R/Python Parity

The following is a list of R functions alongside the equivalent Python ones. Most methods in Python are member methods of the of the H2OFrame class. H2O does not override native Python `all` or `any` methods but these are included as member methods (e.g., `myFrame[0].any()` not `any(myFrame[0])`).

Similarly, model accessor methods are members of their respective classes.

Note: This is not a complete listing of the R or Python H2O API. Please refer to the R or Python documentation.

H2O Algorithms:

R	Python	Function Description
`h2o.deeplearning`	`h2o.estimators.deeplearning`	Creates a Deep Learning model.
`h2o.gbm`	`h2o.estimators.gbm`	Creates a Gradient Boosting Machine model.
`h2o.glm`	`h2o.estimators.glm`	Creates a Generalized Linear model.
`h2o.glrm`	`h2o.estimators.glrm`	Creates a Generalized Low Rank model.
`h2o.kmeans`	`h2o.estimators.kmeans`	Creates a K-means model.
`h2o.naiveBayes`	`h2o.estimators.naive_bayes`	Computes Naive Bayes probabilities on an H2O dataset.
`h2o.randomForest`	`h2o.estimators.random_forest`	Creates a Distributed Random Forest model.

H2OFrame Operations:

R	Python	Function Description
`as.character.H2OFrame`	`ascharacter`	Converts the column to characters.
`as.data.frame.H2OFrame`	`as_data_frame`	Returns the dataset as an R or Python object.
`as.date.H2OFrame`	`as_date`	Return the column with all elements converted to millis since the epoch.
`as.factor`	`asfactor`	Converts a column into a factor column.
`as.matrix.H2OFrame`		Converts an H2OFrame to a matrix.
`as.numeric`	`asnumeric`	Converts factor columns to numbers (numeric columns unchanged).
`as.vector.H2OFrame`		Convers an H2OFrame to a vector.
`colnames`	`col_names`	The column names of an H2OFrame.
`dim.H2OFrame`	`dim`	Returns the number of rows and columns in the H2OFrame.
`dimnames.H2OFrame`	`names`	Column names of an H2OFrame.
`h2o.anyFactor`	`anyfactor`	Returns whether or not the frame has factor columns.
`h2o.assign`	`assign`	Copies the data frame and assigns it the specified key.
`h2o.biases`	`biases`	(for Deep Learning) Returns the frame for the respective bias vector.
`h2o.cbind`	`cbind`	Takes a sequence of H2O datasets and combines them by column.
`h2o.clearLog`	`h2o.clear_log`	Clears all H2O R command and error response logs from the local disk for debugging.

h2o.clusterIsUp	cluster_is_up	Returns `true` if the cluster is up; `false` otherwise.
h2o.clusterStatus	cluster_status	Checks node status for the node you are connected to.
h2o.createFrame	h2o.create_frame	Creates an H2O data frame with real-valued, categorical, integer, and binary columns as specified.
h2o.cut	cut	Cuts a numeric vector into factor “buckets”. Similar to R’s cut method.
h2o.day	day	Returns a new day column from a msec-since-Epoch column.
h2o.dayOfWeek	dayOfWeek	Returns a new Day-of-Week column from a msec-since-Epoch column.
h2o.ddply	ddply	For each subset of an H2O data set, apply a user-specified function, then combine the results. **Caution**: This is an experimental feature.
h2o.describe	describe	Generates an in-depth description of the H2OFrame, including everyting in `summary()` plus the data layout.
h2o.downloadAllLogs	download_all_logs	Downloads H2O log Files to the disk.
h2o.downloadCSV	download_csv	Downloads an H2O data set to a CSV file on the local disk. **Caution**: Files located on the H2O server may be very large! Make sure you have enough hard drive space to accommodate the entire file.
h2o.entropy	entropy	For each string, return the Shannon entropy. If the string is empty, the entropy is 0.
h2o.exportFile	export_file	Export a given H2OFrame (which can be either VA or FV) to a path on the machine this python session is currently connected to.
h2o.filterNACols	filter_na_cols	Filter columns that have a proportion of NAs which is >= *frac*
h2o.getTimezone	h2o.get_timezone	Returns the time zone for the H2O Cloud.
h2o.groupBy	group_by	Returns a new GroupBy object using this frame and the desired grouping columns. The returned groups are sorted by the natural group-by column sort.
h2o.gsub	gsub	Replaces all matches. **Note**: Changes the frame.
h2o.head	head	Returns the first or last rows of an H2OFrame object. Analogous to R’s head call on a data.frame.
h2o.hist	hist	Computes a histogram over a numeric column. If breaks==“FD”, the MAD is used over the IQR in computing bin width.
h2o.hit_ratio_table	hit_ratio_table	Retrieves the Hit Ratios.
h2o.hour	hour	Returns a new Hour-of-Day column from a msec-since-Epoch column.
h2o.ifelse	ifelse	Equivalent to [y if t else n for t,y,n in zip(self,yes,no)]. Based on the booleans in the test vector, the output has the values of the yes and no vectors interleaved (or merged together). All Frames must have the same row count. Single column frames are broadened to match wider Frames. Scalars are allowed, and are also broadened to match wider frames.

