Package 'cheem'

November 8, 2023

Title Interactively Explore Local Explanations with the Radial Tour

Version 0.4.0.0

Description Given a non-linear model, calculate the local explanation.
We purpose view the data space, explanation
space, and model residuals as ensemble graphic interactive on a shiny
application. After an observation of interest is identified, the normalized
variable importance of the local explanation is used as a 1D projection
basis. The support of the local explanation is then explored by changing
the basis with the use of the radial tour <doi:10.32614/RJ-2020-027>;
<doi:10.1080/10618600.1997.10474754>.

Depends R (>= 3.5.0)

License MIT + file LICENSE

URL `https://github.com/nspyrison/cheem/`,
`https://nspyrison.github.io/cheem/`

BugReports `https://github.com/nspyrison/cheem/issues`

Imports spinifex (>= 0.3.3), ggplot2, plotly, magrittr, shiny,
shinythemes, shinycssloaders, DT, conflicted

Suggests methods, tourr, lqmm, mvtnorm, gganimate, dplyr, tidyr,
tictoc, beepr, knitr, testthat (>= 3.0.0), rmarkdown

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Language en-US

Config/testthat/edition 3

NeedsCompilation no

Author Nicholas Spyrison [aut, cre] (<`https://orcid.org/0000-0002-8417-0212`>)

Maintainer Nicholas Spyrison <`spyrison@gmail.com`>

Repository CRAN

Date/Publication 2023-11-08 21:30:02 UTC

R topics documented:

`amesHousing2018` *Ames housing data 2018*

Description

House sales prices from Ames, Iowa, USA between 2006 and 2010. Only complete numeric observations remain.

Usage

```
amesHousing2018

amesHousing2018_raw

amesHousing2018_NorthAmes
```

Format

complete data.frame with 2291 rows and 18 numeric variables, SalesPrice, the response variable, and 3 class variables

An object of class `data.frame` with 2930 rows and 82 columns.

An object of class `data.frame` with 338 rows and 11 columns.

Details

amesHousing2018 Complete data.frame, n = 2291, 18 numeric variable (including 2 temporal: MoSold, YrSold), response variable SalePrice, 3 class factors.

amesHousing2018_NorthAmes A simplified subsample, just North Ames (largest neighborhood). Complete data.frame, n = 338, 9 numeric variables, response variable SalePrice, 1 class factor SubclassMS, a zoning subclass.

amesHousing2018_raw Original data from Kaggle, 2930 rows of 82 variables. Sparse rows (639) and sparse/defaulted columns (64) are removed.

No data dictionary is provided on Kaggle, but amesHousing2018 variables are inferred to be:

- LotFrontage, Length of the front (street facing) side of the lot in yards (0.914m)
- LotArea, Area of the lot in square yards (0.836m^2)
- OverallQual, Overall quality (of the house?), integer in (1, 10)
- OverallCond, Overall condition (of the lot?), integer in (1, 10)
- YearBuild, The year the house was originally built
- BsmtUnfArea, Unfinished basement area, in square yards (0.836m^2)
- TotBsmtArea, Total basement area, in square yards (0.836m^2)
- 1stFlrArea, First (ground) floor living area in square yards (0.836m^2)
- LivingArea, Total living area in square yards (0.836m^2)
- Bathrms, The number of bathrooms
- Bedrms, The number of bedrooms
- TotRms, The total number of rooms
- GarageYrBlt, The year the garage was build
- GarageCars, The number of car spaces in the garage
- GarageArea, The area of the garage in square yards (0.836m^2)
- MoSold, The number of the month of the house sale
- YrSold, The number of the year of the house sale
- SalePrice, The sale of the house in USD (as of the year of sale?)
- SubclassMS, Factor subclass of construction zone, 16 levels
- SubclassMS, Factor major class of construction zone, 7 levels
- Neighborhd, Factor neighborhood of Ames, IA, 28 levels

Source

De Cock, D. (2011). "Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project," *Journal of Statistics Education*, Volume 19, Number 3. http://jse.amstat.org/v19n3/decock/DataDocumentation.txt http://jse.amstat.org/v19n3/decock.pdf

Kaggle, Ames Housing Dataset https://www.kaggle.com/prevek18/ames-housing-dataset

Replicating this dataset:

```
if(FALSE) ## Don't accidentally open the URL.
  browseURL("https://www.kaggle.com/prevek18/ames-housing-dataset")
ames <- readr::read_csv("./buildignore/AmesHousing.csv")
amesHousing2018_raw <- data.frame(ames)
## save(amesHousing2018_raw, file = "./data/amesHousing2018_raw.rda")

## Complete rows and numeric variables
ames1 <- ames[, unlist(lapply(ames, is.numeric))]
ames1$Bathrooms <- ames1$`Full Bath` + ames1$`Half Bath`
ames1 <- ames1[, c(1:18, 38, 19:37)]
col_idx <- !(colnames(ames1) %in% c(
  "Order", "Mas Vnr Area", "BsmtFin SF 1", "BsmtFin SF 2",
  "Bsmt Full Bath", "Bsmt Half Bath", "Fireplaces",
  "Wood Deck SF", "Open Porch SF", "Enclosed Porch",
  "3Ssn Porch", "Screen Porch", "Pool Area", "Misc Val", "2nd Flr SF",
  "Low Qual Fin SF", "Full Bath", "Half Bath", "Kitchen AbvGr"))
row_idx <- !is.na(ames1$"Garage Yr Blt") &
  !is.na(ames1$"Lot Frontage") &
  !is.na(ames1$"Bsmt Unf SF") &
  !is.na(ames1$"Total Bsmt SF")
ames2 <- as.data.frame(ames1[row_idx, col_idx])

## Looking for character classes to keep:
ames_char <- ames[, unlist(lapply(ames, is.character))]
ames_clas <- as.data.frame(lapply(ames_char, factor))[, -1]
ames_clasint <- data.frame(lapply(ames_clas, as.integer))
col_idx_char <- which(names(ames_clas) %in%
                        c("MS.SubClass", "MS.Zoning", "Neighborhood"))
classes <- ames_clas[row_idx, col_idx_char]

amesHousing2018 <- cbind(ames2, classes)
names(amesHousing2018) <- c(
  "LotFrontage", "LotArea","OverallQual", "OverallCond", "YearBuild",
  "YearRemod", "BsmtUnfArea", "TotBsmtArea", "1stFlrArea", "LivingArea",
  "Bathrms", "Bedrms", "TotRms", "GarageYrBlt", "GarageCars", "GarageArea",
  "MoSold", "YrSold", "SalePrice", "SubclassMS", "ZoneMS", "Neighborhd")
## save(amesHousing2018, file = "./data/amesHousing2018.rda")

.thin_col_idx <- names(amesHousing2018) %in% c(
  "LotArea", "OverallQual", "YearBuild",
  "LivingArea", "Bathrms", "Bedrms", "TotRms",
  "GarageYrBlt", "GarageArea", "SalePrice", "SubclassMS")
amesHousing2018_thin <- amesHousing2018[, .thin_col_idx]

## subset to north ames, and only 5 largest subclasses
r_idx <- amesHousing2018$Neighborhd == "NAmes" &
  amesHousing2018$SubclassMS %in% c("020", "050", "080", "090", "060")
amesHousing2018_NorthAmes <- amesHousing2018_thin[r_idx, ]
amesHousing2018_NorthAmes$SubclassMS <- factor(
  amesHousing2018_NorthAmes$SubclassMS,
  unique(amesHousing2018_NorthAmes$SubclassMS))
if(F){ ## Don't accidentally save
 save(amesHousing2018_NorthAmes, file = "./data/amesHousing2018_NorthAmes.rda")
```

