Package 'healthyR'

October 13, 2022

Title Hospital Data Analysis Workflow Tools

Version 0.2.0

Description Hospital data analysis workflow tools, modeling, and automations. This library provides many useful tools to review common administrative hospital data. Some of these include average length of stay, readmission rates, average net pay amounts by service lines just to name a few. The aim is to provide a simple and consistent verb framework that takes the guesswork out of everything.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

URL https://github.com/spsanderson/healthyR

BugReports https://github.com/spsanderson/healthyR/issues

Imports magrittr, rlang (>= 0.1.2), tibble, timetk, ggplot2, dplyr, lubridate, graphics, purrr, stringr, writexl, cowplot, scales, sqldf, tidyr, ggrepel, plotly

Suggests knitr, rmarkdown, roxygen2, pacman, healthyR.data, broom, uwot, tidyselect

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Steven Sanderson [aut, cre], Steven Sanderson [cph]

Maintainer Steven Sanderson <spsanderson@gmail.com>

Repository CRAN

Date/Publication 2022-07-18 19:40:02 UTC

R topics documented:

`category_counts_tbl` *Counts by Category*

Description

Get the counts of a column by a particular grouping if supplied, otherwise just get counts of a column.

Usage

```
category_counts_tbl(.data, .count_col, .arrange_value = TRUE, ...)
```

Arguments

.data	The data.frame/tibble supplied.
.count_col	The column that has the values you want to count.
.arrange_value	Defaults to true, this will arrange the resulting tibble in descending order by .count_col
...	Place the values you want to pass in for grouping here.

Details

- Requires a data.frame/tibble.
- Requires a value column, a column that is going to counted.

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR.data)
library(dplyr)

healthyR_data %>%
  category_counts_tbl(
    .count_col = payer_grouping
    , .arrange = TRUE
    , ip_op_flag
  )

healthyR_data %>%
  category_counts_tbl(
    .count_col = ip_op_flag
    , .arrange_value = TRUE
    , service_line
  )
```

---

color_blind *Provide Colorblind Compliant Colors*

Description

8 Hex RGB color definitions suitable for charts for colorblind people.

Usage

```
color_blind()
```

Details

This function is used in others in order to help render plots for those that are color blind.

Value

A vector of 8 Hex RGB definitions.

Author(s)

Steven P. Sanderson II, MPH

Examples

```
color_blind()
```

---

diverging_bar_plt *Diverging Bar Chart*

Description

Diverging Bars is a bar chart that can handle both negative and positive values. This can be implemented by a smart tweak with `geom_bar()`. But the usage of `geom_bar()` can be quite confusing. That's because, it can be used to make a bar chart as well as a histogram. Let me explain.

By default, `geom_bar()` has the stat set to count. That means, when you provide just a continuous X variable (and no Y variable), it tries to make a histogram out of the data.

In order to make a bar chart create bars instead of histogram, you need to do two things. Set `stat = identity` and provide both `x` and `y` inside `aes()` where, `x` is either character or factor and `y` is numeric. In order to make sure you get diverging bars instead of just bars, make sure, your categorical variable has 2 categories that changes values at a certain threshold of the continuous variable. In below example, the mpg from mtcars data set is normalized by computing the z score. Those vehicles with mpg above zero are marked green and those below are marked red.

Usage

```
diverging_bar_plt(
  .data,
  .x_axis,
  .y_axis,
  .fill_col,
  .plot_title = NULL,
  .plot_subtitle = NULL,
  .plot_caption = NULL,
```

```
  .interactive = FALSE
)
```

Arguments

`.data`	The data to pass to the function, must be a tibble/data.frame.
`.x_axis`	The data that is passed to the x-axis.
`.y_axis`	The data that is passed to the y-axis. This will also equal the parameter `label`
`.fill_col`	The column that will be used to fill the color of the bars.
`.plot_title`	Default is NULL
`.plot_subtitle`	Default is NULL
`.plot_caption`	Default is NULL
`.interactive`	Default is FALSE. TRUE returns a plotly plot

Details

This function takes only a few arguments and returns a ggplot2 object.

Value

A `plotly` plot or a `ggplot2` static plot

Author(s)

Steven P. Sanderson II, MPH

Examples

```
suppressPackageStartupMessages(library(ggplot2))

data("mtcars")
mtcars$car_name <- rownames(mtcars)
mtcars$mpg_z <- round((mtcars$mpg - mean(mtcars$mpg))/sd(mtcars$mpg), 2)
mtcars$mpg_type <- ifelse(mtcars$mpg_z < 0, "below", "above")
mtcars <- mtcars[order(mtcars$mpg_z), ]  # sort
mtcars$car_name <- factor(mtcars$car_name, levels = mtcars$car_name)

diverging_bar_plt(
  .data          = mtcars
  , .x_axis      = car_name
  , .y_axis      = mpg_z
  , .fill_col    = mpg_type
  , .interactive = FALSE
)
```

`diverging_lollipop_plt` *Diverging Lollipop Chart*

Description

This is a diverging lollipop function. Lollipop chart conveys the same information as bar chart and diverging bar. Except that it looks more modern. Instead of geom_bar, I use geom_point and geom_segment to get the lollipops right. Let's draw a lollipop using the same data I prepared in the previous example of diverging bars.

Usage

