**Описание классов и методов**

Оглавление

[Описание классов 2](#_Toc416445893)

[Базовые классы основных сущностей (baseClasses) 2](#_Toc416445894)

[Experiment 2](#_Toc416445895)

[Trial 2](#_Toc416445896)

[Subject 3](#_Toc416445897)

[Factor 3](#_Toc416445898)

[Stimulus 3](#_Toc416445899)

[AOI 4](#_Toc416445900)

[RawDataRecord 4](#_Toc416445901)

[DataRecord 4](#_Toc416445902)

[Классы объектов данных для анализа (baseEyeDataClasses) 5](#_Toc416445903)

[TimeSamples 5](#_Toc416445904)

[FrameSamples 5](#_Toc416445905)

[TrajectorySamples 5](#_Toc416445906)

[PupilSamples 5](#_Toc416445907)

[OtherSamples 6](#_Toc416445908)

[FilterMarkers 6](#_Toc416445909)

[EventMarkers 6](#_Toc416445910)

[FixationsData 6](#_Toc416445911)

[SaccadesData 6](#_Toc416445912)

[GlissadesData 7](#_Toc416445913)

[SmoothPursuitsData 7](#_Toc416445914)

[BlinksData 7](#_Toc416445915)

[AdditionalEventData 7](#_Toc416445916)

[EventData 7](#_Toc416445917)

[EyesData 7](#_Toc416445918)

[AOISequence 8](#_Toc416445919)

[AOITransMatrix 8](#_Toc416445920)

[AOIStatsVector 8](#_Toc416445921)

[Классы объектов-списков и таблиц (listsAndTablesClasses) 9](#_Toc416445922)

[Experiments 9](#_Toc416445923)

[Trials 9](#_Toc416445924)

[Subjects 9](#_Toc416445925)

[AvailableFactors 9](#_Toc416445926)

[Factors 9](#_Toc416445927)

[FactorsData 10](#_Toc416445928)

[Stimuli 10](#_Toc416445929)

[DataSample 10](#_Toc416445930)

[AOISet 10](#_Toc416445931)

[AOISets 10](#_Toc416445932)

[TAS 11](#_Toc416445933)

[RawDataRecords 11](#_Toc416445934)

[Классы настроек и опций (optionsAndSettingsClasses) 11](#_Toc416445935)

[ReadSettings 11](#_Toc416445936)

[AvailableDataFields 11](#_Toc416445937)

[DataFieldNames 12](#_Toc416445938)

[HeaderKeys 12](#_Toc416445939)

[RawDataSettings 12](#_Toc416445940)

[Классы внешних функций (extFunctionsClasses) 13](#_Toc416445941)

[extFunction 13](#_Toc416445942)

[extFunctions 13](#_Toc416445943)

[Описание методов 14](#_Toc416445944)

[Методы добавления объектов в списки 14](#_Toc416445945)

[addExperiment 14](#_Toc416445946)

[addTrial 14](#_Toc416445947)

[addSubject 14](#_Toc416445948)

[addFactorDefinition 15](#_Toc416445949)

[addStimulus 15](#_Toc416445950)

[addAOI 16](#_Toc416445951)

[addAOISet 16](#_Toc416445952)

[addFactorsRecord 16](#_Toc416445953)

[addFactor 16](#_Toc416445954)

[addDataRecord 16](#_Toc416445955)

[Функция createRawDataRec 17](#_Toc416445956)

[addRawDataRecord 18](#_Toc416445957)

[addRawDataRecords 18](#_Toc416445958)

[Методы печати содержимого объектов 19](#_Toc416445959)

[getSubjectCodes 19](#_Toc416445960)

[printDataSampleKeys 19](#_Toc416445961)

# Описание классов

## Базовые классы основных сущностей (baseClasses)

### Experiment

Описание класса:

Класс предназначен для представления основной информации об эксперименте и привязки к нему набора траекторий через ID эксперимента (ID присутствует, а точнее неявно создаётся, в качестве элемента списка экспериментов – объекте класса **Experiments**, который определён в разделе «[Классы объектов-списков и таблиц](#_Классы_объектов-списков_и)»).

Слоты:

name = "character", название эксперимента;

description = "character", краткое описание эксперимента;

instruction = "character", инструкция испытуемому или описание процедуры эксперимента;

experimenters = "character", неуникальные идентификаторы людей, проводивших эксперимент;

conditions = "Conditions", экспериментальные условия, в которых проводился эксперимент, настройки оборудования, а также единицы измерения и другая уточняющая информация, распространяющаяся по умолчанию на все испытания в данном эксперименте (если у соответствующих объектов данный слот не переопределён).

Описание класса "Conditions" см. в разделе «[Классы настроек и опций](#_Классы_настроек_и)».

### Trial

Описание класса: Класс предназначен для представления основной информации об испытании конкретного эксперимента с возможностью переопределения экспериментальных условий.

Слоты:

id = "numeric", идентификатор испытания (почти всегда – порядковый номер испытания);

expID = "numeric", идентификатор эксперимента, в котором проводилось испытание;

name = "character", наименование испытания;

description = "character", описание испытания;

conditions = "Conditions", особые условия/настройки/единицы измерения и т.п., распространяющиеся только на данное испытание.

Описание класса "Conditions" см. в разделе «[Классы настроек и опций](#_Классы_настроек_и)».

