SurveyKN Tutorial

Written by Can Yalnız

Welcome

Welcome to SurveyKN! SurveyKN is an internal tool developed for Area HR to help HR members write reports for the periodic “Soddisfazione degli Associati”. Its main purpose is to automate the most monotonous and downright boring parts of writing the reports. SurveyKN takes the answers given to the survey(in .csv format), and generates report templates(in .docx) format.
The tool is written in Python, however you don’t need to know programming or Python to use it. All you need is a terminal and you’re good to go. Now, let me show you how you, as the user, can use SurveyKN to easily generate report templates just the way you want so that we can save precious time.

Installation & Setup

SurveyKN Folder

Let’s do a quick head count, before we can start with setting up the tool, you need to make sure you have all the necessary files and the correct folders. *Initially*, the directory structure of the SurveyKN folder should be like so:

```
SurveyKN:
    L Current Configuration:
        L config.yml
        L doctree.yml
    L Tutorial:
        L tutorial.md
        L tutorial.pdf
        L tutorial.html
    L generate.py
    L question.py
```

```
L setup.py
L visuals.py
L environment.yml
```

If you have all the files and folders intact, we can move on to setting up Python and the Python environment. If you have missing files to start with, please contact your colleague that pointed you to this tool in the first place and complain.

Important: This tutorial assumes that you are in the SurveyKN folder while running the scripts, if you get an error saying that Python cannot find the file or script, please navigate to the SurveyKN folder.

Python, the Terminal & the Environment

Before we can start running scripts like a true pro, you need to set up Python and the execution environment correctly. If at any point you feel overwhelmed, it would be best to consult someone that has previous experience with Python. It will be worth it once we get through the setup.

Anaconda & Python

You must first install Anaconda Individual Edition, which will *also* setup Python on our machine if it wasn't already installed.

Getting an Anaconda Prompt in Windows

Linux and MacOS users can simply use their terminals.
To proceed with the installation and the usage of SuveyKN, you need to know how to open an Anaconda prompt(we will simply refer to the prompt as a **terminal** for the rest of this tutorial). In Windows, if Anaconda was properly installed, you have two options:

Anaconda Navigator

1. Open up Anaconda Navigator
2. On your left, you will see a navigation menu, go to Environments
3. Pick the environment you want to run in, if this is your first time, you will most likely have only one[base (root)] environment available, if you're done with the setup and wish to start running the scripts, choose SurveyKN

4. After picking the right environment, click the small triangle next to the environment's name, the one that looks like a play button
5. Choose "Open Terminal"

Search Bar

1. Type "anaconda prompt" in your search bar
2. Click on the launcher for the prompt, it could look something like "Anaconda Prompt (anaconda3)"

The Python Environment

Once we have Anaconda nicely setup, you need to create yourself a virtual environment in which the scripts can feel at home. Doing this is fairly easy, just open up a terminal, navigate to your SurveyKN folder, and run the following commands, one after the other

```
conda install -c conda conda-env
conda env create --file environment.yml
```

Perfect! You have created the virtual environment we'll be using to run our scripts. Be careful though, even though we created the virtual environment, we need to step into it by activating it. To do this simply run

```
conda activate SurveyKN
```

This command will put us into the SurveyKN environment. **Always remember to enter the SurveyKN virtual environment before working with SurveyKN. The remainder of this tutorial will assume that you are in the SurveyKN virtual environment.**

Once we're in our environment you need to run one last command to finish setting it up

```
conda install -c plotly plotly-orca
```

Data Root Setup

We can finally start working directly with SurveyKN! In order to function properly, SurveyKN needs to have a data root. The data root is nothing but a directory that will hold

all of the data SurveyKN will generate. To setup the data root and create the necessary directories, run

```
python setup.py fresh
```

At this point, you will be asked to give a path for the data root. This can be any path you'd like that has the appropriate read/write privileges. In most cases, the best thing to do is to have the data root within SurveyKN by just entering a single dot(`.`). It is possible to move your data root later on.

Important: This tutorial assumes that you are in the SurveyKN folder while running the scripts, if you get an error saying that Python cannot find the file or script, please navigate to the SurveyKN folder.

Survey Results & Questions

Once SurveyKN and its requirements are set up, we can move on to the next bit. In this section we will go over how we should format the survey results file and how we need to process the questions that were in the survey.

Survey Results

The survey results should be formatted in a table where each column represents a question in the survey and each row represents a member. So that in the intersection of each row and column we can see the answer that particular member gave to that question. Additionally, the table must have a column named `AREA` where the area of each member is reported. It is also possible to have extra columns in the table for the email, the name, etc. of the members. This table should be saved as a `.csv` file. This file can be anywhere in your computer.