```
diverging_lollipop_plt(
  .data,
  .x_axis,
  .y_axis,
  .plot_title = NULL,
  .plot_subtitle = NULL,
  .plot_caption = NULL,
  .interactive = FALSE
)
```

Arguments

`.data`	The data to pass to the function, must be a tibble/data.frame.
`.x_axis`	The data that is passed to the x-axis. This will also be the `x` and `xend` parameters of the `geom_segment`
`.y_axis`	The data that is passed to the y-axis. This will also equal the parameters of `yend` and `label`
`.plot_title`	Default is NULL
`.plot_subtitle`	Default is NULL
`.plot_caption`	Default is NULL
`.interactive`	Default is FALSE. TRUE returns a plotly plot

Details

This function takes only a few arguments and returns a ggplot2 object.

Value

A `plotly` plot or a `ggplot2` static plot

Author(s)

Steven P. Sanderson II, MPH

Examples

```
suppressPackageStartupMessages(library(ggplot2))

data("mtcars")
mtcars$car_name <- rownames(mtcars)
mtcars$mpg_z <- round((mtcars$mpg - mean(mtcars$mpg))/sd(mtcars$mpg), 2)
mtcars$mpg_type <- ifelse(mtcars$mpg_z < 0, "below", "above")
mtcars <- mtcars[order(mtcars$mpg_z), ]  # sort
mtcars$car_name <- factor(mtcars$car_name, levels = mtcars$car_name)

diverging_lollipop_plt(.data = mtcars, .x_axis = car_name
, .y_axis = mpg_z)
```

---

dx_cc_mapping *Diagnosis to Condition Code Mapping file*

Description

A dataset containing the Diagnosis Code to AHRQ Condition Code Mapping that is used in helping to define service lines for inpatient discharges.

Usage

```
data(dx_cc_mapping)
```

Format

A data frame with 86852 rows and 5 variables

Details

- CC_Code. DX_1, DX_2, ..., DX_n
- CC_Desc. DX_1 = Conduction disorders, DX_n = description
- ICD_Ver_Flag. ICD Version 10 or 9
- ICDCode. ICD-9 ro ICD-10 Code
- Diagnosis. Long QT Syndrome

gartner_magic_chart_plt *Gartner Magic Chart - Plotting of two continuous variables*

Description

Plot a Gartner Magic Chart of two continuous variables

Usage

```
gartner_magic_chart_plt(
  .data,
  .x_col,
  .y_col,
  .point_size_col = NULL,
  .y_lab,
  .x_lab,
  .plt_title,
  .tl_lbl,
  .tr_lbl,
  .br_lbl,
  .bl_lbl
)
```

Arguments

.data	The data set you want to plot
.x_col	The x-axis for the plot
.y_col	The y-axis for the plot
.point_size_col	The default is NULL, if you want to size the dots by a column in the data.frame/tibble then enter the column name here.
.y_lab	The y-axis label
.x_lab	The x-axis label
.plt_title	The title of the plot
.tl_lbl	The top left label
.tr_lbl	The top right label
.br_lbl	The bottom right label
.bl_lbl	The bottom left label

Details

- Supply a data frame with at least two continuous variables to plot against each other

Value

A ggplot plot

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(dplyr)

data_tbl <- tibble(
    x = rnorm(100, 0, 1),
    y = rnorm(100, 0, 1),
    z = abs(x) + abs(y)
 )

gartner_magic_chart_plt(
  .data = data_tbl,
  .x_col = x,
  .y_col = y,
  .point_size = z,
  .x_lab = "los",
  .y_lab = "ra",
  .plt_title = "tst",
  .tr_lbl = "High RA-LOS",
  .tl_lbl = "High RA",
  .bl_lbl = "Leader",
  .br_lbl = "High LOS"
)

gartner_magic_chart_plt(
  .data = data_tbl,
  .x_col = x,
  .y_col = y,
  .point_size = NULL,
  .x_lab = "los",
  .y_lab = "ra",
  .plt_title = "tst",
  .tr_lbl = "High RA-LOS",
  .tl_lbl = "High RA",
  .bl_lbl = "Leader",
  .br_lbl = "High LOS"
)
```

---

hr_scale_color_colorblind

Provide Colorblind Compliant Colors

Description

8 Hex RGB color definitions suitable for charts for colorblind people.

Usage

```
hr_scale_color_colorblind(..., theme = "hr")
```

Arguments

...	Data passed in from a ggplot object
theme	Right now this is hr only. Anything else will render an error.

Details

This function is used in others in order to help render plots for those that are color blind.

Value

A gggplot layer

Author(s)

Steven P. Sanderson II, MPH

---

hr_scale_fill_colorblind

Provide Colorblind Compliant Colors

Description

8 Hex RGB color definitions suitable for charts for colorblind people.

Usage

```
hr_scale_fill_colorblind(..., theme = "hr")
```

Arguments

...	Data passed in from a ggplot object
theme	Right now this is hr only. Anything else will render an error.