### Subject

Описание класса: Класс предназначен для представления основной информации об испытуемом. Такие слоты как, например, возраст, место рождения и т.п. рассматриваются как факторы испытуемого, которые можно задать, добавив соответствующий объект класса [**Factor**](#_Factor)в список факторов **AvailableFactors**, затем определив для него значение в списке факторов **Factors** и связав с конкретным испытуемым через таблицу факторов **FactorsData**. Описание классов **AvailableFactors**, **Factors** и **FactorsData** см. вразделе «[Классы объектов-списков и таблиц](#_Классы_объектов-списков_и)».

Слоты:

code = "character", уникальный код испытуемого (задаётся экспериментатором, либо определяется в процессе импорта данных).

### Factor

Описание класса: Класс предназначен для представления в системе сведений о каком-либо факторе (вещественном, целочисленном, номинальном или порядковом).

Слоты:

varName = "character", название фактора, указываемое экспериментатором или определяемое при загрузке значений факторов из внешнего файла (важно, чтобы оно удовлетворяло требованиям к названиям переменных, которые предъявляются R);

description = "character", расшифровка фактора, то есть его описание задаваемое экспериментатором;

type = "character", перечисляемый тип фактора: «numeric» - вещественное число, «integer» - целое число, «factor» - номинальный фактор, или «ordFactor» - порядковый фактор;

levels = "character", вектор, определяющий уровни фактора, порядок элементов в котором важен в случае фактора типа «ordFactor»; в случае вещественных и целочисленных факторов levels = as.character(NA).

### Stimulus

Описание класса: Класс предназначен для представления основной информации о стимуле, используемом в экспериментах. Стимулы могут быть разного типа, например, изображениями, видеорядами, веб-страницами, наблюдаемыми сценами (фиксируемыми с помощью фронтальной камеры айтрекера).

Слоты:

id = "numeric", идентификатор стимула в системе;

name = "character", название стимула;

description = "character", описание стимульного материала;

path = "character", путь к файлу стимула (необходим для загрузки и отрисовки в случае изображений или воспроизведения в случае видеоряда);

type = "character", перечисляемый тип стимула: «Image» - изображение, «Video» - видеоряд, «Scene» - внешняя наблюдаемая испытуемым сцена, «Web page» - веб-страница;

dim = "integer", целочисленный вектор, определяющий реальную размерность стимула (для изображений это – вектор из двух целых чисел, определяющих число пикселей по горизонтали и вертикали);

dimOnPresentation = "integer", целочисленный вектор, определяющий размерность стимула (по аналогии с dim), актуальную при его демонстрации;

duration = "numeric", длительность стимула (актуально для видеоряда или сцены);

durationUnits = "numeric", единицы измерения длительности в экспоненциальной записи (например, 1E0 для секунд, 1E-3 для миллисекунд), которые используются для сведения исходных единиц измерения в стандартные;

framesCount = "numeric", число кадров в видеоряде (демонстрируемого видео или наблюдаемой внешней сцены).

### AOI

Описание класса: Класс предназначен для представления данных о пространственной области интереса, выделяемой на стимуле, которая может быть динамической. Для динамических зон dispositionData хранит индексированный вектор списков, где каждый список содержит информацию о расположении зоны интереса для соответствующего фрейма в видеоряде.

Слоты:

name = "character", название области;

dynamic = "logical", флаг, указывающий, является ли зона динамической;

type = "character", перечисляемый тип области: «Classic» - классическая область, задаваемая геометрическим объектом, размещённом в пространстве стимула, который определяет её замкнутую область, «Fuzzy» - нечёткая область, задаваемая ядровой функцией, взвешивающей попадания в окрестность некоторой точки стимула;

shape = "character", перечисляемый тип формы области или вид ядровой функции: «Rectangle» - прямоугольная, «Circle» - окружность, «Ellipse» - эллипс, «Polyhedron» - многогранник, «Gaussian» - гауссовское ядро и т.п.

dispositionData = "list", список, содержащий данные о расположении области; в случае прямоугольной или многогранной области это – список пар координат вершин, в случае окружности или эллипса это – список с координатами центра фигуры и радиусом или длиной большой и малой полуосей, в случае гауссовской области это – координаты середины «окна» и величина параметра его ширины (сигма).

### RawDataRecord

Описание класса: Класс предназначен для представления в системе «сырых» экспериментальных данных в том смысле, что они хоть и загружены в систему, но ещё не оформлены в виде соответствующих объектов, траектории не разбиты по испытаниям, а информация из «шапки» данных (код испытуемого, размерность стимула и прочее подобное) не обработана парсером.

Слоты:

filePath = "character", полный путь к файлу с данными траектории;

headerLines = "character", содержимое «шапки» данных, считываемое загрузчиком данных (шапка определяется первыми n строками с начала файла, где n задаётся через слот skip объекта [**ReadSettings**](#_ReadSettings));

data = "data.frame", фрейм с содержимым таблицы из файла данных (все столбцы таблицы считываются как есть), у которого столбцы именуются в соответствии с заголовками столбцов таблицы (если установлен параметр header объекта [**ReadSettings**](#_ReadSettings)**)**

### DataRecord

Описание класса: Класс предназначен для представления «обобщённой записи» данных по составному ключу «ID эксперимента»-«ID испытуемого»-«ID испытания». Под «обобщённой записью» понимается список элементов, который можно представить как стандартную запись лишь виртуально, содержащий помимо значений элементов составного ключа и объекта [EyesData](#_EyesData) другие объекты (по 1 экземляру), являющиеся представителями одного из следующих классов: [EventData](#_EventData), [AOISequence](#_AOISequence), [AOITransMatrix](#_AOITransMatrix), [AOIStatsVector](#_AOIStatsVector), а также слот, содержащий список параметров-статистик, вычисленных для объектов, присутствующих в данной записи, с помощью generic-метода paramEstimator (например, для EyesData это может быть длительность записи, для EventData это может быть, например, число фиксаций, для AOITransMatrix это может быть метка наиболее часто посещаемой области интереса, а для AOIStatsVector – время нахождения взора в наименее часто посещаемой области и т.п.).