`h2o.impute`	`impute`	Perform in-place imputation by filling missing values with aggregates computed on the “na.rm’d” vector. Additionally, it is possible to perform imputation based on groupings of columns from within data; these columns can be passed by index or name to the by parameter. If a factor column is supplied, then the method must be `mode`.
`h2o.insertMissingValues`	`insert_missing_values`	Primarily used for testing. Randomly replaces a user-specified fraction of entries in a H2O dataset with missing values. **Caution**: This will modify the original dataset. Unless this is intended, this function should only be called on a subset of the original.
`h2o.interaction`	`h2o.interaction`	Creates an H2O frame with n-th order interaction features between categorical columns.
`h2o.is.character`	`ischaracter`	Returns `true` if the column is a character column, otherwise `false`. (In Python, this is the same as `isstring`.)
`h2o.is.factor`	`isfactor`	Returns `true` if this vector is a factor.
`h2o.is.numeric`	`isnumeric`	Returns `true` if the column is numeric, otherwise returns `false`.
`h2o.kfold_column`	`kfold_column`	Build a fold assignments column for cross-validation. This call will produce a column having the same data layout as the calling object.
`h2o.levels`	`levels`	Returns the factor levels for this frame and the specified column index.
`h2o.listTimezones`	`h2o.list_timezones`	Returns a list of all the timezones.
`h2o.loadModel`	`h2o.load_model`	Loads a saved H2O model from the disk.
`h2o.ls`	`ls`	Lists Keys on an H2O Cluster.
`h2o.lstrip`	`lstrip`	Strip set from the left, then return a copy of the target column with leading characters removed. The set argument is a string specifying the set of characters to be removed. If omitted, the set argument defaults to removing whitespace.
`h2o.match`	`match`	Creates a column of the positions of the first matches of its first argument in its second.
`h2o.mean`	`mean`	Returns the mean of the column.
`h2o.median`	`median`	Returns the median of this column.
`h2o.merge`	`merge`	Merge two datasets based on common column names. The two datasets must have at least one common column.
`h2o.mktime`	`mktime`	Computes msec since the Unix Epoch.
`h2o.month`	`month`	Returns a new month column from an msec-since-Epoch column.
`h2o.nacnt`	`nacnt`	Returns the number of NAs per column.
`h2o.nchar`	`nchar`	Returns the number of characters in each string of a single-column H2OFrame.

