SideEye Documentation

Release 1.0.0a13

Amanda Doucette

Dec 09, 2019

Contents

1 API Reference **1**
- 1.1 Configuration . . . 1
- 1.2 Data . . . 4
- 1.3 Measures . . . 12
- 1.4 Parser . . . 19
- 1.5 Calculate Measures . . . 20
- 1.6 Output Formatting . . . 21

2 Getting Started **23**
- 2.1 Running the Script . . . 23
- 2.2 .DA1 File Formats . . . 24
- 2.3 Numeric Region File Formats . . . 24
- 2.4 Text Region File Formats . . . 25

3 The Configuration File **27**
- 3.1 wide_format . . . 28
- 3.2 da1_fields . . . 28
- 3.3 region_fields . . . 29
- 3.4 asc_parsing . . . 29
- 3.5 cutoffs . . . 29
- 3.6 region_measures . . . 30
- 3.7 trial_measures . . . 30
- 3.8 region_output . . . 31
- 3.9 trial_output . . . 31
- 3.10 terminal_output . . . 31

4 Default Configuration **33**

5 Version History **35**
- 5.1 1.0.0a13 . . . 35
- 5.2 1.0.0a12 . . . 35
- 5.3 1.0.0a11 . . . 35
- 5.4 1.0.0a10 . . . 35
- 5.5 1.0.0a9 . . . 35
- 5.6 1.0.0a8 . . . 36
- 5.7 1.0.0a7 . . . 36
- 5.8 1.0.0a6 . . . 36

5.9 1.0.0a5 . . . 36
5.10 1.0.0a4 . . . 36
5.11 1.0.0a3 . . . 36
5.12 1.0.0a2 . . . 36
5.13 1.0.0a1 . . . 36

6 Introduction **37**
6.1 Installation . . . 37

Python Module Index **39**

Index **41**

CHAPTER 1

API Reference

1.1 Configuration

Functions for parsing sideeye configurations from JSON files, and validating sideeye configuration dictionaries.

class sideeye.config.**ASCParsingConfig**(*asc_config: Dict[str, Union[int, bool]] = {}*)
ASC parser configuration.

fixation_min_cutoff
Minimum cutoff for including a fixation.

Type Union[int, bool]

max_saccade_dur
Maximum cutoff for saccade duration.

Type Union[int, bool]

blink_max_count
Maximum number of blinks before trial exclusion.

Type Union[int, bool]

blink_max_dur
Maximum blink duration before trial exclusion.

Type Union[int, bool]

Parameters **asc_config** (*Dict*) – ASC configuration dictionary.

class sideeye.config.**Configuration**(*config_file: str = None*)
SideEye configuration.

wide_format
Whether output should be in wide (True) or long (False) format.

Type bool

da1_fields
DA1 file configuration.

Type *DA1Config*

region_fields
Region file configuration.

Type *RegionConfig*

asc_parsing
ASC file configuration.

Type *ASCParsingConfig*

cutoffs
Fixation cutoff configuration.

Type *CutoffsConfig*

measures
Configuration for calculating measures.

Type *MeasuresConfig*

output
Output file configuration.

Type *OutputConfig*

terminal_output
Verbose output level.

Type int

Parameters **config_file** (*Optional[str]*) – Path to configuration JSON file.

class sideeye.config.**CutoffsConfig**(*cutoffs: Dict[str, Union[int, bool]] = {}*)
Fixation cutoff configuration.

min
Minimum cutoff for fixations.

Type int

max
Maximum cutoff for fixations.

Type int

include_fixation
Whether excluded fixations should be included in saccade duration.

Type bool

include_saccades
Whether the saccades into and out of an excluded fixation should be included in saccade duration.

Type bool

Parameters **cutoffs** (*dict*) – Cutoff configuration dictionary.

class sideeye.config.**DA1Config**(*da1_config: Dict[str, int] = {}*)
DA1 parser configuration.

index
Trial index column.

Type int

condition
Item condition column.

Type int

number
Item number column.

Type int

time
Total trial time column.

Type int

fixation_start
Column number of first fixation

Type int

Parameters **da1_config** (*Dict*) – DA1 configuration dictionary.

class sideeye.config.**MeasuresConfig**(*region_measures: Dict[str, Dict[str, Union[int, str]]], trial_measures: Dict[str, Dict[str, Union[int, str]]]*)
Region measure configuration.

region
Output configuration for region measures.

Type Dict[str, *OutputColumnConfig*]

trial
Output configuration for trial measures.

Type Dict[str, *OutputColumnConfig*]

all
Output configuration for all measures.

Type Dict[str, *OutputColumnConfig*]

names
List of all measure names.

Type List[str]

Parameters

- **region_measures** (*Dict[str, Dict]*) – Region measure configuration dictionary.
- **trial_measures** (*Dict[str, Dict]*) – Trial measure configuration dictionary.

class sideeye.config.**OutputColumnConfig**(*measure: str*, *measure_config: Dict[str*, *Union[int*, *str]] = {}*)
Configuration for a single output column.

cutoff
Cutoff for excluding measure from output.

Type int

header
Name of column in output file.

Type str

Parameters

- **measure** (*str*) – Name of measure.
- **measure_config** (*Dict*) – Measure configuration dictionary.

class sideeye.config.**OutputConfig**(*region_output: Dict[str, Dict[str, Union[int, str]]], trial_output: Dict[str, Dict[str, Union[int, str]]]*)
Output column configuration.

class sideeye.config.**RegionConfig**(*region_config: Dict[str, Union[int, bool]] = {}*)
Region parser configuration.

number
Item number column.

Type int

condition
Condition label column.

Type int

boundaries_start
First region boundary column position.

Type int

includes_y
Whether or not y values are included in the region file.

Type bool

Parameters **region_config** (*Dict*) – Region configuration dictionary.

sideeye.config.**validate_key**(*config_dict: Dict[str, Any], key: str, value_type: type, default: Any*)
Returns config_dict[key] if the value exists and if of type value_type, otherwise returns a default value.

1.2 Data

Contents

- *Data*
  - *Experiment*
  - *Trial*
  - *Item*
  - *Region*
  - *Saccade*
  - *Fixation*

– *Point*

The data module contains SideEye's core data structures - Experiments, Trials, Items, Regions, Saccades, Fixations, and Points. Raw data can be parsed into these objects, and measures can be calculated from them.

1.2.1 Experiment

An Experiment is the highest-level data object in the SideEye module, representing a single participant's data.

class sideeye.data.**Experiment**(*name: str, trials: List[sideeye.data.trial.Trial], filename: str = '', date: datetime.datetime = None*)

Represents the data of one participant in an experiment.

name
Name of experiment.

Type str

trials
Dictionary mapping (number, condition) tuples to trials.

Type dict

filename
Name of file.

Type str

date
Date of experiment.

Type Date

trial_indices
A dictionary mapping trial indices to (number, condition) tuples. Used to locate trials.

Type dict

Parameters

- **name** (*str*) – A string name/identifier for the participant.
- **trials** (*List[Trial]*) – Trials in the experiment.
- **filename** (*Optional[str]*) – Optional name of source file.
- **date** (*Optional[datetime]*) – Optional date.

get_trial(*number: str = None, condition: str = None, index: int = None*) → sideeye.data.trial.Trial
Get a trial by item number and condition, or by index if specified.

Parameters

- **number** (*ItemNum*) – Trial number.
- **condition** (*Condition*) – Trial condition.
- **index** (*int*) – Trial index.