Слоты:

expID = "numeric", ID эксперимента, в котором проводилось испытание, к которому относится объект object;

subjectID = "numeric", ID испытуемого, который проходил испытание, к которому относится объект object;

trialID = "numeric", ID испытания, к которому относится объект object;

eyesDataObject = "[EyesData](#_EyesData)", слот, содержащий данные о движении взора;

analysisResults = "list", слот-список, содержащий набор объектов классов [EventData](#_EventData), AOISequence, [AOITransMatrix](#_AOITransMatrix), [AOIStatsVector](#_AOIStatsVector) (по 1 заменяемому при новом расчёте экземпляру);

statistics = "list", список, содержащий набор статистик, рассчитанных как для eyesDataObject, так и для элементов analysisResults.

## Классы объектов данных для анализа (baseEyeDataClasses)

### TimeSamples

Описание класса: Класс предназначен для представления данных о моментах времени отсчётов траектории взора.

Слоты:

time = "list", список с единственным элементом time – вектором моментов времени, соответствующих отсчётам данных глазодвигательной активности.

### FrameSamples

Описание класса: Класс предназначен для представления данных о номерах фреймов (кадров видеоряда), которые обычно доступны для записей, полученных для стимулов-видеозаписей или с помощью мобильного айтрекера.

Слоты:

frame = "list", список с единственным элементом frame – вектором номеров кадров, соответствующих отсчётам данных глазодвигательной активности. На 1 кадр может приходиться несколько отсчётов данных.

### TrajectorySamples

Описание класса: Класс предназначен для представления данных о траектории взора по поверхности стимула (временной ряд пар координат позиции взора).

Слоты:

eyeData = "data.frame", фрейм данных с двумя переменными – PORX и PORY.

### PupilSamples

Описание класса: Класс предназначен для представления данных об изменении во времени размера зрачка (временной ряд величины зрачка по горизонтали и, необязательно, вертикали). В случае аппроксимации зрачка с помощью окружности, в наличии могут быть только данные лишь о радиусе или диаметре зрачка, а фрейм содержит только переменную PUPX, содержащую временной ряд значений радиуса или диаметра зрачка. В случае аппроксимации с помощью эллипса, фрейм содержит обе переменные, PUPX (величина раскрытия зрачка по горизонтали) и PUPY (величина раскрытия зрачка по вертикали).

Слоты:

pupilData = "data.frame", фрейм данных с переменной PUPX и необязательной переменной PUPY.

### OtherSamples

Описание класса: Класс предназначен для представления данных об изменении во времени каких-либо других показателей глазодвигательной активности.

Слоты:

otherData = "data.frame", содержит набор переменных, чьи имена определяются именами либо именами переменных, взятых из заголовка таблицы данных, либо имена переменных, которые указывает пользователь.

### FilterMarkers

Описание класса: Класс предназначен для представления данных о маркерах, устанавливаемых фильтрами траектории взора. Наименования маркеров возможно определять произвольно как строковые константы (используется при отрисовке размеченной траектории, а также при подсчёте числа нежелательных отсчётов разного вида).

Слоты:

filterMarkersData = "character",

markerNames = "list"

Прототип:

markerNames = list(zeroes = "0",

outOfBounds = "Out of bounds",

abnormalSpeed = "Abnormal speed",

abnormalAcceleration = "Abnormal acceleration")

### EventMarkers

Описание класса: Класс предназначен для представления данных о маркерах событий, устанавливаемых детекторами событий. Наименования маркеров возможно определять произвольно как строковые константы (используется при отрисовке размеченной траектории, а также при расчётах характеристик событий).

Слоты:

eventMarkersData = "character",

markerNames = "list"

Прототип:

markerNames = list(fixation = "Fixation",

saccade = "Saccade",

glissade = "Glissade",

smoothPursuit = "Smooth pursuit",

blink = "Blink",

noise = "Noise")

### FixationsData

Описание класса: Класс предназначен для представления данных об окуломоторных событиях – фиксациях взора.

Слоты:

fixations = "data.frame", фрейм данных с переменными position, dispersion, onset, offset, duration.

### SaccadesData

Описание класса: Класс предназначен для представления данных об окуломоторных событиях – саккадах (быстрых движениях взора).

Слоты:

saccades = "data.frame", фрейм данных с переменными startPosition, endPosition, amplitude, onset, offset, duration, meanVelocity, peakVelocity, meanAcceleration, peakAcceleration, asymmetry, curvature, orientation.

### GlissadesData

Описание класса: Класс предназначен для представления данных об окуломоторных событиях – глиссадах (корректирующих движениях взора, возникающих после саккад).

Слоты:

glissades = "data.frame", фрейм данных с переменными startPosition, endPosition, amplitude, onset, offset, duration, meanVelocity, peakVelocity, meanAcceleration, peakAcceleration, asymmetry, curvature, orientation, antecedentSaccadeNumber.

### SmoothPursuitsData

Описание класса: Класс предназначен для представления данных об окуломоторных событиях – плавных преследованиях (такого рода движения взора возникают лишь в случае отслеживания движущихся объектов в стимульном материале).