Examples

```
library(cheem)

## Regression setup:
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

## Cheem list
ames_rf_chm <- cheem_ls(X, Y, ames_rf_shap, ames_rf_pred, clas,
                        label = "North Ames, RF, SHAP")
## Cheem visuals
if(interactive()){
  prim <- 1
  comp <- 2
  global_view(ames_rf_chm, primary_obs = prim, comparison_obs = comp)
  bas <- sug_basis(ames_rf_chm, prim, comp)
  mv  <- sug_manip_var(ames_rf_chm, primary_obs = prim, comp)
  ggt <- radial_cheem_tour(ames_rf_chm, basis = bas, manip_var = mv)
  animate_plotly(ggt)
}

## Save for use with shiny app (expects an rds file)
if(FALSE){ ## Don't accidentally save.
  saveRDS(ames_rf_chm, "./chm_NAmes_rf_tshap.rds")
  run_app() ## Select the saved rds file from the data drop down.
}
```

`ames_rf_pred` *Ames random forest model predictions and shap values*

Description

Predictions and treeshap attribution of a random forest model of North Ames house sales data regressing Sales Price from house and lot variables.

Usage

```
ames_rf_pred

ames_rf_shap
```

Format

`ames_rf_pred` is a n=338 length vector of the prediction of an random forest model predicting the numeric House Sales in North Ames. `ames_rf_shap` is a (338 x 9) data frame of the treeshap SHAP attribution of the random forest model for each observation.

Replicating

```
library(cheem)
library(randomForest)
library(treeshap)
set.seed(135)

## Regression setup
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

## Model and treeSHAP
ames_rf_fit <- randomForest::randomForest(
  X, Y, ntree = 125,
  mtry = ifelse(is_discrete(Y), sqrt(ncol(X)), ncol(X) / 3),
  nodesize = max(ifelse(is_discrete(Y), 1, 5), nrow(X) / 500))
ames_rf_pred <- predict(ames_rf_fit, X)
ames_rf_shap <- treeshap::treeshap(
  treeshap::randomForest.unify(ames_rf_fit, X), X, FALSE, FALSE)
ames_rf_shap <- ames_rf_shap$shaps

if(F){ ## Don't accidentally save
  save(ames_rf_pred, file = "./data/ames_rf_pred.rda")
  save(ames_rf_shap, file = "./data/ames_rf_shap.rda")
  #usethis::use_data(ames_rf_pred)
  #usethis::use_data(ames_rf_shap)
}
```

An object of class `data.frame` with 338 rows and 9 columns.

Examples

```
library(cheem)

## Regression setup
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

## Precomputed predictions and shap attribution
str(ames_rf_pred)
str(ames_rf_shap)

## Cheem
ames_chm <- cheem_ls(X, Y, ames_rf_shap, ames_rf_pred, clas,
                     label = "Ames, random forest, treeshap")

## Save for use with shiny app (expects an rds file)
if(FALSE){ ## Don't accidentally save.
  saveRDS(ames_chm, "./chm_ames_rf_tshap.rds")
  run_app() ## Select the saved rds file from the data dropdown.
}

## Cheem visuals
if(interactive()){
  prim <- 1
  comp <- 2
  global_view(ames_chm, primary_obs = prim, comparison_obs = comp)
  bas <- sug_basis(ames_rf_shap, prim, comp)
  mv  <- sug_manip_var(ames_rf_shap, primary_obs = prim, comp)
  ggt <- radial_cheem_tour(ames_chm, basis = bas, manip_var = mv)
  animate_plotly(ggt)
}
```

`as_logical_index` *Assure a full length logical index*

Description

Suggests a alpha opacity to plot with as a function of the number of observation.

Usage

```
as_logical_index(index, n)
```

Arguments

`index`	A vector, typically a numeric row index of the data to coerce to a logical index.
`n`	Single numeric, the number of rows of the data use as a replicate return length.

Value

A logical index of length n.

See Also

Other cheem utility: `color_scale_of()`, `contains_nonnumeric()`, `is_discrete()`, `is_diverging()`, `linear_tform()`, `logistic_tform()`, `problem_type()`, `rnorm_from()`, `sug_basis()`, `sug_manip_var()`

Examples

```
library(cheem)

## Coerce a numeric index to logical
as_logical_index(c(1, 4:10, 15), nrow(mtcars))

## Logical indexs are unchanged
as_logical_index(mtcars$mpg > 30, nrow(mtcars))
```

`cheem_ls` *Preprocessing for use in shiny app*

Description

Performs the preprocessing steps needs to supply the plot functions `global_view()` and `radial_cheem_tour()` used in the shiny app. The user need to supply the attributions and predictions. For help getting started with this see `vignette("getting-started-with-cheem")`.

