# API Python pour la visualisation interactive de données conversationnelles

# Démontration de son utilisation

> V1.0 22/06/2018
> Pour plus d'informations, voir la documention complète intégrée au code

## 1. Importation de l'API

La visualisation de données commence par l'exécution du fichier contenant l'API pour importer ses classes

In [1]:
```
%run Visualization_API.py
```

BokehJS 0.12.13 successfully loaded.

## 2. Lecture des données

On peut donc maintenant lire les données en créant un objet VisualisationData. Il faut pour cela renseigner en paramètre le répertoire dans lequel se situe les fichiers de données à lire et le format du corpus.

In [2]:
```
demo1 = VisualizationData("X11", corpus_format = "SW")
```

```
4876 data files have been read
```

Il existe 3 formats préenregistrés : "SW", "CID" et "minimalist" qui comprennent chacuns les détails de format aussi paramètrables

In [3]:
```
demo2 = VisualizationData("fp_CID", corpus_format = "CID",
                          metadata_columns = ['id_conv','id_speaker','data_type','corpus'],
                          data_delimiter = "\t",
                          data_head_lines = 0,
                          file_name = ['id_speaker'])
```

```
14 data files have been read
```

Il est aussi possible de filtrer les données lues en renseignant les conversations et les locuteurs souhaités

In [4]:
```
demo3 = VisualizationData("X11", corpus_format = "SW", conversations=["3456", "3034"])
```

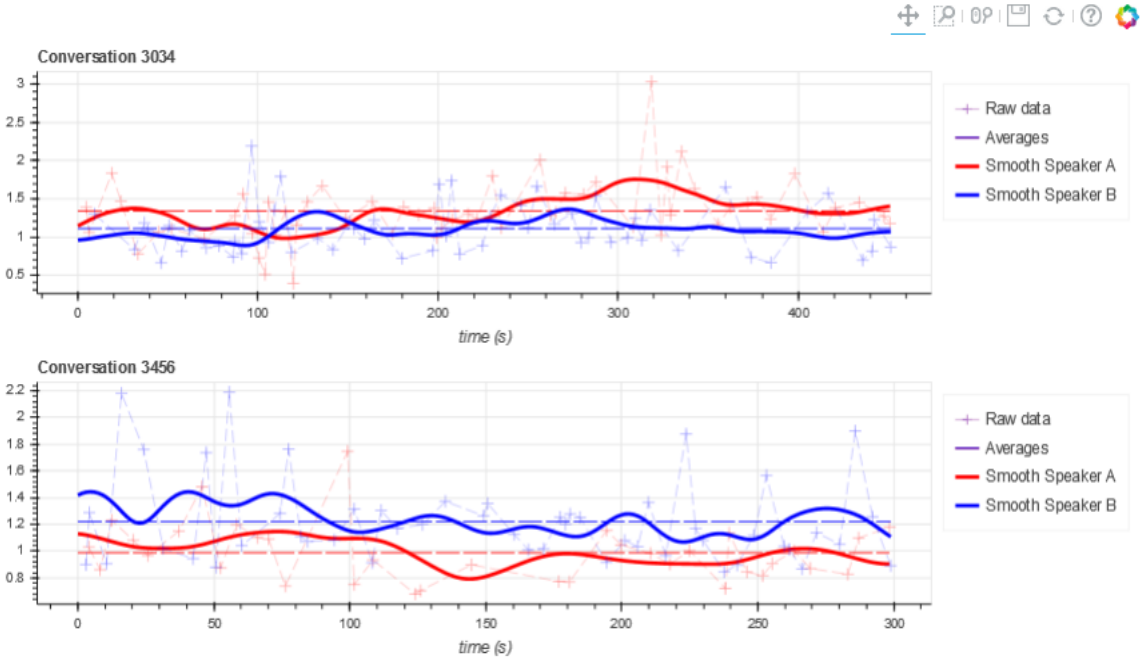
```
4 data files have been read
```

## 3. Visualisation des données

Les objets VisualizationData peuvent ensuite être visualisés par n'importe quelle fonction de la classe abstraite Display.
Pour chaques fonctions, il est possible de cliquer sur la légende pour faire disparaitre et apparaitre les données associées