Слоты:

smoothPursuits = "data.frame", фрейм данных с переменными startPosition, endPosition, amplitude, onset, offset, duration, peakVelocity, peakAcceleration, asymmetry, curvature, orientation. **Набор переменных – характеристик плавного преследования следует уточнить!**

### BlinksData

Описание класса: Класс предназначен для представления данных об окуломоторных событиях – морганиях.

Слоты:

blinks = "data.frame", фрейм данных с переменными onset, offset, duration, eyePosStart, eyePosEnd, pupSizeStart, pupSizeEnd

### AdditionalEventData

Описание класса: Класс предназначен для представления данных о других окуломоторных событиях.

Слоты:

eventName = "character", название окуломоторного события;

additionalEventsData = "data.frame", произвольный фрейм данных, содержащий характеристики всех выявленных событий.

### EventData

Описание класса: Класс предназначен для представления данных об обнаруживаемых событиях в едином объекте.

Слоты:

fixations = "FixationsData", объект класса FixationsData

saccades = "SaccadesData", объект класса SaccadesData

glissades = "GlissadesData", объект класса GlissadesData

smoothPursuits = "SmoothPursuitsData", объект класса SmoothPursuitsData

blinks = "BlinksData", объект класса BlinksData

additionalEvents = "list", список объектов класса AdditionalEventData

### EyesData

Описание класса: Класс предназначен для представления данных о глазодвигательной активности, включая моменты времени, номера кадров, позиции взора, размеры зрачка, дополнительные отсчёты, маркеры фильтров и событий, а также названия временных рядов (для построения графиков) и переопределённые условия conditions эксперимента.

Слоты:

fieldNames = "DataFieldNames", названия временных рядов;

conditions = "Conditions", условия эксперимента, которые возможно задать для конкретной траектории конкретного испытуемого (например, если он слеп на левый глаз, то записывается правый глаз, в отличие от основной массы испытуемых, и т.п.);

time = "TimeSamples", моменты времени, соответствующие зарегистрированным отсчётам;

frame = "FrameSamples", фреймы видеостимула или видеоряда регистрируемой внешней сцены, соответствующие зарегистрированным отсчётам;

leftEyeSamples = "TrajectorySamples", отсчёты траектории взора левого глаза;

rightEyeSamples = "TrajectorySamples", отсчёты траектории взора правого глаза;

leftPupilSamples = "PupilSamples", отсчёты величины зрачка левого глаза;

rightPupilSamples = "PupilSamples", отсчёты величины зрачка правого глаза;

leftAdditionalSamples = "OtherSamples", дополнительные отсчёты для левого глаза;

rightAdditionalSamples = "OtherSamples", дополнительные отсчёты для левого глаза;

leftFilterMarkers = "FilterMarkers", маркеры фильтра, применённого к траектории взора левого глаза;

rightFilterMarkers = "FilterMarkers", маркеры фильтра, применённого к траектории взора правого глаза;

leftEventMarkers = "EventMarkers", маркеры детектора событий, применённого к траектории взора левого глаза;

rightEventMarkers = "EventMarkers", маркеры детектора событий, применённого к траектории взора правого глаза.

### AOISequence

Описание класса: Класс предназначен для представления данных о последовательности имён «посещённых» областей интереса, принадлежащих набору AOISetID.

Слоты:

dynamic = "logical", флаг, указывающий, получена ли последовательность для динамических областей интереса;

sequence = "data.frame", фрейм данных с переменными frame, name, onset, offset и duration;

AOISetID = "numeric", идентификатор набора областей интереса, использовавшемся при построении последовательности.

### AOITransMatrix

Описание класса: Класс предназначен для представления данных о матрице характеристик процесса перемещения взора между областями интереса, принадлежащих набору AOISetID.

Слоты:

type = "character", тип матрицы (frequencies – матрица частот переходов между областями интереса, probabilities – матрица вероятностей переходов, SR – матрица представления преемника для переходов между областями интереса, normalized SR – нормализованная SR матрица);

dynamic = "logical", флаг, указывающий, получена ли матрица для динамических областей интереса;

matrices = "list", список матриц c элементами frame и matrices;

AOISetID = "numeric", идентификатор набора областей интереса, использовавшемся при построении матриц.

### AOIStatsVector

Описание класса: Класс предназначен для представления данных о векторе характеристик попадания взора в области интереса, принадлежащие набору AOISetID.

Слоты:

type = "character", тип вектора (frequencies – вектор частот попадания в области интереса, probabilities – вектор вероятностей попадания, times – вектор суммарных времён пребывания)

dynamic = "logical", флаг, указывающий, получен ли вектор статистик для динамических областей интереса;

vectors = "list", список векторов c элементами frame и vectors;

AOISetID = "numeric", идентификатор набора областей интереса, использовавшемся при построении векторов.

## Классы объектов-списков и таблиц (listsAndTablesClasses)

### Experiments

Описание класса: Класс предназначен для представления данных обо всех экспериментах, определённых в системе.

Слоты:

expList = "list", список с элементами-списками ids (список идентификаторов экспериментов, автоматически инкрементируемых при добавлении эксперимента в список с помощью метода [addExperiment](#_addExperiment)) и experiments (список с объектами класса [Experiment](#_Experiment)).

### Trials

Описание класса: Класс предназначен для представления данных обо всех испытаниях, определённых в системе.

Слоты:

trialsList = "list", список с элементами-списками ids (список идентификаторов испытаний, определяемых при загрузке экспериментальных данных), expIDs (идентификаторы соответствующих испытаниям экспериментов) и trials (список с объектами класса [Trial](#_Trial)).

### Subjects

Описание класса: Класс предназначен для представления данных обо всех испытуемых, определённых в системе.