Questions

In order to function properly SurveyKN needs to know each question by heart. We do this by registering our questions into the *Question Store* where each will be given a unique ID code. First step is determining which questions we're interested in. At present we don't

have to know exactly how we would like to structure the reports, but we should find an answer to the question “Which of the survey questions should show up in the generated reports?”. When we have the answer, we can start working with `question.py`, the script that we will use to interact with the *Question Store*. We shall only go through the most essential options of this script here, if you wish to know the full capabilities and all the things you can do by using it, you can run

```
python question.py help
```

in your terminal. *Make sure you are in the SurveyKN directory and in the SurveyKN virtual environment.*

Registering a single question

To register a single new question into the store you just run

```
python question.py new
```

The script will then ask you to enter the question into the terminal. Here you must pay attention to the spelling of the question and any special characters it might have. What you put in here must match *exactly* with what is in the survey. If the question has a number in the `.csv` file, such as `3) question text?`, then you should only write `question text?`.

Tip: You can paste into most terminals using Ctrl+Shift+V if Ctrl+V isn’t working.

Registering multiple questions simultaneously

In order to register multiple new questions, you must first put these questions into a `.txt` file, where you have one question on each line. Then simply run

```
python question.py fromfile <path/to/file.txt>
```

where the path should point to the `.txt` file.

Deleting the last question you registered

The *Question Store* allows you to delete only one question at a time and this has to be the last question you registered. To do this, you must run

```
python question.py deletelast
```

The program will show you what the last question inserted into the store was along with its identification code and ask for your confirmation. To proceed simply type `yes`.

The Doctree

Now, let me introduce you to the *Doctree*. Our *Doctree* file is what we will be using to tell SurveyKN exactly how to structure the reports. The file `doctree.yml` will have the tree structure of the documents and which questions we wish to include in the final templates. You can find `doctree.yml` in the folder named `Current Configuration`. We shall use the example `doctree.yml` below to learn about the tree structure, the questions and the comments.

```
General Meetings:
    Questions:
        - Quanto hai partecipato alle RIUNIONI GENERALI?
        - Sei soddisfatto della gestione e della struttura delle RIUNIONI GENE
    Comments:
        - Quali sono i tuoi suggerimenti per migliorare le RIUNIONI GENERALI?

Individual Areas:
    Area Meetings:
        Questions:
            - Quanto hai partecipato alle RIUNIONI D'AREA?
            - Sei soddisfatto della gestione e della struttura delle RIUNIONI

    Placement Satisfaction:
        Questions:
            - Attualmente, sei soddisfatto dell'AREA in cui ti trovi?
            - Sei soddisfatto delle ATTIVITÀ e/o dei PROGETTI a cui hai preso
            - Sei soddisfatto del metodo di lavoro del tuo RESPONSABILE?
            - Il tuo RESPONSABILE ascolta le tue richieste e cerca di aiutarti
    Comments:
        - Come pensi che il tuo RESPONSABILE o la tua AREA di appartenenza pos
```

Tree Structure

The best way to organize our thoughts in the report is to use a tree structure, just like in this tutorial you're reading where everything is neatly packed into sections, subsections, subsubsections etc. We can customize the tree structure of our reports by editing the `doctree.yml` file, which is formatted using YAML syntax.

In our example, our *Doctree* has two *sections*: `General Meetings`, `Individual Areas` and `Individual Areas` has two *subsections*: `Area Meetings`, `Placement Satisfaction`. *Although we may not need it very frequently, keep in mind that we can go even deeper into the tree with subsubsections.*

Leaves

Just like every abstract tree out there, notice that our *Doctree* has *leaves*. They are those objects that don't have *children* of their own, they mark the end of branches. In the *Doctree*, it's useful to think that the *leaves* are the Questions and the Comments together with their list of items.

Questions

Each `Questions` "object" in our *Doctree* holds the list of questions we wish to talk about at that specific location in our tree. For example, in the above *Doctree*, we are saying that we want a *section* called `General Meetings` in our document, and we want that *section* to talk about the questions:

```
- Quanto hai partecipato alle RIUNIONI GENERALI?
- Sei soddisfatto della gestione e della struttura delle RIUNIONI GENERALI?
```

Also, we are saying that we want a *section* called `Individual Areas`, we want this *section* to have a *subsection* called `Area Meetings` within this *subsection*, we want to talk about the following questions:

```
- Quanto hai partecipato alle RIUNIONI D'AREA?
- Sei soddisfatto della gestione e della struttura delle RIUNIONI D'AREA?
```

etc.