Details

This function is used in others in order to help render plots for those that are color blind.

Value

A gggplot layer

Author(s)

Steven P. Sanderson II, MPH

`kmeans_mapped_tbl` *K-Means Mapper*

Description

Create a tibble that maps the `kmeans_obj()` using `purrr::map()` to create a nested data.frame/tibble that holds n centers. This tibble will be used to help create a scree plot.

Usage

```
kmeans_mapped_tbl(.data, .centers = 15)
```

Arguments

`.data`	You must have a tibble in the working environment from the `kmeans_user_item_tbl()`
`.centers`	How many different centers do you want to try

Details

Takes in a single parameter of .centers. This is used to create the tibble and map the `kmeans_obj()` function down the list creating a nested tibble.

Value

A nested tibble

Author(s)

Steven P. Sanderson II, MPH

See Also

https://en.wikipedia.org/wiki/Scree_plot

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
```

```
   as_tibble()

ui_tbl <-  kmeans_user_item_tbl(
   .data           = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 )

kmeans_mapped_tbl(ui_tbl)
```

---

kmeans_obj *K-Means Functions*

Description

Takes the output of the kmeans_user_item_tbl() function and applies the k-means algorithm to it using stats::kmeans()

Usage

```
kmeans_obj(.data, .centers = 5)
```

Arguments

.data	The data that gets passed from kmeans_user_item_tbl()
.centers	How many initial centers to start with

Details

Uses the stats::kmeans() function and creates a wrapper around it.

Value

A stats k-means object

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
```

```
  filter(payer_grouping != "?") %>%
  select(service_line, payer_grouping) %>%
  mutate(record = 1) %>%
  as_tibble()

kmeans_user_item_tbl(
  .data           = data_tbl
  , .row_input    = service_line
  , .col_input    =  payer_grouping
  , .record_input = record
) %>%
  kmeans_obj()
```

`kmeans_scree_data_tbl` *K-Means Scree Plot Data Table*

Description

Take data from the `kmeans_mapped_tbl()` and unnest it into a tibble for inspection and for use in the `kmeans_scree_plt()` function.

Usage

```
kmeans_scree_data_tbl(.data)
```

Arguments

`.data`	You must have a tibble in the working environment from the `kmeans_mapped_tbl()`

Details

Takes in a single parameter of .data from `kmeans_mapped_tbl()` and transforms it into a tibble that is used for `kmeans_scree_plt()`. It will show the values (tot.withinss) at each center.

Value

A nested tibble

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

ui_tbl <-  kmeans_user_item_tbl(
   .data           = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 )

kmm_tbl <- kmeans_mapped_tbl(ui_tbl)

kmeans_scree_data_tbl(kmm_tbl)
```

---

kmeans_scree_plt *K-Means Scree Plot*

Description

Create a scree-plot from the `kmeans_mapped_tbl()` function.

Usage

```
kmeans_scree_plt(.data)
```

Arguments

`.data`	The data from the `kmeans_mapped_tbl()` function

Details

Outputs a scree-plot

Value

A ggplot2 plot

Author(s)

Steven P. Sanderson II, MPH

See Also

https://en.wikipedia.org/wiki/Scree_plot

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

ui_tbl <-  kmeans_user_item_tbl(
   .data           = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 )

kmm_tbl <- kmeans_mapped_tbl(ui_tbl)

kmeans_scree_plt(.data = kmm_tbl)
```

---

kmeans_tidy_tbl *K-Means tidy Functions*

Description

K-Means tidy functions

Usage

```
kmeans_tidy_tbl(.kmeans_obj, .data, .tidy_type = "tidy")
```

Arguments

.kmeans_obj	A stats::kmeans() object
.data	The user item tibble created from kmeans_user_item_tbl()
.tidy_type	"tidy","glance", or "augment"

Details

Takes in a k-means object and its associated user item tibble and then returns one of the items asked for. Either: broom::tidy(), broom::glance() or broom::augment(). The function defaults to broom::tidy().

Value

A tibble

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR.data)
library(dplyr)
library(broom)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

uit_tbl <- kmeans_user_item_tbl(
  .data           = data_tbl
  , .row_input    = service_line
  , .col_input    =  payer_grouping
  , .record_input = record
)

km_obj  <- kmeans_obj(uit_tbl)

kmeans_tidy_tbl(
  .kmeans_obj  = km_obj
  , .data      = uit_tbl
  , .tidy_type = "augment"
)

kmeans_tidy_tbl(
  .kmeans_obj  = km_obj
  , .data      = uit_tbl
  , .tidy_type = "glance"
)

kmeans_tidy_tbl(
  .kmeans_obj  = km_obj
  , .data      = uit_tbl
  , .tidy_type = "tidy"
) %>%
  glimpse()
```

`kmeans_user_item_tbl` *K-Means Functions*

Description

Takes in a data.frame/tibble and transforms it into an aggregated/normalized user-item tibble of proportions. The user will need to input the parameters for the rows/user and the columns/items.

Usage

