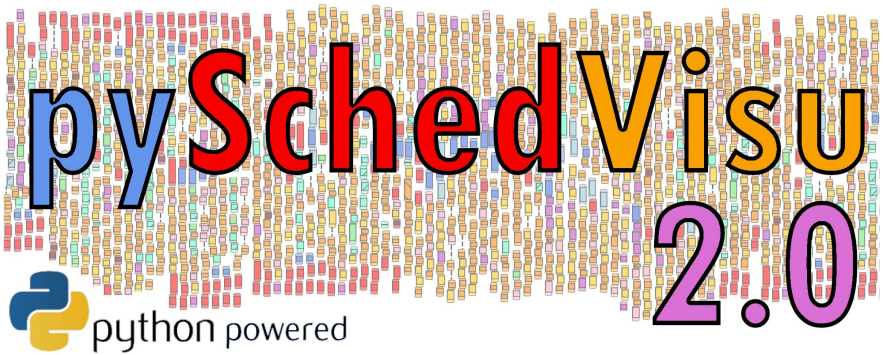


Service de médecine nucléaire
et imagerie moléculaire

pySchedVisu

Documentation

Balazs Laurenczy
CHUV
3 janvier 2020

canton de vaud

Table des matières (avec des liens cliquables!)

- **Qu'est-ce que** pySchedVisu ?
- **Que fait** pySchedVisu ?
- **Que contient un rapport** de pySchedVisu ?
- **Légende pour le planning** créé par pySchedVisu
- **Comment fonctionne** pySchedVisu ?
- **Où est-ce que** pySchedVisu **s'exécute** ?
- **Où se trouve** pySchedVisu ?
- **Comment installer** pySchedVisu sur un **nouveau poste** ?
- **Comment paramétrer** pySchedVisu ?
- **Comment créer une nouvelle version** de pySchedVisu
- **Comment faire pour exécuter** pySchedVisu de manière **récurrente**?

Qu’est-ce que pySchedVisu ?

- pySchedVisu est un logiciel de génération de rapport de planning des scanners PET et SPECT du service de médecine nucléaire du CHUV.
- Les rapports sont générés chaque semaine et contiennent l’utilisation des scanners, la durée et le nombre d’examens, ainsi que d’autres informations.
- Ces rapports sont utiles pour voir l’utilisation *effective* des scanners (le temps d’imagerie réelle), comparé à leur utilisation *planifiée*. Certaines statistiques d’utilisation fournis par le rapport de pySchedVisu (durée moyenne d’examen, nombre de trous, etc.) sont également utiles.

Que fait pySchedVisu ?

pySchedVisu fonctionne en 3 étapes:

1. pySchedVisu récupère les données nécessaires à la création des rapports dans le PACS (base de donnée d'images) du CHUV.
2. pySchedVisu traite ensuite ces données pour en extraire des informations pertinentes (heure de début et fin de chaque examen, type d'examen, etc.).
3. pySchedVisu condense et affiche finalement ces données sous la forme d'un rapport, stocké et envoyé par e-mail.

Que contient un rapport de pySchedVisu ?

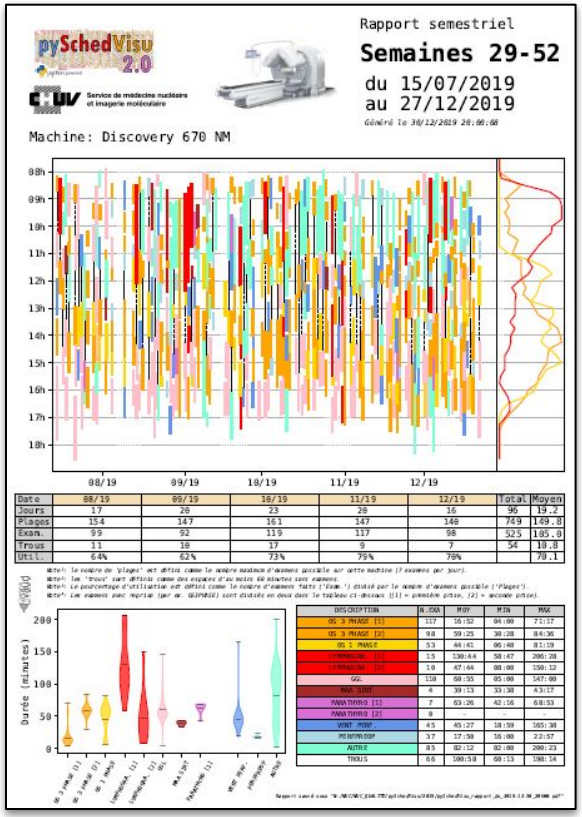
Chaque rapport contient plusieurs pages, avec une page par *scanner* et par *période de temps*. Il y a actuellement 5 scanners (*Discover 670 NM, Intevo, Millennium, PET GE, PET Siemens*) et 7 périodes de temps (*hebdomadaire, bimensuel, mensuel, trimestriel, semestriel, annuel et longue durée*). Il y a donc 35 pages en tout (pour peu que les données soient présentes pour chacune de ces combinaisons).

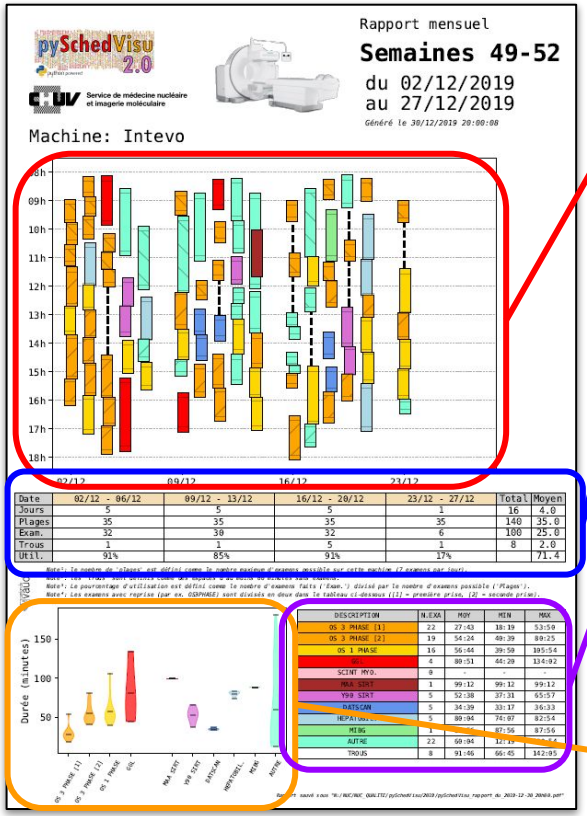
Chaque page du rapport contient 4 sections principales:

1. Le **planning** *effectif*, sous forme d'agenda avec des rectangles représentants chaque examen.
2. Un tableau de statistiques de **comptage** pour chaque jour/semaine/mois/année.
3. Un tableau de statistiques sur la **durée** de chaque type d'examen.
4. Un graphique de type "**violin plot**", représentant visuellement les durées de chaque type d'examen.

Des **informations générales** sur le type de rapport (nom de la machine, dates, etc.) sont également visibles.

Exemple de rapports pySchedVisu





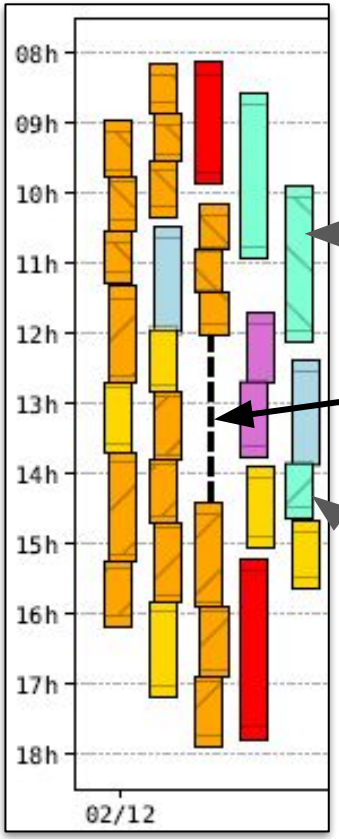
1. Planning effectif

2. Tableau de comptage

3. Tableau de statistiques des durées d'examen

4. Violin plot

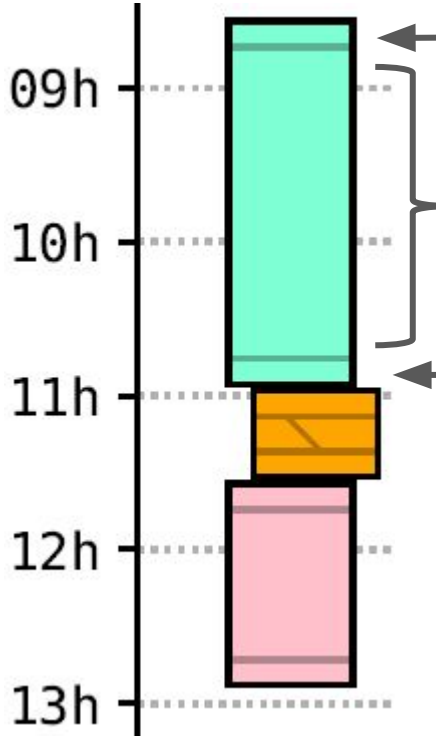
Légende pour le planning créé par pySchedVisu



prise n°1
(hachuré vers la gauche)

“trou” (espace sans examen)

re-prise (n°2) du même patient
(hachuré vers la droite)



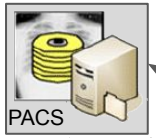
temps de préparation

temps d’examen
(imagerie)

temps de sortie

Comment fonctionne pySchedVisu ? (schéma)

Ceci est un schéma expliquant le fonctionnement de pySchedVisu. Vous trouverez plus d'informations dans les pages qui suivent.



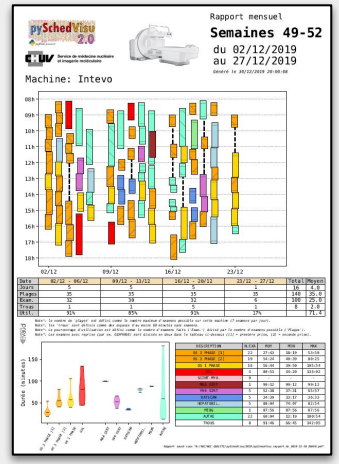
pSV

`df_series`

`df_series`

`df_series`

`df_studies`



1A) Récupération des examens pour chaque jour

1B) Récupération des séries pour chaque examen et chaque jour

1C) Téléchargement des DICOMs et extraction des informations

2) Transformation des séries en examens, avec toutes les informations nécessaires

3) Création du rapport

`config.ini`

0) Chargement du fichier de configuration

`retrieve_data.py`

`extract_data.py`

`create_report.py`

Comment fonctionne pySchedVisu ? (1)

pySchedVisu est un logiciel Python qui est executé chaque semaine à l'aide du Planificateur de Tâches de Windows. Il utilise également la librairie dcm4che afin de communiquer avec le PACS. pySchedVisu utilise également un certain nombres de modules Python afin de communiquer avec le PACS, de manipuler les données et de créer le rapport final. Ces modules sont listés dans le fichier `requirements.txt`.

Comme dit plus haut, pySchedVisu fonctionne en 3 étapes:

1. Récupération des données
2. Extraction des données
3. Création du rapport

Ces trois sections se trouvent dans 3 scripts Python différents et sont décrits dans les pages suivantes.

Comment fonctionne pySchedVisu ? (3)

pySchedVisu exécute d'abord le script `main.py`, qui **organise et coordonne l'exécution du programme entier**, de la récupération des données à l'envoi de l'email contenant le rapport. Ce script appelle donc les autres scripts `retrieve_data.py`, `extract_data.py`, `create_report.py`. Le script `utils.py`, contient des fonctions d'aide.

