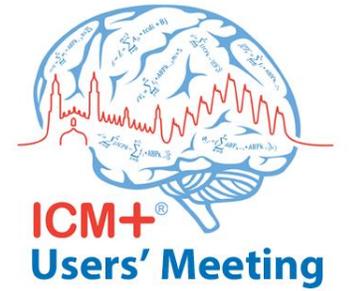




UNIVERSITY OF
CAMBRIDGE



Extending ICM+ with Python scripts with examples using CENTER-TBI data sets

Michał M. Placek

mp963@cam.ac.uk

8th September 2019

Division of Neurosurgery, Department of Clinical Neurosciences

Brain Physics Lab



Python for ICM+ installation

Name	Date modified	Type	Size
 python_env_setup.exe	2019-09-04 18:06	Application	38 018 KB

File description: ICM+ Python Installer Setup
Company: University of Cambridge
File version: 0.0.0.0
Date created: 2019-09-04 18:05
Size: 37.1 MB

Setup - ICM+ Python Installer

Information
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

This Installer will start Python 3.7.4 installation.

Please make sure you tick 'Add Python to PATH'. !

Once python is installed a little batch script will run to download 3 indispensable maths packages: **numpy**, **scipy** and **statsmodels**.

Please NOTE that if you change the default installation location of Python you will need to run the script manually after the setup is done.

Next > Cancel

Python for ICM+ installation

Name	Date modified	Type	Size
python_env_setup.exe	2019-09-04 18:06	Application	38 018 KB

File description: ICM+ Python Installer Setup
Company: University of Cambridge
File version: 0.0.0.0
Date created: 2019-09-04 18:05
Size: 37.1 MB

Setup - ICM+ Python Installer

Information
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

This Installer will start Python 3.7.4 installation.

Please make sure you tick 'Add Python to PATH'. !

Once python is installed a little batch script will run to download 3 indispensable maths packages: **numpy**, **scipy** and **stattools**.

Please NOTE that if you change the default installation location of Python you will need to run the script manually after the setup is done.

Next > Cancel

Python 3.7.4 (32-bit) Setup

Install Python 3.7.4 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\micha\AppData\Local\Programs\Python\Python37-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ **Customize installation**
Choose location and features

Install launcher for all users (recommended)
 Add Python 3.7 to PATH

Cancel

↑ Tick these options!

Python for ICM+ installation

Name	Date modified	Type	Size
python_env_setup.exe	2019-09-04 18:06	Application	38 018 KB

File description: ICM+ Python Installer Setup
Company: University of Cambridge
File version: 0.0.0.0
Date created: 2019-09-04 18:05
Size: 37.1 MB

Setup - ICM+ Python Installer

Information
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

This Installer will start Python 3.7.4 installation.

Please make sure you tick 'Add Python to PATH'. !

Once python is installed a little batch script will run to download 3 indispensable maths packages: **numpy**, **scipy** and **stattools**.

Please NOTE that if you change the default installation location of Python you will need to run the script manually after the setup is done.

Next > Cancel

Python 3.7.4 (32-bit) Setup

Install Python 3.7.4 (32-bit)

Select **Install Now** to install Python with default settings, or choose **Customize** to enable or disable features.

Install Now  **Then, click 'Install Now.'**

C:\Users\micha\AppData\Local\Programs\Python\Python37-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

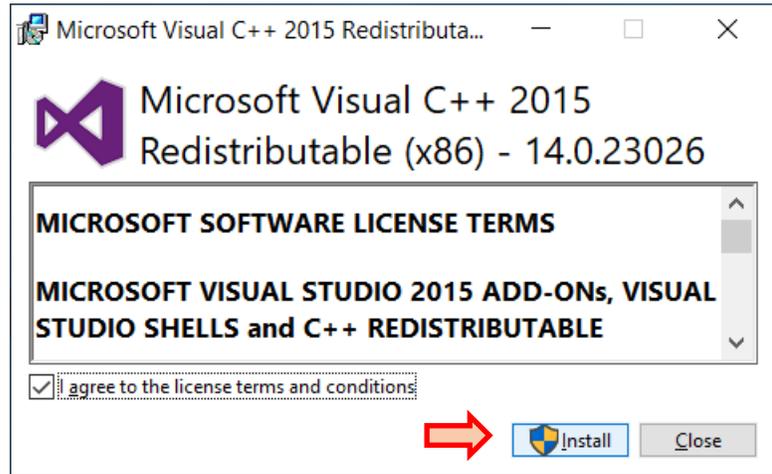
→ **Customize installation**
Choose location and features

