority <fct>
```

glimpse()

```
tibble::glimpse(data_building_complaints())
#> Rows: 4,234
#> Columns: 11
#> $ days_to_disposition <dbl> 72, 1, 41, 45, 16, 62, 56, 11, 35, 38, 39, 106, 1,~
#> $ status              <chr> "ACTIVE", "ACTIVE", "ACTIVE", "ACTIVE", "ACTIVE", ~
#> $ year_entered         <fct> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 20~
#> $ latitude            <dbl> 40.66173, 40.57668, 40.73242, 40.68245, 40.63156, ~
#> $ longitude           <dbl> -73.98297, -74.00453, -73.87630, -73.79367, -73.99~
#> $ borough             <fct> Brooklyn, Brooklyn, Queens, Queens, Brooklyn, Quee~
#> $ special_district     <fct> None, None, None, None, None, None, None, None, No~
#> $ unit                <fct> Q-L, Q-L, SPOPS, Q-L, BKLYN, Q-L, Q-L, SPOPS, Q-L,~
#> $ community_board      <fct> 307, 313, 404, 412, 312, 406, 306, 306, 409, 404, ~
#> $ complaint_category   <fct> 45, 45, 49, 45, 31, 45, 45, 49, 45, 45, 45, 4A, 31~
#> $ complaint_priority   <fct> B, B, C, B, C, B, B, C, B, B, B, B, C, C, B, B, B,~
```

Source

https://data.cityofnewyork.us/Housing-Development/DOB-Complaints-Received/eabe-havv/about_data

Examples

```
data_building_complaints()
```

`data_chimiometrie_2019` *Chimiometrie 2019 Data Set*

Description

Larsen and Clemmensen (2019) state: "This data set was published as the challenge at the Chimiometrie 2019 conference held in Montpellier and is available at the conference homepage. The data consist of 6915 training spectra and 600 test spectra measured at 550 (unknown) wavelengths. The target was the amount of soy oil (0-5.5%), ucerne (0-40%) and barley (0-52%) in a mixture."

The test set included a distribution shift due to the use of a different instrument and this competition was designed to measure how models might be made to be resistant to such a difference. However, since there are no test set outcomes, we only include the training set here.

There are 6,915 rows and 553 columns. The columns whose names start with `wvlgth_` are the spectral values with the numbers in the column names referring to the order (as opposed to the wavenumber). Fernández Pierna (2020) suggest that the wavelengths range from 1300 2nm to 2398 2nm.

The three outcome columns are `"soy_oil"`, `"lucerne"`, and `"barley"`.

Usage

```
data_chimiometrie_2019(...)
```

Arguments

`...`	Arguments passed to `pins::pin_read()`.

Value

A tibble.

glimpse()

```
tibble::glimpse(data_chimiometrie_2019()[, 1:10])
#> Rows: 6,915
#> Columns: 10
#> $ soy_oil    <dbl> 2.1, 2.1, 2.1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,~
#> $ lucerne    <dbl> 23.5712, 23.5712, 23.5712, 25.0000, 25.0000, 25.0000, 25.00~
#> $ barley     <dbl> 0, 0, 0, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,~
#> $ wvlgth_001 <dbl> 0.2076995, 0.2064382, 0.2070081, 0.2057694, 0.2005429, 0.20~
#> $ wvlgth_002 <dbl> 0.2074427, 0.2062003, 0.2067785, 0.2055505, 0.2003232, 0.20~
#> $ wvlgth_003 <dbl> 0.2072212, 0.2059973, 0.2065901, 0.2053678, 0.2001469, 0.20~
#> $ wvlgth_004 <dbl> 0.2070317, 0.2058266, 0.2064396, 0.2052174, 0.2000069, 0.20~
#> $ wvlgth_005 <dbl> 0.2068830, 0.2056964, 0.2063288, 0.2051110, 0.1999092, 0.20~
#> $ wvlgth_006 <dbl> 0.2067773, 0.2056115, 0.2062618, 0.2050571, 0.1998616, 0.20~
#> $ wvlgth_007 <dbl> 0.2067083, 0.2055686, 0.2062386, 0.2050495, 0.1998592, 0.20~
```

License:

No license was given for the data.

Source

https://chemom2019.sciencesconf.org/resource/page/id/13.html

References

J. Larsen and L. Clemmensen (2019) "Deep learning for Chemometric and non-translational data," *arXiv.org*, https://arxiv.org/abs/1910.00391.

J.A. Fernández Pierna, A. Laborde, L. Lakhal, M. Lesnoff, M. Martin, Y. Roggo, and P. Dardenne (2020) "The applicability of vibrational spectroscopy and multivariate analysis for the characterization of animal feed where the reference values do not follow a normal distribution: A new chemometric challenge posed at the 'Chimiométrie 2019' congress," *Chemometrics and Intelligent Laboratory Systems*, vol 202, p. 104026. doi:10.1016/j.chemolab.2020.104026

Examples