La première étape de chaque étape (`retrieve`, `extract`, `report`) consiste à déterminer les dates (jours) qui doivent être traités. En effet, l**e programme ne traite pas à nouveau** tous les jours / toutes les données à chaque exécution, mais il **sauvegarde le résultat** de chaque étape afin qu'il n'y ait plus besoin de les refaire. Ainsi, si un jour a déjà été traité, il est simplement recharger depuis le disque, sans être re-téléchargé ou re-traité.

pySchedVisu utilise un **fichier de configuration** (`config.ini`) et produit des `logs` qui peuvent être utiles pour **comprendre les erreurs** et obtenir des statistiques (nombre d'examens traités par heure, temps d'exécution total, etc.)

Comment fonctionne pySchedVisu ? (4)

pySchedVisu traite des données venant du PACS, dont la nomenclature peut être compliquée, c'est pourquoi il est nécessaire de donner quelques brèves définitions:

- Un **examen** (`study` dans le code) est l'ensemble des **séries** qui ont été faites pour un patient à une date donnée. Parfois, un patient a eu plusieurs examens dans la même journée, on parle alors de *reprise*.

- Une **série** (`series` dans le code) est un ensemble **d'images**, une partie d'examen effectuée par les scanners. Typiquement, un **examen** comporte plusieurs **séries** (CT de reconnaissance, examen PET, etc.) qui sont espacés entre eux dans le temps, mais appartiennent au même **examen**.

- Une **image** (`image` ou `instance` dans le code) est un ensemble de pixels et de méta-données ("**headers**"). Certaines informations clés ne se trouvent qu'à ce niveau, d'où la nécessité de télécharger des fichiers DICOMs.

- Un **scanner** (`machine` dans le code) est un ensemble de scanners, souvent un ensemble PET-CT ou SPECT-CT.

Comment fonctionne la récupération des données de pySchedVisu ?

Les données nécessaires à la génération du rapport sont stockées dans le “**header**” des fichiers DICOMs, eux-mêmes stockés dans le PACS. Afin de les récupérer, pySchedVisu effectue les étapes suivantes (script `retrieve_data.py`):

1. **Pour chaque jour** nécessaire, **trouver** tous les examens de type CT, PT et NM effectués ce jour-là (fonction `find_studies_for_day`). Il y en a typiquement entre 25 et 50, toutes machines confondues.
2. **Pour chaque examen**, **trouver** toutes les séries qu’elle contient (fonction `find_series_for_studies`). Il y en a typiquement entre 100 et 400 par jour, toutes machines et examens confondus.
3. **Pour chaque série**, **trouver** les informations nécessaires (fonction `fetch_info_for_series`):
    a. **Télécharger** les fichiers DICOM associés à chaque série. En fonction de la modalité, ceci peut être un ou plusieurs fichiers, car pour les séries CT & PT, il est suffisant de trouver la première et la dernière image d’une série, alors que pour les séries NM, tout la série doit être téléchargée.
    b. **Lire et stocker** les informations nécessaires se trouvant dans le **header** du fichier DICOM. Les données sont stockées sous forme de DataFrame (`pandas`) dans des fichiers binaire `Pickle`. Le programme crée **un fichier par jour**, contenant la table de toutes les séries pour ce jour-là.

Comment fonctionne l'extraction des données de pySchedVisu ?

Les données de toutes les séries sont maintenant stockées pour chaque jour dans un fichier Pickle de type `XXXXX.pkl` (un fichier par jour, dossier `C:\TEMP\pySchedVisu\data`).

La prochaine étape consiste à **réunir toutes les séries** en un grand DataFrame et les **regrouper en examens**. Pour ce faire, pySchedVisu effectue les étapes suivantes (script `extract_data.py`):

1. **Charger les séries** de chaque jour (fonction `load_data_from_files`)
2. Créer un **consensus** sur le **noms des scanners** (fonction `mark_machine_group`)
3. Marquer les **reprises** (fonction `mark_retakes`)
4. **Regrouper** les séries en examen (`df_series` devient `df_studies`)
5. Créer un **consensus** sur les **descriptions des examens** (fonction `create_description_consensus`)
6. Ajouter des annotations aux examens (fonctions `add_preparation_times` et `add_time_to_prev_and_next`)
7. **Stocker** les deux DataFrame de séries et d'examens (sous `df_series.pkl` et `df_studies.pkl`)

Comment fonctionne la création du rapport de pySchedVisu ?

Tous les examens sont maintenant **stockées ensemble** dans le fichier `studies.pkl` (dossier `C:\TEMP\pySchedVisu\data`). Ces examens doivent maintenant être **affichés dans un rapport** ayant une page par *scanner* et par *période de temps* (rapport hebdomadaire, mensuel, etc.). Pour ce faire, pySchedVisu effectue les étapes suivantes (script `create_report.py`):

1. **Déterminer les dates** de début et fin pour chaque “fréquence” (fonction `create_report`)
2. **Créer un fichier** PDF multi-pages (module `PdfPages` de `matplotlib`)
3. Pour chaque *machine*
    a. Pour chaque *période de temps*
        i. Créer une **page** contenant:
            1. Les informations de la page (fonctions `create_header` et `create_notes`)
            2. L’agenda `create_schedule`)
            3. Le tableau de comptage sous l’agenda (fonction `create_daily_table`)
            4. Le “violin” plot (fonction `create_violin`)
            5. Le tableau des statistiques (fonction `create_stat_table`)

Comment fonctionne pySchedVisu ? (df_series)

	Date	Patient ID	Machine	Machine Group List	Machine Group	Modality	Institution Name	Series Time	Start Time	End Time	Protocol Name	Study Description	i_take	SUID	Study Instance UID	Series Instance UID
0	20160104	[illegible]	Discovery 690	discovery690	PET GE	CT	PET CT CHUV	085445	085503	085503	4.3 PET-CT Rb82 Rest Stress	PET CT COEUR	1	[illegible]	[illegible]	[illegible]
1	20160104	[illegible]	Discovery 690	discovery690	PET GE	CT	PET CT CHUV	085540	085555	085555	4.3 PET-CT Rb82 Rest Stress	PET CT COEUR	1	[illegible]	[illegible]	[illegible]
2	20160104	[illegible]	Discovery 690	discovery690	PET GE	PT	PET CT CHUV	085834	085847	090453	5.5 PT-CT Rb82 Rest Recon	PET CT COEUR	1	[illegible]	[illegible]	[illegible]
3	20160104	[illegible]	Discovery 690	discovery690	PET GE	PT	PET CT CHUV	085834	090047	090047	5.5 PT-CT Rb82 Rest Recon	PET CT COEUR	1	[illegible]	[illegible]	[illegible]
4	20160104	[illegible]	Discovery 690	discovery690	PET GE	PT	PET CT CHUV	091304	091317	091923	5.6 PT-CT Rb82 Stress Recon	PET CT COEUR	1	[illegible]	[illegible]	[illegible]
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
122594	20191223	[illegible]	Tandem_Discovery_670	brightspeed,tandemdiscove...	Discovery 670 NM	NM	CHUV	164025	164025	164525	User&GGL SENT&GGL SEIN D	GGL SEIN D	1	[illegible]	[illegible]	[illegible]
122595	20191223	[illegible]	Tandem_Discovery_670	brightspeed,tandemdiscove...	Discovery 670 NM	NM	CHUV	165331	165331	165831	User&GGL SENT&GGL SEIN D	GGL SEIN D	1	[illegible]	[illegible]	[illegible]
122596	20191223	[illegible]	Tandem_Discovery_670	brightspeed,tandemdiscove...	Discovery 670 NM	NM	CHUV	170459	170459	173459	User&GGL SENT&GGL SEIN D	GGL SEIN D	1	[illegible]	[illegible]	[illegible]
122597	20191223	[illegible]	BrightSpeed	brightspeed,tandemdiscove...	Discovery 670 NM	CT	CHUV	172559	172614	172633	5.10 NUC Thorax Extra Low...	GGL SEIN D	1	[illegible]	[illegible]	[illegible]
122598	20191223	[illegible]	BrightSpeed	brightspeed,tandemdiscove...	Discovery 670 NM	CT	CHUV	172655	172710	172710	5.10 NUC Thorax Extra Low...	GGL SEIN D	1	[illegible]	[illegible]	[illegible]

122599 rows × 16 columns

Ceci est le tableau (*DataFrame*) des **séries**. Il se trouve (normalement) sous `C:/TEMP/pySchedVisu/data/series.pkl` et contient les colonnes suivantes:

- `Date` : date du jour de l'examen
- `Patient ID` : IPP du patient
- `Machine` : nom du scanner sur lequel la série a été enregistrée / créé
- `Machine Group List` & `Machine Group` : noms du groupe de scanner auquel cette série appartient
- `Modality` : modalité de la série (PT, CT, NM)
- `Institution Name` : nom de l'institution de la série (attribut DICOM)
- `Series Time` : temps de création de la série (attribut DICOM)
- `Start Time` & `End Time` : temps de début et de fin de la série (calculé par pySchedVisu)
- `Protocol Name` : nom du protocole de la série
- `Study Description` & `Series Description` : description de l'examen et de la série
- `i_take` : numéro de la reprise (1 = première prise, 2 = reprise)
- `Series Instance UID` : identifiant unique de la série (attribut DICOM)
- `Study Instance UID` : identifiant de l'examen à laquelle la série appartient (attribut DICOM)
- `SUID` identifiants de la série avec reprise

Le tableau contient également les champs supplémentaires suivants qui sont tous des attributs DICOMs propres à la série ou aux images qu'elle contient:

- `AcquisitionTime`, `ActualFrameDuration`, `ContentTime`, `ImageType`, `InstanceNumber`, `Number of Series Related Instances`, `NumberOfFrames`

Comment fonctionne pySchedVisu ? (df_studies)

SUID	Date	Patient ID	Machine	Machine Group	Modality	Description	Study Description	Start Time	End Time	Start Time Prep	End Time Prep
.1600012314.0_1	20160104		PET GE	PET GE	CT/PT	RB82 COEUR	PET CT COEUR	085503	092247	084503	092817
.1600010126.0_1	20160104		PET GE	PET GE	CT/PT	FDG STAND.	PET CT TRONC	093347	095214	092817	095807
.1600011035.0_1	20160104		PET GE	PET GE	CT/PT	FDG STAND.	PET CT TRONC	100401	102208	095807	102745
.1600011265.0_1	20160104		Discovery 670 NM	Discovery 670 NM	CT/NM	PERFPREOP	PERFUSION pre-op	103003	110025	102003	111025
.1500139807.0_1	20160104		PET GE	PET GE	CT/PT	FDG STAND.	PET CT TRONC	103322	105240	102745	105630
...	...		...	...	...	...	...	...	...	...	...
.1900283253.0_1	20191223		Intevo	Intevo	NM/CT	OS 1 PHASE	Os_1Phase	145841	154940	145338	155657
.1900284992.0_1	20191223		PET Siemens	PET Siemens	CT/PT	GA68 PSMA	PET^3_Pet_Ga68_PSMA_Tronc_Flow (Adult)	154753	160214	153753	160724
349500000005_1	20191223		Intevo	Intevo_NoCT	NM	OTHER	VENTILATION TC	160415	161914	155657	162914
.1900283114.0_1	20191223		Discovery 670 NM	Discovery 670 NM	NM/CT	GGL	GGL SEIN D	161217	173459	160217	174459
.1900285831.0_1	20191223		PET Siemens	PET Siemens	CT/PT	GA68 PSMA	PET^3_Pet_Ga68_PSMA_Tronc_Flow (Adult)	161234	162456	160724	163456

28021 rows × 11 columns

Ceci est le tableau (*DataFrame*) des **examens**. Il se trouve (normalement) sous `C:/TEMP/pySchedVisu/data/studies.pkl` et contient les colonnes suivantes:

- `Date` : date du jour de l'examen
- `Patient ID` : IPP du patient
- `Machine Group` & `Machine` : scanner sur lequel l'examen a été fait. Ces champs sont créés à partir des séries de chaque examen.
- `Modality` : modalité de la série (PT, CT, NM)
- `Description` : description standardisée de l'examen (après création du consensus)
- `Study Description` : description de l'examen
- `Start Time` & `End Time` : temps de début et de fin de l'examen (calculé par `pySchedVisu` à partir des temps de début et de fin des séries de chaque examen)
- `Start Time Prep` & `End Time Prep` : temps de début et de fin de l'examen en incluant le temps de préparation des patients (10 min, configurable)

Où est-ce que pySchedVisu s'exécute ?