Usage

```
cheem_ls(
  x,
  y = NULL,
  attr_df,
  pred = NULL,
  class = NULL,
  label = "label",
  verbose = getOption("verbose")
)
```

Arguments

`x`	The explanatory variables of the model.
`y`	The target variable of the model.
`attr_df`	A data frame or matrix of (n, p) local explanation attributions.
`pred`	A (n, 1) vector, the predictions from the associated model.
`class`	Optional, (n, 1) vector, a variable to group points by. This can be the same as or different from `y`, the target variable.

`label`	Optionally provide a character label to store reminder text for the type of model and local explanation used. Defaults to "label".
`verbose`	Logical, if start time and run duration should be printed. Defaults to getOption("verbose").

Value

A list of data.frames needed for the `shiny` application.

See Also

`global_view()` `radial_cheem_tour()` `radial_cheem_tour()`

Other cheem preprocessing: `subset_cheem()`

Examples

```
library(cheem)

## Classification setup
X    <- spinifex::penguins_na.rm[, 1:4]
Y    <- spinifex::penguins_na.rm$species
clas <- spinifex::penguins_na.rm$species

## Bring your own attributions and predictions,
## for help review the vignette or package down site;
if(FALSE){
  vignette("getting-started-with-cheem")
 browseURL("https://nspyrison.github.io/cheem/articles/getting-started-with-cheem.html")
}

## Cheem
peng_chm <- cheem_ls(X, Y, penguin_xgb_shap, penguin_xgb_pred, clas,
                     label = "Penguins, xgb, shapviz")

## Save for use with shiny app (expects an rds file)
if(FALSE){ ## Don't accidentally save.
  saveRDS(peng_chm, "./chm_peng_xgb_shapviz.rds")
  run_app() ## Select the saved rds file from the data dropdown.
}

## Cheem visuals
if(interactive()){
  prim <- 1
  comp <- 2
  global_view(peng_chm, primary_obs = prim, comparison_obs = comp)
  bas <- sug_basis(penguin_xgb_shap, prim, comp)
  mv  <- sug_manip_var(penguin_xgb_shap, primary_obs = prim, comp)
  ggt <- radial_cheem_tour(peng_chm, basis = bas, manip_var = mv)
  animate_plotly(ggt)
}

## Regression example
```

```
library(cheem)

## Regression setup
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

#' ## Bring your own attributions and predictions,
## for help review the vignette or package down site;
if(FALSE){
  vignette("getting-started-with-cheem")
 browseURL("https://nspyrison.github.io/cheem/articles/getting-started-with-cheem.html")
}

## Cheem list
ames_rf_chm <- cheem_ls(X, Y, ames_rf_shap, ames_rf_pred, clas,
                        label = "North Ames, RF, treeshap")

## Save for use with shiny app (expects an rds file)
if(FALSE){ ## Don't accidentally save.
  saveRDS(ames_rf_chm, "./chm_NAmes_rf_tshap.rds")
  run_app() ## Select the saved rds file from the data drop down.
}

## Cheem visuals
if(interactive()){
  prim <- 1
  comp <- 2
  global_view(ames_rf_chm, primary_obs = prim, comparison_obs = comp)
  bas <- sug_basis(ames_rf_shap, prim, comp)
  mv  <- sug_manip_var(ames_rf_shap, primary_obs = prim, comp)
  ggt <- radial_cheem_tour(ames_rf_chm, basis = bas, manip_var = mv)
  animate_plotly(ggt)
}
```

`chocolates` *Chocolates dataset*

Description

The chocolates data was compiled by students at Iowa State University of STAT503 (circa 2015) taught by Dianne Cook. Nutrition label information on the chocolates as listed on manufacturer websites. All numbers were normalized to be equivalent to a 100g serving. Units of measurement are listed in the variable name.

Usage

```
chocolates
```

Format

A complete data.frame with 88 observations and 10 numeric variables, name of the chocolate, manufacturer, country, and type of the chocolate.

- Name, the name of the chocolate
- MFR, chocolate manufacturer
- Country, the country the manufacturer is incorporated.
- Type, the type of chocolate according to the website, either 'Dark' or 'Milk"
- Calories, the number of calories per 100 grams
- CalFat, calories from fat per 100 grams
- TotFat_g, grams of total fat per 100 grams
- SatFat_g, grams of saturated fat per 100 grams
- Chol_mg, milligrams of cholesterol per 100 grams
- Na_mg, milligrams of sodium (salt) per 100 grams
- Carbs_g, grams of carbohydrates per 100 grams
- Fiber_g, grams of fiber per 100 grams
- Sugars_g, grams of sugar per 100 grams
- Protein_g, grams of sugar per 100 grams

Source

Monash University, Introduction to Machine Learning course https://iml.numbat.space/

Replicating this dataset:

```
if(FALSE) ## Don't accidentally open the URL.
  browseURL("https://iml.numbat.space/")
## Accessed Jan 2022
chocolates <- readr::read_csv("https://iml.numbat.space/data/chocolates.csv")
chocolates <- data.frame(chocolates)
chocolates[, 2] <- factor(chocolates[, 2])
chocolates[, 3] <- factor(chocolates[, 3])
chocolates[, 4] <- factor(chocolates[, 4])
if(F){ ## Don't accidentally save
  save(chocolates, file = "./data/chocolates.rda")
```

Examples

```
library(cheem)

## Classification setup
X    <- chocolates[, 5:14]
Y    <- chocolates$Type
clas <- chocolates$Type

## Cheem
```

```
choc_chm <- cheem_ls(X, Y, chocolates_svm_shap, chocolates_svm_pred, clas,
                     label = "Chocolates, LM, shap")

## Save for use with shiny app (expects an rds file)
if(FALSE){ ## Don't accidentally save.
  saveRDS(choc_chm, "./chm_chocolates_svm_shap.rds")
  run_app() ## Select the saved rds file from the data dropdown.
}

## Cheem visuals
if(interactive()){
  prim <- 1
  comp <- 2
  global_view(peng_chm, primary_obs = prim, comparison_obs = comp)
  bas <- sug_basis(penguin_xgb_shap, prim, comp)
  mv  <- sug_manip_var(penguin_xgb_shap, primary_obs = prim, comp)
  ggt <- radial_cheem_tour(peng_chm, basis = bas, manip_var = mv)
  animate_plotly(ggt)
}
```

chocolates_svm_pred *Chocolate svm model predictions and shap values*

Description

Predictions and DALEX shap attribution of an svm model of Chocolate data classifying type of chocolate (light/dark).