h2o.nlevels	nlevels	Returns the number of factor levels for this frame and the specified column index.
h2o.num_valid_substrings	num_valid_substrings	Returns the count of all possible substrings >= 2 chars that are contained in the specified line-separated text file.
h2o.quantile	quantile	Computes quantiles over a given H2OFrame.
h2o.rbind	rbind	Combines H2O Datasets by Rows; takes a sequence of H2O data sets and combines them by rows.
h2o.relevel	relevel	Reorders levels of an H2O factor, similarly to standard R's relevel(). The levels of a factor are reordered such that the reference level is at level 0, remaining levels are moved down as needed.
h2o.rep_len	rep_len	Replicates the values in `data` in the H2O backend.
h2o.round	round	Returns the rounded values in the H2OFrame to the specified number of decimal digits.
h2o.rstrip	rstrip	Strip set from the right, then return a copy of the target column with leading characters removed. The set argument is a string specifying the set of characters to be removed. If omitted, the set argument defaults to removing whitespace.
h2o.runif	runif	Returns new H2OVec filled with doubles sampled uniformly from (0,1).
h2o.scale	scale	Centers and/or scales the columns of the H2OFrame.
sd	sd	Returns the standard deviation of column data.
h2o.setLevel	set_level	Sets all column values to one of the levels.
h2o.setLevels	set_levels	Applicable on a single categorical column. New domains must be aligned with the old domains. **Note**: This call does not copy the file, it changes the column in place.
h2o.setTimezone	set_timezone	Sets the time zone for the H2O Cloud.
h2o.signif	signif	Returns the rounded values in the H2OFrame to the specified number of significant digits.
h2o.splitFrame	split_frame	Splits a frame into distinct subsets of size determined by the given ratios. The number of subsets is always 1 more than the number of ratios given.
h2o.strsplit	strsplit	Splits the strings in the target column using the specifed pattern.
h2o.sub	sub	Substitutes the first occurrence of pattern in a string with replacement. **Note**: Changes the frame.
h2o.substring	substring	For each string, return a new string that is a substring of the original string. If end_index is not specified, then the substring extends to the end of the original string. If the start_index is longer than the length of the string or is greater than or equal to the end_index, an empty string is returned. Negative start_index is coerced to 0.
h2o.summary	summary	Summarizes the columns of an H2O data frame or subset of columns and rows using vector notation

		(e.g. dataset[row, col]. Summary includes min/mean/max/sigma and other rollup data.
`h2o.table`	`table`	Compute the counts of values appearing in a column, or co-occurence counts between two columns, then returns a frame of the counts at each combination of factor levels.
`h2o.tail`	`tail`	Displays a digestible chunk of the H2OFrame starting from the end. Analogous to R's tail call on a data.frame.
`h2o.tolower`	`tolower`	Translates characters from upper to lower case for a particular column. **Note**: Changes the frame.
`h2o.toupper`	`toupper`	Translates characters from lower to upper case for a particular column. **Note**: Changes the frame.
`h2o.trim`	`trim`	(Applicable only to frames with one column) Trims the edge-spaces in a column of strings.
`h2o.unique`	`unique`	Extracts the unique values in a column.
`h2o.var`	`var`	Returns the variance or covariance matrix of the columns in this H2OFrame.
`h2o.week`	`week`	Converts the entries of an H2OFrame object from milliseconds to weeks of the week year (starting from 1).
`h2o.which`	`h2o.which`	Returns the H2OFrame of 1 column filled with 0-based indices for which the condition is True.
`h2o.year`	`year`	Convert the entries of an H2OFrame object from milliseconds to years, indexed starting from 1900.
`is.na`	`isna`	Returns a new boolean H2OVec.
`na.omit.H2OFrame`	`na_omit`	Removes rows with NAs from the H2OFrame.
`names.H2OFrame`	`names`	Column names of an H2OFrame.
`ncol.H2OFrame`	`ncol`	Returns the number of columns in this H2OFrame.
`nrow.H2OFrame`	`nrow`	Returns the number of rows in this H2OFrame.
`str.H2OFrame`	`structure`	Compactly display the structure of this H2OFrame.

H2O Model Operations:

R	Python	Function Description
`h2o.accuracy`	`accuracy`	Returns the accuracy for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.aic`	`aic`	Retrieves the AIC for this set of metrics.

`h2o.anomaly`	`anomaly`	Obtain the reconstruction error for the input test_data.
`h2o.auc`	`auc`	Retrieves the AUC for this set of metrics.
`h2o.betweenss`	`betweenss`	Returns the between cluster sum of squares. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.centers`	`centers`	Returns the centers for the kmeans model.
`h2o.centersSTD`	`centers_std`	Returns the standardized centers for the kmeans model.
`h2o.centroid_stats`	`centroid_stats`	Returns the centroid statistics for each cluster. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.coef`	`coef`	Returns the coefficients for this model.
`h2o.coef_norm`	`coef_norm`	Returns the normalized coefficients.
`h2o.confusionMatrix`	`confusion_matrix`	Returns the confusion matrix for the specified metrics/thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.cross_validation_fold_assignment`	`cross_validation_fold_assignment`	Retrieves the cross-validation fold assignment for all rows in the training data.