Install launcher for all users (recommended)
 Add Python 3.7 to PATH

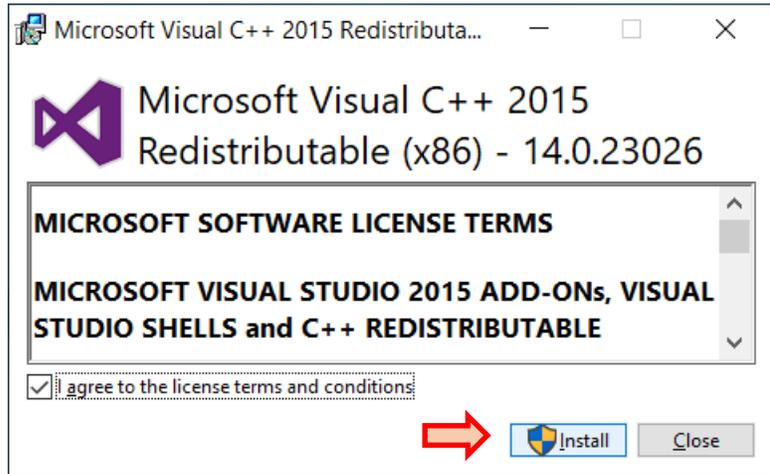
Cancel

 **Tick these options!**

Python for ICM+ installation: cont'd

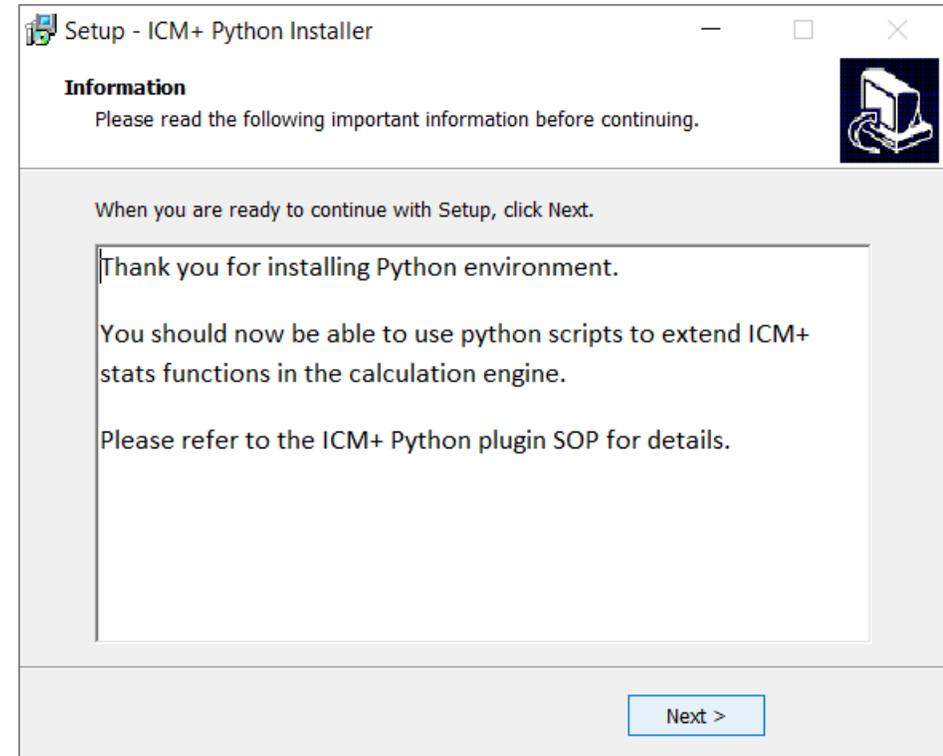
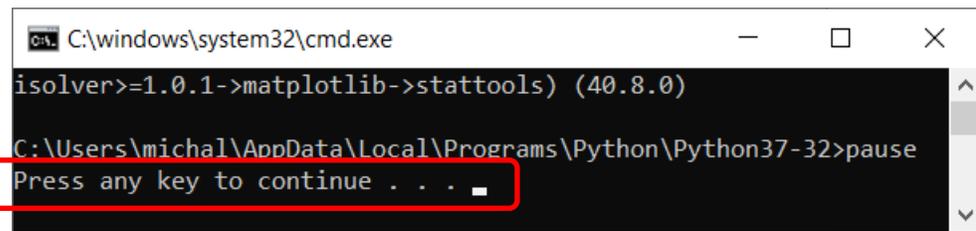
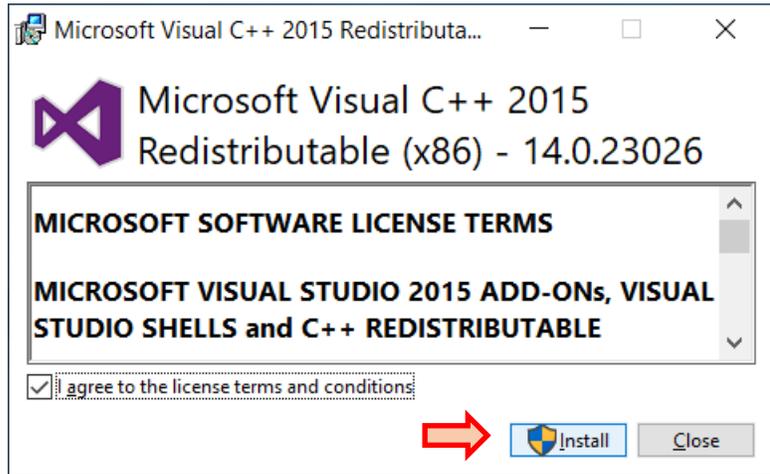