Usage

```
chocolates_svm_pred

chocolates_svm_shap
```

Format

chocolate_svm_pred is a n=88 length vector of the prediction of an svm model predicting the number of the factor level of the species of penguin. chocolate_svm_shap is a (88 x 10) data frame of the DALEX SHAP attribution of the svm model for each observation.

Replicating

```
library(cheem)
library(e1071)
library(DALEX)
set.seed(135)

## Classification setup
X    <- chocolates[, 5:14]
```

```
Y    <- chocolates$Type
clas <- chocolates$Type

## Model and predict
choc_svm_fit <- svm(
  formula = Y ~ ., data = data.frame(Y, X),
  type = 'C-classification', kernel = 'linear', probability = TRUE)
chocolates_svm_pred <- predict(choc_svm_fit, data.frame(Y, X))

## SHAP via DALEX, versatile but slow
choc_svm_exp <- explain(choc_svm_fit, data = X, y = Y,
                        label = "Chocolates, svm")
## Note that cheem expects a full [n, p] attribution space
## Shap takes about ~30-40 sec for me
chocolates_svm_shap <- matrix(NA, nrow(X), ncol(X)) ## init a df of the same structure
sapply(1:nrow(X), function(i){
  pps <- predict_parts_shap(choc_svm_exp, new_observation = X[i, ])
  ## Keep just the [n, p] local explanations
  chocolates_svm_shap[i, ] <<- tapply(
    pps$contribution, pps$variable, mean, na.rm = TRUE) %>% as.vector()
})
chocolates_svm_shap <- as.data.frame(chocolates_svm_shap)

if(F){ ## Don't accidentally save
  save(chocolates_svm_pred, file = "./data/chocolates_svm_pred.rda")
  save(chocolates_svm_shap, file = "./data/chocolates_svm_shap.rda")
  #usethis::use_data(chocolates_svm_pred)
  #usethis::use_data(chocolates_svm_shap)
}
```

An object of class `data.frame` with 88 rows and 10 columns.

Examples

```
library(cheem)

## Classification setup
X    <- chocolates[, 5:14]
Y    <- chocolates$Type
clas <- chocolates$Type

## Precomputed predictions and shap attribution
str(chocolates_svm_pred)
str(chocolates_svm_shap)

## Cheem
choc_chm <- cheem_ls(X, Y, chocolates_svm_shap,
                     chocolates_svm_pred, clas,
                     label = "Chocolates, SVM, shap")
```

```
## Save for use with shiny app (expects an rds file)
if(FALSE){ ## Don't accidentally save.
  saveRDS(choc_chm, "./cmh_chocolates_svm_shap.rds")
  run_app() ## Select the saved rds file from the data dropdown.
}

## Cheem visuals
if(interactive()){
  prim <- 1
  comp <- 2
  global_view(choc_chm, primary_obs = prim, comparison_obs = comp)
  bas <- sug_basis(chocolates_svm_shap, prim, comp)
  mv  <- sug_manip_var(chocolates_svm_shap, primary_obs = prim, comp)
  ggt <- radial_cheem_tour(choc_chm, basis = bas, manip_var = mv)
  animate_plotly(ggt)
}
```

---

color_scale_of — *Suggest a color and fill scale.*

Description

Whether or not a vector is a diverges a value, returns a logical. Used to help default a scale_color for ggplot2.

Usage

```
color_scale_of(x, mid_pt = 0, limits = NULL, ...)
```

Arguments

`x`	A vector to to scale the color to.
`mid_pt`	A single number checking divergence from. Defaults to 0.
`limits`	A vector of the min and max values for the scale. Useful for setting an absolute range, such as (-1, 1) for attribution/y correlation of each point. Points outside of limits show as default grey. Defaults to NULL; the range of x.
`...`	Optional other arguments passed to ggplot2::continuous_scale or ggplot2::discrete_scale.

Value

A list containing a scale_color, and scale_fill; the suggested color/fill scale for a ggplot.

See Also

Other cheem utility: `as_logical_index()`, `contains_nonnumeric()`, `is_discrete()`, `is_diverging()`, `linear_tform()`, `logistic_tform()`, `problem_type()`, `rnorm_from()`, `sug_basis()`, `sug_manip_var()`

Examples

```
library(cheem)
library(ggplot2)
g <- ggplot(mtcars, aes(disp, mpg))

## Discrete
g + geom_point(aes(color = factor(vs))) +
  color_scale_of(mtcars$vs)
## Sequential increasing
g + geom_point(aes(color = mpg)) +
  color_scale_of(mtcars$mpg)
## Dummy sequential decr
g + geom_point(aes(color = -1 *mpg)) +
  color_scale_of(-1 * mtcars$mpg)
## Dummy diverging
g + geom_point(aes(color = mpg - median(mpg))) +
  color_scale_of(mtcars$mpg - median(mtcars$mpg))
## Dummy limits
g + geom_point(aes(color = mpg - median(mpg))) +
  color_scale_of(mtcars$mpg - median(mtcars$mpg), limits = c(-5, 5))
```

`contains_nonnumeric` *Check if a vector contains non-numeric character*

Description

Returns a logical, whether or not a vector contains any non-numeric characters. Typically used to test if row names hold non-index information.

Usage

```
contains_nonnumeric(x)
```

Arguments

`x`	A vector to be tested for existence of non-numeric characters.

Value

Logical, whether or not x contains any non-numeric characters.

See Also

Other cheem utility: `as_logical_index()`, `color_scale_of()`, `is_discrete()`, `is_diverging()`, `linear_tform()`, `logistic_tform()`, `problem_type()`, `rnorm_from()`, `sug_basis()`, `sug_manip_var()`

Examples

```
library(cheem)

contains_nonnumeric(mtcars$mpg)
contains_nonnumeric(rownames(mtcars)) ## Meaningful info to use in tooltip
contains_nonnumeric(rownames(cars)) ## Assume no meaningful info to use in tooltip
```

`devMessage` *Development message*

Description

Send a message if the 4th chunk of the package version is 9000.