`h2o.cross_validation_holdout_predictions`	`cross_validation_holdout_predictions`	Retrieves the (out-of-sample) holdout predictions of all cross-validation models on the training data. This is equivalent to summing up all H2OFrames returned by cross_validation_predictions.
`h2o.cross_validation_models`	`cross_validation_models`	Retrieves a list of cross-validation models.
`h2o.cross_validation_predictions`	`cross_validation_predictions`	Retrieves the (out-of-sample) holdout predictions of all cross-validation models on their holdout data. Note that the predictions are expanded to the full number of rows of the training data, with 0 fill-in.
`h2o.deepfeatures`	`deepfeatures`	Returns hidden layer details.
`h2o.download_pojo`	`download_pojo`	Downloads the POJO for this model to the specified path directory - do not use a trailing slash. If path is `""`, then dump to screen.
`h2o.F0point5`	`F0point5`	Returns the F0.5 for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.F1`	`F1`	Returns the F1 for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.F2`	`F2`	Returns the F2 for a set of thresholds. If all are False (default), then return the

		training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.find_threshold_by_max_metric`	`find_threshold_by_max_metric`	If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.fnr`	`fnr`	Returns the False Negative Rates for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.fpr`	`fpr`	Returns the False Positive Rates for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.gainsLift`	`gains_lift`	Get the Gains/Lift table for the specified metrics. If all are False (default), then return the training metric Gains/Lift table. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.getGLMFullRegularizationPath`	`getGLMRegularizationPath`	Extract full regularization path explored during lambda search from glm model.
`h2o.getGrid`	`get_grid`	Get a grid object from H2O distributed K/V store.

`h2o.giniCoef`	`giniCoef`	Retrieves the Gini coefficeint for this set of metrics.
`h2o.logloss`	`logloss`	Retrieves the log loss for this set of metrics.
`h2o.makeGLMModel`	`makeGLMModel`	allows setting betas of an existing GLM model.
`h2o.maxPerClassError`	`max_per_class_error`	Returns the max per-class error for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.mcc`	`mcc`	Returns the Matthews correlation coefficient (MCC) for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.mean_residual_deviance`	`mean_residual_deviance`	Returns the mean residual deviance for a set of metrics.
`h2o.metric`	`metric`	Returns the metric value for a set of thresholds. If all are False (default), then return the training metric value. If more than one option is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.mse`	`mse`	Retrieves the MSE for this set of metrics.
`h2o.null_deviance`	`null_deviance`	Returns the null deviance if the model has residual deviance, or None if no null deviance.
`h2o.null_dof`	`null_degrees_of_freedom`	Returns the null degrees of freedom if the model has residual deviance, or

		None if no null degrees of freedom.
`h2o.num_iterations`	`num_iterations`	Returns the number of iterations required for convergence or to reach max iterations.
`h2o.performance`	`model_performance`	Generates model metrics for this model on `test_data`.
`h2o.predict`	`predict`	Predicts on a dataset.
`h2o.precision`	`precision`	Returns the precision for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.proj_archetypes`	`proj_archetypes`	Project each archetype in an H2O GLRM model into the corresponding feature space from the H2O training frame.
`h2o.r2`	`r2`	Retrieves the R^2 coefficient for this set of metrics.
`h2o.reconstruct`	`reconstruct`	Reconstruct the training data from the GLRM model and impute all missing values.
`h2o.residual_deviance`	`residual_deviance`	Returns the residual deviance if the model has residual deviance, or None if no residual deviance.
`h2o.residual_dof`	`residual_degrees_of_freedom`	Returns the residual degrees of freedom if the model has residual deviance, or None if no residual degrees of freedom.
`h2o.saveModel`	`save_model`	Saves an H2O model object to the disk.
`h2o.scoreHistory`	`scoring_history`	Retrieves model score history.
`show`	`show`	Returns the rounded values in the H2OFrame

		to the specified number of significant digits.
`h2o.size`	`size`	Returns the sizes of each cluster. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`summary`	`summary`	Generates a summary of the frame on a per-Vec basis.
`h2o.tnr`	`tnr`	Returns the True Negative Rate for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.totss`	`totss`	Returns the total sum of squares to grand mean. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.tot_withinss`	`tot_withinss`	Returns the total within cluster sum of squares. If all are False (default), then return the training metric value. If more than one options is set to True, then return a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.tpr`	`tpr`	Returns the True Positive Rate for a set of thresholds. If all are False (default), then return the training metric value. If more than one options is set to True, then return a

		dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`h2o.varimp`	`varimp`	Prettyprints the variable importances or returns them in a list ordered from most important to least important. Each entry in the list is a 4-tuple of (variable, relative_importance, scaled_importance, percentage).
`h2o.weights`	`weights`	Returns the frame for the respective weight matrix.
`h2o.withinss`	`withinss`	Returns the within cluster sum of squares for each cluster. If all are False (default), then returns the training metric value. If more than one options is set to True, then returns a dictionary of metrics where the keys are `train`, `valid`, and `xval`.
`plot.H2OModel`	`plot`	Plots training set scoring history (and validataion set if available) for an H2O Model.
`plot.H2OTabulate`		Plots the simple co-occurrence based tabulation of X vs Y as a heatmap, where X and Y are two Vecs in a given dataset
`predict.H2OModel`	`predict`	Predict on a dataset.
`predict_leaf_node_assignment.H2OModel`	`predict_leaf_node_assignment`	Predict on a dataset and return the leaf node assignment (only for tree-based models).
`print.H2OTable`		Returns a truncated view of the table if there are more than 20 rows.
`summary,H2OGrid-method`	`summary`	Format the grid object in a user-friendly way.
`summary,H2OModel-method`	`summary`	Return a detailed summary of the model.

Other Methods:

R	Python	Function Description

`h2o.clusterInfo`	`cluster_info`	Performs node status check for connected node.
`h2o.getFrame`	`get_frame`	Obtains a handle to the frame in H2O with the `frame_id` key.
`h2o.getModel`	`get_model`	Returns the specified model.
`h2o.grid`	`H2OGridSearch`	Provides a set of functions to launch a grid search of a hyper-parameter space for a model get its results.
`h2o.importFile`	`import_file`	Imports files into an H2O cloud. The default behavior is to pass-through to the parse phase automatically.
`h2o.import_sql_select`	`import_sql_select`	Imports the SQL table that is the result of the specified SQL query to H2OFrame in memory. Currently supported SQL databases are MySQL, PostgreSQL, MariaDB, and Netezza.
`h2o.import_sql_table`	`import_sql_table`	Import SQL table to H2OFrame in memory. Assumes that the SQL table is not being updated and is stable. Currently supported SQL databases are MySQL, PostgreSQL, and MariaDB.
`h2o.init`	`h2o.init`	Initiates an H2O connection to the specified IP address and port.
`h2o.logAndEcho`	`log_and_echo`	Sends a message to H2O for logging and/or debugging purposes.
`h2o.networkTest`	`network_test`	View network speed with various file sizes.
`h2o.no_progress`	`no_progress`	Disable the progress bar from flushing to stdout. The completed progress bar is printed when a job is complete so as to demarcate a log file.
`h2o.openLog`	`open_log`	Opens existing logs of H2O R POST commands and error responses on the local disk.
`h2o.parseRaw`	`parse_raw`	Used in conjunction with `import_file` and `parse_setup` in order to make alterations before parsing.
`h2o.parseSetup`	`parse_setup`	During parse setup, the H2O cluster will make several guesses about the attributes of the data. This method allows a user to perform corrective measures by updating the returning dictionary from this method. This dictionary is then fed into `parse_raw` to produce the H2OFrame instance.
`h2o.removeAll`	`remove_all`	Removes all objects from H2O.
`h2o.rm`	`remove`	Removes the specified object from H2O. This is a “hard” delete of the object and removes all subparts.
`h2o.show_progress`	`show_progress`	Enables the progress bar. (Progress bar is enabled by default).
`h2o.shutdown`	`shutdown`	Shuts down the specified instance. All data will be lost.
`h2o.startLogging`	`h2o.start_logging`	Starts logging H2O R POST commands and error responses to the local disk.
`h2o.stopLogging`	`h2o.stop_logging`	Stops logging of H2O R POST commands and error responses to the local disk.
`h2o.uploadFile`	`upload_file`	Uploads a dataset at the path given from the local machine to the H2O cluster.