```
data_chimiometrie_2019()
```

---

data_detectors *Predictions from GPT Detectors*

Description

Data derived from the paper *GPT detectors are biased against non-native English writers.* The study authors carried out a series of experiments passing a number of essays to different GPT detection models. Juxtaposing detector predictions for papers written by native and non-native English writers, the authors argue that GPT detectors disproportionately classify real writing from non-native English writers as AI-generated.

Usage

```
data_detectors(...)
```

Arguments

...	Arguments passed to pins::pin_read().

Details

A data frame with 6,185 rows and 9 columns:

kind Whether the essay was written by a `"Human"` or `"AI"`.

.pred_AI The class probability from the GPT detector that the inputted text was written by AI.

.pred_class The uncalibrated class prediction, encoded as `if_else(.pred_AI > .5, "AI", "Human")`

detector The name of the detector used to generate the predictions.

native For essays written by humans, whether the essay was written by a native English writer or not. These categorizations are coarse; values of `"Yes"` may actually be written by people who do not write with English natively. `NA` indicates that the text was not written by a human.

name A label for the experiment that the predictions were generated from.

model For essays that were written by AI, the name of the model that generated the essay.

document_id A unique identifier for the supplied essay. Some essays were supplied to multiple detectors. Note that some essays are AI-revised derivatives of others.

prompt For essays that were written by AI, a descriptor for the form of "prompt engineering" passed to the model.

Value

tibble

tibble print

```
data_detectors()
#> # A tibble: 6,185 x 9
#>   kind  .pred_AI .pred_class detector    native name  model document_id prompt
#>    <fct>    <dbl> <fct>      <chr>       <chr>  <chr> <chr>       <dbl> <chr>
#>  1 Human 1.00     AI          Sapling      No     Real~ Human         497 <NA>
#>  2 Human 0.828    AI          Crossplag    No     Real~ Human         278 <NA>
#>  3 Human 0.000214 Human       Crossplag    Yes    Real~ Human         294 <NA>
#>  4 AI    0        Human       ZeroGPT      <NA>   Fake~ GPT3          671 Plain
#>  5 AI    0.00178  Human       Originality~ <NA>   Fake~ GPT4          717 Eleva~
#>  6 Human 0.000178 Human       HFOpenAI     Yes    Real~ Human         855 <NA>
#>  7 AI    0.992    AI          HFOpenAI     <NA>   Fake~ GPT3          533 Plain
#>  8 AI    0.0226   Human       Crossplag    <NA>   Fake~ GPT4          484 Eleva~
#>  9 Human 0        Human       ZeroGPT      Yes    Real~ Human         781 <NA>
#> 10 Human 1.00     AI          Sapling      No     Real~ Human         460 <NA>
#> # i 6,175 more rows
```

glimpse()

```
tibble::glimpse(data_detectors())
#> Rows: 6,185
#> Columns: 9
#> $ kind       <fct> Human, Human, Human, AI, AI, Human, AI, AI, Human, Human, ~
#> $ .pred_AI   <dbl> 9.999942e-01, 8.281448e-01, 2.137465e-04, 0.000000e+00, 1.~
```