Usage

```
devMessage(text)
```

Arguments

`text`	A character string to message() if package version is 9000.

`global_view` *Linked* `plotly` *display, global view of data and attribution space.*

Description

from a cheem_ls() list, create a linked `plotly` of the global data- and attribution- spaces. Typically consumed directly by shiny app.

Usage

```
global_view(
  cheem_ls,
  primary_obs = NULL,
  comparison_obs = NULL,
  color = c("default", "residual", "log_maha.data", "cor_attr_proj.y"),
  height_px = 480,
  width_px = 1440,
  as_ggplot = FALSE
)
```

Arguments

`cheem_ls`	A return from `cheem_ls()`, a list of data frames.
`primary_obs`	The rownumber of the primary observation. Its local attribution becomes the 1d projection basis, and the point it highlighted as a dashed line. Defaults to NULL, no highlighting applied.
`comparison_obs`	The rownumber of the comparison observation. Point is highlighted as a dotted line. Defaults to NULL, no highlighting applied.
`color`	The name of the column in cheem_ls$global_view_df to map to color. Expects c("default", "residual", "log_maha.data", "cor_attr_proj.y"). Defaults to "default"; predicted_class for classification, dummy class for regression.
`height_px`	The height in pixels of the returned `plotly` plot. Defaults to 480, does nothing when `as_ggplot` is TRUE.
`width_px`	The width in pixels of the returned `plotly` plot. Defaults to 1440, does nothing when `as_ggplot` is TRUE.
`as_ggplot`	Logical, if TRUE returns the plots before being passed to `plotly` functions.

Value

A `plotly` plot, an interactive html widget of the global view, first two components of the basis of the data- and attribution- spaces.

See Also

`run_app()`

Other cheem consumers: `global_view_legwork()`, `radial_cheem_tour()`, `run_app()`

Examples

```
library(cheem)

## Regression setup:
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

## global_view()
ames_rf_chm <- cheem_ls(X, Y, ames_rf_shap, ames_rf_pred, clas,
                        label = "North Ames, RF, SHAP")
if(interactive()){
  global_view(ames_rf_chm, as_ggplot = TRUE) ## early return of ggplot
  global_view(ames_rf_chm) ## uses ggplot facets %>% plotly

  ## Different color mappings, especially for regression
  global_view(ames_rf_chm, color = "residual")
  global_view(ames_rf_chm, color = "log_maha.data")
  global_view(ames_rf_chm, color = "cor_attr_proj.y")
}
## Also consumed by: ?run_app()
```

`global_view_df_1layer` *Create the plot data.frame for the global linked plotly display.*

Description

Internal function consumed in the cheem workflow. Produces the plot data.frame of 1 layer. consumed downstream in cheem_ls.

Usage

```
global_view_df_1layer(x, class = NULL, label = "label")
```

Arguments

`x`	The explanatory variables of the model.
`class`	Optional, (n, 1) vector, a variable to group points by. This can be the same as or different from `y`, the target variable.
`label`	Optionally provide a character label to store reminder text for the type of model and local explanation used. Defaults to "label".

Value

A data.frame, for the global linked **plotly** display.

`global_view_legwork` *The legwork behind the scenes for the global view*

Description

Not meant for end user. This is an internal legwork function to keep from repeating global_view code.

Usage

```
global_view_legwork(
  cheem_ls,
  primary_obs = NULL,
  comparison_obs = NULL,
  color = c("default", "residual", "log_maha.data", "cor_attr_proj.y")
)
```

Arguments

`cheem_ls`	A return from `cheem_ls()`, a list of data frames.
`primary_obs`	The rownumber of the primary observation. Its local attribution becomes the 1d projection basis, and the point it highlighted as a dashed line. Defaults to NULL, no highlighting applied.
`comparison_obs`	The rownumber of the comparison observation. Point is highlighted as a dotted line. Defaults to NULL, no highlighting applied.
`color`	The name of the column in cheem_ls$global_view_df to map to color. Expects c("default", "residual", "log_maha.data", "cor_attr_proj.y"). Defaults to "default"; predicted_class for classification, dummy class for regression.

Value

a `ggplot2` object, waiting to be rendered with `global_view` or `global_view_subplots`.

See Also

Other cheem consumers: `global_view()`, `radial_cheem_tour()`, `run_app()`

`ifDev` *Evaluate if development*

Description

Evaluate the expression if the 4th chunk of the package version is 9000.

Usage

```
ifDev(expr)
```

Arguments

`expr`	A character string to message() if package version is 9000.

`is_discrete` *Check if a vector is discrete*

Description

Whether or not a vector is a discrete variable, returns a logical. Typically used on the Y variable of a model.

Usage

```
is_discrete(x, na.rm = TRUE)
```

Arguments

`x`	A vector to check the discreteness of.
`na.rm`	Whether or not to remove NA values before testing discreteness. Defaults to TRUE.

Value

Logical, whether or not `x` is a discrete variable.

See Also

Other cheem utility: `as_logical_index()`, `color_scale_of()`, `contains_nonnumeric()`, `is_diverging()`, `linear_tform()`, `logistic_tform()`, `problem_type()`, `rnorm_from()`, `sug_basis()`, `sug_manip_var()`

Examples

```
library(cheem)

is_discrete(mtcars$mpg) ## Numeric column, with more than 25 unique values.
is_discrete(mtcars$cyl) ## Numeric column, labeled as discrete, because less than 25 unique values
is_discrete(letters)    ## Characters and factors labeled discrete.
```

`is_diverging` *Check if a vector diverges a value*

Description

Whether or not a vector is a diverges a value, returns a logical. Used to help default a scale_color for ggplot2.

Usage

```
is_diverging(x, mid_pt = 0)
```

Arguments

`x`	A vector to check the divergence of.
`mid_pt`	A single number checking divergence from. Defaults to 0.

Value

Logical, whether or not `x` is a diverges `mid_pt`.

See Also

Other cheem utility: `as_logical_index()`, `color_scale_of()`, `contains_nonnumeric()`, `is_discrete()`, `linear_tform()`, `logistic_tform()`, `problem_type()`, `rnorm_from()`, `sug_basis()`, `sug_manip_var()`

Examples

```
library(cheem)

is_diverging(-10:10)
is_diverging(-10:-5)
is_diverging(mtcars$mpg, 25)
is_diverging(mtcars$mpg, 40)
```

`linear_tform` *Linear function to help set alpha opacity*

Description

Suggests a alpha opacity to plot with as a function of the number of observation.