```
kmeans_user_item_tbl(.data, .row_input, .col_input, .record_input)
```

Arguments

`.data`	The data that you want to transform
`.row_input`	The column that is going to be the row (user)
`.col_input`	The column that is going to be the column (item)
`.record_input`	The column that is going to be summed up for the aggregattion and normalization process.

Details

This function should be used before using a k-mean model. This is commonly referred to as a user item matrix because "users" tend to be on the rows and "items" (e.g. orders) on the columns. You must supply a column that can be summed for the aggregation and normalization process to occur.

Value

A aggregated/normalized user item tibble

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

 kmeans_user_item_tbl(
```

```
  .data           = data_tbl
  , .row_input    = service_line
  , .col_input    =  payer_grouping
  , .record_input = record
)
```

`los_ra_index_plt` *Plot LOS and Readmit Index with Variance*

Description

Plot the index of the length of stay and readmit rate against each other along with the variance

Usage

```
los_ra_index_plt(.data)
```

Arguments

`.data`	The data supplied from `los_ra_index_summary_tbl()`

Details

- Expects a tibble
- Expects a Length of Stay and Readmit column, must be numeric
- Uses `cowplot` to stack plots

Value

A `patchwork` `ggplot2` plot

Author(s)

Steven P. Sanderson II, MPH

Examples

```
suppressPackageStartupMessages(library(dplyr))

data_tbl <- tibble(
  "alos"                = runif(186, 1, 20)
  , "elos"              = runif(186, 1, 17)
  , "readmit_rate"      = runif(186, 0, .25)
  , "readmit_rate_bench" = runif(186, 0, .2)
)

los_ra_index_summary_tbl(
```

```
  .data = data_tbl
  , .max_los       = 15
  , .alos_col      = alos
  , .elos_col      = elos
  , .readmit_rate  = readmit_rate
  , .readmit_bench = readmit_rate_bench
) %>%
  los_ra_index_plt()

los_ra_index_summary_tbl(
  .data = data_tbl
  , .max_los       = 10
  , .alos_col      = alos
  , .elos_col      = elos
  , .readmit_rate  = readmit_rate
  , .readmit_bench = readmit_rate_bench
) %>%
  los_ra_index_plt()
```

`los_ra_index_summary_tbl` *Make LOS and Readmit Index Summary Tibble*

Description

Create the length of stay and readmit index summary tibble

Usage

```
los_ra_index_summary_tbl(
  .data,
  .max_los = 15,
  .alos_col,
  .elos_col,
  .readmit_rate,
  .readmit_bench
)
```

Arguments

`.data`	The data you are going to analyze.
`.max_los`	You can give a maximum LOS value. Lets say you typically do not see los over 15 days, you would then set .max_los to 15 and all values greater than .max_los will be grouped to .max_los
`.alos_col`	The Average Length of Stay column
`.elos_col`	The Expected Length of Stay column
`.readmit_rate`	The Actual Readmit Rate column
`.readmit_bench`	The Expected Readmit Rate column

Details

- Expects a tibble
- Expects the following columns and there should only be these 4
  - Length Of Stay Actual - Should be an integer
  - Length Of Stacy Benchmark - Should be an integer
  - Readmit Rate Actual - Should be 0/1 for each record, 1 = readmitted, 0 did not.
  - Readmit Rate Benchmark - Should be a percentage from the benchmark file.
- This will add a column called visits that will be the count of records per length of stay from 1 to .max_los
- The .max_los param can be left blank and the function will default to 15. If this is not a good default and you don't know what it should be then set it to 75 percentile from the `stats::quantile()` function using the defaults, like so .max_los = `stats::quantile(data_tbl$alos)[[4]]`
- Uses all data to compute variance, if you want it for a particular time frame you will have to filter the data that goes into the .data argument. It is suggested to use `timetk::filter_by_time()`
- The index is computed as the excess of the length of stay or readmit rates over their respective expectations.

Value

A tibble

Author(s)

Steven P. Sanderson II, MPH

Examples

```
suppressPackageStartupMessages(library(dplyr))

data_tbl <- tibble(
  "alos"            = runif(186, 1, 20)
  , "elos"          = runif(186, 1, 17)
  , "readmit_rate"  = runif(186, 0, .25)
  , "readmit_bench" = runif(186, 0, .2)
)

los_ra_index_summary_tbl(
  .data = data_tbl
  , .max_los       = 15
  , .alos_col      = alos
  , .elos_col      = elos
  , .readmit_rate  = readmit_rate
  , .readmit_bench = readmit_bench
)

los_ra_index_summary_tbl(
  .data = data_tbl
```

```
  , .max_los       = 10
  , .alos_col      = alos
  , .elos_col      = elos
  , .readmit_rate  = readmit_rate
  , .readmit_bench = readmit_bench
)
```

---

named_item_list *Tibble to named list*

Description

Takes in a data.frame/tibble and creates a named list from a supplied grouping variable. Can be used in conjunction with save_to_excel() to create a new sheet for each group of data.