Слоты:

subjectsList = "list", список с элементами-списками ids (список идентификаторов испытуемых, автоматически инкрементируемых при добавлении испытуемого в список с помощью метода [addSubject](#_addSubject)) и subjects (список с объектами класса [Subject](#_Subject)).

### AvailableFactors

Описание класса: Класс предназначен для представления данных о факторах, определённых в системе. Факторы могут быть привязаны к таким объектам как [Subject](#_Subject), [Trial](#_Trial) и [Stimulus](#_Stimulus) посредством указания значения соответствующего поля owner.

Слоты:

availableFactors = "data.frame", фрейм данных с переменными id (идентификатор фактора, автоматически инкрементируемый при добавлении фактора), varName (имя факторной переменной), description (описание фактора), type (шкала измерения фактора: номинальный «factor», порядковый «ordFactor», целочисленный «integer» и вещественный «numeric»), levels (уровни фактора: для целочисленного и вещественного – NA, для порядкового – упорядоченный вектор наименований уровней, для номинального – вектор наименований уровней, чей порядок неважен) и owner ("subject", "trial" or "stimulus")..

### Factors

Описание класса: Класс предназначен для представления данных о значениях факторов, определённых в системе для конкретного объекта. Позволяет задать набор значений конкретных факторов для какого-либо объекта, привязка к которому осуществляется через объект класса [FactorsData](#_FactorsData).

Слоты:

ownerId = "numeric", идентификатор объекта, к которому относятся значения факторов;

factors = "numeric", список идентификаторов факторов, относящихся к объекту с ownerId;

values = "list", список значений соответствующих факторов; значения должны быть приведены в соответствующий формат (numeric, integer; factor и ordered с указанием соответствующих уровней).

### FactorsData

Описание класса: Класс предназначен для представления данных о значениях всех определённых факторов конкретных объектов.

Слоты:

ownersIds = "numeric", вектор идентификаторов объектов, к которым привязаны факторы;

factorsList = "list", список объектов класса [Factors](#_Factors).

### Stimuli

Описание класса: Класс предназначен для представления данных обо всех стимулах, определённых в системе.

Слоты:

stimuliList = "list", список с элементами-списками ids (список идентификаторов стимулов, автоматически инкрементируемых при добавлении стимула в список с помощью метода [addStimulus](#_addStimulus)) и stimuli (список с объектами класса [Stimulus](#_Stimulus)).

### DataSample

Описание класса: Класс предназначен для представления экспериментальных данных и результатах их анализа по составному ключу expID, subjectID, trialID.

Слоты:

keys = "data.frame", фрейм данных, содержащий значения составных ключей, с переменными expID, subjectID, trialID;

eyesDataObjectsList = "list", список объектов класса [EyesData](#_EyesData), соответствующих значениям составных ключей;

analysisResults = "list", список объектов, заполняемый объектами классов [EventData](#_EventData), [AOISequence](#_AOISequence), [AOITransMatrix](#_AOITransMatrix_1), [AOIStatsVector](#_AOIStatsVector) (по 1 экземпляру, заменяемому при пересчёте);

statistics = "list", список элементов, содержащих именованные интегральные статистики (например, результаты подсчёта фиксаций, общая длительность записи, номер кластера и т.п.)

### AOISet

Описание класса: Класс предназначен для представления данных о наборе областей интереса.

Слоты:

name = "character", название набора областей интереса;

AOIs = "list", список объектов класса [AOI](#_AOI).

### AOISets

Описание класса: Класс предназначен для представления данных о наборах областей интереса, определённых в системе.

Слоты:

AOISetsList = "list", список с элементами-списками ids (список идентификаторов наборов областей интереса, автоматически инкрементируемых при добавлении набора в список с помощью метода [addAOISet](#_addAOISet)) и AOISets (список с объектами класса [AOISet](#_AOISet)).

### TAS

Описание класса: Класс предназначен для представления данных о связях испытаний, стимулов и зон интереса

Слоты:

trialID = "numeric", идентификатор испытания;

AOISetID = "numeric", идентификатор набора зон интереса, привязанных к стимулу/испытанию;

stimulusID = "numeric", идентификатор стимула, привязанного к испытанию.

### RawDataRecords

Описание класса: Класс предназначен для представления данных о содержимом файлов сырых данных, загруженных в систему.

Слоты:

rawDataRecordsList = "list", список с элементами-списками fileNumbers и rawDataRecords (список с объектами класса [RawDataRecord](#_RawDataRecord))

## Классы настроек и опций (optionsAndSettingsClasses)

### ReadSettings

Описание класса: Класс предназначен для представления данных о настройках считывания текстовых файлов данных с разделителями.

Слоты:

readSettings = "list", список настроек считывания файлов данных;

Прототип:

encoding = "UTF-8", кодировка;

sep = "\t", разделитель столбцов;

skip = 20, число строк шапки данных;

comment.char = "#", символ, указывающий на строку в файле данных как на комментарий (обычно это строки шапки);

header = T, наличие строки заголовков полей непосредственно над таблицей данных;

splitBy = "Trial", указатель поля ("Trial" или "Stimname"), по которому траектория разбивается на подтраектории, соответствующие отдельным испытаниям;

### AvailableDataFields

Описание класса: Класс предназначен для представления данных о расположении полей данных в таблице «сырых» данных, указываемых пользователем. Другими словами, это список соответствий, указывающий, из каких полей считывать те или иные данные (например, для поля time может быть указан номер «1», если в таблице «сырых» данных последовательность моментов времени, в которые были зарегистрированы отсчёты, располагается именно в 1-м столбце).