```
#> $ .pred_class <fct> AI, AI, Human, Human, Human, Human, AI, Human, Human, AI, ~
#> $ detector    <chr> "Sapling", "Crossplag", "Crossplag", "ZeroGPT", "Originali~
#> $ native      <chr> "No", "No", "Yes", NA, NA, "Yes", NA, NA, "Yes", "No", NA,~
#> $ name        <chr> "Real TOEFL", "Real TOEFL", "Real College Essays", "Fake C~
#> $ model       <chr> "Human", "Human", "Human", "GPT3", "GPT4", "Human", "GPT3"~
#> $ document_id <dbl> 497, 278, 294, 671, 717, 855, 533, 484, 781, 460, 591, 11,~
#> $ prompt      <chr> NA, NA, NA, "Plain", "Elevate using technical", NA, "Plain~
```

Source

https://simonpcouch.github.io/detectors/ doi:10.1016/j.patter.2023.100779

Examples

```
data_detectors()
```

---

data_elevators *elevators data set*

Description

A data set containing information of a subset of the elevators in NYC. The data set has been filtered to contain active elevators with non-missing speed.

Usage

```
data_elevators(...)
```

Arguments

...	Arguments passed to pins::pin_read().

Details

device_number Unique identify number for the elevator

bin Building Identification Number

borough Regional subdivisions of NYC. One of "Manhattan", "Bronx", "Brooklyn", "Queens", or "Staten Island"

tax_block Id for tax block. Smaller than borough

tax_lot Id for tax block. Smaller than tax_block

house_number House number, very poorly parsed. Use with caution

street_name Street name, very poorly parsed. Use with caution

zip_code Zip code, formatted to 5 digits. 0 and 99999 are marked as NA

device_type Type of device. Most common type is "Passenger Elevator"

lastper_insp_date Date, refers to the last periodic inspection by the Department of Buildings. These dates will no longer be accurate, as they were collected by November 2015

approval_date Date of approval for elevator

manufacturer Name of manufacturer, poorly cleaned. Most assigned NA

travel_distance Distance travelled, not cleaned. Mixed formats

speed_fpm Speed in feet/minute

capacity_lbs Capacity in lbs

car_buffer_type Buffer type. A buffer is a device designed to stop a descending car or counterweight beyond its normal limit and to soften the force with which the elevator runs into the pit during an emergency. Takes values "Oil", "Spring", and NA

governor_type Governor type, An overspeed governor is an elevator device which acts as a stopping mechanism in case the elevator runs beyond its rated speed

machine_type Machine type, labels unknown.

safety_type Safety type, labels unknown.

mode_operation Operation mode, labels unknown.

floor_from Lowest floor, not cleaned. Mixed formats

floor_to Highest floor, not cleaned. Mixed formats

latitude Latitude of elevator

longitude Longitude of elevator

elevators_per_building number of elevators in building ...

Value

tibble

tibble print

```
data_elevators()
#> # A tibble: 35,042 x 25
#>   device_number bin    tax_block tax_lot house_number street_name    zip_code
#>    <chr>         <chr>   <chr>     <chr>   <chr>        <chr>           <chr>
#>  1 1D10028       1024795 1021      26      1614         BROADWAY        10019
#>  2 1D10094       1041822 1392      25      53           E 77TH ST       10021
#>  3 1D10097       1038223 1323      1       201          E 49 ST         10017
#>  4 1D10146       1080443 1274      6       40           CENTRAL PARK S~ <NA>
#>  5 1D10200       1085777 1074      24      651          TENTH AVENUE    <NA>
#>  6 1D10301       1002075 181       16      179          FRANKLIN STREET 10013
#>  7 1D10302       1010518 606       4       121          WEST 10 STREET  10011
#>  8 1D10303       1085955 1329      1       915          3 AVENUE        10022
#>  9 1D10304       1044058 1430      5       220          E. 76 ST        10021
#> 10 1D10305       1087468 1951      4       133          MORNINGSIDE AV~ <NA>
#> # i 35,032 more rows
#> # i 18 more variables: borough <fct>, device_type <chr>,
#> #   lastper_insp_date <date>, approval_date <date>, manufacturer <chr>,
```