Actuellement (3 janvier 2020), pySchedVisu est installé comme **tâche hebdomadaire** sur le poste HOS 51499. pySchedVisu s'exécute tous les samedis matins à 01:00 du matin.

Afin que l'exécution s'effectue correctement chaque semaine, il est nécessaire que la **machine ne soit pas éteinte**. De plus, l'utilisateur qui est inscrit comme propriétaire de la tâche dans le planificateur (actuellement David Viertl) doit avoir une **session ouverte** sur la machine.

Veuillez voir ici pour plus d'informations concernant la création de la tâche récurrente ou pour **changer l'heure et/ou le jour d'exécution**.

Où se trouve pySchedVisu ?

Le code de pySchedVisu se trouve dans le dépôt de code GitHub suivant:

https://github.com/blaurenczy/pyschedvisu

Le programme compilé (exécutable sans Python) se trouve également dans divers dossiers du répertoire `NUC`:

1. `N:\NUC\COMMUN\CIVILISTE\SchedVisu\pySchedVisu`
2. `N:\NUC\NUC_QUALITE\pySchedVisu`

Le code et le programme compilé se trouvent également dans le dossier `C:\TEMP\pySchedVisu` du poste où le programme s'exécute.

Comment installer pySchedVisu sur un nouveau poste ?

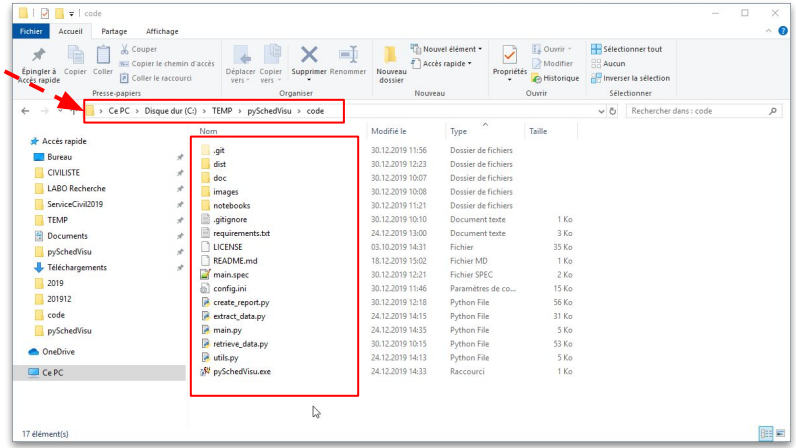
Pour "*installer*" (déployer) pySchedVisu, il vous faut effectuer les étapes suivantes:

1. Récupérer la dernière version de pySchedVisu, sous forme de fichier ZIP s'appelant **pySchedVisu_v2_1.zip** (ou une autre version plus récente). Voir ici pour plus d'informations.

2. Extraire le logiciel dans votre dossier `C:\TEMP` de manière à ce que le dossier contenant tous les fichiers soit appelé `C:\TEMP\pySchedVisu`.

3. Modifier le fichier de configuration (`config.ini`) afin que les paramètres correspondent à la nouvelle machine (paramètres de connexion au PACS, chemins d'accès aux fichiers, etc.). Voir ici pour plus de détails (et également ici).

4. Créer une tâche dans le Planificateur de Tâches Windows. Voir ici pour plus de détails.



Comment paramétrer pySchedVisu ?

pySchedVisu fonctionne grâce à un fichier de configuration (`config.ini`), grâce auquel un grand nombre de paramètres peuvent être changés. Ainsi, de nombreux changements peuvent être faits, sans avoir besoin de toucher au code Python (et donc sans avoir besoin de re-compiler ou de changer l'exécutable).

Tous les paramètres du fichier de configuration sont documentés dans le fichier lui-même. Certains paramètres méritent toutefois une explication plus détaillée. Ils sont décrits dans les pages qui suivent.

Note: Les lignes commençant par un `#` ne sont pas interprétées et servent juste à documenter / commenter ce fichier de configuration.

Comment paramétrer pySchedVisu ? (main 1)

```
[main]

report_range = hebdomadaire,bimensuel,mensuel,trimestriel,semestriel,annuel,longueduree
```

pySchedVisu crée un rapport pour différentes périodes de temps (la dernière semaine, le dernier mois, etc.). Ce paramètre permet de changer lesquelles de ces périodes se trouvent dans le rapport. Par exemple, en utilisant la ligne suivante:

```
report_range = hebdomadaire,mensuel,annuel
```

Seul le rapport de la dernière semaine, du dernier mois, et de l'année en cours seront inclus dans le rapport.

Note: afin de définir de nouvelle période de temps (les 3 dernières semaines par exemple), des modifications au code doivent être effectués. Ce paramètre permet seulement d'inclure ou non des périodes déjà définies, pas d'en créer de nouveau à la volée.

Comment paramétrer pySchedVisu ? (main 2)

```
[main]

report_year_start = 2016
```

pySchedVisu récupère les données depuis le début de l'année 2016. Ce paramètre permet de changer l'année à partir de laquelle les données sont récupérées et incluses dans le rapport.

Note: Lors du changement de ce paramètre vers une date antérieure à 2016, des nouvelles données peuvent être requise (par exemple tout 2015), ce qui peut surcharger le PACS, ainsi que potentiellement nécessiter un supplément de temps d'exécution de pySchedVisu.

Comment paramétrer pySchedVisu ? (main 3)

```
[main]

debug_level = WARNING
```

pySchedVisu crée des fichiers de logs (normalement stockés dans `C:/TEMP/pySchedVisu/logs`). Ces logs contiennent uniquement les informations les plus importantes sur l'exécution du logiciel. Des logs plus ou moins détaillés peuvent être obtenus en changeant ce paramètre pour un autre niveau (`DEBUG`, `INFO`, `WARNING`, `ERROR`).

Si des fichiers de log ne sont pas nécessaires, ils peuvent être librement effacés, afin de créer de la place ou de réduire le nombre de fichiers sur le poste où pySchedVisu est executé.

Note: un logging de bas niveau (`DEBUG` ou même `INFO`) peut causer une augmentation significative de la taille des fichiers de logs, ainsi qu'un potentiellement ralentissement de l'exécution de pySchedVisu.

Comment paramétrer pySchedVisu ? (email)

```
[email]
body = …
subject = Rapport hebdomadaire pySchedVisu
recipients_email = Prenom.Nom@chuv.ch,exemple@gmail.com
```

pySchedVisu envoie les rapports par e-mails. Ces paramètres permettent de changer le sujet, le texte et les destinataires de ces e-mails.

Les destinataires doivent être spécifiés comme une liste séparée par des virgules, sans espaces entre les adresses.

Le texte de l'e-mail ("body") peut contenir des balises HTML. Il contient également des balises de remplacement de texte (par exemple `{__REPORT_PATH__}`) qui ne doivent pas être changées (à moins d'être certain de comprendre ce que l'on fait).

Comment paramétrer pySchedVisu ? (path)

```
[path]
dcm4che_path = C:/TEMP/pySchedVisu/dcm4che-5.19.1/bin
dicom_temp_dir = C:/TEMP/pySchedVisu/DICOMs
log_dir = C:/TEMP/pySchedVisu/logs
data_dir = C:/TEMP/pySchedVisu/data
studies_db_save_path = C:/TEMP/pySchedVisu/data/studies.pkl
series_db_save_path = C:/TEMP/pySchedVisu/data/series.pkl
output_dir = N:/NUC/NUC_QUALITE/pySchedVisu
```

pySchedVisu stocke des fichiers (logs, DICOMs, Pickle, etc.) et utilise d'autres ressources (dcm4che, dossier temporaire pour les DICOMs, etc.). Le logiciel a besoin de savoir où ces éléments se trouvent, mais il n'est pas nécessaire qu'ils soient toujours tous au même endroit prédéfini. Il est donc **possible de changer n'importe lequel de ces chemins**, ***pour peu que le chemin existe et qu'il soit possible d'y accéder***. Cependant, pySchedVisu ne peut être exécuté que depuis le dossier `C:/TEMP` (restriction du CHUV). Ainsi, tout ce qui est exécutable doit se trouver dans ce dossier-là.

Veuillez respecter les conventions de notations: pas de *slash* `/` final à la fin des chemins des dossiers et n'utilisez des *backslashs* `\` que si c'est vraiment nécessaire.

Comment paramétrer pySchedVisu ? (PACS)

```
[PACS]
local_ae_title = CIVILISTENUC2
local_host = 155.105.54.51
local_port = 104
remote_ae_title = csps1FIR
remote_host = 155.105.3.105
remote_port = 104
```

pySchedVisu a besoin de se connecter au PACS afin de télécharger les fichiers DICOMs contenant les données nécessaires pour créer le rapport. **Les paramètres de connexions sont différents pour chaque machine!** Ainsi, il est nécessaire de récupérer/établir ces paramètres pour chaque nouvelle installation. Veuillez entrer en contact avec le service de PACS TRM de Radiologie pour obtenir ces identifiants pour un nouveau poste (*rad.trm.pacs at chuv.ch*).

Comment paramétrer pySchedVisu ? (machines 1)

pySchedVisu traite les données de plusieurs scanners différents et doit donc savoir quelles scanners existent. Ceci est paramétré à plusieurs endroits du fichier de configuration. **Afin d'ajouter un nouveau scanner, tous les champs liés aux scanners doivent être paramétrés!** Les pages suivantes (machines 1-5) décrivent comment procéder.

```
[machines]
PET Siemens = biograph64,biograph64vision600
PET GE = discovery690
Discovery 670 NM = brightspeed,tandemdiscovery670
Discovery 670 NM_NoCT = tandemdiscovery670
Millennium = millenniummpr
Intevo = encore2,symbiaintevo16
Intevo_NoCT = encore2
```

La section [machines] (extrait ci-dessus) définit les noms des différents "groupes" de scanners. En effet, un scanner est la plupart du temps constitué d'une paire de scanner (CT + PT ou CT + SPECT). Ainsi, le nom affiché dans le rapport comprend les examens effectués sur le CT **ET** le PT (ou le SPECT) pour un scanner donné.
Ces noms de scanners doivent correspondre à ce qui se trouve dans les fichiers DICOMs et donc dans le tableau des séries df_series (voir ici). Les noms de groupes vont déterminer ce qui se trouve dans le tableau des études df_studies (voir ici).
Il est parfois nécessaire de définir un "groupe" nommé _NoCT afin d'inclure les examens où aucun CT n'a été effectué (i.e. examen avec uniquement des séries de type PT ou NM). Ces groupes sont ensuite intégrés automatiquement au groupe avec CT.