1.2.2 Trial

A Trial represents the data from a participant reading one item. An individual Trial is identified by an index, and contains the total time spent reading the Item, lists of Fixations and Saccades associated with the Trial, and a dictionary of trial and region measures calculated for the Trial.

The list of saccades for a Trial is generated using the list of fixations. There are two parameters that affect how saccades are defined: *include_fixation* and *include_saccades*. If a Fixation is excluded from calculations, there are several ways to define a saccade. If *include_fixation* is true, the excluded fixation will be included in the duration of the Saccade. If *include_saccades* is true, the saccades surrounding an excluded fixation are included in the duration of the Saccade.

In the following example, the excluded fixation and surrounding saccades are included in the Saccade, which has a total duration of 30ms.

```
include_saccades: True
include_fixations: True

Time:      0        10      20           30      40       50
Fixations: |  fix1  |       | excl. fix. |       |  fix3  |
           ------------------------------------------------
Saccades:  |        |          saccade1          |        |

Trial.fixations = [fix1, fix2]
Trial.saccades = [saccade1]
```

If the excluded fixation is included in the saccade, but the surrounding saccades are not, the Saccade has a total duration of 10ms.

```
include_saccades: False
include_fixations: True

Time:      0        10      20           30      40       50
Fixations: |  fix1  |       | excl. fix. |       |  fix3  |
           ------------------------------------------------
Saccades:  |                |  saccade1  |                |

Trial.fixations = [fix1, fix3]
Trial.saccades = [saccade1]
```

If the surrounding saccades are included, but the excluded fixation is not, the total duration of the Saccade is 20ms - the sum of the durations of the two surrounding saccades.

```
include_saccades: True
include_fixations: False

Time:      0        10      20           30      40       50
Fixations: |  fix1  |       | excl. fix. |       |  fix3  |
           ------------------------------------------------
Saccades:  |        | sacc1 |  EXCLUDED  | sacc1 |        |

Trial.fixations = [fix1, fix3]
Trial.saccades = [sacc1(both parts combined)]
```

If the excluded fixation and surrounding saccades are both not included, no Saccade is added to the Trial. This is the default.

```
include_saccades: False
include_fixations: False

Time:      0        10      20           30      40       50
Fixations: |  fix1  |       | excl. fix. |       |  fix3  |
           ------------------------------------------------
Saccades:  |        |          EXCLUDED          |        |

Trial.fixations = [fix1, fix3]
Trial.saccades = []
```

The Trial object also contains all measures that have been calculated so far. *trial_measures* is a dictionary mapping measure names to calculation outputs. *region_measures* is a dictionary mapping region numbers to dictionaries mapping measure names to calculation outputs. Region measure outputs are in the format *{'value': output_value, 'fixations': fixation_in_calculation}*:

```
trial_measures = {
  'measure_1': False,
  'measure_2': 5000,
  ...
}

region_measures = {
  0: {
    'region_measure_1': {'value': 50, 'fixations': [fix1, fix3]},
    'region_measure_2': {'value': True, 'fixations': [fix4]},
    ...
  },
  1: {
  'region_measure_1': {'value': 148, 'fixations': [fix2, fix5, fix6]},
  'region_measure_2': {'value': False, 'fixations': None},
    ...
  },
  ...
}
```

class sideeye.data.**Trial**(*index: int, time: int, item: sideeye.data.item.Item, fixations: List[sideeye.data.fixation.Fixation], include_fixation: bool = False, include_saccades: bool = False*)

Represents an individual Trial in an experiment.

index
Trial index.

Type int

time
Total time of trial in milliseconds.

Type int

item
Item corresponding to trial data.

Type *Item*

fixations
List of fixations in trial.

Type List[*Fixation*]

saccades
A list of saccades in the trial.

Type List[*Saccade*]

trial_measures
Trial measures that have been calculated for the trial.

Type dict

region_measures
Region measures that have been calculated for the trial.

Type dict

Parameters

- **index** (*int*) – An identifier for the Trial. Must be greater than or equal to 0.
- **time** (*int*) – Total time of the Trial in milliseconds.
- **item** (*Item*) – An Item corresponding to the Trial.
- **fixations** (*List[Fixation]*) – A list of Fixations in the Trial.
- **include_fixation** (*bool*) – Boolean indicating whether an excluded fixation should be included in a saccade.
- **include_saccades** (*bool*) – Boolean indicating whether saccades surrounding an excluded fixation should be included in a saccade.

fixation_count() → int
Return the number of fixations in the item.

1.2.3 Item

An Item represents the text that is displayed to a participant during a trial. A number and condition identify the item, which consists of a list of regions, and optionally a list of labels for the regions.

class sideeye.data.**Item**(*number: str, condition: str, regions: List[sideeye.data.region.Region], labels: Sequence[Union[int, str]] = None*)
Represents an Item in an experiment.

number
An identifier for the Item.

Type int

condition
An identifier for the condition of the Item.

Type int

regions
A list of Regions in the Item.

Type List[*Region*]

labels
A list of labels for the regions. If labels are not provided, integer indices are used. All labels are unique.

Type List[Union[int, str]]

Parameters

- **number** (*ItemNum*) – An identifier for the Item.
- **condition** (*Condition*) – An identifier for the condition of the Item.
- **regions** (*List[Region]*) – A list of Region objects in the Item. All regions must be unique.
- **labels** (*List[Union[int, str]]*) – A list of labels for regions. If not provided, integer indices will be used. All labels must be unique.

find_region(*x_pos: int*, *y_pos: int*) → sideeye.data.region.Region
Get the region containing position (x_pos, y_pos).

Parameters

- **x_pos** (*int*) – X (character) position of a location.
- **y_pos** (*int*) – Y (line) position of a location.

get_region(*label: Union[str, int]*) → sideeye.data.region.Region
Get Region with matching label.

Parameters **label** (*Union[str, int]*) – A region label.

region_count() → int
Get the number of Regions in the Item.

1.2.4 Region

A Region represents the boundaries of a region in an Item as (character, line) points in the text of the Item. A region also has a label, number, and optionally the text contained in the region.

class sideeye.data.**Region**(*start: sideeye.data.point.Point*, *end: sideeye.data.point.Point*, *length: int = None*, *text: str = ''*, *label: Union[str, int] = 'undefined'*, *number: int = None*)
Represents a Region as x and y coordinates for character positions of the beginning and end of the Region.

start
The character and line position of the beginning of the Region.

Type *Point*

end
The character and line position of the end of the Region.

Type *Point*

length
Length of region in characters.

Type Optional[int]

text
Text contained in the region.

Type Optional[str]

label
A label for the region.

Type Optional[Union[str, int]]

number
A number identifier for the region.

Type Optional[int]

Parameters

- **start** (Point) – The character and line position of the beginning of the Region.
- **end** (Point) – The character and line position of the end of the Region.
- **length** (*Optional[int]*) – Length of region in characters.
- **text** (*Optional[str]*) – Text contained in the region.
- **label** (*Optional[Union[str, int]]*) – A label for the region.
- **number** (*Optional[int]*) – A number identifier for the region.

1.2.5 Saccade

A Saccade is the period of time between two fixations. The start of the Saccade is the Fixation before the Saccade, and the end is the Fixation after the Saccade. If the location of the end Fixation is earlier in the Item the location of the start Fixation, the Saccade is a regression.

class sideeye.data.**Saccade**(*duration: int*, *regression: bool*, *start: sideeye.data.fixation.Fixation*, *end: sideeye.data.fixation.Fixation*)
Represents a saccade as the time in milliseconds between two fixations.