Python for ICM+ installation: cont'd



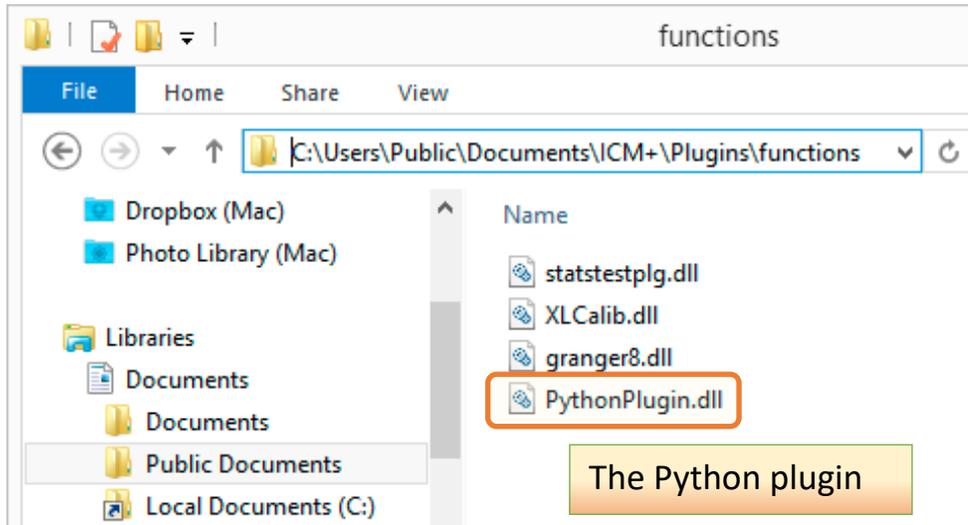
```
C:\windows\system32\cmd.exe
isolver>=1.0.1->matplotlib->stattools) (40.8.0)
C:\Users\michal\AppData\Local\Programs\Python\Python37-32>pause
Press any key to continue . . .
```