Слоты:

availableFields = "list", список соответствий полей и номеров столбцов таблицы данных, изначально состоит из неопределённых значений NA.

Прототип:

availableFields = list(time = NA, trial = NA, frame = NA, stimname = NA, smptype = NA,

lporx = NA, lpory = NA, rporx = NA, rpory = NA,

lpupxsize = NA, lpupysize = NA, rpupxsize = NA, rpupysize = NA,

additionalFields = list(lrawx = NA, lrawy = NA))

Здесь, time – внутреннее (системное) имя поля, содержащего отсчёты моментов времени; trial – внутреннее имя поля, содержащего указатели испытаний, в которых были получены те или иные отсчёты параметров; stimname – внутреннее имя поля, содержащего указатели стимулов, для которых были получены те или иные отсчёты параметров (используется так же как и trial для разбиения траектории на части, соответствующие разным испытаниям); smptype – внутреннее имя поля, содержащего указатели типа отсчёта (например, полезные отсчёты значений регистрируемых параметров в SMI-файлах помечаются как «SMP», а записи о системных сообщениях программы производителя айтрекера – как «MSG»), **при этом все отсчёты с метками, отличными от «SMP» удаляются из данных парсером.**

### DataFieldNames

Описание класса: Класс предназначен для представления данных о наименованиях параметров, временные ряды значений которых имеются в данных. Предполагается, что в основном данные имена могут использоваться при построении читаемых графиков.

Слоты:

fieldNames = "list", список соответствий полей и имён параметров, временные ряды значений которых имеются в данных

Прототип:

fieldNames = list(time = "Time", trial = "Trial", frame = "Frame", stimname = "StimulusName", smptype = "SampleType”, lporx = "LPORX", lpory = "LPORY", rporx = "RPORX", rpory = "RPORY", lpupxsize = "LPupilXSize", lpupysize = "LPupilYSize", rpupxsize = "RPupilXSize", rpupysize = "RPupilYSize", additionalFields = list(lrawx = "LRawX", lrawy = "LRawY"))

### HeaderKeys

Описание класса: Класс предназначен для представления данных о ключах шапки данных, по которым парсер с помощью регулярных выражений извлекает различные существенные свойства данных (например, код испытуемого, частота дискретизации и т.п.)

Слоты:

subjectCode = "character", строка с указателем значения ключа для кода испытуемого;

sampleRate = "character", строка с указателем значения ключа для частоты дискретизации;

stimDim = "character", строка с указателем значения ключа для актуального размера стимула при экспозиции;

headDist = "character", строка с указателем значения ключа для горизонтального расстояния от головы испытуемого до стимульного экрана или расстояния до плоскости, использованной при калибровке мобильного айтрекера;

otherKeys = "list", список других указателей для некоторых более редко используемых conditions;

Прототип (пример значений ключей для файлов SMI):

subjectCode = "Subject",

sampleRate = "Sample Rate",

stimDim = "Stimulus Dimension",

headDist = "Head Distance"

### RawDataSettings

Описание класса: Класс предназначен для представления данных обо всех настройках, существенных для использования при загрузке данных из файлов и последующем парсинге загруженных данных.

Слоты:

rawReadSettings = "ReadSettings", объект класса ReadSettings;

availableFields = "AvailableDataFields", объект класса AvailableDataFields;

fieldNames = "DataFieldNames", объект класса DataFieldNames;

headerKeys = "HeaderKeys", объект класса HeaderKeys;

eyeTrackerModel = "character", название модели айтрекера;

eyeTrackerType = "character", тип айтрекера ("tower", "remote" или "head-mounted").

Conditions

Описание класса: Класс предназначен для представления данных о важнейших параметрах, определяющих условия эксперимента (режим записи: бинокулярная или монокулярная, по левому или правому глазу; частота дискретизации; расстояния, определяющие положение испытуемого; характеристики камеры айтрекера; единицы измерения времени и размеров; способ аппроксимации зрачка и задания его размеров).

Слоты:

conditions = "list", список настроек;

Прототип:

eye = NA, режим записи (левый глаз "left", правый глаз "right" или бинокулярный режим "both");

sampleRate = NA, частота дискретизации в Гц;

screenDistance = NA, расстояние от глаз испытуемого до плоскости стимула или калибровочной плоскости;

pupilShape = NA, способ аппроксимации зрачка;

pupilDataType = NA, способ задания величины раскрытия зрачка ("radius" или "diameter");

timeUnits = NA, единицы измерения времени в экспоненциальной записи (например, 1E-3 для миллисекунд);

distanceUnits = NA, единицы измерения расстояний в экспоненциальной записи (например, 1E-2 для миллиметров);

pupilSizeUnits = NA, единицы измерения размера зрачка (пиксели "px" или размеры в стандартных единицах (мм, см) в экспоненциальной записи, например, 1E-2 для миллиметров)

## Классы внешних функций (extFunctionsClasses)

### extFunction

Описание класса: Класс предназначен для представления данных о внешних функциях обработки объектов системы, которые могут быть подключены пользователем (т.е. что-то вроде «плагинов»).

Слоты:

id = "integer", идентификатор функции;

name = "character", наименование функции;

applyToClass = "character", наименование класса, к которому применима функция;

fun = "function", непосредственно объект-функция;

settings = "list", список настроек, используемых функцией.

### extFunctions

Описание класса: Класс предназначен для представления данных обо всех внешних функциях, определённых в системе.

Слоты:

functionsList = "list", список объектов класса [extFunction](#_extFunction).