Parameters

- **duration** (*int*) – Duration in milliseconds of the saccade.
- **regression** (*bool*) – True if the saccade is a regression, false otherwise.
- **start** (Fixation) – The fixation before the saccade.
- **end** (Fixation) – The fixation after the saccade.

1.2.6 Fixation

A Fixation represents a period of time where a participant fixated on an item in a trial. Fixation position is represented by character and line position in the item. A Fixation can also hold information about the region of the item it occurred in, and whether or not it has been excluded by fixation cutoff parameters.

class sideeye.data.**Fixation**(*position: sideeye.data.point.Point*, *start: int*, *end: int*, *index: int*, *region: sideeye.data.region.Region*, *excluded: bool = False*)
Represents a single Fixation by location, start time, and end time.

char
Character position of the fixation. None of position is negative.

Type Optional[int]

line
Line position of the fixation. None if position is negative.

Type Optional[int]

start
Start time of Fixation in milliseconds since trial start.

Type int

end
End time of Fixation in milliseconds since trial start.

Type int

index
Index of where the Fixation occurred in a trial.

Type int

region
Region the Fixation occurred in.

Type *Region*

excluded
Whether or not the fixation will be excluded from calculations.

Type bool

Parameters

- **position** (Point) – The character and line position of the Fixation, zero-indexed.
- **start** (*int*) – Start time for the Fixation, in milliseconds.
- **end** (*int*) – End time for the Fixation, in milliseconds.
- **index** (*int*) – An index of where the fixation occurred in a trial.
- **region** (Region) – Region the Fixation occurs in.
- **excluded** (*Optional[boolean]*) – Whether the fixation should be excluded from calculations.

assign_region(*region: sideeye.data.region.Region*)
Assign a Region object to the Fixation.

Parameters **region** (Region) – Region to assign to the fixation.

duration() → int
Fixation duration in ms.

1.2.7 Point

A Point represents an (x, y) location. It is used to represent fixation positions and region boundaries.

class sideeye.data.**Point**(*x: int*, *y: int*)
An (x, y) location.

x
x location.

Type int

y
y location.

Type int

1.3 Measures

Contents

- *Measures*
  - *Trial Measures*
  - *Region Measures*

1.3.1 Trial Measures

Trial-based eye-tracking measures. These measures are calculated for each trial of an experiment.

`sideeye.measures.trial.`**`average_backward_saccade`**(*trial: sideeye.data.trial.Trial*)

Average saccade duration between fixations moving backward through the sentence, or the average duration of a regression saccade.

```
backward_saccades = 0
total_time = 0

for saccade in trial:
    if saccade is regression:
        backward_saccades += 1
        total_time += saccade.duration

if backward_saccades is 0:
    return 0
else:
    return total_time / backward_saccades
```

`sideeye.measures.trial.`**`average_forward_saccade`**(*trial: sideeye.data.trial.Trial*)

Average saccade duration between fixations moving forward through the sentence.

```
forward_saccades = 0
total_time = 0

for saccade in trial:
    if saccade is not regression:
        forward_saccades += 1
        total_time += saccade.duration

if forward_saccades is 0:
    return 0
else:
    return total_time / forward_saccades
```

`sideeye.measures.trial.`**`fixation_count`**(*trial: sideeye.data.trial.Trial*)

Total number of non-excluded fixations in a trial.

```
return length of [non-excluded fixations in trial]
```

`sideeye.measures.trial.`**`latency_first_regression`**(*trial: sideeye.data.trial.Trial*)

Time until the end of the fixation before the first regression saccade in the trial, where a regression is defined as

a saccade where the position of the fixation at the beginning of the saccade is later in the item than the position of the fixation at the end of the saccade. None if there are no regressions.

```
for saccade in trial:
    if saccade is regression:
        return saccade.start_fixation.end_time
    else:
        return None
```

sideeye.measures.trial.**location_first_regression**(*trial: sideeye.data.trial.Trial*)

(x, y) character position of the last fixation before the first regression saccade in the trial, where a regression is defined as a saccade where the position of the fixation at the beginning of the saccade is later in the item than the position of the fixation at the end of the saccade. None if there are no regressions.

```
for saccade in trial:
    if saccade is regression:
        return (saccade.start_fixation.char, saccade.start_fixation.line)
    else:
        return None
```

sideeye.measures.trial.**percent_regressions**(*trial: sideeye.data.trial.Trial*)

Proportion of saccades that are regressions from the location of the previous fixation, where a regression is defined as a saccade where the position of the fixation at the beginning of the saccade is later in the item than the position of the fixation at the end of the saccade. None if there are no saccades.

```
regressions = 0

for saccade in trial:
    if saccade is regression:
        regressions += 1

return regressions / (length of [trial.saccades])
```

sideeye.measures.trial.**trial_total_time**(*trial: sideeye.data.trial.Trial*)

Total time in the trial.

```
return total trial time from .DA1 file OR end time of last non-excluded fixation
```

1.3.2 Region Measures

Region-based eye-tracking measures. These measures are calculated for each region of each trial in an experiment.

Many of these measures use a list of first pass fixations in the region in calculating the measure. First pass fixations are defined as:

```
def get_first_pass_fixations(trial, region_number):
    first_pass_fixations = []

    for fixation in trial:
        if fixation.region_number > region_number:
            break
        if length of first_pass_fixations > 0 and fixation.region_number is not␣
↪region_number:
            break
        if fixation.region_number is region_number and fixation is not excluded:
            first_pass_fixations += [fixation]
```

(continues on next page)

(continued from previous page)

```
    return first_pass_fixations
```

sideeye.measures.region.**skip**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

True if the reader fixates a region beyond the target region before fixating the target region or the target region is never fixated, False otherwise.

```
if length of get_first_pass_fixations(trial, region_number) is 0:
    return True
else return False
```

sideeye.measures.region.**first_pass_regressions_in**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

This measure is True if the reader made a fixation in the region after fixating on any region to the right of it, False if they only fixated on the region after fixating on regions to the left, and None if the region was never fixated.

```
region_fixations = [all non-excluded fixations in region]

if length of region_fixations is 0:
    return None

for fixation in region_fixations:
    if previous_fixation.region_number > region_number:
        return True

return False
```

sideeye.measures.region.**first_pass_regressions_out**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

This measure is True if the reader made a regression out of the region on the first pass - that is, exited a region to the left after the first pass. The measure is False if they exited to the right, and None if the region was not fixated during first pass reading.

```
fp_fixations = get_first_pass_fixations(trial, region_number)

if length of fp_fixations is 0:
    return None

next_fixation = next non-excluded fixation after last fixation in fp_fixations

if next_fixation.region_number < region_number:
    return True
else:
    return False
```

sideeye.measures.region.**landing_position**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The location relative to the beginning of the region (in number of characters) of the first fixation during the first pass, where the first character in the region is at position (0, 0). If the region was skipped, this measure is None.

```
fp_fixations = get_first_pass_fixations(trial, region_number)
```

(continues on next page)