Usage

```
named_item_list(.data, .group_col)
```

Arguments

.data	The data.frame/tibble.
.group_col	The column that contains the groupings.

Details

- Requires a data.frame/tibble and a grouping column.

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR.data)

df <- healthyR_data
df_list <- named_item_list(.data = df, .group_col = service_line)
df_list
```

opt_bin *Get the optimal binwidth for a histogram*

Description

Gives the optimal binwidth for a histogram given a data set, it's value and the desired amount of bins

Usage

```
opt_bin(.data, .value_col, .iters = 30)
```

Arguments

.data	The data set in question
.value_col	The column that holds the values
.iters	How many times the cost function loop should run

Details

Modified from Hideaki Shimazaki Department of Physics, Kyoto University shimazaki at ton.scphys.kyoto-u.ac.jp Feel free to modify/distribute this program.

- Supply a data.frame/tibble with a value column. from this an optimal binwidth will be computed for the amount of binds desired

Value

A tibble of histogram breakpoints

Examples

```
suppressPackageStartupMessages(library(purrr))
suppressPackageStartupMessages(library(dplyr))

df_tbl <- rnorm(n = 1000, mean = 0, sd = 1)
df_tbl <- df_tbl %>%
  as_tibble() %>%
  set_names("value")

df_tbl %>%
  opt_bin(
    .value_col = value
    , .iters = 100
  )
```

px_cc_mapping *Procedure to Condition Code Mapping file*

Description

A dataset containing the Procedure Code to AHRQ Condition Code Mapping that is used in helping to define servce lines for inpatient discharges.

Usage

```
data(px_cc_mapping)
```

Format

A data frame with 79721 rows and 5 variables

Details

- CC_Code. PX_1, PX_2, ..., PX_n
- CC_Desc. PX_1 = Genitourinary incontinence procedures
- ICD_Ver_Flag. 10 or 9
- ICDCode. ICD-9 or ICD-10 Code
- Procedure. Inject Implant Urethra

save_to_excel *Save a file to Excel*

Description

Save a tibble/data.frame to an excel `.xlsx` file. The file will automatically with a save_dtime in the format of 20201109_132416 for November 11th, 2020 at 1:24:16PM.

Usage

```
save_to_excel(.data, .file_name)
```

Arguments

`.data`	The tibble/data.frame that you want to save as an `.xlsx` file.
`.file_name`	the name you want to give to the file.

Details

- Requires a tibble/data.frame to be passed to it.

Value

A saved excel file

Author(s)

Steven P. Sanderson II, MPH

`service_line_augment` *Service Line Grouper Augment Function*

Description

Takes a few arguments from a data.frame/tibble and returns a service line augmented to a data.frame/tibble for a set of patients.

Usage

```
service_line_augment(.data, .dx_col, .px_col, .drg_col)
```

Arguments

`.data`	The data being passed that will be augmented by the function.
`.dx_col`	The column containing the Principal Diagnosis for the discharge.
`.px_col`	The column containing the Principal Coded Procedure for the discharge. It is possible that this could be blank.
`.drg_col`	The DRG Number coded to the inpatient discharge.

Details

This is an augment function in that appends a vector to an data.frame/tibble that is passed to the `.data` parameter. A data.frame/tibble is required, along with a principal diagnosis column, a principal procedure column, and a column for the DRG number. These are needed so that the function can join the dx_cc_mapping and px_cc_mapping columns to provide the service line. This function only works on visits that are coded using ICD Version 10 only.

Lets take an example discharge, the DRG is 896 and the Principal Diagnosis code maps to DX_660, then this visit would get grouped to `alcohol_abuse`

DRG 896: ALCOHOL, DRUG ABUSE OR DEPENDENCE WITHOUT REHABILITATION THERAPY WITH MAJOR COMPLICATION OR COMORBIDITY (MCC)

DX_660 Maps to the following ICD-10 Codes ie F1010 Alcohol abuse, uncomplicated:

```
library(healthyR)
dx_cc_mapping %>%
  filter(CC_Code == "DX_660", ICD_Ver_Flag == "10")
```

Value

An augmented data.frame/tibble with the service line appended as a new column.

Author(s)

Steven P. Sanderson II, MPH

Examples

```
df <- data.frame(
  dx_col = "F10.10",
  px_col = NA,
  drg_col = "896"
)

service_line_augment(
  .data = df,
  .dx_col = dx_col,
  .px_col = px_col,
  .drg_col = drg_col
)
```

---

service_line_vec *Service Line Grouper Vectorized Function*

Description

Takes a few arguments from a data.frame/tibble and returns a service line vector for a set of patients.

Usage