Ops Group

This group includes:

- **Arith**, for performing arithmetic on numeric or complex vectors

- **Compare**, for comparing values
- **Logic**, for logical operations

R	Python	Type
+	+	Arith
-	-	Arith
*	*	Arith
^	^	Arith
/	/	Arith
%%	mod	Arith
%/%	intDiv	Arith
==	==	Compare
!=	!=	Compare
<	<	Compare
>	>	Compare
<=	<=	Compare
>=	>=	Compare
&	&	Logic
|	|	Logic
!	!	Logic

Math Group

This group includes:

- **Miscellaneous**, which contains the absolute value and square root functions
- **Rounding**, which allows rounding of numbers
- **Logarithms/Exponentials**, which compute logarithmic and exponential functions
- **Trigonometric**, for trigonometric functions
- **Hyperbolic**, for hyperbolic functions
- **Sign**, which returns a vector with the signs of the corresponding elements of x (does not work for complex vectors)
- **Special**, which contains gamma functions
- **Cumulative**, which returns the cumulative sums, products, minima, or maxima

R	Python	Type
abs	abs	Miscellaneous
sqrt	sqrt	Miscellaneous
floor	floor	Rounding
ceiling	ceil	Rounding
trunc	trunc	Rounding
exp	exp	Log/Exp
expm1	expm1	Log/Exp
log	log	Log/Exp
log10	log10	Log/Exp
log2	log2	Log/Exp
log1p	log1p	Log/Exp

`cos`	`cos`	Trigonometric
`sin`	`sin`	Trigonometric
`tan`	`tan`	Trigonometric
`acos`	`acos`	Trigonometric
`asin`	`asin`	Trigonometric
`atan`	`atan`	Trigonometric
`cospi`	`cospi`	Trigonometric two-argument
`sinpi`	`sinpi`	Trigonometric two-argument
`tanpi`	`tanpi`	Trigonometric two-argument
`cosh`	`cosh`	Hyperbolic
`sinh`	`sinh`	Hyperbolic
`tanh`	`tanh`	Hyperbolic
`acosh`	`acosh`	Hyperbolic
`asinh`	`asinh`	Hyperbolic
`atanh`	`atanh`	Hyperbolic
`sign`	`sign`	Sign
`round`	`round`	Sign
`signif`	`signif`	Sign
`lgamma`	`lgamma`	Special
`gamma`	`gamma`	Special
`digamma`	`digamma`	Special
`trigamma`	`trigamma`	Special
`cumsum`	`cumsum`	Cumulative
`cumprod`	`cumprod`	Cumulative
`cummax`	`cummax`	Cumulative
`cummin`	`cummin`	Cumulative

Summary Group

This group includes:

- **Maxima/Minima**, which returns the maxima and minima
- **Range**, which returns the range of a column
- **Product**, which returns the product
- **Sum**, which returns the sum
- **All**, which tells the user if all values are true
- **Any**, which tells the user if any values are true

R	Python
`max`	`max`
`min`	`min`
`range`	
`prod`	`prod`
`sum`	`sum`
`all`	`all`

any	any

Non Group Generic

This group includes:

- **Extract/Replace**, for extracting or replacing part of an object
- **Matrix Multiplication**, for multiplying two matrices
- **Value Matching**, for returning matching vectors

R	Python	Type
`[`		Extract/Replace
`[[`		Extract/Replace
`[[<-`		Extract/Replace
`[<-`		Extract/Replace
`$<-`		Extract/Replace
`%/%`	`//`	Matrix Multiplication
`%x%`	`mult`	Matrix Multiplication
`%in%`	`in`	Value Matching

Python H2OFrame / Pandas DataFrame Munging Converion Table

Note: A blank under the `Pandas Equivalent Method` means the method is equivalent to H2O. (Parenthesis are not always shown when need).

H2OFrame Method	Pandas Equivalent Method
.abs	
.acos	.apply(lambda x: numpy.arccos(x), axis = 0)
.acosh	.apply(lambda x: numpy.arccosh(x), axis = 0)
.all	
.any	
.any_na_rm	
.anyfactor	
.apply	.apply
.as_data_frame	
.as_date	.to_datetime
.ascharacter	astype(str)
.asfactor	.astype('category') or .astype('object')
.asin	.apply(lambda x: numpy.arcsin(x), axis = 0)
.asinh	.apply(lambda x: numpy.arcsinh(x), axis = 0)
.asnumeric	astype(numpy.float) or apply(numpy.float)
.atan	.apply(lambda x: numpy.arctan(x), axis = 0)
.atanh	.apply(lambda x: numpy.arctanh(x), axis = 0)
.categories	.unique()
.cbind	.concat()
.ceil	.apply(numpy.ceil)
.col_names	.columns
.columns	
.columns_by_type	.select_dtypes()
.concat	
.cor	.corr
.cos	.apply(lambda x: numpy.arccoh(x), axis = 0)
.cosh	.apply(lambda x: numpy.arccos(x), axis = 0)
.cospi	.apply(lambda x: numpy.cos(numpy.pi * x), axis = 0)
.count	
.countmatches	.str.contains()
.cummax	
.cummin	