(continued from previous page)

```
if (length of fp_fixations is not 0
        and fp_fixations[0].char is not None
        and fp_fixations[0].line is not None):
    return (fp_fixations[0].char - region.start.char,
            fp_fixations[0].line - region.start.line)

else:
    return None
```

sideeye.measures.region.**launch_site**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The location of the last fixation prior to the first fixation in the region. This measure returns an (x, y) tuple, where x is either the number of characters from the the beginning of the target region and y is the number of lines from the beginning of the region. -1 indicates the last character of the region preceding the target region, or the preceding line, and increasing negative numbers indicate further launch sites. If the region was skipped this measure is None.

```
fixations = [fixation for fixation in trial if fixation is not excluded]

for (index, fixation) in enumerate(fixations):
    if fixation.region_number > region_number:
        break
    if fixation.region_number == region_number:
        if index == 0:
            break

        if fixations[index - 1].char is None:
            launch_char = fixations[index - 1].char
        else:
            launch_char = fixations[index - 1].char - fixation.region.start.char

        if fixations[index - 1].line is None:
            launch_line = fixations[index - 1].line
        else:
            launch_line = fixations[index - 1].line - fixation.region.start.line

        return (launch_char, launch_line)

return None
```

sideeye.measures.region.**first_pass_fixation_count**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The number of fixations made in a region before leaving it during first pass. If the region was skipped this measure is None.

```
fp_fixations = get_first_pass_fixations(trial, region_number)

if length of fp_fixations is 0:
    return None
else:
    return length of fp_fixations
```

sideeye.measures.region.**first_fixation_duration**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The duration of the first fixation in a region during first pass reading (i.e., before the reader fixates areas beyond

the region). If this region is skipped during first pass, this measure is None.

```
fp_fixations = get_first_pass_fixations(trial, region_number)

if length of fp_fixations is 0:
    return None
else:
    return duration of first fixation in fp_fixations
```

sideeye.measures.region.**single_fixation_duration**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

If there is only one fixation on the region during first pass reading, this measure is the duration of that fixation. If the region is skipped during first pass, the measure is None.

```
fp_fixations = get_first_pass_fixations(trial, region_number)

if length of fp_fixations is 1:
    return duration of fixation in fp_fixations
else:
    return None
```

sideeye.measures.region.**first_pass**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The summed duration of all fixations in a region during first pass (i.e., before the reader fixates areas beyond the region). If this region is skipped during first pass, this measure is None.

```
fp_fixations = get_first_pass_fixations(trial, region_number)

if length of fp_fixations is 0:
    return None

total = 0

for fixation in fp_fixations:
    total += fixation.duration()

return total
```

sideeye.measures.region.**go_past**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

Also known as regression path duration. The summed duration of all fixations starting from the first fixation in the region, including any fixations in prior regions until the reader goes past the region (i.e., before the reader fixates areas beyond the region). If this region is skipped during first pass, this measure is None.

```
if length of get_first_pass_fixations(trial, region_number) is 0:
    return None

total = 0

for fixation in trial:
    if fixation is not excluded:
        if fixation.region_number > region_number:
            break
        if total is not 0 or fixation.region_number is region_number:
            total += fixation.duration()

return total
```

sideeye.measures.region.**total_time**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The summed duration of all fixations in the region that occur at any time during the trial. If this region is never fixated this measure is 0.

```
total = 0

for fixation in trial:
    if fixation.region_number is region_number and fixation is not excluded:
        total += fixation.duration()

return total
```

sideeye.measures.region.**right_bounded_time**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The summed duration of all fixations starting from the first fixation in the region, excluding any fixations in prior regions until the reader goes past the region (i.e., until the reader fixates areas beyond the region). If this region is skipped during first pass, this measure is None.

```
if length of get_first_pass_fixations(trial, region_number) is 0:
    return None

total = 0

for fixation in trial:
    if fixation is not excluded:
        if fixation.region_number > region_number:
            break
        if fixation.region_number is region_number:
            total += fixation.duration()

return total
```

sideeye.measures.region.**reread_time**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The summed duration of all fixations in the region that occur after a region to the right has been fixated. If this region is never fixated this measure is 0.

```
total = 0
reread = False

for fixation in trial:
    if fixation is not excluded:
        if fixation.region_number > region_number:
            reread = True
        if reread and fixation.region_number is region_number:
            total += fixation.duration()

return total
```

sideeye.measures.region.**second_pass**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The summed duration of all fixations in the region that occur on a region after that region has been exited in either direction for the first time. If this region is never fixated this measure is 0.

```
total = 0
first_pass = False
exited = False
```

(continues on next page)

(continued from previous page)

```
for fixation in trial:
    if fixation is not excluded:
        if fixation.region_number is region_number:
            first_pass = True
        if first_pass and fixation.region_number is not region_number:
            exited = True
        if first_pass and exited and fixation.region_number is region_number:
            total += fixation.duration()

return total
```

sideeye.measures.region.**spillover_time**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The duration of fixations on the region immediately following the region of interest, where the previous fixation was on the region of interest. If there are no such fixations, the measure is None.

```
total = 0
visited_region = False

for fixation in trial:
    if fixation is not excluded:
        if visited_region and fixation.region_number is not region_number + 1:
            visited_region = False
        if fixation.region_number is region_number:
            visited_region = True
        if visited_region and fixation.region_number is region_number + 1:
            total += fixation.duration()

if total is 0:
    return None

return total
```

sideeye.measures.region.**refixation_time**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

The sum of all first-pass fixations excluding the first fixation. If there was a single fixation in the region or it was skipped this measure is None.

```
fp_fixations = get_first_pass_fixations(trial, region_number)

if length of fp_fixations is 0 or 1:
    return None

total = 0

for fixation in fp_fixations[1:]:
    total += fixation.duration()

return total
```

sideeye.measures.region.**go_back_time_char**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

Go-back time is the time (in ms) until the first regression is made after encountering a region. For the purposes of go-back time, any fixation which lands to the left of the preceding fixation is considered a regression; the landing site regression does not need to precede the critical region. If a region was fixated, it is measured from the onset of the first fixation in that regreion. If a region was skipped, it is measured from the offset of the

preceding fixation. The end point is the end of the fixation that precedes the regression. If no regression is made after the critical region, this measure is 'None.'

go_back_time_char defines a regression character by character: any fixation which lands on a character to the left of the preceding fixation counts as a regression.

sideeye.measures.region.**go_back_time_region**(*trial: sideeye.data.trial.Trial*, *region_number: int*) → sideeye.types.RegionMeasure

Go-back time is the time (in ms) until the first regression is made after encountering a region. For the purposes of go-back time, any fixation which lands to the left of the preceding fixation is considered a regression; the landing site regression does not need to precede the critical region. If a region was fixated, it is measured from the onset of the first fixation in that regreion. If a region was skipped, it is measured from the offset of the preceding fixation. The end point is the end of the fixation that precedes the regression. If no regression is made after the critical region, this measure is 'None.'

go_back_time_region defines a regression region by region: any fixation which lands on a region to the left of the preceding fixation counts as a regression.

1.4 Parser

1.4.1 Experiment Parser

This module contains functions for parsing experiments.

sideeye.parser.experiment.**parse**(*experiment_file: str*, *region_file: str*, *config: sideeye.config.Configuration = <sideeye.config.Configuration object>*) → sideeye.data.experiment.Experiment

Given a DA1 file and region file, and config file, parse an Experiment. If config is not provided, default config will be used.

Parameters

- **experiment_file** (*str*) – Name of DA1 or ASC file.
- **region_file** – Name of region file (.cnt, .reg, or .txt).
- **config** (Configuration) – Configuration.

1.4.2 ASC Parser

A parser for .ASC files.

sideeye.parser.asc.**parse**(*asc_file: str, items: Dict[str, Dict[str, sideeye.data.item.Item]], config: sideeye.config.ASCParsingConfig = <sideeye.config.ASCParsingConfig object>*)

Parses a .ASC file into an Experiment object.