```
service_line_vec(.data, .dx_col, .px_col, .drg_col)
```

Arguments

.data	The data being passed that will be augmented by the function.
.dx_col	The column containing the Principal Diagnosis for the discharge.
.px_col	The column containing the Principal Coded Procedure for the discharge. It is possible that this could be blank.
.drg_col	The DRG Number coded to the inpatient discharge.

Details

This is a vectorized function in that it returns a vector. It can be applied inside of a `mutate` statement when using `dplyr` if desired. A data.frame/tibble is required, along with a principal diagnosis column, a principal procedure column, and a column for the DRG number. These are needed so that the function can join the dx_cc_mapping and px_cc_mapping columns to provide the service line. This function only works on visits that are coded using ICD Version 10 only.

Lets take an example discharge, the DRG is 896 and the Principal Diagnosis code maps to DX_660, then this visit would get grouped to `alcohol_abuse`

DRG 896: ALCOHOL, DRUG ABUSE OR DEPENDENCE WITHOUT REHABILITATION THERAPY WITH MAJOR COMPLICATION OR COMORBIDITY (MCC)

DX_660 Maps to the following ICD-10 Codes ie F1010 Alcohol abuse, uncomplicated:

```
library(healthyR)
dx_cc_mapping %>%
  filter(CC_Code == "DX_660", ICD_Ver_Flag == "10")
```

Value

A vector of service line assignments.

Author(s)

Steven P. Sanderson II, MPH

Examples

```
df <- data.frame(
  dx_col = "F10.10",
  px_col = NA,
  drg_col = "896"
)

service_line_vec(
  .data = df,
  .dx_col = dx_col,
  .px_col = px_col,
  .drg_col = drg_col
)
```

`sql_left` *Use SQL LEFT type function*

Description

Perform an SQL LEFT() type function on a piece of text

Usage

```
sql_left(.text, .num_char)
```

Arguments

.text	A piece of text/string to be manipulated
.num_char	How many characters do you want to grab

Details

- You must supply data that you want to manipulate.

Examples

```
sql_left("text", 3)
```

---

sql_mid *Use SQL MID type function*

Description

Perform an SQL SUBSTRING type function

Usage

```
sql_mid(.text, .start_num, .num_char)
```

Arguments

.text	A piece of text/string to be manipulated
.start_num	What place to start at
.num_char	How many characters do you want to grab

Details

- You must supply data that you want to manipulate.

Examples

```
sql_mid("this is some text", 6, 2)
```

sql_right *Use SQL RIGHT type functions*

Description

Perform an SQL RIGHT type function

Usage

```
sql_right(.text, .num_char)
```

Arguments

.text	A piece of text/string to be manipulated
.num_char	How many characters do you want to grab

Details

- You must supply data that you want to manipulate.

Examples

```
sql_right("this is some more text", 3)
```

top_n_tbl *Top N tibble*

Description

Get a tibble returned with n records sorted either by descending order (default) or ascending order.

Usage

```
top_n_tbl(.data, .n_records, .arrange_value = TRUE, ...)
```

Arguments

.data	The data you want to pass to the function
.n_records	How many records you want returned
.arrange_value	A boolean with TRUE as the default. TRUE sorts data in descending order
...	The columns you want to pass to the function.

Details

- Requires a data.frame/tibble
- Requires at least one column to be chosen inside of the ...
- Will return the tibble in sorted order that is chosen with descending as the default

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR.data)

df <- healthyR_data

df_tbl <- top_n_tbl(
  .data = df
  , .n_records = 3
  , .arrange_value = TRUE
  , service_line
  , payer_grouping
)

print(df_tbl)
```

---

ts_alos_plt *Plot ALOS - Average Length of Stay*

Description

Plot ALOS - Average Length of Stay

Usage

```
ts_alos_plt(.data, .date_col, .value_col, .by_grouping, .interactive)
```

Arguments

.data	The time series data you need to pass
.date_col	The date column
.value_col	The value column
.by_grouping	How you want the data summarized - "sec", "min", "hour", "day", "week", "month", "quarter" or "year"
.interactive	TRUE or FALSE. TRUE returns a plotly plot and FALSE returns a static ggplot2 plot

Details

- Expects a tibble with a date time column and a value column
- Uses `timetk` for underlying sumarization and plot
- If .by_grouping is missing it will default to "day"
- A static ggplot2 object is return if the .interactive function is FALSE otherwise a `plotly` plot is returned.

Value

A timetk time series plot

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR)
library(healthyR.data)
library(timetk)
library(dplyr)
library(purrr)

# Make A Series of Dates ----
data_tbl <- healthyR_data

df_tbl <- data_tbl %>%
    filter(ip_op_flag == "I") %>%
    select(visit_end_date_time, length_of_stay) %>%
    summarise_by_time(
        .date_var = visit_end_date_time
        , .by     = "day"
        , visits  = mean(length_of_stay, na.rm = TRUE)
    ) %>%
    filter_by_time(
        .date_var     = visit_end_date_time
        , .start_date = "2012"
        , .end_date   = "2019"
    ) %>%
    set_names("Date","Values")

ts_alos_plt(
  .data = df_tbl
  , .date_col = Date
  , .value_col = Values
  , .by = "month"
  , .interactive = FALSE
)
```

`ts_census_los_daily_tbl`

Time Series - Census and LOS by Day

Description

Sometimes it is important to know what the census was on any given day, or what the average length of stay is on given day, including for those patients that are not yet discharged. This can be easily achieved. This will return one record for every account so the data will still need to be summarized. If there are multiple entries per day then those records will show up and you will therefore have multiple entries in the column `date` in the resulting `tibble`. If you want to aggregate from there you should be able to do so easily.

If you have a record where the `.start_date_col` is filled in but the corresponding `end_date` is null then the end date will be set equal to `Sys.Date()`

If a record has a `start_date` that is `NA` then it will be discarded.

This function can take a little bit of time to run while the join comparison runs.

Usage

