TP1_Corrige

February 13, 2025

##

Polytechnique Montréal Département Génie Informatique et Génie Logiciel INF8008 – Prétraitement de données . TP1 - Analyse descriptive des données Hiver 2025 . 20 janvier 2025

[6]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

0.1 Introduction

Le TD1 porte sur l'analyse descriptive de données. Nous survolons l'utilisation de fonctions de base de Pandas et de l'analyse de données numériques et de leur visualisation.

Les données du fichier ntsb-accidents.csv proviennent d'une base de données sur les accidents d'aviation provenant du National Transportation Safety Board . Nous nous en inspirerons pour le travail des TPs de cette session.

Les champs du fichier de données ntsb-accidents.csv sont les suivants :

- **event_id** : Identifiant unique de l'événement (souvent un code de référence pour suivre chaque cas).
- **ntsb_make** : Fabricant de l'appareil impliqué dans l'événement (exemple : BELL, ROBINSON).
- **ntsb_model** : Modèle spécifique de l'appareil (exemple : R22 BETA, R44).
- **ntsb_number** : Code d'identification de l'incident assigné par la NTSB.
- **year** : Année où l'événement a eu lieu.
- **date** : Date et heure exactes de l'événement (année/mois/jour heure:minute:seconde).
- **city** : Ville où l'événement s'est produit.
- **state** : État ou province où l'événement s'est produit.
- **country** : Pays où l'événement a eu lieu.
- **total_fatalities** : Nombre total de décès associés à l'incident.
- **latimes_make** : Fabricant simplifié utilisé dans les rapports (exemple : BELL, ROBINSON).
- **latimes_model** : Modèle simplifié utilisé dans les rapports (exemple : R22, 369).
- **latimes_make_and_model** : Combinaison du fabricant et du modèle pour décrire l'appareil de manière concise (exemple : BELL 407, ROBINSON R44).

Les librairies python qui seront à utiliser pour ce TP sont les suivantes: - pandas - matplotlib

[7]:
```
import pandas as pd
```

0.2 1. Analyse des données numériques

0.2.1 Q1

À l'aide de Pandas, chargez les données dans une variable nommée 'df'. Quelle est la dimension de 'df'? Combien y a-t-il de lignes et de colonnes? (2 points)

[8]:
```
#TODO: Chargement et affichage de df
df = pd.read_csv('/content/drive/MyDrive/INF8008/Hiver 2025/ntsb-accidents.csv')
df
```

[8]:

	event_id	ntsb_make	ntsb_model	ntsb_number	\
0	20061222X01838	BELL	407	NYC07FA048	
1	20060817X01187	ROBINSON	R22 BETA	LAX06LA257	
2	20060111X00044	ROBINSON	R44	MIA06FA039	
3	20060419X00461	ROBINSON	R44 II	DFW06FA102	
4	20060208X00181	ROBINSON	R44	SEA06LA052	
..	…	…	…	…	
158	20160711X32921	BELL HELICOPTER TEXTRON CANADA	407	ERA16FA248	
159	20160804X45514	SCHWEIZER	269C 1	CEN16FA304	
160	20160404X74644	BELL	206	ERA16FA144	
161	20160507X31120	AIRBUS	AS350	ANC16FA023	
162	20160612X85856	ROBINSON HELICOPTER COMPANY	R44 II	CEN16FA215	

	year	date	city	state	country	\
0	2006	12/14/2006 00:00:00	DAGSBORO	DE	USA	
1	2006	08/10/2006 00:00:00	TUCSON	AZ	USA	
2	2006	01/01/2006 00:00:00	GRAND RIDGE	FL	USA	
3	2006	04/13/2006 00:00:00	FREDERICKSBURG	TX	USA	
4	2006	02/06/2006 00:00:00	HELENA	MT	USA	
..	…	…	…	…	…	
158	2016	07/11/2016 00:00:00	HICKORY	KY	USA	
159	2016	08/03/2016 00:00:00	JEANERETTE	LA	USA	
160	2016	04/04/2016 00:00:00	PIGEON FORGE	TN	USA	
161	2016	05/06/2016 00:00:00	SKAGWAY	AK	USA	
162	2016	06/12/2016 00:00:00	JONESBORO	AR	USA	