Parameters

- **asc_file** (*string*) – Path to .ASC file.
- **items** (*Dict[str, Dict[str,* Item*]]*) – List of items in experiments.
- **config** (ASCParsingConfig) – Configuration for .ASC parsing.

1.4.3 DA1 Parser

A file parser for DA1 data files.

sideeye.parser.da1.**parse**(*filename: str, items: Dict[str, Dict[str, sideeye.data.item.Item]], config: sideeye.config.Configuration = <sideeye.config.Configuration object>, da1_type: str = None*) → sideeye.data.experiment.Experiment

Parses DA1-like files into sideeye Experiment objects, given column positions.

Parameters

- **filename** (*str*) – DA1 file.
- **items** (*Dict[ItemNum, Dict[Condition, Item]]*) – List of items in the experiment.
- **da1_type** (*str*) – Type of DA1 file - *timdrop*, *robodoc*, or *None* for any other type.

1.4.4 Region Parser

A file parser for region data files.

sideeye.parser.region.**textfile**(*filename: str, verbose: int = 0*) → Dict[str, Dict[str, sideeye.data.item.Item]]

A parser for a region text file, with regions separated by `/` and lines within an item separated by `\n`. `\n` is ignored in counting the length of a region.

Each line should contain one Item, with the item number, followed by a space or tab (`\t`), the item condition, another space or tab, and the region string.

For example: `1 2 This is item one /condition two.\n/There are two lines /and four regions.`

sideeye.parser.region.**file**(*filename: str, config: sideeye.config.Configuration = <sideeye.config.Configuration object>, verbose: int = 0*) → Dict[str, Dict[str, sideeye.data.item.Item]]

Parses a .reg or .cnt file into a dictionary of sideeye Item objects.

Parameters

- **filename** (*str*) – Region filename.
- **config** (Configuration) – Configuration.

sideeye.parser.region.**text**(*region_string: str*) → List[sideeye.data.region.Region]

A parser for region strings, with regions separated by `/`.

Single-line Example: `This is region 1/ This is region 2/ This is region 3/ This is region 4.`

Multi-line Example: `This is region 1/ This is region 2/ This is region 3/ This is region 4.`

1.5 Calculate Measures

This module contains functions for calculating measures on parsed experiments. Calculated measures are saved as a dictionary in the *measures* attribute of the experiment. These functions do not return anything, they only calculate the measure or measures on each trial or region of the experiments.

sideeye.calculate.**calculate_measure**(*experiments: List[sideeye.data.experiment.Experiment], measure: str, verbose: int = 0*)

Given an array of experiments and the name of a measure, calculate the measure for every trial in the experiment.

Parameters

- **experiments** (*List[Experiment]*) – List of experiments to calculate measures for.
- **measure** (*str*) – Name of measure to calculate.
- **verbose** (*int*) – Debugging output level.

sideeye.calculate.**calculate_all_measures**(*experiments: List[sideeye.data.experiment.Experiment], output_file: str = None, config: sideeye.config.Configuration = <sideeye.config.Configuration object>*)

Given an array of experiments and config file, calculate all measures specified in the config file for the experiment, and optionally output the results as a csv.

Parameters

- **experiments** (*List[Experiment]*) – List of experiments to calculate measures for.
- **output_file** (*str*) – Name of output file. if *None*, no file is produced.
- **config** (Configuration) – SideEye configuration.

1.6 Output Formatting

This module contains functions to generate csv reports of measures calculated for experiments.

sideeye.output.**measure_output**(*measure: str, cutoff: int, columns: Dict[str, sideeye.config.OutputColumnConfig], experiment: sideeye.data.experiment.Experiment, trial: sideeye.data.trial.Trial, region: Optional[sideeye.data.region.Region]*) → str

Generates a formatted output string for an individual measure.

Parameters

- **measure** (*str*) – Name of measure.
- **cutoff** (*int*) – Cutoff value for measure. Use -1 for no cutoff.
- **columns** (*Dict[str, Dict]*) – Dict of columns to output.
- **experiment** (Experiment) – Experiment to generate output string for.
- **trial** (Trial) – Trial to generate output for.
- **region** (*Optional[Region]*) – Region to generate output for.

sideeye.output.**generate_region_output**(*experiments: List[sideeye.data.experiment.Experiment], config: sideeye.config.Configuration = <sideeye.config.Configuration object>*) → str

Generates a string in csv format of a list of experiments' region measures using columns specified in config file.

Parameters

- **experiments** (*List[Experiment]*) – List of experiments.
- **config** (Configuration) – Configuration.

sideeye.output.**generate_trial_output**(*experiments: List[sideeye.data.experiment.Experiment], config: sideeye.config.Configuration = <sideeye.config.Configuration object>*) → str

Generates a string in csv format of list of experiments' trial measures using columns and measures specified in config file.

Parameters

- **experiments** (*List[Experiment]*) – List of experiments.
- **config** (Configuration) – Configuration.

sideeye.output.**generate_all_output**(*experiments: List[sideeye.data.experiment.Experiment], config: sideeye.config.Configuration = <sideeye.config.Configuration object>*) → str

Generates a string in csv format of all measures specified in config file for a list of experiments. :param experiments: List of experiments. :type experiments: List[Experiment] :param config: Configuration. :type config: Configuration

sideeye.output.**generate_all_output_wide_format**(*experiments: List[sideeye.data.experiment.Experiment], config: sideeye.config.Configuration = <sideeye.config.Configuration object>*) → str

Generates a string in csv format of all measures specified in config file for a list of experiments, with all measures as columns.

Parameters

- **experiments** (*List[Experiment]*) – List of experiments.
- **config** (Configuration) – Configuration.

CHAPTER 2

Getting Started

While the calculations and outputs generated by SideEye can be customized, eye tracking experiment files can easily be processed with the `sideeye` script without additional configuration. To use this script you will need:

- A region file (.cnt or .reg)
- One or more .DA1 or .ASC files
- A *Configuration* JSON file (optional)

2.1 Running the Script

After installing SideEye, open a Terminal or Command Prompt window and run the following command:

```
sideeye
```

A file selection window will prompt you for each of the necessary files. First is the configuration JSON file. If your DA1, ASC, or region files differ from the formats specified in the *default configuration*, use a custom configuration file. See *The Configuration File* for a full explanation. Otherwise, click "Cancel" and the default configuration will be used.

Next you will be prompted to select DA1 or ASC files. Select one or more files associated with the same experiment. See below for more detail on DA1 file formats.

The next file selection prompt is for a region file. Select the .reg or .cnt file associated with the experiment. See below for more detail on the types of region files.

Finally, enter an output file name. If SideEye is successful in processing your experiment, a CSV of all the measures calculated for each trial will be generated.

2.2 .DA1 File Formats

A .DA1 file describes the position and timing of fixations. Each line of the file represents a single trial in an experiment. The first fields in the line contain information identifying the trial. After these fields, fixations are represented by groups of four numbers - x-position, y-position, start time, and end time. Five fields are necessary for parsing a .DA1 file:

`index`: An integer identifying a single trial, usually the first number in the line.

`condition`: A value identifying the condition of the trial.

`number`: A value identifying the item of the trial.

`time`: The total time of a trial, sometimes either not included in DA1 files or at the end of every line. If this is true, this parameter should be `-1`.

`fixation_start`: The position in a line where fixation data starts.