```
ts_census_los_daily_tbl(
  .data,
  .keep_nulls_only = FALSE,
  .start_date_col,
  .end_date_col,
  .by_time = "day"
)
```

Arguments

`.data`	The data you want to pass to the function
`.keep_nulls_only`	A boolean that will keep only those records that have a NULL end date, meaning the patient is still admitted. The default is FALSE which brings back all records.
`.start_date_col`	The column containing the start date for the record
`.end_date_col`	The column containing the end date for the record.
`.by_time`	How you want the data presented, defaults to day and should remain that way unless you need more granular data.

Details

- Requires a dataset that has at least a start date column and an end date column
- Takes a single boolean parameter

Value

A tibble object

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(healthyR)
library(healthyR.data)
library(dplyr)

df <- healthyR_data

df_tbl <- df %>%
  filter(ip_op_flag == "I") %>%
  select(visit_start_date_time, visit_end_date_time) %>%
  timetk::filter_by_time(.date_var = visit_start_date_time, .start_date = "2020")

ts_census_los_daily_tbl(
   .data              = df_tbl
   , .keep_nulls_only = FALSE
   , .start_date_col  = visit_start_date_time
   , .end_date_col    = visit_end_date_time
)
```

---

ts_median_excess_plt *Create a plot showing the excess of the median value*

Description

Plot out the excess +/- of the median value grouped by certain time parameters.

Usage

```
ts_median_excess_plt(
  .data,
  .date_col,
  .value_col,
  .x_axis,
  .ggplot_group_var,
  .years_back
)
```

Arguments

.data	The data that is being analyzed, data must be a tibble/data.frame.
.date_col	The column of the tibble that holds the date.
.value_col	The column that holds the value of interest.
.x_axis	What is the be the x-axis, day, week, etc.
.ggplot_group_var	The variable to group the ggplot on.
.years_back	How many yeas back do you want to go in order to compute the median value.

Details

- Supply data that you want to view and you will see the excess +/- of the median values over a specified time series tibble.

Value

A ggplot2 plot

Examples

```
suppressPackageStartupMessages(library(timetk))

ts_signature_tbl(
  .data       = m4_daily
  , .date_col = date
) %>%
ts_median_excess_plt(
  .date_col           = date
  , .value_col        = value
  , .x_axis           = month
  , .ggplot_group_var = year
  , .years_back       = 1
)
```

---

ts_plt *Time Series Plot*

Description

This is a warpper function to the timetk::plot_time_series() function with a limited functionality parameter set. To see the full reference please visit the timetk package site.

Usage

```
ts_plt(
  .data,
  .date_col,
  .value_col,
  .color_col = NULL,
  .facet_col = NULL,
  .facet_ncol = NULL,
  .interactive = FALSE
)
```

Arguments

.data	The data to pass to the function, must be a tibble/data.frame.
.date_col	The column holding the date.
.value_col	The column holding the value.
.color_col	The column holding the variable for color.
.facet_col	The column holding the variable for faceting.
.facet_ncol	How many columns do you want.
.interactive	Return a plotly plot if set to TRUE and a static ggplot2 plot if set to FALSE. The default is FALSE.

Details

This function takes only a few of the arguments in the function and presets others while choosing the defaults on others. The smoother functionality is turned off.

Value

A plotly plot or a ggplot2 static plot

Author(s)

Steven P. Sanderson II, MPH

See Also

https://business-science.github.io/timetk/reference/plot_time_series.html

Examples

```
suppressPackageStartupMessages(library(dplyr))
library(timetk)
library(healthyR.data)

healthyR.data::healthyR_data %>%
  filter(ip_op_flag == "I") %>%
  select(visit_end_date_time, service_line) %>%
```

```
  filter_by_time(
    .date_var = visit_end_date_time
    , .start_date = "2020"
    ) %>%
  group_by(service_line) %>%
  summarize_by_time(
    .date_var = visit_end_date_time
    , .by = "month"
    , visits = n()
  ) %>%
 ungroup() %>%
 ts_plt(
   .date_col = visit_end_date_time
   , .value_col = visits
   , .color_col = service_line
 )
```

`ts_readmit_rate_plt` *Plot Readmit Rate*

Description

Plot Readmit Rate

Usage

```
ts_readmit_rate_plt(.data, .date_col, .value_col, .by_grouping, .interactive)
```

Arguments

`.data`	The data you need to pass.
`.date_col`	The date column.
`.value_col`	The value column.
`.by_grouping`	How you want the data summarized - "sec", "min", "hour", "day", "week", "month", "quarter" or "year".
`.interactive`	TRUE or FALSE. TRUE returns a `plotly` plot and FALSE returns a static `ggplot2` plot.