Note: We need to be careful about how we write the questions in our Doctree, the questions must exactly match what we've registered into the Question Store.

Note: Every branch of our tree must end with a `Questions` object.

Comments

The second type of *leaves* that we have are `Comments` "objects". These contain a list of "questions" in the survey, that asked our members for their comments. For example, we want to include our members' comments in the `General Meetings` section, so we put a `Comments` object with the following list

```
- Quali sono i tuoi suggerimenti per migliorare le RIUNIONI GENERALI?
```

Keep in mind that you can have more than one item in these lists. Also, notice that the `Comments` object within the section `Individual Areas` belongs to the *section* itself and not to any of the *subsections*. Both *sections* and *subsections* can have their own `Comments` objects.

*Note: Having a `Comments` object is **not** mandatory for sections, subsections or subsubsections.*

Configuration & Customization

No that we have the structure of our reports in order, we can move on to configuring and customizing the output. We do this by editing the `config.yml` file.

Areas

Using SurveyKN we can generate individual reports for each area. To let SurveyKN know which areas there are, we use the `Areas` field in the config file.

Note: There must be at least one area for the program to work.

Document Configuration

Our config file also lets us customize various aspects of the reports. You can change the title, date and disclaimer of the reports and customize all of the fonts, sizes and properties of different parts of the report.

Conclusion Tree

In the config file you will see that you have the option of including a `Conclusion Tree` in the reports. If you choose this option by setting it to `True`, the generated reports will have a summarized version of the *Doctree* where you can write the conclusion for each section individually.

Custom Charts

One of the most important pieces of our reports are the data visualizations. By default, every question in the *Doctree* gets its own *pie chart*. This pie chart visually represents the answers reported by the members of the area to that specific question. However, SurveyKN also makes it possible to go above and beyond just the default pie charts.

Global Pie Chart

The `Global Pie Chart` is drawn from the answers of all the members that completed the current survey. It is useful for comparing the area of the report to the general standing of our association.

Past Survey Pie Chart

The `Past Survey Pie Chart` is drawn from the answers of the members of the same area, but this pie chart plots the answers reported in a previous survey. You should put the *survey id* of the survey whose results you wish to plot in the `Parameters` field of the `Past Survey Pie` in `config.yml`. However, if you wish to draw this chart for a past survey, that survey must have been processed by SurveyKN in the past. The *survey id* of any survey is in the format `date-month` (i.e. 2020-01).

Past Survey Bar Chart

The `Past Survey Bar Chart` takes it a step further by providing a way for us to compare multiple past survey results with the current one. The surveys you put in the `Parameters` field of the `Past Survey Bar` in `config.yml` will be plotted in a bar chart that neatly shows how the answers progressed over time. It is recommended to put the *survey ids* in chronological order, oldest to newest.

Choosing and Ordering Custom Charts

All of the above custom charts are at your disposal and it's up to you to decide which of these should appear in the reports and in what order. To configure this, you just need to edit the list of `Custom Charts Draw` in `config.yml`. Only those custom charts whose names appear in this list will be drawn for the questions. The list should be in the order you wish to see the charts in the report.

Note: If you don't want any custom charts, then instead of leaving this field empty in the config file, you must have it as `Custom Charts Draw: Null`.

Exceptions

You may want to exclude some of the questions from the custom charts. If you want a question to have **only** the default pie chart, you need to put its *question id* into the list of `Custom Charts Exceptions` in `config.yml`.

Note: If you don't wish to exclude any of the questions from the custom charts, then instead of leaving this field empty in the config file, you must have it as `Custom Charts Exceptions: Null`.

Note: If you're not sure about the question id of the questions you wish to exclude, just run

```
python question.py listall
```

this will list all of the questions in store along with their three letter question ids.

Chart Styles, Structure & Survey Answers

Available Options

We can customize almost everything about our charts. One of the most important aspects of any data visualization graphic is the color scheme. We can individually set the colors of all the answers to our multiple choice questions under the `Available Choices` field in `config.yml`. Each entry in this field should look like this

```
Molto:
    Hex: '#171C42'
```

```
    Label: Molto
```

Where `Molto` is one of the available choices to at least one of our questions, `Hex` is the hex value of the color we want to assign to this option and `Label` is what we want to show up in the charts to represent this answer. As you can see, for `Molto` we can assign it directly to its label, however, for an option that's quite lengthy like this

```
Dovrò lasciarvi al termine della sessione esami:
    Hex: '#19D3F3'
    Label: Arrivaderci
```

It could be more visually appealing on the graphs to write `Arrivaderci` instead of `Dovrò lasciarvi al termine della sessione esami`, since it's so lengthy.