The default configuration for .DA1 files is:

```
"da1_fields" : {
  "index": 0,
  "condition": 1,
  "number": 2,
  "time": -1,
  "fixation_start": 8
}
```

This default configuration matches a .DA1 file with the following format:

```
Position:   0   |   1   |   2   | 3 | 4 | 5 | 6 | 7 |   8   |   9   | ... |  -1  |
-------------------------------------------------------------------------------------
Value:    index | cond. | num.  |   |   |   |   |   | fix.  | fix.  | ... | time |
```

If your .DA1 files do not match this format, copy *default_config.json* into a new file in the same directory, and edit the `da1_fields` section to match your .DA1 file format.

2.3 Numeric Region File Formats

Region files (typically .cnt or .reg) describe regions of interest in items of the experiment. Regions are defined by a character position of the beginning and ending of the region. Character positions can either be a single integer for single-line items, or a pair (line, character) of integers for multi-line items. A line in a region file contains the number and condition of an item, followed by the beginning and end positions of the regions. Four fields are necessary for parsing a region file.

`number`: Item identifier.

`condition`: Item condition.

`boundaries_start`: The position where region boundaries start in a line.

`includes_y`: True for multi-line items where regions are bounded by character and line position pairs, false for single-line items where regions are bounded by single integer character positions.

The default configuration for region files is:

```
"region_fields": {
  "number": 0,
  "condition": 1,
  "boundaries_start": 3,
  "includes_y": false
}
```

The default configuration matches a region file with the following format:

```
Position:   0   |   1   | 2 |   3   |   4   |   5   |   6   | ... |
-------------------------------------------------------------------
Value:    num.  | cond. |   |  r1   |  r1   |  r2   |  r2   | ... |
```

If your region file does not match this format, copy *default_config.json* into a new file in the same directory, and edit the `region_fields` section to match your region file format.

2.4 Text Region File Formats

Text region files are a new format for use with SideEye. This format can be used instead of a .cnt or .reg file, and allows for region length to be calculated automatically, and for region text to be included in the experiment output. There is only one format for this type of region file, so the `region_fields` configuration section is ignored. In this format, each line represents an item.

In this format, each line should have a number or string identifying the item number, followed by a tab or space, a number or string identifying the item condition, another tab or space, and then the text of the item. In the region text, regions are separated by the `/` character, and lines are separated by `\n`.

For example, the following line represents an item with number 1, condition 3, and four regions on two lines:

```
1    3    This is an item/ with two lines/\nand four/ regions.
```

The item displayed by the eye tracker would be:

```
This is an item with two lines
and four regions.
```

CHAPTER 3

The Configuration File

Contents

- *The Configuration File*
  - *wide_format*
  - *da1_fields*
  - *region_fields*
  - *asc_parsing*
  - *cutoffs*
  - *region_measures*
  - *trial_measures*
  - *region_output*
  - *trial_output*
  - *terminal_output*

The configuration file is a JSON file specifying the parameters used by SideEye's parsing and output functions. There is a *default configuration* used by SideEye if a custom configuration is not provided. A custom configuration file must contain the same fields as the default configuration file. To use a custom configuration, copy *default_config.json*, change values to your configuration, and provide the file name in functions requiring a configuration in your code. If any section is excluded from a custom configuration, default values will be used.

The sections of the config file are:

`wide_format`: Whether the output file should be in wide or long format.

`da1_fields`: The locations of values in each line of a .DA1 file.

`region_fields`: The locations of values in each line of a region file.

`asc_parsing`: Cutoff values for ASC parsing.

`cutoffs`: Fixation duration cutoffs.

`region_measures`: Used in region measure calculation and output.

`trial_measures`: Used in trial measure calculation and output.

`region_output`: Reported columns for region measures in output file.

`trial_output`: Reported columns for trial measures in output file.

`terminal_output`: Amount of status/debugging information to output to terminal.

3.1 wide_format

A boolean (`true/false`) indicating whether the output should be in wide or long format. If `true`, the output will be in wide format, with each measure as a column of the csv. If `false`, the output will be in long format, with each measure as a separate row of the csv.

For example, the column headers in long format will be:

```
experiment_name,trial_id,trial_total_time,item_id,item_condition,region_number,
↪measure,value
```

The column headers in wide format will be:

:: experiment_name,trial_id,trial_total_time,item_id,item_condition,region_number,skip,first_pass_regressions_out, ... ,average_forward_saccade,average_backward_saccade

3.2 da1_fields

This section contains parameters needed for parsing .DA1 files. A .DA1 file describes the position and timing of fixations. Each line of the file represents a single trial in an experiment. The first fields in the line contain information identifying the trial. After these fields, fixations are represented by groups of four numbers - x-position, y-position, start time, and end time. Five fields are necessary for parsing a .DA1 file:

`index`: An integer identifying a single trial, usually the first number in the line.

`condition`: A value identifying the condition of the trial.

`number`: A value identifying the item of the trial.

`time`: The total time of a trial, sometimes either not included in DA1 files or at the end of every line. If this is true, this parameter should be `-1`.

`fixation_start`: The position in a line where fixation data starts.

The default configuration for .DA1 files is:

```
"da1_fields" : {
  "index": 0,
  "condition": 1,
  "number": 2,
  "time": -1,
  "fixation_start": 8
}
```

This default configuration matches a .DA1 file with the following format:

```
Position:   0   |   1   |   2   | 3 | 4 | 5 | 6 | 7 |   8   |   9   | ... |  -1  |
-------------------------------------------------------------------------------------
Value:    index | cond. | num.  |   |   |   |   |   | fix.  | fix.  | ... | time |
```

3.3 region_fields

Similar to `da1_fields`, this section specifies the positions of values in each line of a region file. If the newer text region format is used, this section of the configuration is ignored. Region files (typically .cnt or .reg) describe regions of interest in items of the experiment. Regions are defined by a character position of the beginning and ending of the region. Character positions can either be a single integer for single-line items, or a pair (line, character) of integers for multi-line items. A line in a region file contains the number and condition of an item, followed by the beginning and end positions of the regions. Four fields are necessary for parsing a region file.

`number`: Item identifier.

`condition`: Item condition.

`boundaries_start`: The position where region boundaries start in a line.

`includes_y`: True for multi-line items where regions are bounded by character and line position pairs, false for single-line items where regions are bounded by single integer character positions.

The default configuration for region files is:

```
"region_fields": {
  "number": 0,
  "condition": 1,
  "boundaries_start": 3,
  "includes_y": false
}
```

The default configuration matches a region file with the following format:

```
Position:   0   |   1   | 2 |   3   |   4   |   5   |   6   | ... |
-----------------------------------------------------------------
Value:    num.  | cond. |   |  r1   |  r1   |  r2   |  r2   | ... |
```

3.4 asc_parsing

This section contains parameters for parsing ASC files. If only DA1 files are being processed, this section can be ignored. The following four optional fields are used for fixation and trial exclusion in ASC parsing:

`fixation_min_cutoff`: Minimum cutoff for including a fixation.

`max_saccade_dur`: Maximum cutoff for saccade duration.

`blink_max_count`: Maximum number of blinks before trial exclusion.

`blink_max_dur`: Maximum blink duration before trial exclusion.

3.5 cutoffs

This section contains cutoffs for fixations. If a fixation's duration is less than *min* or greater than *max*, it will be excluded from measure calculations. *include_fixation* and *include_saccades* describe how excluded fixations should

be handled when calculating saccades. For more information, see the examples in *Trials*.

`min`: Minimum cutoff time for fixation duration.