Comment paramétrer pySchedVisu ? (machines 2)

```
[description_discovery670nm]
OS 3 PHASE = os3phases,os3phase,scintiosseuse3phases
OS 1 PHASE = os1phase,scintiosseuse1phase
LYMPHOGRA. = lymphographie
GGL = gglseind,gglseing,gglseins,gglmelanome,gglgyneco,gglseinbilat,gglseinbilateral
MAA SIRT = maafoie
PARATHYRO = parathyroide
VENT PERF. = ventilationtc,perfusionembolie,ventilation
PERFPREOP = perfusionpreop
```

La section `[description_XXXXX]` (extrait ci-dessus), définit le *consensus* sur les descriptions des examens pour chaque scanner. Les noms à gauche du signe égal `=` sont les noms affichés dans le rapport, tandis que sur la droite, on trouve une liste séparée par des virgules des descriptions d'examens, tout en minuscule. Ces descriptions doivent correspondre à ce qui se trouve dans le tableau des séries `df_series` (voir ici). Les noms des descriptions vont déterminer ce qui se trouve dans le tableau des études `df_studies` (voir ici).

Il est recommandé de spécifier un maximum de **9** types d'examens différents, typiquement les plus courants. Les autres types d'examens moins fréquents seront regroupés dans une catégorie "`AUTRE`" dans le rapport final.

Comment paramétrer pySchedVisu ? (machines 3)

La section `[draw]` contient plusieurs paramètres liés aux scanners, définissant comment regrouper / afficher les études de chaque scanner.

```
[draw]
…
n_study_per_day_discovery670nm = 7,7,7,7,7
n_study_per_day_intevo = 7,7,7,7,7
n_study_per_day_millennium = 1,7,6,3,2
n_study_per_day_petge = 15,15,15,15,15
n_study_per_day_petsiemens = 15,15,15,15,15
```

Les champs `n_study_per_day_XXXXX` spécifient le nombre d'examens "normal" par jour pour chaque scanner, pour chaque jour de la semaine. Ce chiffre est utilisé pour le calcul du pourcentage d'utilisation dans le rapport.

Ce chiffre peut être le même pour tous les jours de la semaine (par exemple 15 examens pour chaque jour de la semaine: `15,15,15,15,15`) ou il peut différer pour chaque jour dans le cas d'un scanner qui est utilisé différemment au cours de la semaine (par exemple pour la Millennium, comme ci-dessus).

Ce paramètre doit être spécifié en tant que liste de 5 chiffres (un par jour de la semaine) séparés par des virgules et sans espaces.

Comment paramétrer pySchedVisu ? (machines 4)

La section `[draw]` contient plusieurs paramètres liés aux scanners, définissant comment regrouper / afficher les études de chaque scanner.

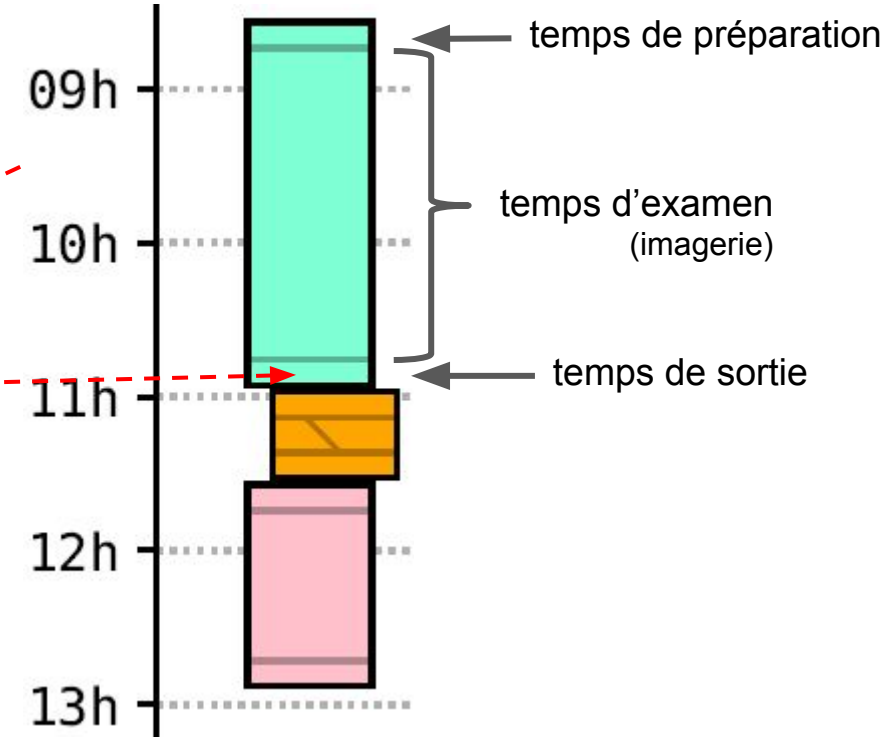
```
[draw]
…
gap_dur_minutes_discovery670nm = 60
gap_dur_minutes_intevo = 60
gap_dur_minutes_millennium = 60
gap_dur_minutes_petge = 30
gap_dur_minutes_petsiemens = 30
```

Les champs `gap_dur_minutes_XXXXX` spécifient le temps (en minutes) minimum pour déclarer un espace entre deux examens comme un *trou*. Les *trous* sont des espaces de temps sans examens au cours de la journée, où un autre examen aurait pu avoir lieu.

Comment paramétrer pySchedVisu ? (machines 5)

```
[draw]
...
prep_time_discovery670nm = 10
prep_time_intevo = 10
prep_time_millennium = 10
prep_time_petge = 10
prep_time_petsiemens = 10
```

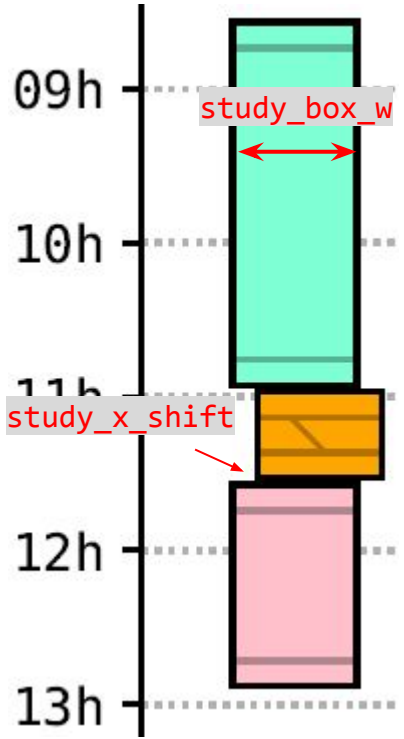
Les champs `prep_time_XXXXX` spécifient le temps (en minutes) nécessaire **avant** et **après** chaque examen pour préparer et faire sortir le patient. Ce temps est ajouté au temps de début et de fin de chaque examen (`Start Time` & `End Time`) afin de donner une vision plus réaliste de la densité d'utilisation des scanners.



Comment paramétrer pySchedVisu ? (draw)

En général, la section `[draw]` contient les paramètres liés à l'affichage du rapport, comme les couleurs des examens, la taille des rectangles, la résolution des images, etc.

```
[draw]
colors = orange,gold,red,pink,...
dpi = 300
sched_start_hour = 8
sched_end_hour = 18
study_box_w = 0.6
study_box_w_prep = 0.6
study_x_shift = 0.06
```



De nouvelles couleurs peuvent être spécifiées selon la liste qui se trouve ici:
https://matplotlib.org/3.1.0/gallery/color/named_colors.html

rosybrown	goldenrod	mediumturquoise	mediumpurple
lightcoral	cornsilk	azure	rebeccapurple
indianred	gold	lightcyan	blueviolet
brown	lemonchiffon	paleturquoise	indigo
firebrick	khaki	darkslategray	darkorchid
maroon	palegoldenrod	darkslategrey	darkviolet
darkred	darkkhaki	teal	mediumorchid

Comment créer une nouvelle version de pySchedVisu

Même si il est possible de changer un certain nombre d'aspects de pySchedVisu uniquement grâce au paramètrage du fichier de configuration, il peut être nécessaire de changer le code et de modifier pySchedVisu afin d'y apporter de plus grands changements, des améliorations ou de corriger des bugs.

N'oubliez pas de faire une sauvegarde du dossier pySchedVisu avant de toucher à quoi que ce soit, afin d'être sûr de pouvoir rétablir la génération du rapport tel qu'elle existe maintenant!

Une fois les changements du code **effectués**, **testés** et **validés**, il est nécessaire de recompiler pySchedVisu afin d'en créer un exécutable. Ceci est faisable à l'aide du module **`pyinstaller`**. La procédure est décrite dans les pages suivantes.

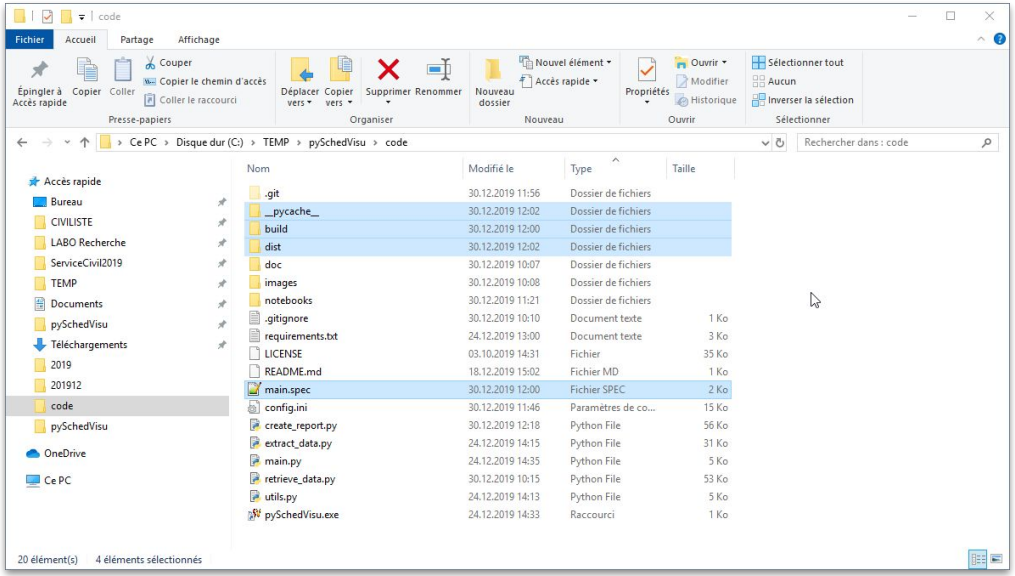
Note importante: Une fois que les changements du code ont été effectués, testés et validés, veuillez les **soumettre** (`commit`) dans le dépot de code GitHub.