Usage

```
linear_tform(n, appox_max_n = 5000, ceiling = 1, floor = 0.3)
```

Arguments

`n`	Number of observations to plot.
`appox_max_n`	The number of observation to reach floor opacity.
`ceiling`	The highest number returned. Defaults to 1.
`floor`	The lowest number returned. Defaults to 0.3.

Value

A scalar numeric, suggested value to set alpha opacity.

See Also

Other cheem utility: `as_logical_index()`, `color_scale_of()`, `contains_nonnumeric()`, `is_discrete()`, `is_diverging()`, `logistic_tform()`, `problem_type()`, `rnorm_from()`, `sug_basis()`, `sug_manip_var()`

Examples

```
library(cheem)

## Suggest an opacity to use in plotting:
(my_alpha <- linear_tform(nrow(spinifex::penguins_na.rm)))

## Visualize
x <- 1:2000
plot(x, sapply(x, linear_tform), col = "blue")
```

---

logistic_tform *Logistic function to help set alpha opacity*

Description

Suggests a alpha opacity to plot with as a function of the number of observation.

Usage

```
logistic_tform(n, mid_pt = 600, k_attenuation = 5, ceiling = 1, floor = 0.3)
```

Arguments

n	Number of observations to plot.
mid_pt	Inflection point that the logistic curve. Defaults to 600.
k_attenuation	The steepness of the transition, larger is a sharper transition. Quite sensitive and defaults to 5.
ceiling	The highest number returned. Defaults to 1.
floor	The lowest number returned. Defaults to 0.3.

Value

A scalar numeric, suggested value to set alpha opacity.

See Also

Other cheem utility: as_logical_index(), color_scale_of(), contains_nonnumeric(), is_discrete(), is_diverging(), linear_tform(), problem_type(), rnorm_from(), sug_basis(), sug_manip_var()

Examples

```
library(cheem)

## Suggest an opacity to use in plotting:
(my_alpha <- logistic_tform(nrow(spinifex::penguins_na.rm)))

## Visualize
x <- 1:2000
plot(x, logistic_tform(x), col = "blue")
```

`model_performance` *Extract higher level model performance statistics*

Description

Internal function, used downstream in cheem_ls.

Usage

```
model_performance(y, pred, label = "label")
```

Arguments

`y`	Observed response, required by ranger models.
`pred`	A (n, 1) vector of the predicted values for each observation. Typically obtainable with `stats::predict()`. Exact syntax and arguments may change across model types.
`label`	Optionally provide a character label to store reminder text for the type of model and local explanation used. Defaults to "label".

Value

A data.frame of model performance statistics.

`penguin_xgb_pred` *Penguins xgb model predictions and shap values*

Description

Predictions and shapviz attribution of an xgb model of Penguin data classifying penguin species.

Usage

```
penguin_xgb_pred

penguin_xgb_shap
```

Format

`penguin_xgb_pred` is a n=333 length vector of the prediction of an xgb model predicting the number of the factor level of the species of penguin. `penguin_xgb_shap` is a (333 x 4) data frame of the shapviz SHAP attribution of the xgb model for each observation.

Replicating

```
library(cheem)
library(xgboost)
library(shapviz)
set.seed(135)

## Classification setup
X    <- spinifex::penguins_na.rm[, 1:4]
Y    <- spinifex::penguins_na.rm$species
clas <- spinifex::penguins_na.rm$species

## Model and predict
peng_train    <- data.matrix(X) %>%
  xgb.DMatrix(label = Y)
peng_xgb_fit  <- xgboost(data = peng_train, max.depth = 3, nrounds = 5)
penguin_xgb_pred <- predict(peng_xgb_fit, newdata = peng_train)

## shapviz
penguin_xgb_shap <- shapviz(peng_xgb_fit, X_pred = peng_train, X = X)
penguin_xgb_shap <- penguin_xgb_shap$S

if(F){ ## Don't accidentally save
  save(penguin_xgb_pred, file = "./data/penguin_xgb_pred.rda")
  save(penguin_xgb_shap, file = "./data/penguin_xgb_shap.rda")
  #usethis::use_data(penguin_xgb_pred)
  #usethis::use_data(penguin_xgb_shap)
}
```

An object of class `matrix` (inherits from `array`) with 333 rows and 4 columns.

Examples

```
library(cheem)

## Classification setup
X    <- spinifex::penguins_na.rm[, 1:4]
Y    <- spinifex::penguins_na.rm$species
clas <- spinifex::penguins_na.rm$species

## Precomputed predictions and shap attribtion
str(penguin_xgb_pred)
str(penguin_xgb_shap)

## Cheem
peng_chm <- cheem_ls(X, Y, penguin_xgb_shap, penguin_xgb_pred, clas,
                     label = "Penguins, xgb, shapviz")

## Save for use with shiny app (expects an rds file)
if(FALSE){ ## Don't accidentally save.
  saveRDS(peng_chm, "./chm_peng_xgb_shapviz.rds")
  run_app() ## Select the saved rds file from the data dropdown.
}

## Cheem visuals
if(interactive()){
  prim <- 1
  comp <- 2
  global_view(peng_chm, primary_obs = prim, comparison_obs = comp)
  bas <- sug_basis(penguin_xgb_shap, prim, comp)
  mv  <- sug_manip_var(penguin_xgb_shap, primary_obs = prim, comp)
  ggt <- radial_cheem_tour(peng_chm, basis = bas, manip_var = mv)
  animate_plotly(ggt)
}
```

`problem_type` *The type of model for a given Y variable*

Description

Whether the Y is a "classification", "regression" or ill-defined problem. Returns a character: "classification", "regression", or an error for strange classes. Minor redundancy with is_discrete, though explicit. Could be useful for `DALEX::explain(type)` as it also expects "classification" or "regression".

Usage

```
problem_type(y)
```

Arguments

`y`	Response variable to be modeled

Value

Character either c("classification", "regression") specifying the assumed model task based on the discreteness of y.