`max`: Maximum cutoff time for fixation duration.

`include_fixation`: When true, if a fixation is excluded by cutoffs, its duration is included in the duration of the saccade between the previous and next non-excluded fixations.

`include_saccades`: When true, if a fixation is excluded by cutoffs, the duration of the saccade into and out of the fixation is included in the saccade between the previous and next non-excluded fixations.

3.6 region_measures

This section contains parameters for region measure calculation and output. It is a list of all calculated region measure, each with two parameters:

`cutoff`: An optional cutoff value for the measure. If the calculated measure is greater than this value, its value in the output report is `CUTOFF`. For some measures, where the value is not numerical, this parameter is ignored. If cutoff is not included in the configuration, or is set to `None` or `-1`, it will not be used.

`exclude`: A boolean (true/false) value specifying whether the measure should be excluded from the output report. If true, the measure will be excluded. If false or not included in the configuration, the measure will be included.

`header`: A string used as a header for the measure in wide output format. If not set, the function name will be used as the header.

The measures in this section of the config file are:

```
skip
first_pass_regressions_out
first_pass_regressions_in
first_fixation_duration
single_fixation_duration
first_pass
go_past
total_time
right_bounded_time
reread_time
second_pass
spillover_time
refixation_time
landing_position
launch_site
first_pass_fixation_count
go_back_time_region
go_back_time_char
```

3.7 trial_measures

This section contains parameters for trial measure calculation and output. Each measure has the same parameters as `region_measures`. The measures included in this section are:

```
location_first_regression
latency_first_regression
```

(continues on next page)

(continued from previous page)

```
fixation_count
percent_regressions
trial_total_time
average_forward_saccade
average_backward_saccade
```

3.8 region_output

This section specifies the columns that should be included in the output file for region measures. Each output column has two parameters:

`exclude`: A boolean (true/false) value specifying whether the column should be excluded from the output report. If true, the column will be excluded. If false or not included in the configuration, the column will be included.

`header`: A title for the header of the column. If not specified, the default name will be used.

Columns included in this section are:

`experiment_name`: Name of experiment.

`filename`: Filename of DA1 file.

`date`: Date of DA1 file if specified, or date file was parsed if not.

`trial_id`: Trial identifier.

`trial_total_time`: Total time of trial.

`item_id`: Item identifier.

`item_condition`: Condition of item.

`region_label`: Label for region.

`region_number`: Region number (beginning with 0).

`region_text`: Text included in region, if specified.

`region_start`: Character location of beginning of region.

`region_end`: Character location of end of region.

3.9 trial_output

This section specifies the columns that should be included in the output file for trial measures. Each column has the same parameters as `region_output`. The columns are the same, but with columns beginning with `region_` excluded.

3.10 terminal_output

A number specifying the level of detail in terminal output. This option is useful for debugging, for example, finding out if an error is being caused by a specific input file, or by calculating a specific measure.

`0`: Errors only.

`1`: File-level information (which file is currently being parsed).

2: Item and trial-level parsing information.

3: Measure-level calculation information.

4: Trial-level calculation information.

5: All output information.

CHAPTER 4

Default Configuration

```
{
  "wide_format": true,
  "da1_fields": {
    "index": 0,
    "condition": 1,
    "number": 2,
    "time": -1,
    "fixation_start": 8
  },
  "region_fields": {
    "number": 0,
    "condition": 1,
    "boundaries_start": 3,
    "includes_y": false
  },
  "asc_parsing": {
    "fixation_min_cutoff": false,
    "max_saccade_dur": false,
    "blink_max_count": false,
    "blink_max_dur": false
  },
  "cutoffs": {
    "min": -1,
    "max": -1,
    "include_fixation": false,
    "include_saccades": false
  },
  "region_measures": {
    "skip": {},
    "first_pass_regressions_out": {},
    "first_pass_regressions_in": {},
    "first_fixation_duration": {},
    "single_fixation_duration": {},
    "first_pass": {},
```

(continues on next page)

(continued from previous page)

```
    "go_past": {},
    "total_time": {},
    "right_bounded_time": {},
    "reread_time": {},
    "second_pass": {},
    "spillover_time": {},
    "refixation_time": {},
    "landing_position": {},
    "launch_site": {},
    "first_pass_fixation_count": {},
    "go_back_time_region": {},
    "go_back_time_char": {}
  },
  "trial_measures": {
    "location_first_regression": {},
    "latency_first_regression": {},
    "fixation_count": {},
    "percent_regressions": {},
    "trial_total_time": {},
    "average_forward_saccade": {},
    "average_backward_saccade": {}
  },
  "region_output": {
    "experiment_name": {},
    "filename": { "exclude": true },
    "date": { "exclude": true },
    "trial_id": {},
    "trial_total_time": {},
    "item_id": {},
    "item_condition": {},
    "region_label": { "exclude": true },
    "region_number": {},
    "region_text": { "exclude": true },
    "region_start": { "exclude": true },
    "region_end": { "exclude": true }
  },
  "trial_output": {
    "experiment_name": {},
    "filename": { "exclude": true },
    "date": { "exclude": true },
    "trial_id": {},
    "trial_total_time": {},
    "item_id": {},
    "item_condition": {}
  },
  "terminal_output": 0
}
```

CHAPTER 5

Version History

5.1 1.0.0a13

- Bug fix: Only include trials ending in D0 in ASC parsing.

5.2 1.0.0a12

- Bug fix: Escape newlines in output CSV files
- Bug fix: Quote region start and end coordinates in output CSV files
- Bug fix: Don't classify saccades after data loss as regressions

5.3 1.0.0a11

- Bug fix: Only include fixations in ASC files after start of trial

5.4 1.0.0a10

- Parse all item and condition labels as strings, not numbers

5.5 1.0.0a9

- Added *sideeye* script
- Added ASC parsing (experimental feature)
- Improved configuration file parsing

5.6 1.0.0a8

- Added textfile region parser
- Added mypy type annotations
- Fixed go-back-time-region measure bug
- Added length to Region class

5.7 1.0.0a7

- Added wide output format and changed default output format to wide.

5.8 1.0.0a6

- Added go-back time by region and character.

5.9 1.0.0a5

- Ignore non-DA1 files when parsing an experiment.
- Update documentation.

5.10 1.0.0a4

5.11 1.0.0a3

5.12 1.0.0a2

5.13 1.0.0a1

- Initial release.

SideEye is a Python package for processing eye tracking data that interfaces with EyeTrack and Experiment Builder.

CHAPTER 6

Introduction

SideEye is a Python package for processing data from eye-tracking while reading experiments. The package contains parsers that accept fixation data in .DA1 or .ASC file format, and region definitions in several formats. SideEye calculates eye-tracking measures for parsed data, and can write output in .csv format.