	total_fatalities	latimes_make	latimes_model	latimes_make_and_model
0	2	BELL	407	BELL 407
1	1	ROBINSON	R22	ROBINSON R22
2	3	ROBINSON	R44	ROBINSON R44
3	2	ROBINSON	R44	ROBINSON R44
4	1	ROBINSON	R44	ROBINSON R44
..	…	…	…	…
158	1	BELL	407	BELL 407
159	1	SCHWEIZER	269	SCHWEIZER 269

```
160                5         BELL           206              BELL 206
161                1       AIRBUS           350            AIRBUS 350
162                1     ROBINSON           R44          ROBINSON R44

[163 rows x 13 columns]
```

[9]:
```
#TODO: Complétez ce code afin d'afficher les bonnes informations dans les␣
 ↪"print".

print("df est de dimension", df.shape)
print("nombre de lignes", df.shape[0])
print("nombre de colonnes", df.shape[1])
```

```
df est de dimension (163, 13)
nombre de lignes 163
nombre de colonnes 13
```

0.2.2 Q2

Quelle est l'intervalle d'années et l'ensemble des fabricants et modèles d'hélicoptères compris dans le jeu de données?(1 point)

[10]:
```
#TODO:
intervalle = df.year.unique()
print("L'intervalle d'années est: ", intervalle)
intervalle_range = (intervalle.min(), intervalle.max())
print("L'intervalle d'années est: ", intervalle_range)
fabricants_modeles = df.latimes_make_and_model.unique()
print("L'ensemble des fabricants et modèles est: ", fabricants_modeles)
```

```
L'intervalle d'années est:  [2006 2007 2008 2009 2010 2012 2013 2014 2015 2016]
L'intervalle d'années est:  (2006, 2016)
L'ensemble des fabricants et modèles est:  ['BELL 407' 'ROBINSON R22' 'ROBINSON
R44' 'AIRBUS 350' 'HUGHES 369'
 'SCHWEIZER 269' 'AIRBUS 135' 'bell 206' 'BELL 206'
 'MCDONNELL DOUGLAS 369' 'SIKORSKY 76' 'AGUSTA 109' 'AIRBUS 130']
```

0.2.3 Q3

Combien de décès y a-t-il en moyenne? (1 point)

[11]:
```
#TODO:
deces = df.total_fatalities.mean()
print("Le nombre moyen de décès est: ", deces)
```

```
Le nombre moyen de décès est:  2.061349693251534
```

0.2.4 Q4

Combien d'**accidents** y a-t-il eu pour chaque combinaison de fabricant et modèle d'hélicoptère? Créez un tableau de fréquence du nombre d'accidents par combinaison de fabricant et modèle. (3 points)

[12]:
```
#TODO:

freq_table = df.groupby('latimes_make_and_model')['event_id'].nunique().
 ↪reset_index()
#differente maniere:
# freq_table = df.latimes_make_and_model.value_counts()
print("Tableau de fréquence du nombre d'accidents par combinaison de fabricant␣
 ↪et modèle: ")
print(freq_table)
```

```
Tableau de fréquence du nombre d'accidents par combinaison de fabricant et
modèle:
   latimes_make_and_model  event_id
0              AGUSTA 109         2
1              AIRBUS 130         1
2              AIRBUS 135         4
3              AIRBUS 350        28
4                BELL 206        28
5                BELL 407        12
6              HUGHES 369        13
7   MCDONNELL DOUGLAS 369         6
8            ROBINSON R22        20
9            ROBINSON R44        38
10         SCHWEIZER 269         5
11           SIKORSKY 76         2
12              bell 206         2
```

0.2.5 Q5

Déterminez la moyenne, la médiane et l'écart-type du nombre de décès. Dans un premier temps, fournissez les statistiques avec AIRBUS 350. Puis, dans un second temps, fournissez les statistiques avec ROBINSON R44.(3 points)