```
#> #   travel_distance <chr>, speed_fpm <dbl>, capacity_lbs <dbl>,
#> #   car_buffer_type <chr>, governor_type <chr>, machine_type <chr>,
#> #   safety_type <chr>, mode_operation <chr>, floor_from <chr>, floor_to <chr>,
#> #   latitude <dbl>, longitude <dbl>, elevators_per_building <int>
```

glimpse()

```
tibble::glimpse(data_elevators())
#> Rows: 35,042
#> Columns: 25
#> $ device_number        <chr> "1D10028", "1D10094", "1D10097", "1D10146", "1D~
#> $ bin                  <chr> "1024795", "1041822", "1038223", "1080443", "10~
#> $ tax_block            <chr> "1021", "1392", "1323", "1274", "1074", "181", ~
#> $ tax_lot              <chr> "26", "25", "1", "6", "24", "16", "4", "1", "5"~
#> $ house_number         <chr> "1614", "53", "201", "40", "651", "179", "121",~
#> $ street_name          <chr> "BROADWAY", "E 77TH ST", "E 49 ST", "CENTRAL PA~
#> $ zip_code             <chr> "10019", "10021", "10017", NA, NA, "10013", "10~
#> $ borough              <fct> Manhattan, Manhattan, Manhattan, Manhattan, Man~
#> $ device_type          <chr> "Dumbwaiter", "Dumbwaiter", "Dumbwaiter", "Dumb~
#> $ lastper_insp_date    <date> 2015-09-18, 2015-08-07, 2015-04-02, 2014-10-15~
#> $ approval_date        <date> 2006-03-07, 2006-05-15, 1998-09-21, 2010-08-02~
#> $ manufacturer         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ travel_distance      <chr> "16'4\"", NA, "23", "8'", "24 FT", "9'0", "12'0~
#> $ speed_fpm            <dbl> 50, 25, 50, 50, 50, 50, 50, 50, 50, 100, 100, 5~
#> $ capacity_lbs         <dbl> 500, 500, 500, 500, NA, 500, 300, 500, 500, 500~
#> $ car_buffer_type      <chr> "Spring", NA, NA, NA, NA, NA, "Spring", NA, NA,~
#> $ governor_type        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ machine_type         <chr> NA, "OD", "BD", "BD", NA, "OD", "OD", "BD", "OG~
#> $ safety_type          <chr> "I", NA, "I", NA, NA, "I", "I", NA, "I", NA, NA~
#> $ mode_operation       <chr> "A", "P", "A", "A", NA, "A", "A", "A", "A", "P"~
#> $ floor_from           <chr> "B", "SB", "B", "B", "C", "BAS", "B", "C", "BMT~
#> $ floor_to             <chr> "1", "3", "2", "1", "G", "1", "1", "2", "4", "5~
#> $ latitude             <dbl> 40.76088, 40.77502, 40.75518, 40.76500, 40.7622~
#> $ longitude            <dbl> -73.98391, -73.96256, -73.97079, -73.97573, -73~
#> $ elevators_per_building <int> 11, 2, 1, 1, 2, 2, 1, 1, 1, 5, 5, 1, 2, 1, 1, 2~
```

Source

https://github.com/datanews/elevators

Examples

```
data_elevators()
```

`data_hotel_rates` *Daily Hotel Rate Data*

Description

A data set to predict the average daily rate for a hotel in Lisbon Portugal.

Usage

```
data_hotel_rates(...)
```

Arguments

`...`	Arguments passed to `pins::pin_read()`.

Details

Data are originally described in Antonio, de Almeida, and Nunes (2019). This version of the data is filtered for one hotel (the "Resort Hotel") and is intended as regression data set for predicting the average daily rate for a room. The data are post-2016; the 2016 data were used to have a predictor for the historical daily rates. See the `hotel_rates.R` file in the `data-raw` directory of the package to understand other filters used when creating this version of the data.

The `agent` and `company` fields were changed from random characters to use a set of random names.

The outcome column is `avg_price_per_room`.

License:

No license was given for the data; See the reference below for source.

Value

A tibble.

Source

https://github.com/rfordatascience/tidytuesday/tree/master/data/2020/2020-02-11

References

Antonio, N., de Almeida, A., and Nunes, L. (2019). Hotel booking demand datasets. *Data in Brief*, 22, 41-49.

Examples