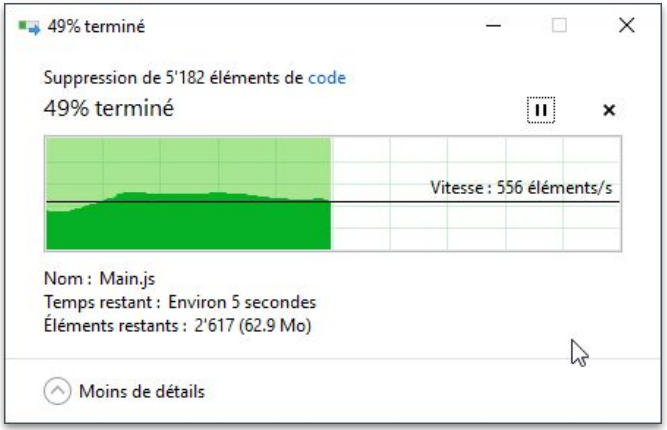
Comment créer une nouvelle version de pySchedVisu (1)

1) Supprimer les fichiers / dossiers suivants, qui se trouvent dans le dossier `C:\TEMP\pySchedVisu` :

`__pycache__`, `build`, `dist`, `main.spec`

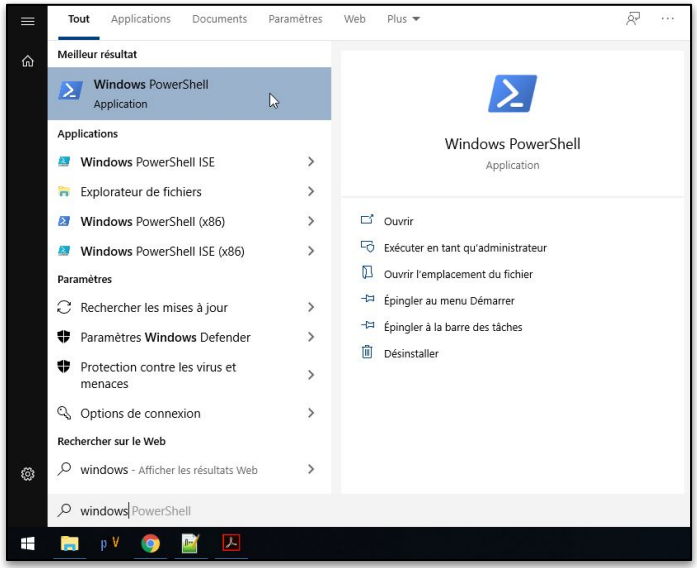
Cela peut prendre un peu de temps, car il y a beaucoup de fichiers dans ces sous-dossiers.

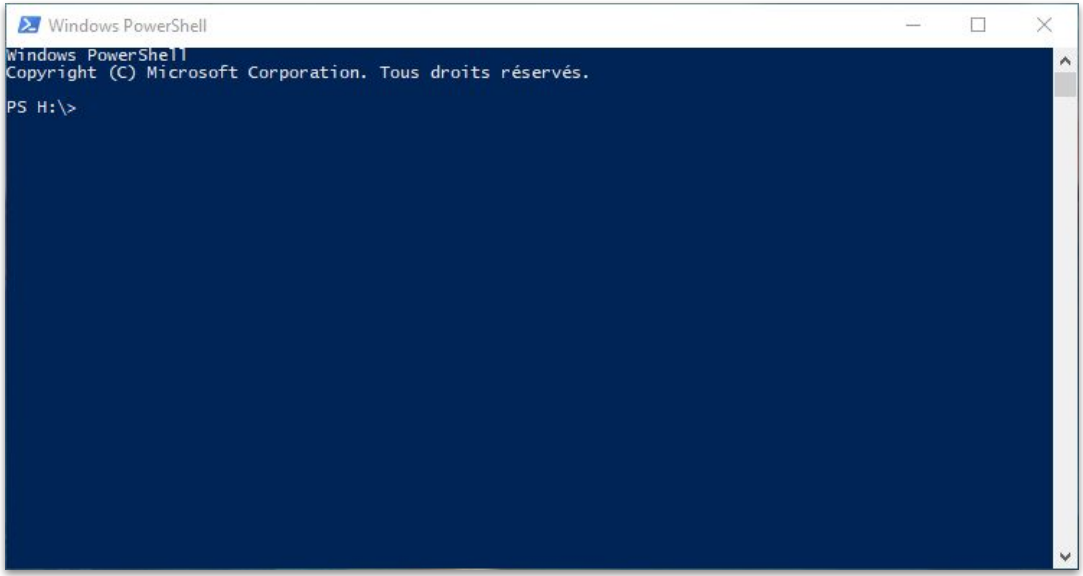




Comment créer une nouvelle version de pySchedVisu (2)

2) Ouvrez le terminal Windows (Windows Power Shell)

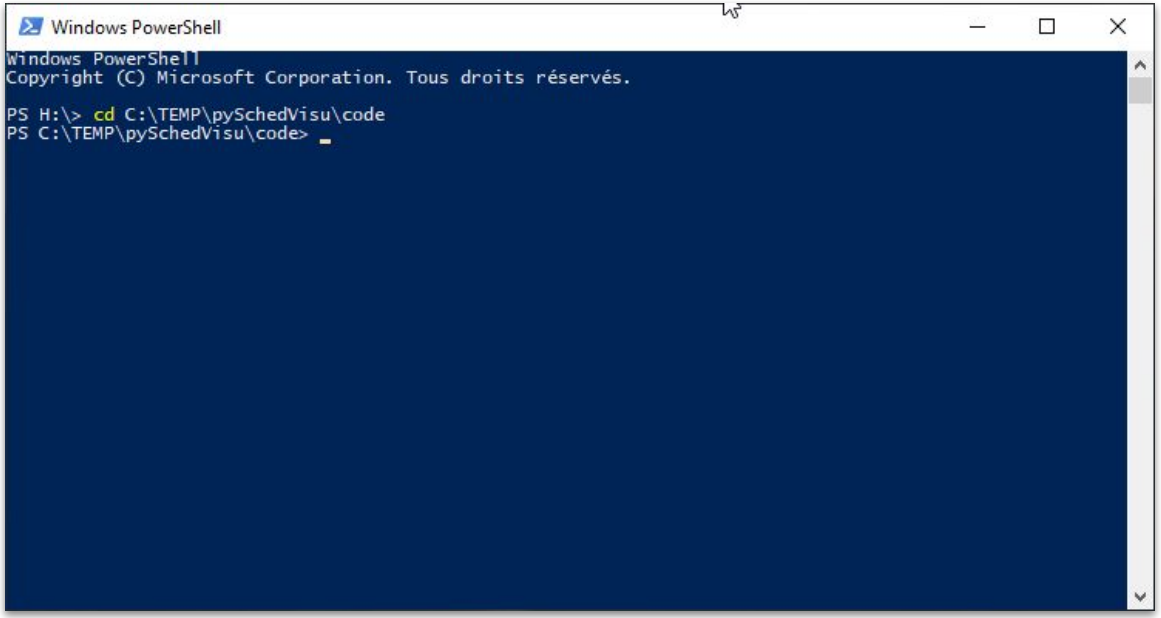




Comment créer une nouvelle version de pySchedVisu (3)

3) Naviguez dans le dossier où se trouve le code en tapant (ou copiant) la ligne suivante dans le terminal:

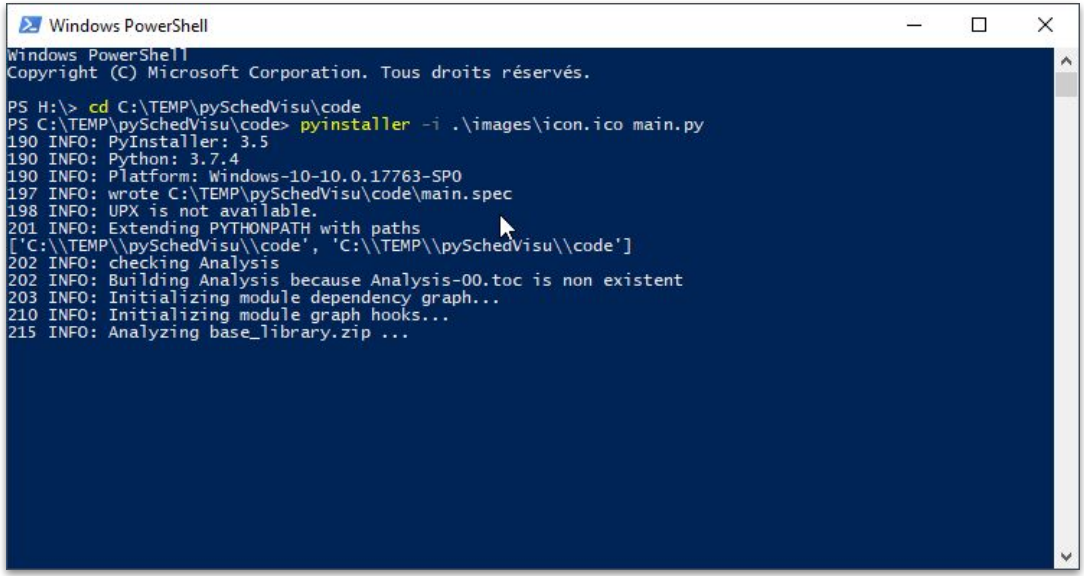
```
cd C:\TEMP\pySchedVisu\code
```



Comment créer une nouvelle version de pySchedVisu (4)

4) Lancez la compilation en tapant (ou copiant) la ligne suivante

```
pyinstaller -i .\images\icon.ico main.py
```

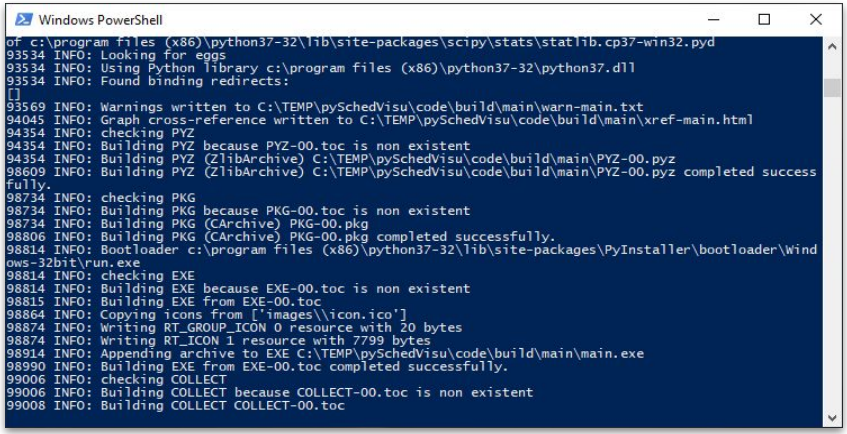


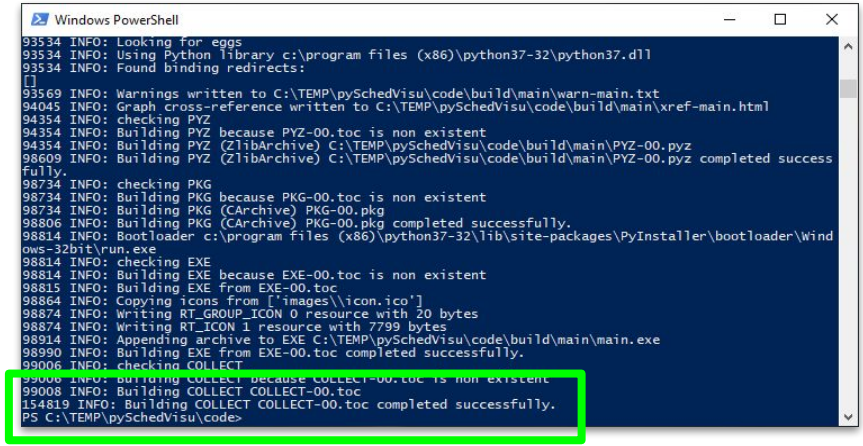
Comment créer une nouvelle version de pySchedVisu (5)

5) Attendez la fin de l'exécution de la commande. Parfois, il est nécessaire de taper la touche "Entrée" si l'exécution se bloque pendant plus de 10 minutes. L'exécution peut prendre quelques minutes et peut afficher des avertissements (`warnings`) de type "`Failed to import XXXXXX`". Ces avertissement ne sont pas grave et peuvent être ignorés (en principe).