6.1 Installation

SideEye requires Python >= 3.6.

```
pip install sideeye
```

Python Module Index

S

sideeye, 36
sideeye.calculate, 20
sideeye.config, 1
sideeye.data, 5
sideeye.measures.region, 13
sideeye.measures.trial, 12
sideeye.output, 21
sideeye.parser.asc, 19
sideeye.parser.da1, 20
sideeye.parser.experiment, 19
sideeye.parser.region, 20

Index

A

all (*sideeye.config.MeasuresConfig attribute*), 3
asc_parsing (*sideeye.config.Configuration attribute*), 2
ASCParsingConfig (*class in sideeye.config*), 1
assign_region() (*sideeye.data.Fixation method*), 11
average_backward_saccade() (*in module sideeye.measures.trial*), 12
average_forward_saccade() (*in module sideeye.measures.trial*), 12

B

blink_max_count (*sideeye.config.ASCParsingConfig attribute*), 1
blink_max_dur (*sideeye.config.ASCParsingConfig attribute*), 1
boundaries_start (*sideeye.config.RegionConfig attribute*), 4

C

calculate_all_measures() (*in module sideeye.calculate*), 21
calculate_measure() (*in module sideeye.calculate*), 20
char (*sideeye.data.Fixation attribute*), 10
condition (*sideeye.config.DA1Config attribute*), 3
condition (*sideeye.config.RegionConfig attribute*), 4
condition (*sideeye.data.Item attribute*), 8
Configuration (*class in sideeye.config*), 1
cutoff (*sideeye.config.OutputColumnConfig attribute*), 3
cutoffs (*sideeye.config.Configuration attribute*), 2
CutoffsConfig (*class in sideeye.config*), 2

D

da1_fields (*sideeye.config.Configuration attribute*), 1
DA1Config (*class in sideeye.config*), 2
date (*sideeye.data.Experiment attribute*), 5
duration() (*sideeye.data.Fixation method*), 11

E

end (*sideeye.data.Fixation attribute*), 11
end (*sideeye.data.Region attribute*), 9
excluded (*sideeye.data.Fixation attribute*), 11
Experiment (*class in sideeye.data*), 5

F

file() (*in module sideeye.parser.region*), 20
filename (*sideeye.data.Experiment attribute*), 5
find_region() (*sideeye.data.Item method*), 9
first_fixation_duration() (*in module sideeye.measures.region*), 15
first_pass() (*in module sideeye.measures.region*), 16
first_pass_fixation_count() (*in module sideeye.measures.region*), 15
first_pass_regressions_in() (*in module sideeye.measures.region*), 14
first_pass_regressions_out() (*in module sideeye.measures.region*), 14
Fixation (*class in sideeye.data*), 10
fixation_count() (*in module sideeye.measures.trial*), 12
fixation_count() (*sideeye.data.Trial method*), 8
fixation_min_cutoff (*sideeye.config.ASCParsingConfig attribute*), 1
fixation_start (*sideeye.config.DA1Config attribute*), 3
fixations (*sideeye.data.Trial attribute*), 7

G

generate_all_output() (*in module sideeye.output*), 22
generate_all_output_wide_format() (*in module sideeye.output*), 22
generate_region_output() (*in module sideeye.output*), 21

generate_trial_output() (*in module sideeye.output*), 21
get_region() (*sideeye.data.Item method*), 9
get_trial() (*sideeye.data.Experiment method*), 5
go_back_time_char() (*in module sideeye.measures.region*), 18
go_back_time_region() (*in module sideeye.measures.region*), 19
go_past() (*in module sideeye.measures.region*), 16

H

header (*sideeye.config.OutputColumnConfig attribute*), 3

I

include_fixation (*sideeye.config.CutoffsConfig attribute*), 2
include_saccades (*sideeye.config.CutoffsConfig attribute*), 2
includes_y (*sideeye.config.RegionConfig attribute*), 4
index (*sideeye.config.DA1Config attribute*), 2
index (*sideeye.data.Fixation attribute*), 11
index (*sideeye.data.Trial attribute*), 7
Item (*class in sideeye.data*), 8
item (*sideeye.data.Trial attribute*), 7

L

label (*sideeye.data.Region attribute*), 9
labels (*sideeye.data.Item attribute*), 8
landing_position() (*in module sideeye.measures.region*), 14
latency_first_regression() (*in module sideeye.measures.trial*), 12
launch_site() (*in module sideeye.measures.region*), 15
length (*sideeye.data.Region attribute*), 9
line (*sideeye.data.Fixation attribute*), 10
location_first_regression() (*in module sideeye.measures.trial*), 13

M

max (*sideeye.config.CutoffsConfig attribute*), 2
max_saccade_dur (*sideeye.config.ASCParsingConfig attribute*), 1
measure_output() (*in module sideeye.output*), 21
measures (*sideeye.config.Configuration attribute*), 2
MeasuresConfig (*class in sideeye.config*), 3
min (*sideeye.config.CutoffsConfig attribute*), 2

N

name (*sideeye.data.Experiment attribute*), 5
names (*sideeye.config.MeasuresConfig attribute*), 3
number (*sideeye.config.DA1Config attribute*), 3
number (*sideeye.config.RegionConfig attribute*), 4
number (*sideeye.data.Item attribute*), 8
number (*sideeye.data.Region attribute*), 9

O

output (*sideeye.config.Configuration attribute*), 2
OutputColumnConfig (*class in sideeye.config*), 3
OutputConfig (*class in sideeye.config*), 4

P

parse() (*in module sideeye.parser.asc*), 19
parse() (*in module sideeye.parser.da1*), 20
parse() (*in module sideeye.parser.experiment*), 19
percent_regressions() (*in module sideeye.measures.trial*), 13
Point (*class in sideeye.data*), 11

R

refixation_time() (*in module sideeye.measures.region*), 18
Region (*class in sideeye.data*), 9
region (*sideeye.config.MeasuresConfig attribute*), 3
region (*sideeye.data.Fixation attribute*), 11
region_count() (*sideeye.data.Item method*), 9
region_fields (*sideeye.config.Configuration attribute*), 2
region_measures (*sideeye.data.Trial attribute*), 8
RegionConfig (*class in sideeye.config*), 4
regions (*sideeye.data.Item attribute*), 8
reread_time() (*in module sideeye.measures.region*), 17
right_bounded_time() (*in module sideeye.measures.region*), 17

S

Saccade (*class in sideeye.data*), 10
saccades (*sideeye.data.Trial attribute*), 7
second_pass() (*in module sideeye.measures.region*), 17
sideeye (*module*), 36
sideeye.calculate (*module*), 20
sideeye.config (*module*), 1
sideeye.data (*module*), 5
sideeye.measures.region (*module*), 13
sideeye.measures.trial (*module*), 12
sideeye.output (*module*), 21
sideeye.parser.asc (*module*), 19
sideeye.parser.da1 (*module*), 20
sideeye.parser.experiment (*module*), 19
sideeye.parser.region (*module*), 20
single_fixation_duration() (*in module sideeye.measures.region*), 16
skip() (*in module sideeye.measures.region*), 14

spillover_time() (*in module sideeye.measures.region*), 18
start (*sideeye.data.Fixation attribute*), 10
start (*sideeye.data.Region attribute*), 9

T

terminal_output (*sideeye.config.Configuration attribute*), 2
text (*sideeye.data.Region attribute*), 9
text() (*in module sideeye.parser.region*), 20
textfile() (*in module sideeye.parser.region*), 20
time (*sideeye.config.DA1Config attribute*), 3
time (*sideeye.data.Trial attribute*), 7
total_time() (*in module sideeye.measures.region*), 16
Trial (*class in sideeye.data*), 7
trial (*sideeye.config.MeasuresConfig attribute*), 3
trial_indices (*sideeye.data.Experiment attribute*), 5
trial_measures (*sideeye.data.Trial attribute*), 8
trial_total_time() (*in module sideeye.measures.trial*), 13
trials (*sideeye.data.Experiment attribute*), 5

V

validate_key() (*in module sideeye.config*), 4

W

wide_format (*sideeye.config.Configuration attribute*), 1

X

x (*sideeye.data.Point attribute*), 11

Y

y (*sideeye.data.Point attribute*), 11