[13]:
```
# Calcul des statistiques pour AIRBUS 350
airbus_stats = [func(df[df['latimes_make_and_model'] == 'AIRBUS␣
 ↪350']['total_fatalities']) for func in [pd.Series.mean, pd.Series.median, pd.
 ↪Series.std]]
print("AIRBUS 350 - Moyenne: {}, Médiane: {}, Écart-type: {}".
 ↪format(*airbus_stats))

# Calcul des statistiques pour ROBINSON R44
```

```
robinson_stats = [func(df[df['latimes_make_and_model'] == 'ROBINSON␣
↪R44']['total_fatalities']) for func in [pd.Series.mean, pd.Series.median, pd.
↪Series.std]]
print("ROBINSON R44 - Moyenne: {}, Médiane: {}, Écart-type: {}".
↪format(*robinson_stats))
```

```
AIRBUS 350 - Moyenne: 2.793103448275862, Médiane: 3.0, Écart-type:
1.6339983901204604
ROBINSON R44 - Moyenne: 1.868421052631579, Médiane: 2.0, Écart-type:
0.9055699304881247
```

[14]:
```
#TODO:
airbus = df[df['latimes_make_and_model'] == 'AIRBUS 350']

avg_a = airbus['total_fatalities'].mean()
std_a = airbus['total_fatalities'].std()
med_a = airbus['total_fatalities'].median()

print("La moyenne du nombre de décès pour AIRBUS 350 est: ", avg_a)
print("L'écart-type du nombre de décès pour AIRBUS 350 est: ", std_a)
print("La médiane du nombre de décès pour AIRBUS 350 est: ", med_a)


robinson = df[df['latimes_make_and_model'] == 'ROBINSON R44']

avg_r = robinson['total_fatalities'].mean()
std_r = robinson['total_fatalities'].std()
med_r = robinson['total_fatalities'].median()

print()
print("La moyenne du nombre de décès pour ROBINSON R44 est: ", avg_r)
print("L'écart-type du nombre de décès pour ROBINSON R44 est: ", std_r)
print("La médiane du nombre de décès pour ROBINSON R44 est: ", med_r)
```

```
La moyenne du nombre de décès pour AIRBUS 350 est:  2.793103448275862
L'écart-type du nombre de décès pour AIRBUS 350 est:  1.6339983901204604
La médiane du nombre de décès pour AIRBUS 350 est:  3.0

La moyenne du nombre de décès pour ROBINSON R44 est:  1.868421052631579
L'écart-type du nombre de décès pour ROBINSON R44 est:  0.9055699304881247
La médiane du nombre de décès pour ROBINSON R44 est:  2.0
```

0.2.6 Q6

Combien d'accidents ont enregistré plus d'un décès? Combien de décès se sont produits en Californie en 2015? (1 point)

[15]:
```
#TODO:

ss = df[df['total_fatalities'] >1]['event_id'].nunique()
print("Le nombre d'accidents avec plus d'un décès est de : ", ss)


avg_time = df[(df['year'] ==  2015) & (df['state']== 'CA')]
print("Le nombre de décès en 2015 à Californie est de :  ", avg_time.
 ↪total_fatalities.mean())
```

```
Le nombre d'accidents avec plus d'un décès est de :  87
Le nombre de décès en 2015 à Californie est de :   3.0
```