La compilation est finie lorsque la ligne suivante s'affiche:

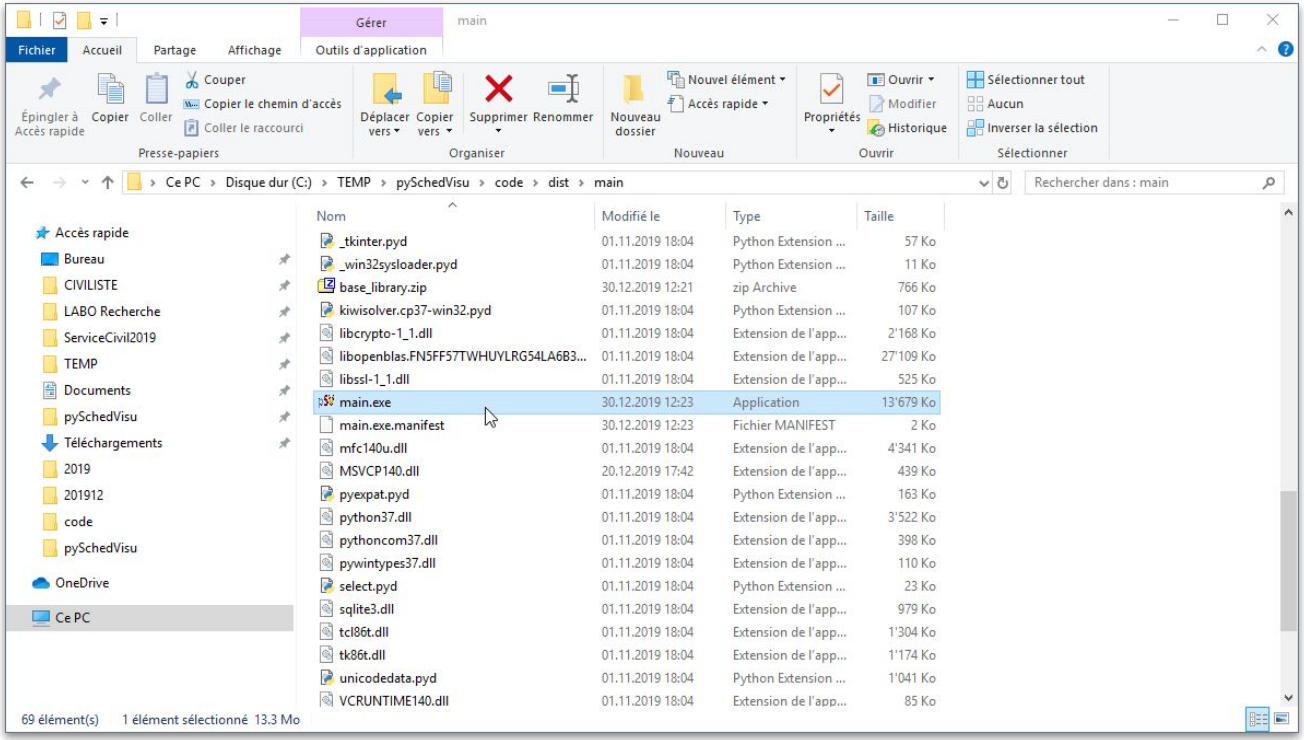
```
Building COLLECT COLLECT-00.toc completed successfully.
```





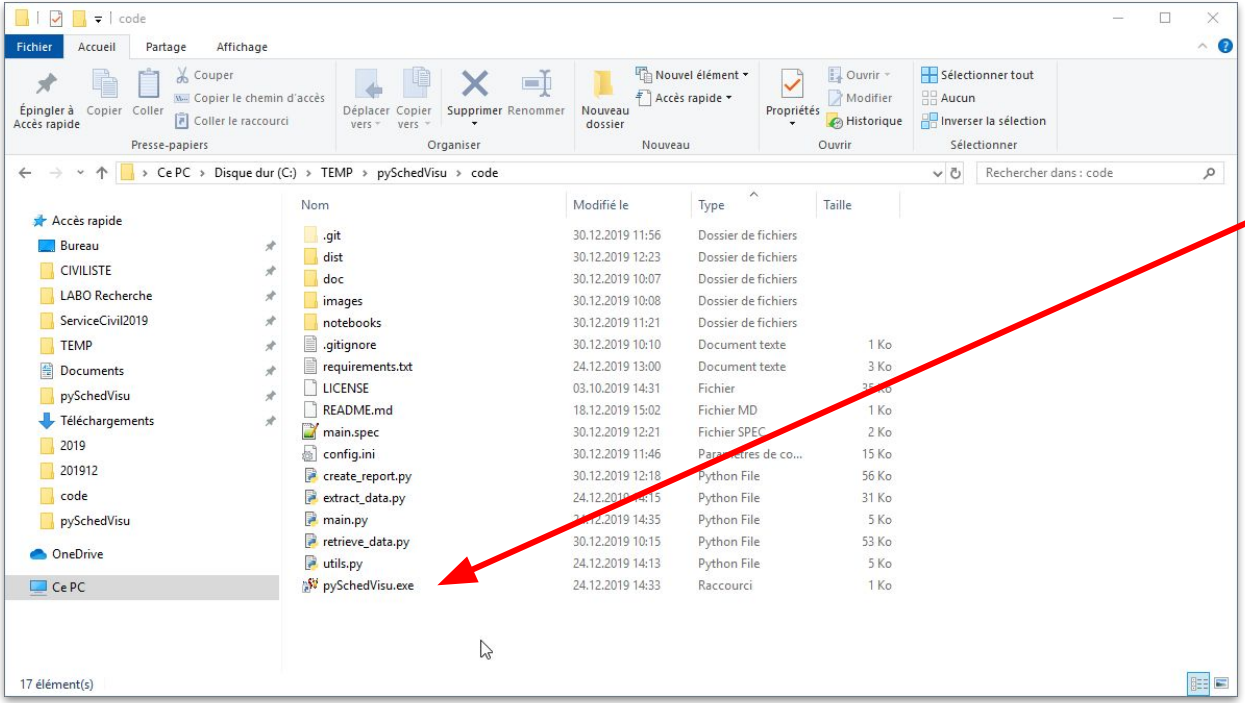
Comment créer une nouvelle version de pySchedVisu (6)

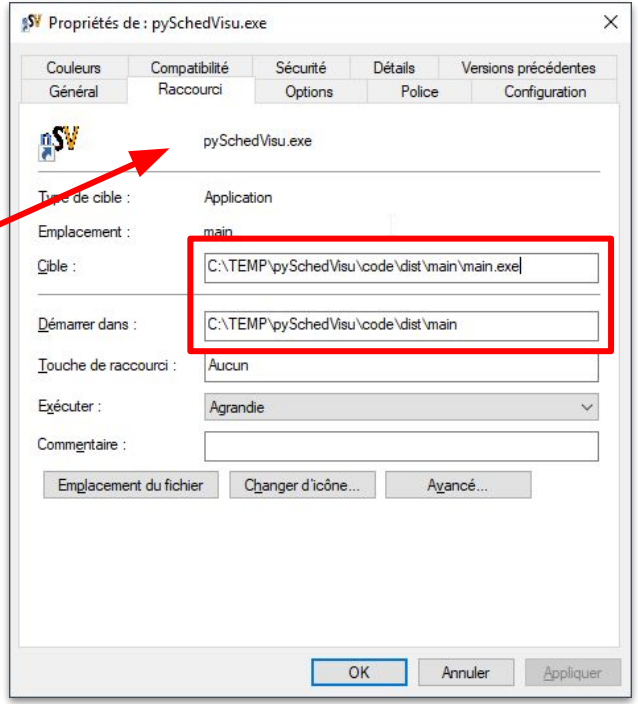
6) Vérifiez que le fichier `main.exe` a bien été créé dans le dossier `C:\TEMP\pySchedVisu\code\dist\main`.



Comment créer une nouvelle version de pySchedVisu (7)

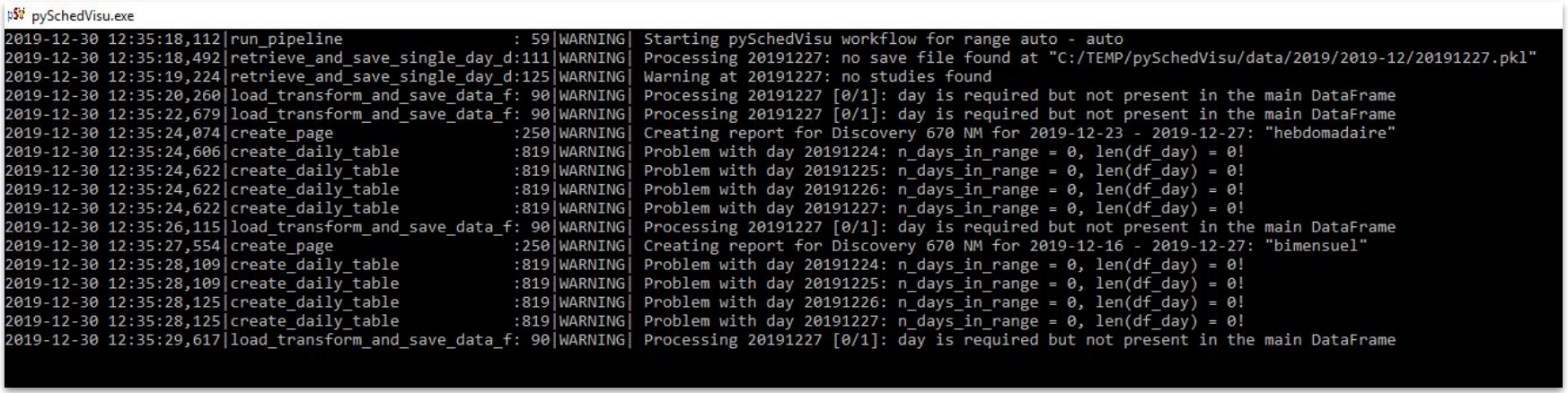
7) Vérifiez que le raccourci `pySchedVisu.exe` dans le dossier où se trouve le code pointe bien sur cet exécutable.





Comment créer une nouvelle version de pySchedVisu (8)

8) Vous pouvez tester le nouvel exécutable en double cliquant dessus, ou en cliquant sur le raccourci cité plus haut.

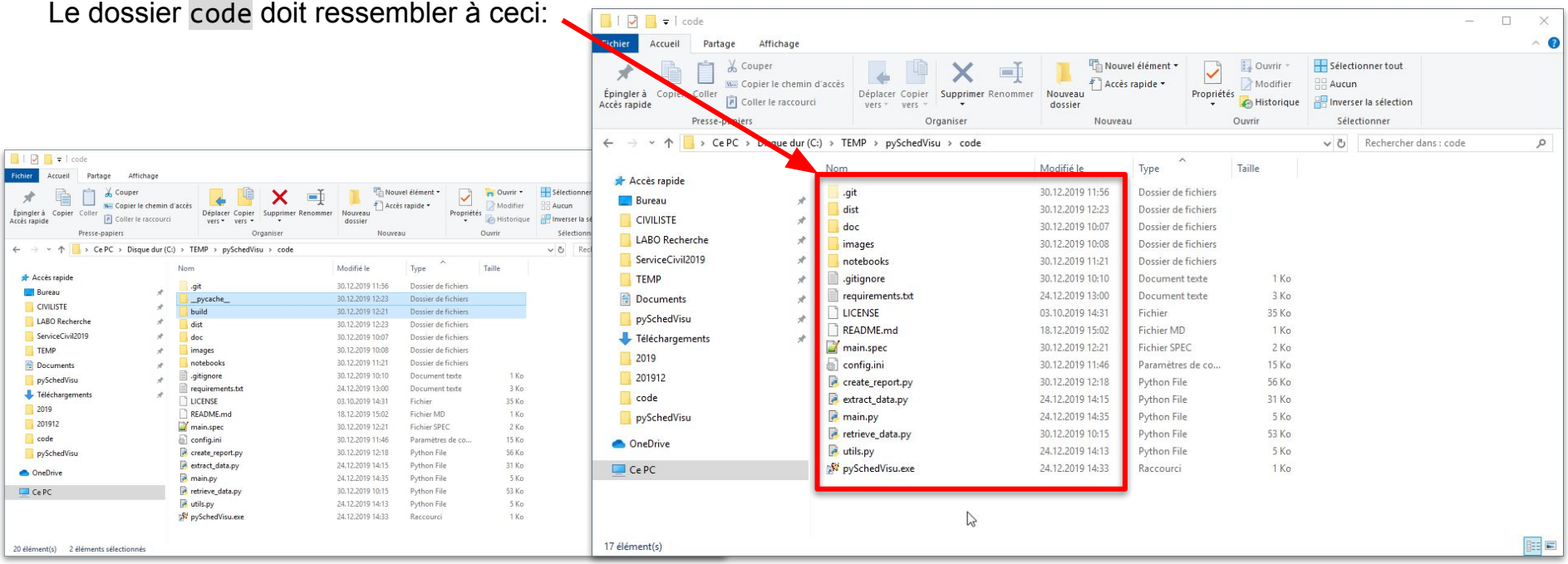


ETAPE NON OBLIGATOIRE

Comment créer une nouvelle version de pySchedVisu (9)