```
data_hotel_rates()
```

`data_pharma_bioreactors` *Pharmaceutical manufacturing monitoring data set*

Description

Samples were collected each day from all bioreactors and glucose was measured using both spectroscopy and the traditional manner. The goal is to create models on the data from the more numerous small-scale bioreactors and then evaluate if these results can accurately predict what is happening in the large-scale bioreactors (see details below).

Usage

```
data_pharma_bioreactors(...)
```

Arguments

...	Arguments passed to `pins::pin_read()`.

Details

Experimental Background:

Pharmaceutical companies use spectroscopy measurements to assess critical process parameters during the manufacturing of a biological drug. Models built on this process can be used with real-time data to recommend changes that can increase product yield. In the example that follows, Raman spectroscopy was used to generate the data. These data were generated from real data, but have been distinctly modified to preserve confidentiality and achieve illustration purposes.

To manufacture the drug being used for this example, a specific type of protein is required and that protein can be created by a particular type of cell. A batch of cells are seeded into a *bioreactor* which is a device that is designed to help grow and maintain the cells. In production, a large bioreactor would be about 2000 liters and is used to make large quantities of proteins in about two weeks.

Many factors can affect product yield. For example, because the cells are living, working organisms, they need the right temperature and sufficient food (glucose) to generate drug product. During the course of their work, the cells also produce waste (ammonia). Too much of the waste product can kill the cells and reduce the overall product yield. Typically key attributes like glucose and ammonia are monitored daily to ensure that the cells are in optimal production conditions. Samples are collected and off-line measurements are made for these key attributes. If the measurements indicate a potential problem, the manufacturing scientists overseeing the process can tweak the contents of the bioreactor to optimize the conditions for the cells.

One issue is that conventional methods for measuring glucose and ammonia are time consuming and the results may not come in time to address any issues. Spectroscopy is a potentially faster method of obtaining these results if an effective model can be used to take the results of the spectroscopy assay to make predictions on the substances of interest (i.e., glucose and ammonia).

However, it is not feasible to do experiments using many large-scale bioreactors. Two parallel experimental systems were used:

- 15 small-scale (5 liters) bioreactors were seeded with cells and were monitored daily for 14 days.
- Three large-scale bioreactors were also seeded with cells from the same batch and monitored daily for 14 days

Notes on Data:

The intensity values have undergone signal processing up to smoothing. See the reference for more details.

License:

```
data_pharma_bioreactors()
#> # A tibble: 664,524 x 6
#>    reactor_id   day glucose wave_number intensity size
#>    <chr>      <int>   <dbl>       <dbl>     <dbl> <chr>
#>  1 S_01           1    24.7         407   0.909   small
#>  2 S_01           1    24.7         408   0.858   small
#>  3 S_01           1    24.7         409   0.766   small
#>  4 S_01           1    24.7         410   0.627   small
#>  5 S_01           1    24.7         411   0.448   small
#>  6 S_01           1    24.7         412   0.236   small
#>  7 S_01           1    24.7         413   0.00707 small
#>  8 S_01           1    24.7         414  -0.222   small
#>  9 S_01           1    24.7         415  -0.438   small
#> 10 S_01           1    24.7         416  -0.629   small
#> # i 664,514 more rows
```

Value

tibble

glimpse()

```
tibble::glimpse(data_pharma_bioreactors())
#> Rows: 664,524
#> Columns: 6
#> $ reactor_id  <chr> "S_01", "S_01", "S_01", "S_01", "S_01", "S_01", "S_01", "S~
#> $ day         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ glucose    <dbl> 24.74713, 24.74713, 24.74713, 24.74713, 24.74713, 24.74713~
#> $ wave_number <dbl> 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418~
#> $ intensity  <dbl> 0.909439216, 0.857607637, 0.766150467, 0.626862221, 0.4480~
#> $ size       <chr> "small", "small", "small", "small", "small", "small", "sma~
```

Source

Kuhn, Max, and Kjell Johnson. *Feature engineering and selection: A practical approach for predictive models*. Chapman and Hall/CRC, 2019.

https://bookdown.org/max/FES/illustrative-data-pharmaceutical-manufacturing-monitoring.html

Examples