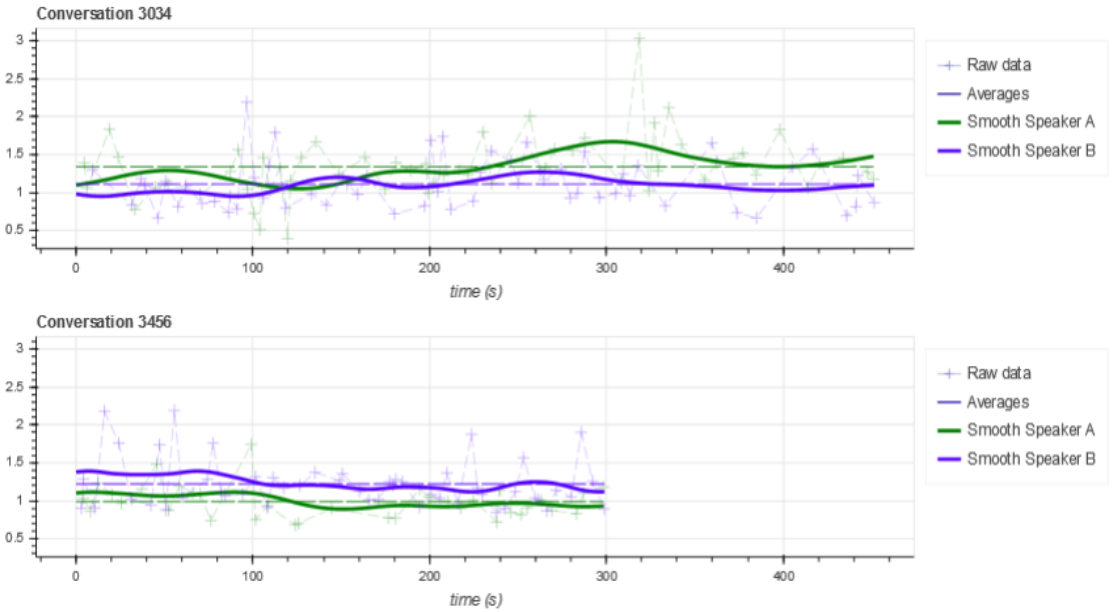
- **Display.conversation** affiche les données brutes, lissées et moyennes de chaque locuteurs, regroupés par conversations

In [5]:
```
Display.conversation(demo3)
```




Plusieurs paramètres optionnels sont possibles: *Linked* synchronise les axes de chaque conversations à True et *color_palette* définit les couleurs utilsées pour les courbes. Il est aussi possible de paramétrer la finesse du lissage en renseignant *smoothing_window* (pourcentage) et *points_number* (nombre de points affichés) ou dynamiquement avec des widgets si l'on passe le paramètre *interactive* à True

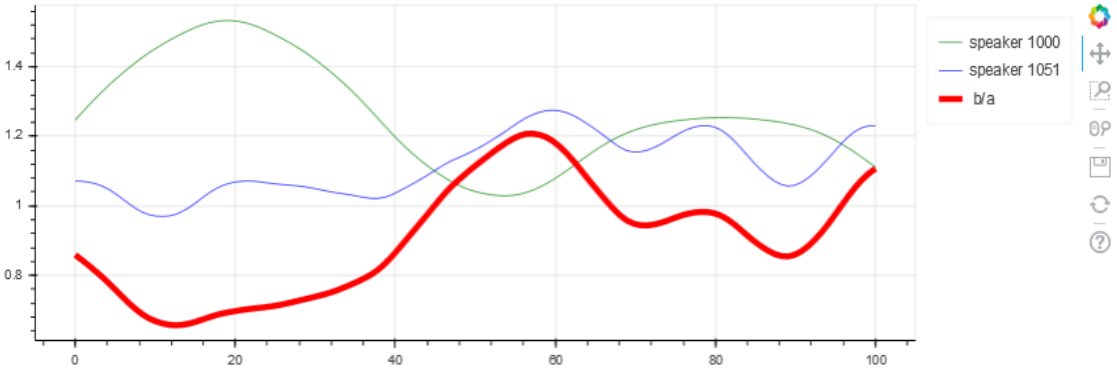
In [6]:
```
1 Display.conversation(demo3, linked = True, color_palette = ["green", "#5500ff"], smoothing_window = 90)
```




- **Display.aggregation** affiche le résultat d'une opération entre deux interlocuteurs, comme la différence ou le ratio. Il faut pour cela renseigner l'objet VisualizationData qui contient les deux donnéee (pas plus) et en second l'opération sous la forme lambda a,b: a+b. Les paramètres optionnel *color_palette* et *smoothing_window* sont aussi disponible.

In [7]:
```
1 demo4 = VisualizationData ("X11", corpus_format = "SW", conversations = ["2051"])
2 Display.aggregation(demo4, lambda a,b: b/a, smoothing_window=70)
```