9) Il est maintenant possible de créer le fichier ZIP pour empaqueter tout le code et pouvoir le distribuer, si nécessaire. Il faut d'abord effacer les dossiers inutiles suivants: `__pycache__` & `build`

Le dossier `code` doit ressembler à ceci:



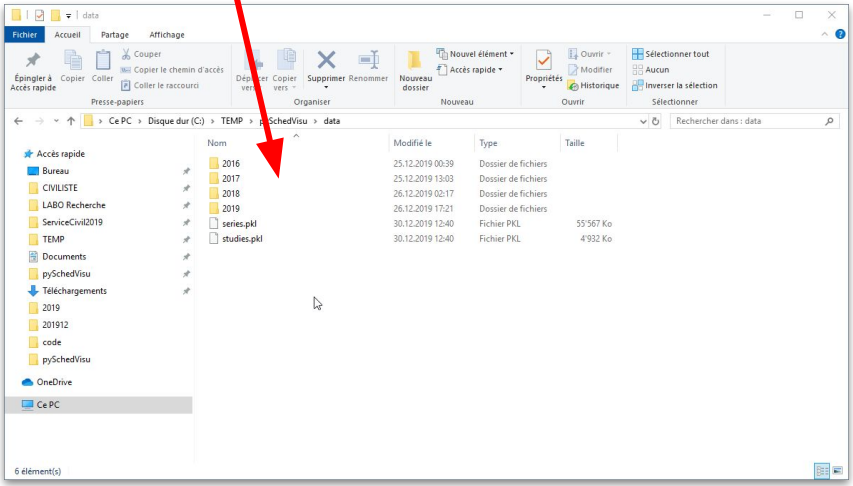
ETAPE NON OBLIGATOIRE

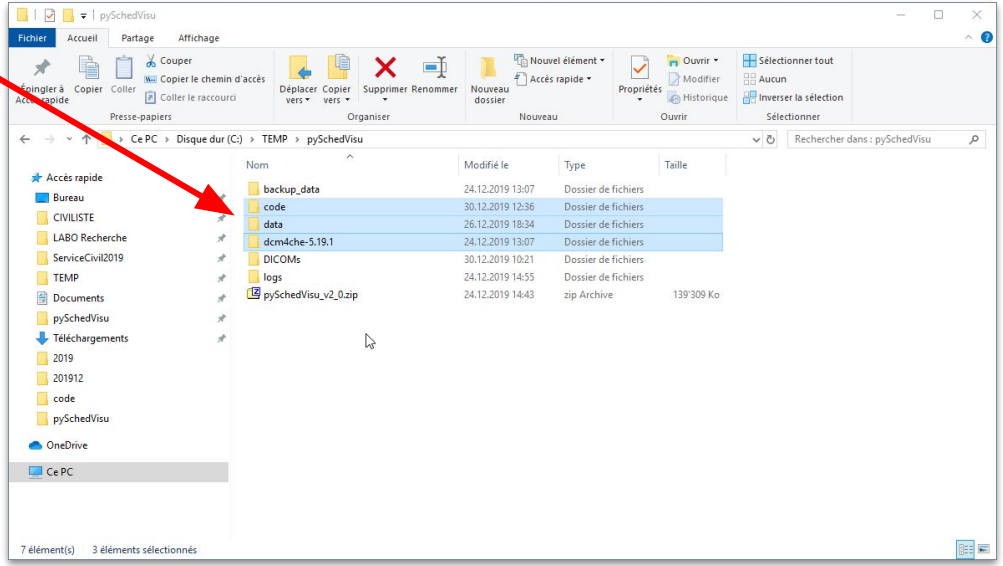
Comment créer une nouvelle version de pySchedVisu (10-11)

10) Vérifiez que toutes les données sont bien dans le dossier `data`

11) Il faut maintenant `zipper` les 3 dossiers suivants *(les autres dossiers ne sont pas nécessaires)*:

- `code`
- `data`
- `dcm4che-5.19.1`

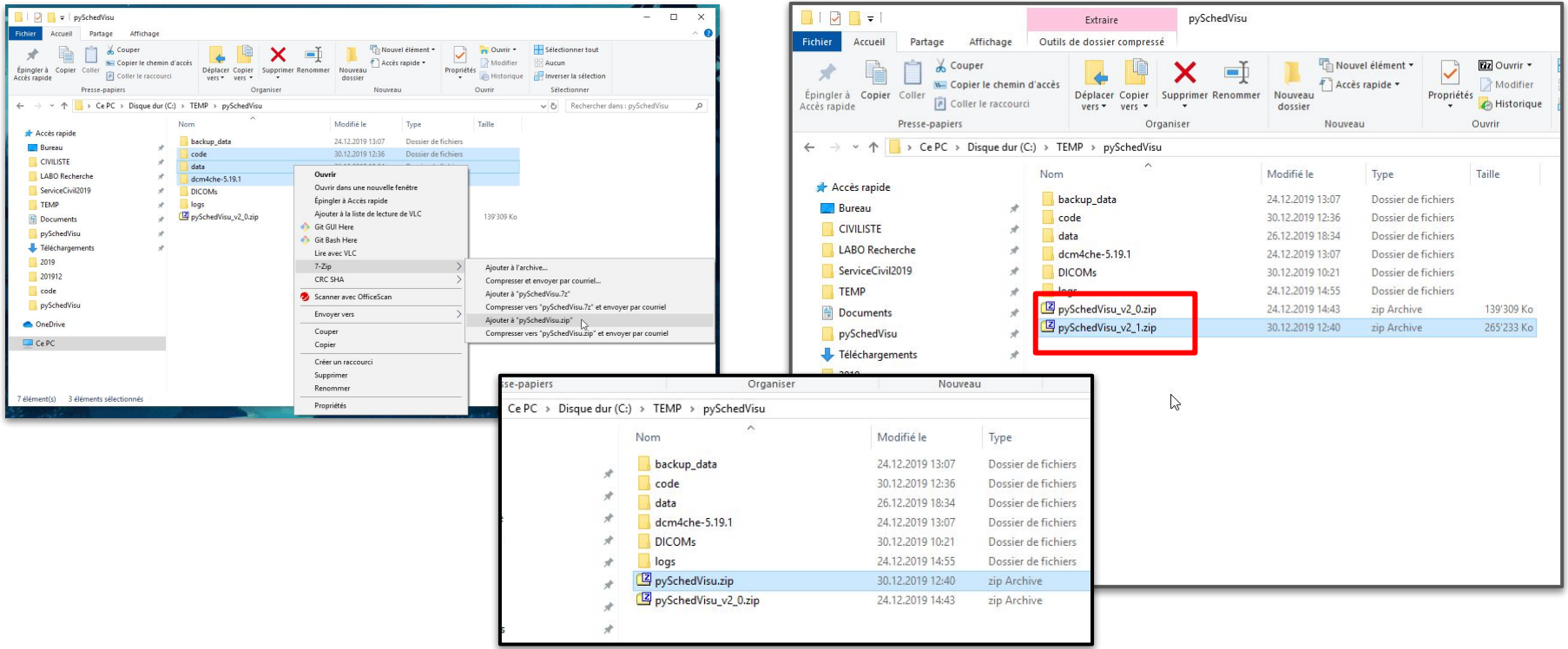




ETAPE NON OBLIGATOIRE

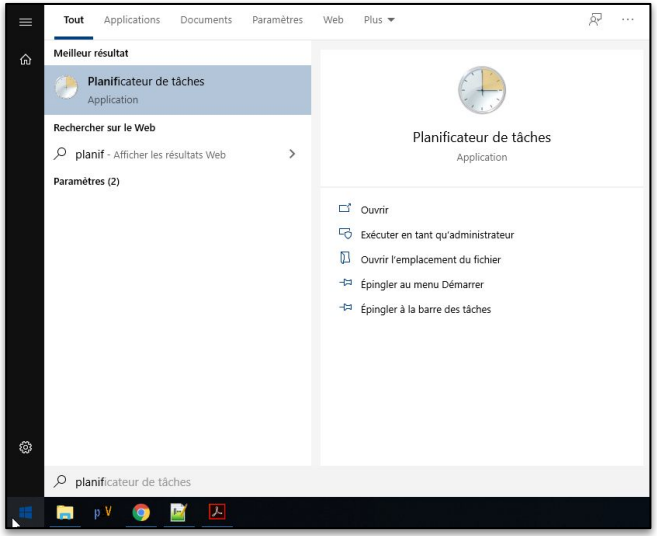
Comment créer une nouvelle version de pySchedVisu (12)

12) Zippez les 3 dossiers avec 7-Zip et renommez le fichier `.zip` avec le nouveau numéro de version.

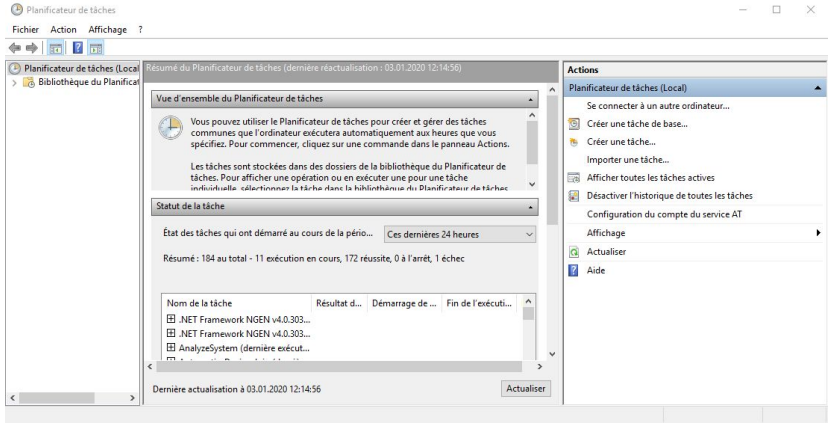


Comment faire pour exécuter pySchedVisu de manière récurrente? (1)

pySchedVisu est un logiciel qui s'exécute de manière récurrente, une fois par semaine (voir ici pour plus d'informations). Ceci est fait grâce au planificateur de tâches Windows. Les pages suivantes montrent comment créer et paramétrer le planificateur afin qu'il lance pySchedVisu chaque semaine de manière fiable.