.cumprod	
.cumsum	
.cut	
.day	Series.dt.day
.dayOfWeek	DatetimeIndex(pandas_dataframe[time_column]).dayofweek
.ddply	
.describe	
.difflag1	.diff
.digamma	scipy.special.digamma()
.dim	.shape
.drop	
.entropy	NA
.exp	numpy.exp()
.expm1	numpy.expm1()
.filter_na_cols	NA
.flatten	
.floor	.apply(numpy.floor)
.frame	NA
.frame_id	NA
.from_python	NA
.gamma	scipy.special.gamma()
.get_frame	NA
.get_frame_data	similar to the purpose of to_csv()
.getrow	list(pandas_dataframe.loc[0,:])
.group_by	.groupby()
.gsub	.replace()
.head	
.hist	
.hour	DatetimeIndex(pandas_dataframe[time_column]).year
.ifelse	numpy.where()
.impute	NA
.insert_missing_values	NA
.interaction	NA
.isax	NA
.ischaracter	.isinstance(pandas_column, object)
.isfactor	NA
.isin	
.isna	.isnull

.isnumeric	NA
.isstring	.isinstance(pandas_column, object)
.kfold_column	NA
.kurtosis	
.levels	.cat.categories, .unique()
.lgamma	scipy.special.gammaln()
.log	numpy.log()
.log10	numpy.log10()
.log1p	numpy.log1p()
.log2	numpy.log2()
.logical_negation	numpy.logical_not()
.lstrip	.str.lstrip(’’)
.match	
.max	
.mean	
.median	
.merge	
.min	
.mktime	
.mode	NA
.modulo_kfold_column	NA
.moment	pd.to_datetime()
.month	Series.dt.month
.mult	.dot
.na_omit	.dropna()
.nacnt	.isnull().sum()
.names	.columns
.nchar	.str.len()
.ncol	.shape[1]
.ncols	.shape[1]
.nlevels	.nunique()
.nrow	.shape[0]
.nrows	.shape[0]
.num_valid_substrings	
.pop	
.prod	
.quantile	

.rbind	
.refresh	
.relevel	NA
.rep_len	NA
.round	
.rstrip	.str.rstrip()
.runif	numpy.random.uniform()
.scale	sklearn.preprocessing.StandardScaler()
.sd	.std
.set_level	NA
.set_levels	NA
.set_name	.rename()
.set_names	.rename()
.shape	
.show	NA
.sign	numpy.sign()
.signif	NA
.sin	.apply(lambda x: numpy.sin(x), axis = 0)
.sinh	.apply(lambda x: numpy.sinh(x), axis = 0)
.sinpi	.apply(lambda x: numpy.sin(numpy.pi * x, axis = 0)
.skewness	.skew
.split_frame	NA
.sqrt	.apply(lambda x: numpy.sqrt(x), axis = 0)
.ss	NA
.stratified_kfold_column	sklearn.model_selection.StratifiedKFold
.stratified_split	sklearn.model_selection.StratifiedShuffleSplit
.strsplit	.str.split
.structure	NA
.sub	.str.replace()
.substring	.str.slice()
.sum	
.summary	.describe()
.table	.value_counts()
.tail	
.tan	.apply(lambda x: numpy.tan(x), axis = 0)
.tanh	.apply(lambda x: numpy.tanh(x), axis = 0)
.tanpi	.apply(lambda x: numpy.tan(numpy.pi * x, axis = 0)
.tolower	
.toupper	.apply(lambda x: x.upper(), inplace=True)

.transpose	
.trigamma	scipy.special.polygamma(x,3)
.trim	.str.strip
.trunc	
.type	.dtype
.types	.dtypes
.unique	
.var	
.week	Series.dt.week
.which	NA
.year	Series.dt.year