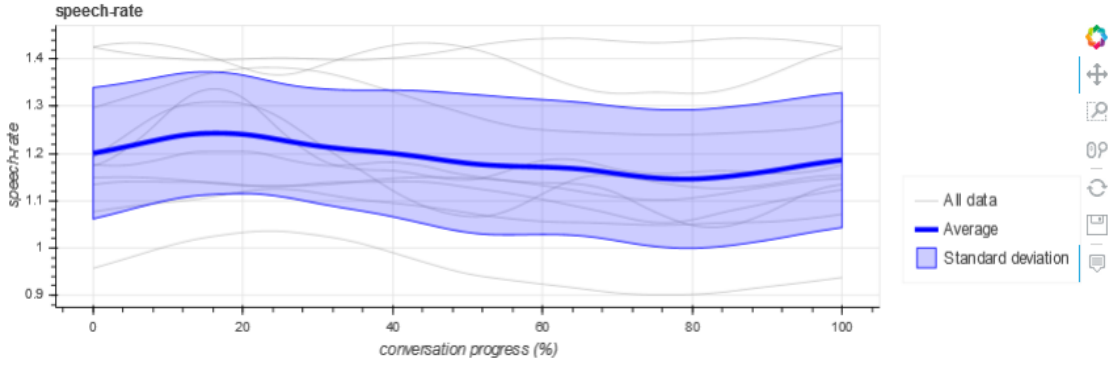
```
2 data files have been read
```




- **Display.average** affiche la courbe moyenne de toutes les données de l'objet VisualizationData passé en paramètre, la répartition de ces données via une bande écart-type, et toutes les données brutes. Pour chaques courbes tracées, survoler la ligne avec le curseur permet d'afficher ses méta-informations.

In [8]:
```
1 demo5 = VisualizationData("X11", corpus_format = "SW",
2                           conversations = ["2222", "2456", "2333", "2444", "2323"])
3 Display.average(demo5)
```

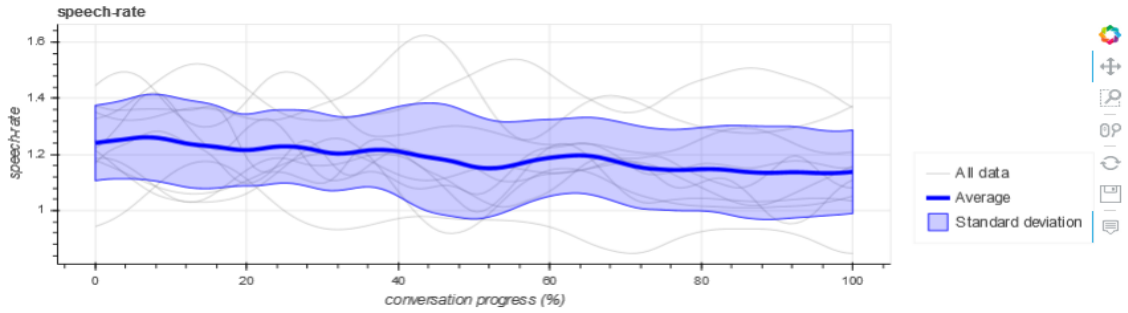
```
10 data files have been read
```




Si l'on renseigne le parmaètre interactive à True, des widgets pour chaques méta-information des données apparaissent (listes déroulantes ou double curseurs linéaire) pour permettre de filtrer dynamiquement les données affiché.

```
In [9]: 1 Display.average(demo5, interactive = True)
```

smoothness 30
id 0 – 4875
id_conv 2001 – 494(
id_caller
id_speaker 1000 – 170
id_topic
sex
age 20 – 71
geography
level_study 0 – 9
corpus
data_type



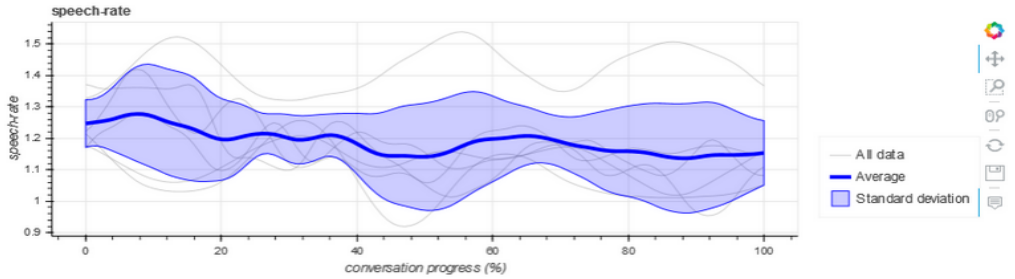

Curent filtered data name | Add to output | Clear output

Les widgets en dessous du graphique permettent d'exporter les données filtrées dans une variable de sortie output, pour ensuite pouvoir comparer différentes données filtrées sur un même graphique. La variable output n'est pas accéssible si l'on importe l'API comme dans cette démonstration. Pour pouvoir utiliser cette dernière fonction de visualisation il faut fonctionner avec l'API sous forme de Notebook Jupyter en ouvrant le fichier *notebook_API.ipynb* et en exécutant les fonctions en bas. Le résultat est comme ci-dessous.

```
In [4]: 1 demo6 = VisualizationData("X11", corpus_format = "SW",
        2                          conversations = ["2222", "2456", "2333", "2444", "2323"])
        3 Display.average(demo6, interactive = True)
```

```
10 data files have been read
```

smoothness 30
id 0 – 4875
id_conv 2001 – 494(
id_caller
id_speaker 1000 – 170
id_topic
sex MALE
age 20 – 71
geography
level_study 0 – 9
corpus
data_type




men | Add to output | Clear output

```
output contient 2 éléments
```

```
In [5]: 1 Display.comparison(output, color_palette = ["#ff0055", "blue"])
```