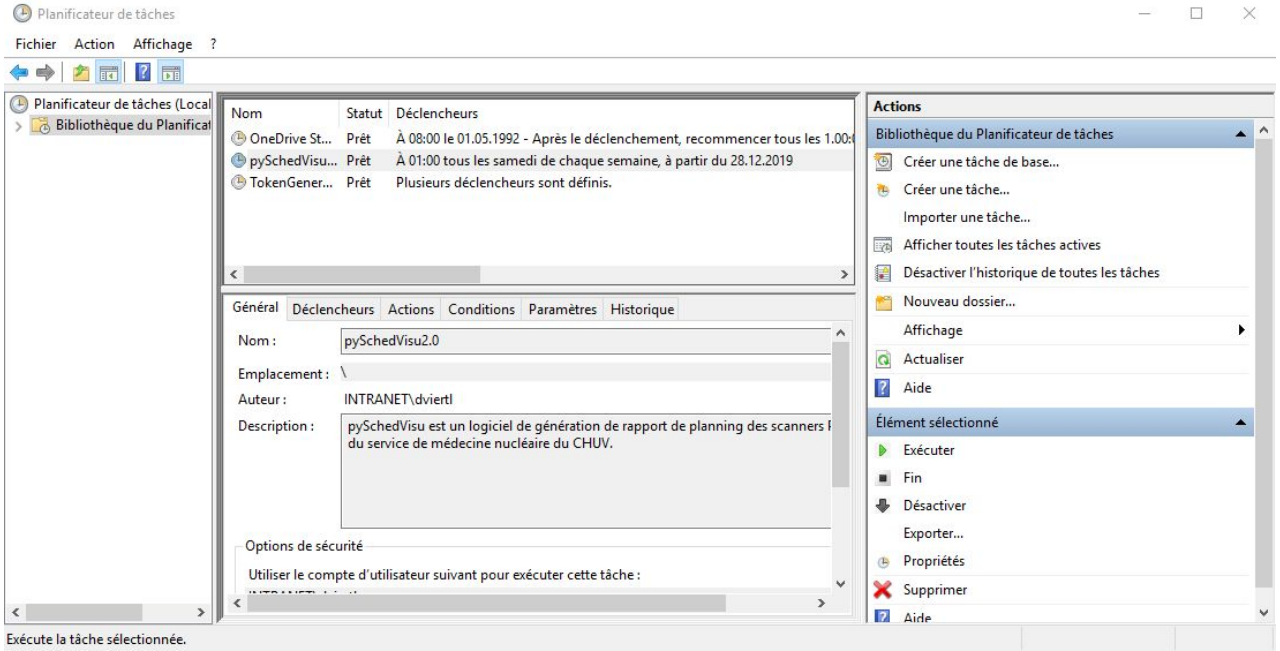


1) Commencez par ouvrir le planificateur de tâche Windows, depuis la session de l'utilisateur qui sera "propriétaire" de la tâche.



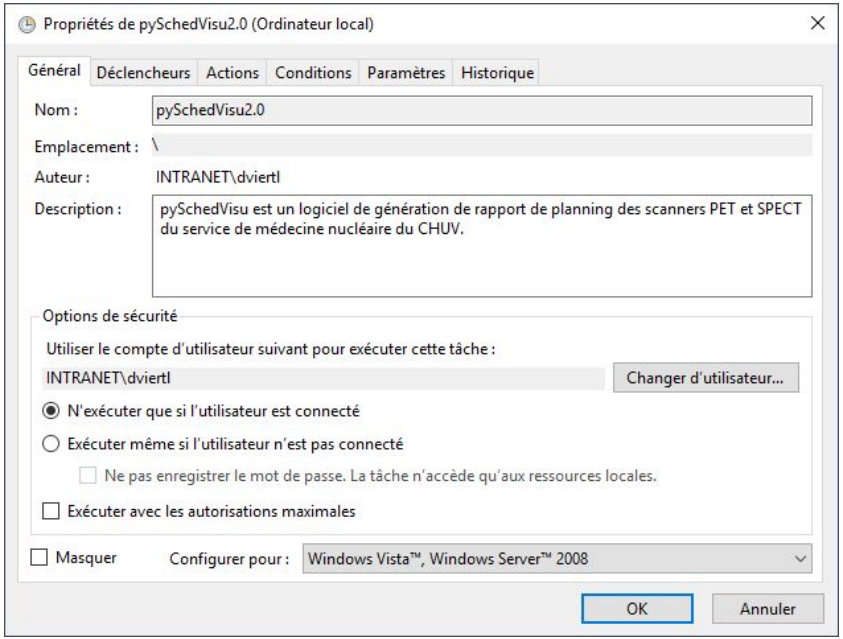
Comment faire pour exécuter pySchedVisu de manière récurrente? (2)

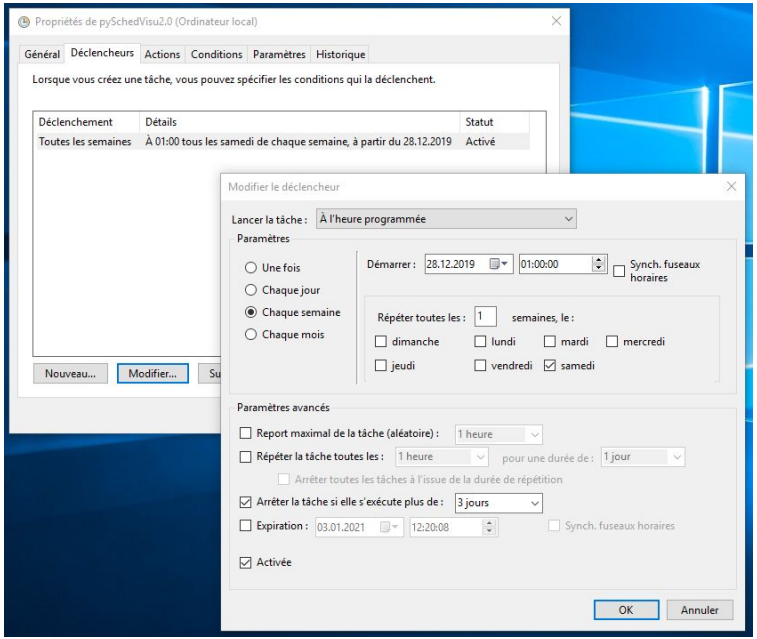
2) Sélectionnez la tâche existante pySchedVisu dans la bibliothèque des tâches, ou créez une nouvelle tâche si elle n'existe pas encore.



Comment faire pour exécuter pySchedVisu de manière récurrente? (3)

3) Modifiez les paramètres dans les différents onglets comme ci-dessous





Comment faire pour exécuter pySchedVisu de manière récurrente? (4)

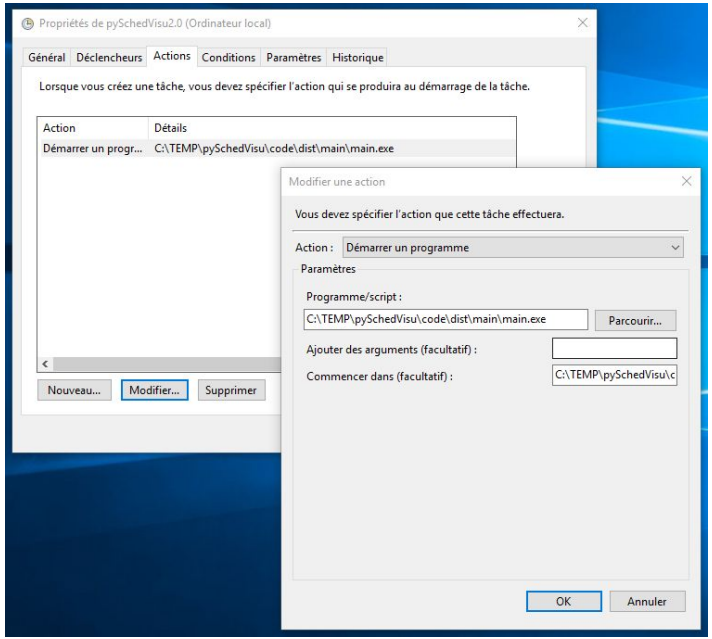
4) Modifiez les paramètres dans l'onglet "Actions" comme ci-dessous.

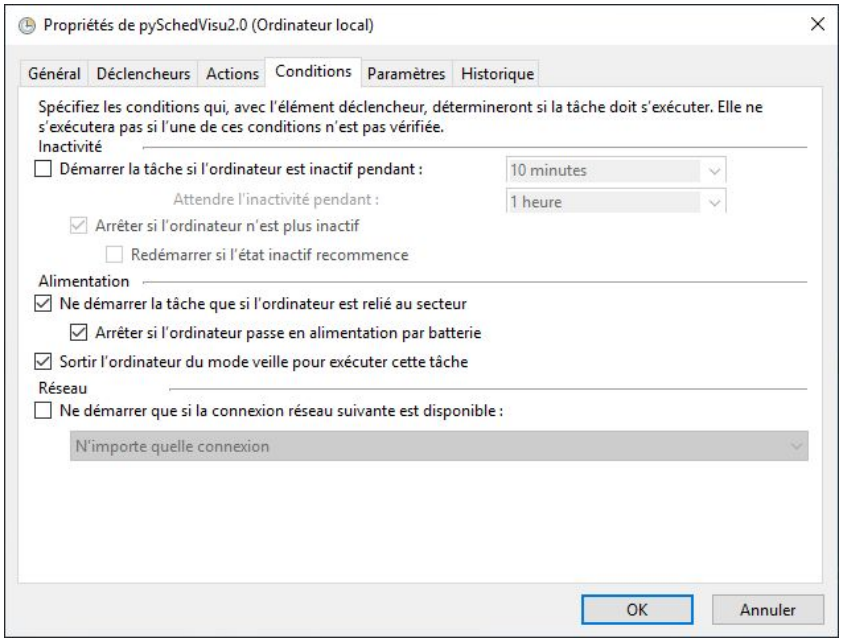
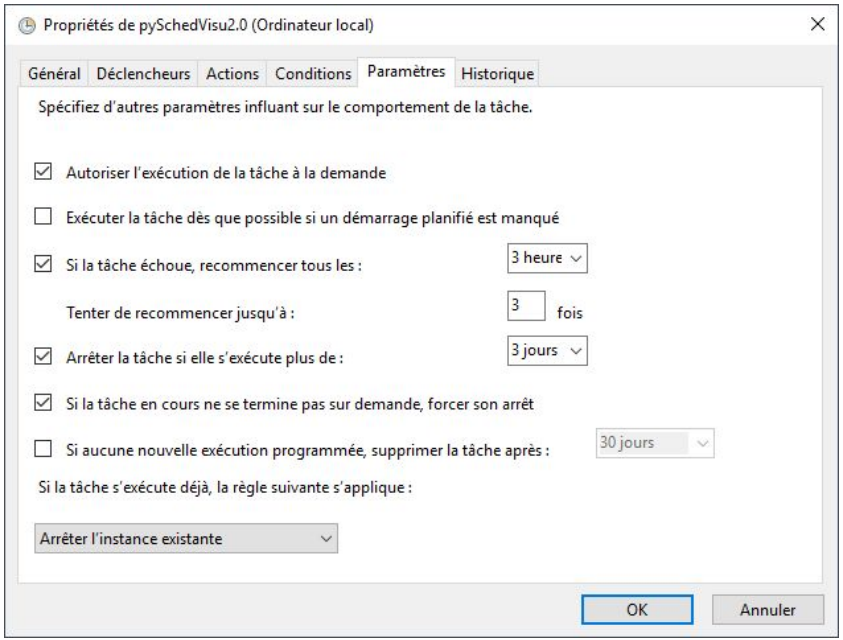
Le champ "*Programme/script*" doit pointer sur l'exécutable pySchedVisu:

```
C:\TEMP\pySchedVisu\code\dist\main\main.exe
```

Le champ "*Commencer dans*" doit pointer sur le dossier du code:

```
C:\TEMP\pySchedVisu\code
```



Comment faire pour exécuter pySchedVisu de manière récurrente? (5)

5) Modifiez les paramètres dans les différents onglets comme ci-dessous