# Описание методов

## Методы добавления объектов в списки

### addExperiment

setGeneric("addExperiment", function(self, expObject){standardGeneric("addExperiment")})

# Method adds an experiment object into Experiments list with ids and experiments sublists

## Method increments ids of experiments

## Method doesn't prevent duplicates in Experiments because their presence is not critical due to user's control

setMethod("addExperiment", "Experiments",

function(self, expObject)

{

expCnt <- length(self@expList$ids)

if (expCnt == 0)

{

self@expList$ids <- 1

self@expList$experiments <- list(expObject)

return(self)

}

id <- self@expList$ids[[expCnt]] + 1

self@expList$ids <- c(self@expList$ids, id)

self@expList$experiments <- c(self@expList$experiments, expObject)

return(self)

}

)

### addTrial

setGeneric("addTrial", function(self, trialObject){standardGeneric("addTrial")})

# Method adds a trial object into Trials list with ids and trials sublists

## Method doesn't increment ids because trials ids are obtained from datafiles

## (?) TO DO: method should check for duplicate trials

setMethod("addTrial", "Trials",

function(self, trialObject)

{

id <- trialObject@id

self@trialsList$ids <- c(self@trialsList$ids, id)

self@trialsList$trials <- c(self@trialsList$trials, trialObject)

return(self)

}

)

### addSubject

setGeneric("addSubject", function(self, subjectObject){standardGeneric("addSubject")})

# Method adds a subject object into the Subjects list with ids and subjects sublists

## Method prevents duplicating subject codes

setMethod("addSubject", "Subjects",

function(self, subjectObject)

{

subjCnt <- length(self@subjectsList$ids)

if (subjCnt == 0)

{

self@subjectsList$ids <- 1

self@subjectsList$subjects <- list(subjectObject)

return(self)

}

if (subjectObject@code %in% getSubjectCodes(self))

{

stop(paste("The subject with code", subjectObject@code, "already exists!"))

}

id <- tail(self@subjectsList$ids, n = 1) + 1

self@subjectsList$ids <- c(self@subjectsList$ids, id)

self@subjectsList$subjects <- c(self@subjectsList$subjects, subjectObject)

return(self)

}

)

### addFactorDefinition

setGeneric("addFactorDefinition", function(self, factor){standardGeneric("addFactorDefinition")})

# Method adds a factor definition (object of the class Factor) into the availableFactors data frame

## Method prevents adding factors with duplicate names

setMethod("addFactorDefinition", "AvailableFactors",

function(self, factor)

{

varName <- factor@varName

description <- factor@description

type <- factor@type

owner <- factor@owner

if (length(factor@levels) == 0) {levels = NA} else {levels <- factor@levels}

facCnt <- nrow(self@availableFactors)

if (facCnt == 0)

{

self@availableFactors <- data.frame(id = 1, varName = varName, description = description, type = type, levels = I(list(levels)), owner = owner, stringsAsFactors = F)

colnames(self@availableFactors) <- c("id", "varName", "description", "type", "levels", "owner")

return(self)

}

if (any(self@availableFactors$varName == varName) & any(self@availableFactors$owner == owner))

{

warning(paste("A factor with name", varName, "already exists for object class", owner))

} else

{

newFactorDef <- list(id = self@availableFactors[facCnt,1]+1, varName = varName, description = description, type = type, levels = I(list(levels)), owner = owner)

self@availableFactors <- rbind(self@availableFactors, newFactorDef)

}

return(self)

}

)

### addStimulus

setGeneric("addStimulus", function(self, stimulusObject){standardGeneric("addStimulus")})

# Method adds a Stimulus object into the Stimuli list with ids and stimuli sublists

## TO DO: method should prevent duplicates in Stimuli

setMethod("addStimulus", "Stimuli",

function(self, stimulusObject)

{

stimCnt <- length(self@stimuliList$ids)

if (stimCnt == 0)

{

self@stimuliList$ids <- 1

self@stimuliList$stimuli <- list(stimulusObject)

return(self)

}

id <- tail(self@stimuliList$ids, n = 1) + 1

self@stimuliList$ids <- c(self@stimuliList$ids, id)

self@stimuliList$stimuli <- c(self@stimuliList$stimuli, stimulusObject)

return(self)

}

)

### addAOI

setGeneric("addAOI", function(self, AOIObject){standardGeneric("addAOI")})

# Method adds an AOI object into the AOISet object

## TO DO: method should prevent duplicates in AOISet

setMethod("addAOI", "AOISet",

function(self, AOIObject)

{

if (length(self@AOIs) == 0)

{

self@AOIs <- AOIObject

}

self@AOIs <- c(self@AOIs, AOIObject)

return(self)

}

)

### addAOISet

setGeneric("addAOISet", function(self, AOISetObject, orderIndex){standardGeneric("addAOISet")})

# Method adds an AOISet object with into the AOISet object

## TO DO: method should prevent duplicates in AOISet

setMethod("addAOISet", "AOISets",

function(self, AOISetObject)

{

setsCnt <- length(self@AOISetsList$ids)

if (setsCnt == 0)

{

self@AOISetsList$ids <- 1

self@AOISetsList$AOISets <- list(AOISetObject)

return(self)

}

id <- tail(self@AOISetsList$ids, n = 1) + 1

self@AOISetsList$ids <- c(self@AOISetsList$ids, id)

self@AOISetsList$AOISets <- c(self@AOISetsList$AOISets, AOISetObject)

return(self)

}

)

### addFactorsRecord

setGeneric("addFactorsRecord", function(self, owner, ownerId, factors) {standardGeneric("addFactorsRecord")})