```
data_pharma_bioreactors()
```

`data_taxi` *Chicago taxi data set*

Description

A data set containing information on a subset of taxi trips in the city of Chicago in 2022.

Usage

```
data_taxi(...)
```

Arguments

`...`	Arguments passed to `pins::pin_read()`.

Details

The source data are originally described on the linked City of Chicago data portal. The data exported here are a pre-processed subset motivated by the modeling problem of predicting whether a rider will tip or not.

tip Whether the rider left a tip. A factor with levels "yes" and "no".

distance The trip distance, in odometer miles.

company The taxi company, as a factor. Companies that occurred few times were binned as "other".

local Whether the trip started in the same community area as it began. See the source data for community area values.

dow The day of the week in which the trip began, as a factor.

month The month in which the trip began, as a factor.

hour The hour of the day in which the trip began, as a numeric.

Previous releases of this data (with `version = "20230630T214846Z-643d0"`) included additional columns:

id A unique identifier for the trip, as a factor.

duration The trip duration, in seconds.

fare The cost of the trip fare, in USD

tolls The cost of tolls for the trip, in USD.

extras The cost of extra charges for the trip, in USD.

total_cost The total cost of the trip, in USD. This is the sum of the previous three columns plus tip.

payment_type Type of payment for the trip. A factor with levels "Credit Card", "Dispute", "Mobile", "No Charge", "Prcard", and "Unknown".

Value

tibble

tibble print

```
data_taxi()
#> # A tibble: 10,000 x 7
#>    tip   distance company                      local dow   month  hour
#>    <fct>    <dbl> <fct>                        <fct> <fct> <fct> <int>
#>  1 yes      17.2  Chicago Independents         no    Thu   Feb      16
#>  2 yes       0.88 City Service                 yes   Thu   Mar       8
#>  3 yes      18.1  other                        no    Mon   Feb      18
#>  4 yes      20.7  Chicago Independents         no    Mon   Apr       8
#>  5 yes      12.2  Chicago Independents         no    Sun   Mar      21
#>  6 yes       0.94 Sun Taxi                     yes   Sat   Apr      23
#>  7 yes      17.5  Flash Cab                    no    Fri   Mar      12
#>  8 yes      17.7  other                        no    Sun   Jan       6
#>  9 yes       1.85 Taxicab Insurance Agency Llc no    Fri   Apr      12
#> 10 yes       1.47 City Service                 no    Tue   Mar      14
#> # i 9,990 more rows
```

glimpse()

```
tibble::glimpse(data_taxi())
#> Rows: 10,000
#> Columns: 7
#> $ tip      <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, yes, y~
#> $ distance <dbl> 17.19, 0.88, 18.11, 20.70, 12.23, 0.94, 17.47, 17.67, 1.85, 1~
#> $ company  <fct> Chicago Independents, City Service, other, Chicago Independen~
#> $ local    <fct> no, yes, no, no, no, yes, no, no, no, no, no, no, no, yes, no~
#> $ dow      <fct> Thu, Thu, Mon, Mon, Sun, Sat, Fri, Sun, Fri, Tue, Tue, Sun, W~
#> $ month    <fct> Feb, Mar, Feb, Apr, Mar, Apr, Mar, Jan, Apr, Mar, Mar, Apr, A~
#> $ hour     <int> 16, 8, 18, 8, 21, 23, 12, 6, 12, 14, 18, 11, 12, 19, 17, 13, ~
```

Source

https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew

Examples

```
data_taxi()
```

internal_board *Internal pins board*

Description

Pins board used internally to manage download, reading, and caching of data sets.

Usage

```
internal_board()
```

Value

a pins board

Examples

```
internal_board()
```

small_fine_foods *small_fine_foods data sets*

Description

Training and testing data set of fine food reviews.

Usage