See Also

Other cheem utility: `as_logical_index()`, `color_scale_of()`, `contains_nonnumeric()`, `is_discrete()`, `is_diverging()`, `linear_tform()`, `logistic_tform()`, `rnorm_from()`, `sug_basis()`, `sug_manip_var()`

Examples

```
library(cheem)

problem_type(mtcars$mpg) ## Numeric, many values
problem_type(mtcars$cyl) ## Numeric column, labeled as discrete, because less than 25 unique values
problem_type(letters) ## Character to classification
problem_type(factor(letters)) ## Factor to classification
```

proto_basis1d_distribution

Adds the distribution of the row local attributions to a ggtour

Description

A spinifex proto_*-like function, that adds the distribution of orthonormalized row values of the specified local explanation `attr_df`. Does not draw the basis bars; use in conjunction with `proto_basis1d()`.

Usage

```
proto_basis1d_distribution(
  attr_df,
  primary_obs = NULL,
  comparison_obs = NULL,
  position = c("top1d", "floor1d", "bottom1d", "off"),
  group_by = as.factor(FALSE),
  pcp_shape = c(3, 142, 124),
  do_add_pcp_segments = TRUE,
  inc_var_nms = NULL,
  row_index = NULL
)
```

Arguments

`attr_df`	An data frame, the attributions of a local explanation.
`primary_obs`	The rownumber of the primary observation. Its local attribution becomes the 1d projection basis, and the point it highlighted as a dashed line. Defaults to NULL, no highlighting.
`comparison_obs`	The rownumber of the comparison observation. Point is highlighted as a dotted line. Defaults to NULL, no highlighting.
`position`	The position for the basis, one of: c("top1d", "floor1d", "bottom1d", "off"). Defaults to "top1d"; basis above the density curves.
`group_by`	Vector to group densities by. Originally *predicted* class.
`pcp_shape`	The number of the shape character to add. Expects 3, 142, or 124, '+', '|' in `plotly`, and '|' in `gganimate`, respectively. Defaults to 3, '+' in either output.
`do_add_pcp_segments`	Logical, whether or not to add to add faint parallel coordinate lines on the 1D basis. Defaults to TRUE.
`inc_var_nms`	A character vector, the names of the variables to keep. Defaults to NULL, all variables kept.
`row_index`	A numeric or logical vector, the index of the rows to keep. Defaults to NULL, all rows kept.

Value

A ggplot object of the the distribution of the local explanation's attributions.

See Also

radial_cheem_tour()

Examples

```
library(cheem)
library(spinifex)

## Regression setup
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

## Basis, manipulation var, manual tour path, & predictions to fix to y-axis
bas     <- sug_basis(ames_rf_shap, 1)
mv      <- sug_manip_var(ames_rf_shap, 1, 2)
mt_path <- manual_tour(bas, mv)
## Also consumed by: ?radial_cheem_tour()

## Compose and animate the tour
ggt <- ggtour(mt_path, scale_sd(X), angle = .3) +
  append_fixed_y(fixed_y = scale_sd(Y)) +
  proto_point(list(color = clas, shape = clas)) +
  proto_basis1d_distribution(
    attr_df = ames_rf_shap,
    primary_obs = 1, comparison_obs = 2,
    position = "top1d", group_by = clas) +
  proto_basis1d(position = "bottom1d") +
  proto_origin()

if(interactive()){
  animate_plotly(ggt)
}
```

---

radial_cheem_tour *Cheem tour; 1D manual tour on the selected attribution*

Description

Create a linked plotlyof the global data- and attribution- spaces. Typically consumed directly by shiny app.

Usage

```
radial_cheem_tour(
  cheem_ls,
  basis,
  manip_var,
  primary_obs = NULL,
  comparison_obs = NULL,
  do_add_pcp_segments = TRUE,
  pcp_shape = c(3, 142, 124),
  angle = 0.15,
  row_index = NULL,
  inc_var_nms = NULL,
  do_center_frame = TRUE,
  do_add_residual = FALSE
)
```

Arguments

`cheem_ls`	A return from `cheem_ls()`, a list of data frames.
`basis`	A 1D projection basis, typically a return of `sug_basis()`.
`manip_var`	The , *number* of the manipulation variable.
`primary_obs`	The rownumber of the primary observation. Its local attribution becomes the 1d projection basis, and the point it highlighted as a dashed line. Defaults to NULL, no primary observation highlighted.
`comparison_obs`	The rownumber of the comparison observation. Point is highlighted as a dotted line. Defaults to NULL, no comparison observation highlighted.
`do_add_pcp_segments`	Logical, whether or not to add parallel coordinate line segments to the basis display.
`pcp_shape`	The number of the shape character to add. Expects 3, 142, or 124, '+', '|' in `plotly`, and '|' in `gganimate`, respectively. Defaults to 3, '+' in either output.
`angle`	The step size between interpolated frames, in radians. Defaults to .15.
`row_index`	Numeric index of selected observations. Defaults to TRUE; 1:n.
`inc_var_nms`	A vector of the names of the variables to include in the projection.
`do_center_frame`	Whether or not to scale by standard deviations away from the mean within each frame or not. Defaults to TRUE, helping to keep the animation centered.
`do_add_residual`	Whether of not to add a facet with a fixed y on residual. Doing so may cause issues with animation. Defaults to FALSE.

Value

ggtour (`ggplot2` object with frame info) animation frames of a radial tour manipulating the contribution of a selected tour. Animated with `spinifex::animate_*` functions.

See Also

run_app()

Other cheem consumers: global_view_legwork(), global_view(), run_app()

Examples

```
library(cheem)

## Regression setup:
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

## radial_cheem_tour()
ames_rf_chm <- cheem_ls(X, Y, ames_rf_shap, ames_rf_pred, clas,
                        label = "North Ames, RF, SHAP")
bas <- sug_basis(ames_rf_shap, 1)
mv  <- sug_manip_var(ames_rf_shap, 1, 2)
ggt <- radial_cheem_tour(ames_rf_chm, basis = bas, manip_var = mv)
if(interactive()){
  ## As a plotly html widget
  spinifex::animate_plotly(ggt)

  ## As a gganimation
  spinifex::animate_gganimate(ggt, render = gganimate::av_renderer())
}
## Also used in: ?run_app()
```

`rnorm_from` *Draw new samples from the supplied data given its mean and covariances.*

Description

Creates new observation of the data given its specific means and shapes. typically applied to a cluster subset of data. *ie* draw from cluster 'a', then assign to cluster 'b'.