**Метод не реализован**

### addFactor

setGeneric("addFactor", function(self, factorId, value){standardGeneric("addFactor")})

**Метод не реализован**

### addDataRecord

setGeneric("addDataRecord", function(self, dataRecord){standardGeneric("addDataRecord")})

# Method adds a dataRecord object into the DataSample object

## Method prevents duplicates in DataSample (by a composite key: expID, subjectID, trialID)

setMethod("addDataRecord", "DataSample",

function(self, dataRecord)

{

expID <- dataRecord@expID

subjectID <- dataRecord@subjectID

trialID <- dataRecord@trialID

complexKey <- list(expID = expID, subjectID = subjectID, trialID = trialID)

if (tail(duplicated(rbind(self@keys,complexKey)),1))

{

warning("Such a record already exists!")

return(self)

} else

{

if (nrow(self@keys) == 0)

{

self@keys <- data.frame(complexKey)

self@eyesDataObjectsList <- append(self@eyesDataObjectsList, dataRecord@eyesDataObject)

self@analysisResultsList <- append(self@analysisResultsList, dataRecord@analysisResults)

self@statisticsList <- append(self@statisticsList, dataRecord@statistics)

return(self)

}

self@keys <- rbind(self@keys, complexKey)

self@eyesDataObjectsList <- append(self@eyesDataObjectsList, dataRecord@eyesDataObject)

self@analysisResultsList <- append(self@analysisResultsList, dataRecord@analysisResults)

self@statisticsList <- append(self@statisticsList, dataRecord@statistics)

return(self)

}

}

)

### Функция createRawDataRec

createRawDataRec <- function(filePath, readSettings, useExt, extFun)

{

if (!file.exists(filePath))

{

stop("Datafile not found!")

}

else

{

if (useExt)

{

# implement data loading using extFun

headerLines <- "NA"

asIsData <- as.data.frame(NA)

rawDataRecord <- new(Class = "RawDataRecord",

filePath = filePath,

headerLines = headerLines,

data = asIsData)

}

else

{

settings <- readSettings@readSettings

headerLines <- readLines(con = filePath, n = settings$skip, encoding = settings$encoding)

asIsData <- read.csv(filePath, sep = settings$sep,

skip = settings$skip,

comment.char = settings$comment.char,

header = settings$header,

blank.lines.skip = T,

check.names = F,

stringsAsFactors = F)

rawDataRecord <- new(Class = "RawDataRecord",

filePath = filePath,

headerLines = headerLines,

data = asIsData)

}

}

rawDataRecord

}

### addRawDataRecord

setGeneric("addRawDataRecord", function(self, filepath, readSettings, useExt, extFun){standardGeneric("addRawDataRecord")})

## TO DO: prevent creating duplicate records

setMethod("addRawDataRecord", "RawDataRecords",

function(self, filepath, readSettings, useExt, extFun)

{

newRawDataRec <- createRawDataRec(filePath = filepath, readSettings = readSettings, useExt = useExt, extFun = extFun)

rawDataRecCnt <- length(self@rawDataRecordsList$fileNumbers)

if (rawDataRecCnt == 0)

{

self@rawDataRecordsList$fileNumbers <- 1

self@rawDataRecordsList$rawDataRecords <- list(newRawDataRec)

return(self)

}

fileNum <- tail(self@rawDataRecordsList$fileNumbers, n = 1) + 1

self@rawDataRecordsList$fileNumbers <- c(self@rawDataRecordsList$fileNumbers, fileNum)

self@rawDataRecordsList$rawDataRecords <- c(self@rawDataRecordsList$rawDataRecords, newRawDataRec)

return(self)

}

)

### addRawDataRecords

setGeneric("addRawDataRecords", function(self, filesFolder, readSettings, useExt, extFun){standardGeneric("addRawDataRecords")})

## TO DO: prevent creating duplicate records

setMethod("addRawDataRecords", "RawDataRecords",

function(self, filesFolder, readSettings, useExt, extFun)

{

if (!file.exists(filesFolder))

{

stop("Data folder not found!")

}

filesToRead <- list.files(path = filesFolder, pattern = NULL, all.files = FALSE,

full.names = TRUE, recursive = FALSE,

ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)

rawDataRecords <- lapply(filesToRead, FUN = createRawDataRec, readSettings = readSettings, useExt = useExt, extFun = extFun)

rawDataRecCnt <- length(self@rawDataRecordsList$fileNumbers)

if (rawDataRecCnt == 0)

{

firstFileNum <- 1

}

else

{

firstFileNum <- tail(self@rawDataRecordsList$fileNumbers, n = 1) + 1

}

filesCnt <- length(rawDataRecords)

fileNumbers <- seq(firstFileNum, length.out = filesCnt)

self@rawDataRecordsList$fileNumbers <- c(self@rawDataRecordsList$fileNumbers, fileNumbers)

self@rawDataRecordsList$rawDataRecords <- c(self@rawDataRecordsList$rawDataRecords, rawDataRecords)

return(self)

}

)

## Методы печати содержимого объектов

### getSubjectCodes

setGeneric("getSubjectCodes", function(self){standardGeneric("getSubjectCodes")})

# Method returns a vector of subject codes

setMethod("getSubjectCodes", "Subjects",

function(self)

{

sapply(self@subjectsList$subjects, function(x) {return(x@code)})

}

)

### printDataSampleKeys

setGeneric("printDataSampleKeys", function(self){standardGeneric("printDataSampleKeys")})

# Method returns a data frame with data records composite keys values (for further use for data filtering)

setMethod("printDataSampleKeys", "DataSample",

function(self)

{

return(self@keys)

}

)