0.3 2. Visualisation de données

[16]:
```
import matplotlib.pyplot as plt
```

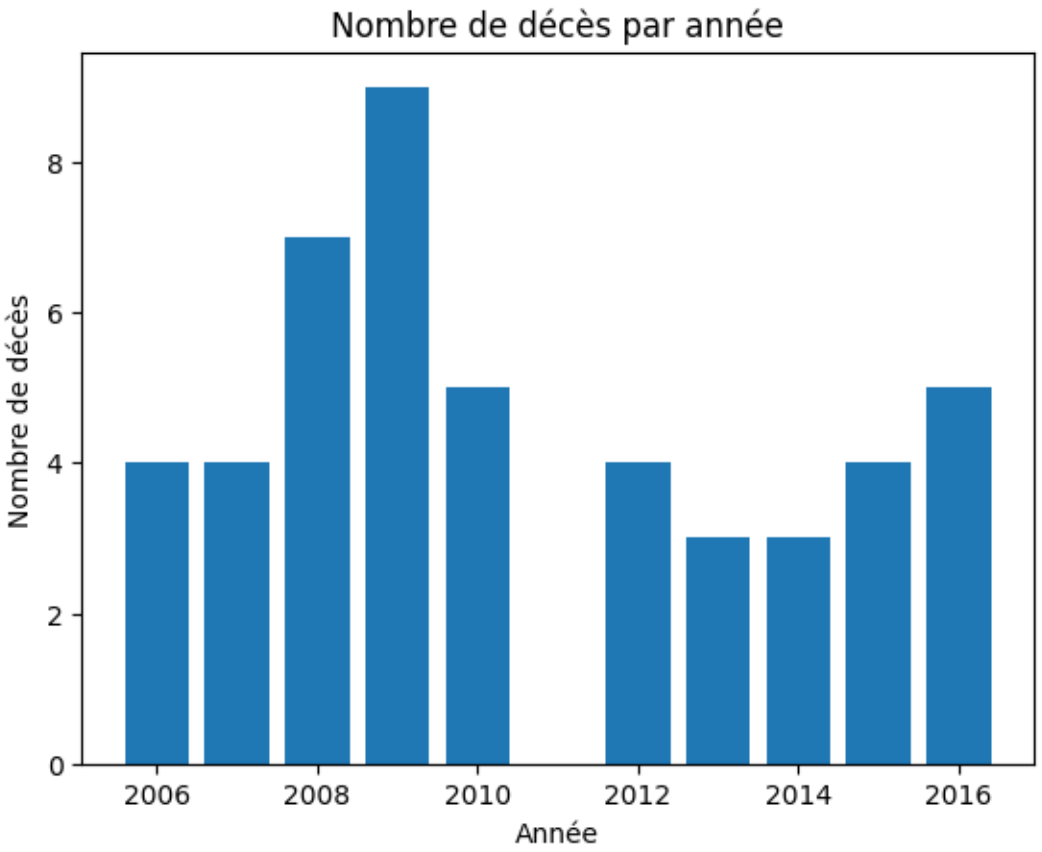
0.3.1 Q7

Affichez le nombre de décès par année. Nommez vos axes et donnez un titre à votre graphique. Qu'observez-vous? Quelle année marque le plus de décès? Quelle année marque le moins?(3 points)

[17]:
```
#TODO:

plt.bar(df['year'], df['total_fatalities'])
plt.xlabel("Année")
plt.ylabel("Nombre de décès")
plt.title('Nombre de décès par année')
```

[17]:
```
Text(0.5, 1.0, 'Nombre de décès par année')
```



2009: le plus de décès

2011: le moins de décès

0.3.2 Q8

En utilisant la fonction subplot de matplotlib:

1) Affichez le nombre d'accidents par année dans un diagramme à barres.

2) Affichez le nombre de décès par année dans un diagramme à barres.

Nommez vos axes et donnez un titre à votre graphique. En comparant ceux-ci, observez-vous une relation entre le nombre de décès et d'accidents? Pouvez-vous justifier la cause du nombre de décès minimal de l'année évoquée dans la question 7? (5 points)