Usage

```
rnorm_from(
  data,
  n_obs = 1,
  var_coeff = 1,
  method = c("pearson", "kendall", "spearman")
)
```

Arguments

`data`	A data.frame or matrix to sample from.
`n_obs`	Number of new observations to draw. Defaults to 1.
`var_coeff`	Variance coefficient, closer to 0 make points near the median, above 1 makes more points further away from the median. Defaults to 1.
`method`	The method of the covariance matrix. Expects "person" (continuous numeric), "kendall" or "spearman" (latter two are ranked based ordinal).

Value

A data.frame, sampled observations given the means and covariance of the data based on with column names kept.

See Also

Other cheem utility: `as_logical_index()`, `color_scale_of()`, `contains_nonnumeric()`, `is_discrete()`, `is_diverging()`, `linear_tform()`, `logistic_tform()`, `problem_type()`, `sug_basis()`, `sug_manip_var()`

Examples

```
library(cheem)

sub <- mtcars[mtcars$cyl == 6, ]
## Draw 3 new observations in the shape of 6 cylinder vehicles, with reduced variance.
rnorm_from(data = sub, n_obs = 3, var_coeff = .5)
```

`run_app` *Runs a shiny app demonstrating manual tours*

Description

Runs a local shiny app that demonstrates manual tour and comparable traditional techniques for static projections of multivariate data sets.

Usage

```
run_app(app_nm = "cheem", ...)
```

Arguments

`app_nm`	Name of the shiny app to run. Expects "cheem_initial".
`...`	Other arguments passed into `shiny::runApp()`. Such as display.mode = "showcase".

Value

Runs a locally hosted shiny app.

See Also

Other cheem consumers: global_view_legwork(), global_view(), radial_cheem_tour()

Examples

```
## Only run this example in interactive R sessions
if(interactive()){
  ## Runs the app
  run_app("cheem")

  ## Run with app code displayed
  run_app(app_nm = "cheem", display.mode = "showcase")
}
```

---

subset_cheem *Subset a cheem list*

Description

Given a numerical index of rownumbers of the original data, subset the correct elements of a cheem list.

Usage

```
subset_cheem(cheem_ls, rownumbers = 1:500)
```

Arguments

cheem_ls	The return of a cheem_ls() call.
rownumbers	A vector of the numeric index of rownumbers to keep. To use a logical index, pass it to which(), eg. idx <- which(mtcars$mpg > 30).

Value

A subset of the supplied cheem_ls.

See Also

cheem_ls()

Other cheem preprocessing: cheem_ls()

Examples

```
library(cheem)

## Classification setup
X    <- spinifex::penguins_na.rm[, 1:4]
Y    <- spinifex::penguins_na.rm$species
clas <- spinifex::penguins_na.rm$species

## Cheem
peng_chm <- cheem_ls(X, Y, penguin_xgb_shap, penguin_xgb_pred, clas,
                     label = "Penguins, xgb, shapviz")
lapply(peng_chm, NROW)

## subset a cheem list
num_idx <- which(spinifex::penguins_na.rm$flipper_length_mm > 185)
peng_chm_sub <- subset_cheem(peng_chm, num_idx)
lapply(peng_chm_sub, NROW)
## Notice that $global_view_df and $decode_df have fewer rows.
```

`sug_basis` *Suggest a 1D Basis*

Description

Extract and formats the 1D attribution basis from the provided local explanation.

Usage

```
sug_basis(attr_df, rownum)
```

Arguments

`attr_df`	A data frame of local explanation attributions.
`rownum`	The rownumber of the observation. Typically primary or comparison observations.

Value

A matrix of the 1D basis.

See Also

`radial_cheem_tour()`

Other cheem utility: `as_logical_index()`, `color_scale_of()`, `contains_nonnumeric()`, `is_discrete()`, `is_diverging()`, `linear_tform()`, `logistic_tform()`, `problem_type()`, `rnorm_from()`, `sug_manip_var()`

Examples

```
library(cheem)

## Attribution basis of the primary instance
sug_basis(ames_rf_shap, rownum = 1)


## This can be used to find a basis to start the radial tour.
# ?radial_cheem_tour

## Regression setup:
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

## radial_cheem_tour()
ames_rf_chm <- cheem_ls(X, Y, ames_rf_shap, ames_rf_pred, clas,
                        label = "North Ames, RF, SHAP")
bas <- sug_basis(ames_rf_shap, 1)
mv  <- sug_manip_var(ames_rf_shap, 1, 2)
ggt <- radial_cheem_tour(ames_rf_chm, basis = bas, manip_var = mv)
if(interactive()){
  ## As a plotly html widget
  spinifex::animate_plotly(ggt)

  ## As a gganimation
  spinifex::animate_gganimate(ggt, render = gganimate::av_renderer())
}
## radial_cheem_tour is also used in: ?run_app()
```

sug_manip_var *Suggest a manipulation variable*

Description

Suggest the number of the variable with the largest difference in attribution between the primary and comparison observations.

Usage

```
sug_manip_var(attr_df, primary_obs, comparison_obs)
```

Arguments

attr_df	A data frame of local explanation attributions.
primary_obs	The rownumber of the primary observation. Its local attribution becomes the 1d projection basis, and the point it highlighted as a dashed line.
comparison_obs	The rownumber of the comparison observation. Point is highlighted as a dotted line.

Value

A single number of the variable with the largest difference of attribution (basis) variables.

See Also

radial_cheem_tour()

Other cheem utility: as_logical_index(), color_scale_of(), contains_nonnumeric(), is_discrete(), is_diverging(), linear_tform(), logistic_tform(), problem_type(), rnorm_from(), sug_basis()

Examples

```
library(cheem)

## Regression setup
dat  <- amesHousing2018_NorthAmes
X    <- dat[, 1:9]
Y    <- dat$SalePrice
clas <- dat$SubclassMS

## Suggest the number of a variable to manipulate
sug_manip_var(ames_rf_shap, primary_obs = 1, comparison_obs = 2)
## This can be used to find a basis to start the radial tour.
# ?radial_cheem_tour
```

Index