Important: The options listed under the `Available Options` field should have the same spelling as all of the instances of that answer that appear in the survey. If you have both `Più no che sì` and `Piu no che sì` in the survey results, you shouldn't add two entries for the different spellings to the `Available Options` field, instead you should keep the spelling consistent in the survey results.

Pie Chart Style

Lines

You can even customize the lines that make up the edges of our pie charts. You can change the color and width of these lines by editing the `Line Color` and the `Line Width` fields of the `Pie Chart Style` field in `config.yml` respectively.

Text Information on the Charts

When it comes to how you would like to support what the pie chart has visualized using colors and proportions, you have three ingredients you can use:

1. Value: The count of the answers (i.e. if three people picked `Molto`, on the slice that represents `Molto` you will see a `3`)
2. Percent: The percentage of the answers (i.e. if three people picked `Molto` and there are a total of 6 people, on the slice that represents `Molto` you will see `50%`)
3. Label: The label of the answers (the mapping between the labels and the colors in the charts are shown in the legend by default, but if you also want to see the labels of the

answers on each slice of the pie, then you should enable this)

These three can be enabled or disabled independently of each other. All of those you have enabled will show up on the slices. You can enable\disable these in `Pie Chart Style` in `config.yml` by setting the respective fields to True\False

Generating Reports

At last we're here. Now that we've setup our Python environment, our data root, registered our questions, created our *Doctree* and updated the configuration, we are ready to generate the reports. Since we've done most of the work, generating the reports is going to be quite straightforward. Just run

```
python generate.py <path/to/survey.csv>
```

Where `<path/to/survey.csv>` is the path to the `.csv` file of the survey you wish to generate the reports of. The program will ask for your confirmation and then ask you for the year of the survey, then the month of the survey in order to assign its survey ID as `year-month`. Once you provide this information, the program will go through your *Doctree* to see if there are any questions missing from the *Question Store*. If there are questions in your *Doctree* that you haven't registered, the program will ask you if you'd like to register these questions before proceeding. Once this is done, the program will generate the reports. You can find the generated reports under SurveyKN/SurveyKN-dataroot/Templates/<survey_id> where <survey_id> is `year-month` of the survey(i.e 2020-01). In this folder you'll find the generated reports as well as copies of the config and doctree files that were used to generate these reports for future reference.

Data Root Directory Structure

Now, let's quickly go over the directory structure of SurveyKN's data root.

```
SurveyKN-dataroot:
    L AppData:
        L question-store.yml
    L Surveys:
```

```
        L <survey_id-original.csv>
        L <survey_id.csv>
    L Templates:
        L <survey_id>
            L config.yml
            L doctree.yml
            L <area.docx>
    L Visuals:
        L <survey_id>:
            L <area>:
                L <question_id-chart_id.png>
```

AppData

Contains the *Question Store*.

Surveys

Contains copies of the survey `.csv` files used for generating the reports. The `.csv` file with `original` in its name is an identical copy of the original survey file, the `.csv` file **without** `original` in its name is modified to facilitate processing.

Templates

Contains one directory per survey, each one of these directories hold the generated report templates as well as their config and doctree files.

Visuals

Contains one directory per survey, each of these directories contain one directory per area, each of these directories contain the generated data visualization images in `.png` format.

Handing SurveyKN Down

SurveyKN not only makes it easier to write our periodic reports, but it also keeps a consistent record of all of the surveys that we've carried out as Area HR. Therefore it is important to make sure no data is lost while SurveyKN is handed down from one area head to the next.

The first thing to do is to have the new area head follow the steps under the `Installation & Setup` section above. They must do everything except for the `Data Root Setup`.

Once the `Installation & Setup` is done at the destination, there are only two folders that need to change hands: `SurveyKN` and `SurveyKN-dataroot`. If during the hand-down, your `SurveyKN-dataroot` folder is somewhere inside your `SurveyKN` folder, it is sufficient to hand over only the `SurveyKN` folder and its contents. However, if your data root is located somewhere else on your system, you must make sure both folders are transferred correctly. Once the folders are received on the other end, the new area head must place `SurveyKN-dataroot` somewhere they see fit, and then run

```
python setup.py redirect
```

to tell SurveyKN where the new data root on their system is.