```
attach_small_fine_foods(envir = parent.frame(), quiet = FALSE, ...)
```

Arguments

envir	Environment to load data sets into. Defaults to `parent.frame()`.
quiet	Logical, should function announce what data sets are loaded.
...	Arguments passed to `pins::pin_read()`.

Details

These data are from Amazon, who describe it as "This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plaintext review."

A subset of the data are contained here and are split into a training and test set. The training set sampled 10 products and retained all of their individual reviews. Since the reviews within these products are correlated, we recommend resampling the data using a leave-one-product-out approach. The test set sampled 500 products that were not included in the training set and selected a single review at random for each.

There is a column for the product, a column for the text of the review, and a factor column for a class variable. The outcome is whether the reviewer gave the product a 5-star rating or not.

Value

tibble

tibble print

```
attach_small_fine_foods()
#> The following data sets have been loaded:
#> `training_data`, `testing_data`
#> Silence this message by setting `quiet = TRUE`.

training_data
#> # A tibble: 4,000 x 3
#>    product    review                                                  score
#>    <chr>      <chr>                                                   <fct>
#>  1 B000J0LSBG "this stuff is  not stuffing  its  not good at all  save yo~ other
#>  2 B000EYLDYE "I absolutely LOVE this dried fruit.  LOVE IT.  Whenever I ~ great
#>  3 B0026LIO9A "GREAT DEAL, CONVENIENT TOO.  Much cheaper than WalMart and~ great
#>  4 B00473P8SK "Great flavor, we go through a ton of this sauce! I discove~ great
#>  5 B001SAWTNM "This is excellent salsa/hot sauce, but you can get it for ~ great
#>  6 B000FAG90U "Again, this is the best dogfood out there.  One suggestion~ great
#>  7 B006BXTCEK "The box I received was filled with teas, hot chocolates, a~ other
#>  8 B002GWH5OY "This is delicious coffee which compares favorably with muc~ great
#>  9 B003R0MFYY "Don't let these little tiny cans fool you.  They pack a lo~ great
#> 10 B001EO5ZXI "One of the nicest, smoothest cup of chai I've made. Nice m~ great
#> # i 3,990 more rows
testing_data
#> # A tibble: 1,000 x 3
#>    product    review                                                  score
#>    <chr>      <chr>                                                   <fct>
#>  1 B005GXFP60 "These are the best tasting gummy fruits I have ever eaten.~ great
#>  2 B000G7V394 "I have been a consumer of Snyders hard sourdough pretzels ~ great
#>  3 B004WJAULO "This tastes so bad, I'm considering throwing it away.  But~ other
#>  4 B003D4MBOS "This product is way too pricey to have so little chocolate~ other
#>  5 B0030Z95B2 "I bought this for my Mom as a gift to accompany her Dolce ~ great
#>  6 B000LRH4WE "This thing is 7 dollars in US?I know its exported from Cyp~ other
```

```
#>  7 B000Z91SZW "This tea tastes like hot cocoa.  Very pleasant experience.~ other
#>  8 B00563VNEI "This product is great for a quick cup of coffee. If you us~ great
#>  9 B0085NFX2O "Grilled out brats, chicken, and burgers for the entire fam~ great
#> 10 B000LRH7XK "I ordered 4 cans of this product.  The product is fine, bu~ other
#> # i 990 more rows
```

glimpse()

```
tibble::glimpse(training_data)
#> Rows: 4,000
#> Columns: 3
#> $ product <chr> "B000J0LSBG", "B000EYLDYE", "B0026LIO9A", "B00473P8SK", "B001S~
#> $ review  <chr> "this stuff is  not stuffing  its  not good at all  save your ~
#> $ score   <fct> other, great, great, great, great, great, other, great, great,~
tibble::glimpse(testing_data)
#> Rows: 1,000
#> Columns: 3
#> $ product <chr> "B005GXFP60", "B000G7V394", "B004WJAULO", "B003D4MBOS", "B0030~
#> $ review  <chr> "These are the best tasting gummy fruits I have ever eaten. Ca~
#> $ score   <fct> great, great, other, other, great, other, other, great, great,~
```

Source

https://snap.stanford.edu/data/web-FineFoods.html

Examples

```
attach_small_fine_foods()
```

Index