Details

- Expects a tibble with a date time column and a value column
- Uses `timetk` for underlying sumarization and plot
- If .by_grouping is missing it will default to "day"

Value

A `timetk` time series plot that is interactive

Author(s)

Steven P. Sanderson II, MPH

Examples

```
set.seed(123)

suppressPackageStartupMessages(library(timetk))
suppressPackageStartupMessages(library(purrr))
suppressPackageStartupMessages(library(dplyr))

ts_tbl <- tk_make_timeseries(
  start = "2019-01-01"
  , by = "day"
  , length_out = "1 year 6 months"
)
values <- arima.sim(
  model = list(
    order = c(0, 1, 0))
    , n = 547
    , mean = 1
    , sd = 5
)

df_tbl <- tibble(
  x = ts_tbl
  , y = values
  ) %>%
  set_names("Date","Values")

ts_readmit_rate_plt(
  .data = df_tbl
  , .date_col = Date
  , .value_col = Values
  , .by = "month"
  , .interactive = FALSE
)
```

---

ts_signature_tbl *Make a Time Enhanced Tibble*

Description

Returns a tibble that adds the time series signature from the `timetk::tk_augment_timeseries_signature()` function. All added from a chosen date column defined by the `.date_col` parameter.

Usage

```
ts_signature_tbl(.data, .date_col, .pad_time = TRUE, ...)
```

Arguments

.data	The data that is being analyzed.
.date_col	The column that holds the date.
.pad_time	Boolean TRUE/FALSE. If TRUE then the timetk::pad_by_time() function is called and used on the data.frame before the modification. The default is TRUE.
...	Grouping variables to be used by dplyr::group_by() before using timetk::pad_by_time()

Details

- Supply data with a date column and this will add the year, month, week, week day and hour to the tibble. The original date column is kept.
- Returns a time-series signature tibble.
- You must know the data going into the function and if certain columns should be dropped or kept when using further functions

Value

A tibble

Examples

```
library(timetk)

ts_signature_tbl(
  .data       = m4_daily
  , .date_col = date
  , .pad_time = TRUE
  , id
)
```

---

umap_list *UMAP Projection*

Description

Create a umap object from the uwot::umap() function.

Usage

```
umap_list(.data, .kmeans_map_tbl, .k_cluster = 5)
```

Arguments

.data	The data from the kmeans_user_item_tbl() function.
.kmeans_map_tbl	The data from the kmeans_mapped_tbl().
.k_cluster	Pick the desired amount of clusters from your analysis of the scree plot.

Details

This takes in the user item table/matix that is produced by `kmeans_user_item_tbl()` function. This function uses the defaults of `uwot::umap()`.

Value

A list of tibbles and the umap object

Author(s)

Steven P. Sanderson II, MPH

See Also

- https://cran.r-project.org/package=uwot (CRAN)
- https://github.com/jlmelville/uwot (GitHub)
- https://github.com/jlmelville/uwot (arXiv paper)

Examples

```
library(healthyR.data)
library(healthyR)
library(dplyr)
library(broom)

data_tbl <- healthyR_data %>%
    filter(ip_op_flag == "I") %>%
    filter(payer_grouping != "Medicare B") %>%
    filter(payer_grouping != "?") %>%
    select(service_line, payer_grouping) %>%
    mutate(record = 1) %>%
    as_tibble()

uit_tbl <- kmeans_user_item_tbl(
   .data           = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 )

kmm_tbl <- kmeans_mapped_tbl(uit_tbl)

umap_list(.data = uit_tbl, kmm_tbl, 3)
```

umap_plt *UMAP and K-Means Cluster Visualization*

Description

Create a UMAP Projection plot.

Usage

```
umap_plt(.data, .point_size = 2, .label = TRUE)
```

Arguments

.data	The data from the umap_list() function.
.point_size	The desired size for the points of the plot.
.label	Should ggrepel::geom_label_repel() be used to display cluster user labels.

Details

This takes in umap_kmeans_cluster_results_tbl from the umap_list() function output.

Value

A ggplot2 UMAP Projection with clusters represented by colors.

Author(s)

Steven P. Sanderson II, MPH

See Also

- https://cran.r-project.org/package=uwot (CRAN)
- https://github.com/jlmelville/uwot (GitHub)
- https://github.com/jlmelville/uwot (arXiv paper)

Examples

```
library(healthyR.data)
library(healthyR)
library(dplyr)
library(broom)
library(ggplot2)

data_tbl <- healthyR_data %>%
    filter(ip_op_flag == "I") %>%
    filter(payer_grouping != "Medicare B") %>%
    filter(payer_grouping != "?") %>%
    select(service_line, payer_grouping) %>%
```

```
    mutate(record = 1) %>%
    as_tibble()

uit_tbl <- kmeans_user_item_tbl(
   .data           = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 )

kmm_tbl <- kmeans_mapped_tbl(uit_tbl)

ump_lst <- umap_list(.data = uit_tbl, kmm_tbl, 3)

umap_plt(.data = ump_lst, .point_size = 3)
```

Index