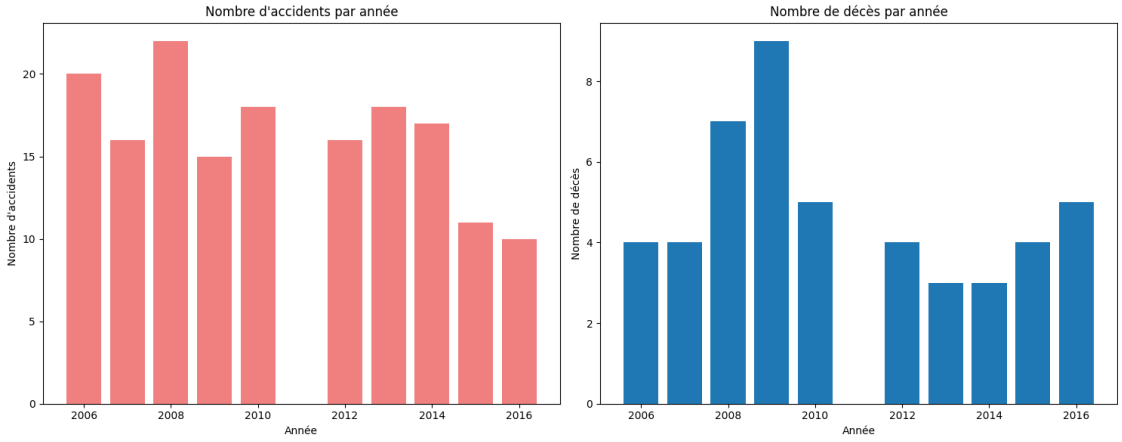
[18]:
```
#TODO: Filtrez votre dataframe

#TODO: Partie 1
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
event_counts_per_year = df['year'].value_counts().sort_index()
```

```
plt.bar(event_counts_per_year.index, event_counts_per_year.values,␣
 ↪color='lightcoral')
plt.xlabel("Année")
plt.ylabel("Nombre d'accidents")
plt.title("Nombre d'accidents par année")

#TODO: Partie 2
plt.subplot(1, 2, 2)
plt.bar(df['year'], df['total_fatalities'])
plt.xlabel("Année")
plt.ylabel("Nombre de décès")
plt.title("Nombre de décès par année")

plt.tight_layout()
plt.show()
```



L'année 2009 ayant le plus de décès n'a pas le plus d'accidents. Le nombre de décès n'est donc pas relié au nombre d'accidents.

L'année 2011 n'a pas de données et de ce fait a le moins de décès enregistrés.

0.4 3. LIVRABLES

Vous devez remettre sur Moodle un fichier compressé .zip contenant :

1) Le code : Un Jupyter notebook en Python qui contient le code tel implanté avec les librairies minimales demandées pour ce TP (Python, Pandas, Matplotlib). Le code doit être exécutable sans erreur et accompagné des commentaires appropriés dans le notebook de manière. Tous vos résultats doivent être reproductibles avec le code dans le notebook. *Attention, en aucun cas votre code ne doit avoir été copié de d'ailleurs.*

2) Un fichier pdf représentant votre notebook complètement exécuté sous format pdf (obtenu via latex ou imprimé en pdf avec le navigateur). Assurez-vous que le PDF est entièrement lisible. Tutoriel youtube

ATTENTION: assurez-vous que votre fichier compressé .zip ne dépasse pas la taille limite acceptée sur Moodle.

ÉVALUATION Votre TP sera évalué sur les points suivants :

Critères : 1. Implantation correcte et efficace 2. Qualité du code (noms significatifs, structure, performance, gestion d'exception, etc.) (1 point) 3. Réponses correctes/sensées aux questions de réflexion ou d'analyse

CODE D'HONNEUR - **Règle 1**: Le plagiat de code est bien évidemment interdit. Toute utilisation de code doit être référencée adéquatement. Vous **ne pouvez pas** soumettre un code, écrit par quelqu'un d'autre. Dans le cas contraire, cela sera considéré comme du plagiat. - **Règle 2**: Vous êtes libres de discuter avec d'autres équipes. Cependant, vous ne pouvez en aucun cas incorporer leur code dans votre TP. - **Règle 3**: Vous ne pouvez pas partager votre code publiquement (par exemple, dans un dépôt GitHub public) tant que le cours n'est pas fini.

0.4.1 Conversion en PDF sur Google Colab

[19]:
```
%%capture
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
```

Assurez vous d'avoir téléchargé le TP complété en notebook sur votre ordinateur, puis importé ce fichier dans le répertoire "content" avant de rouler la ligne suivante.

[20]:
```
!jupyter nbconvert --to pdf /content/TP1.ipynb
```

```
[NbConvertApp] WARNING | pattern '/content/TP1.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
```

```
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--coalesce-streams
    Coalesce consecutive stdout and stderr outputs into one stream (within each
cell).
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--CoalesceStreamsPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
```

```
    Whether to allow downloading chromium if no suitable version is found on the
system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful
for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook',
'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
```

```
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each notebook.
To recover
                                    previous default behaviour (outputting to the
current
                                    working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
```

```
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides',
'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```