Python for ICM+ installation: cont'd



Python Plugin important directories

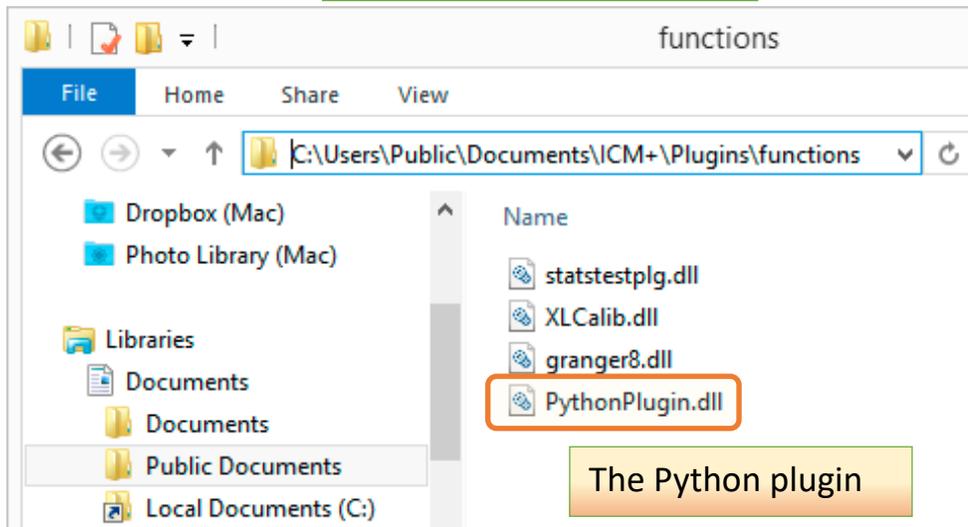
The ICM+ plugins folder



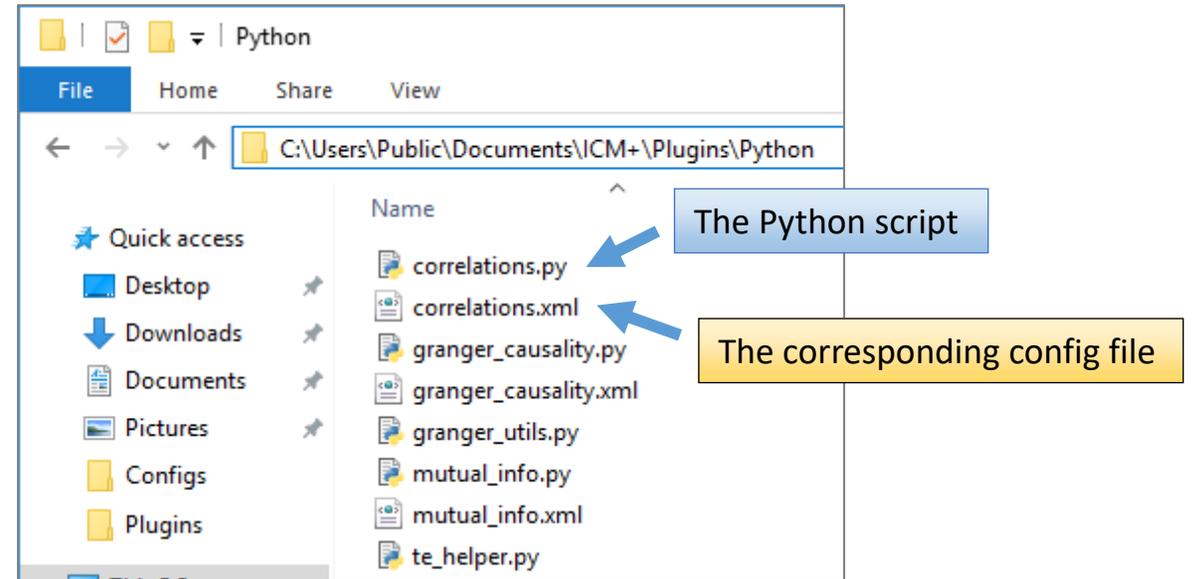
The Python plugin

Python Plugin important directories

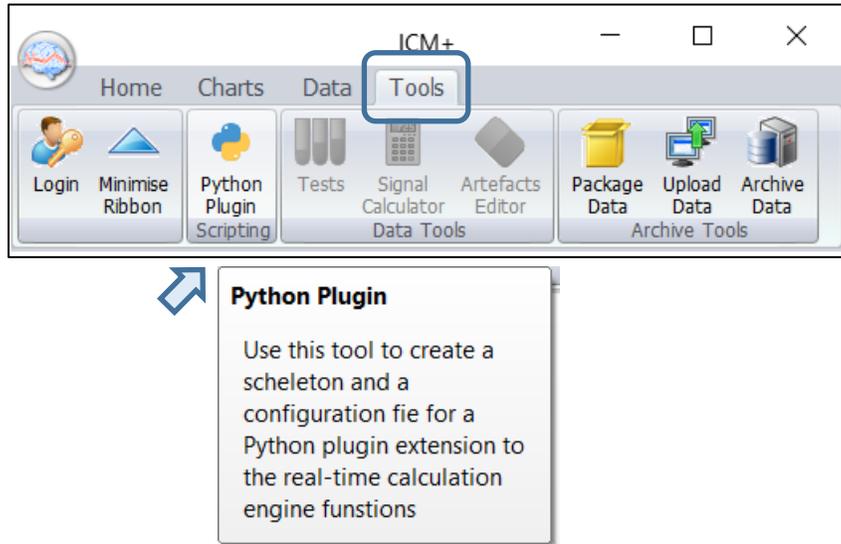
The ICM+ plugins folder



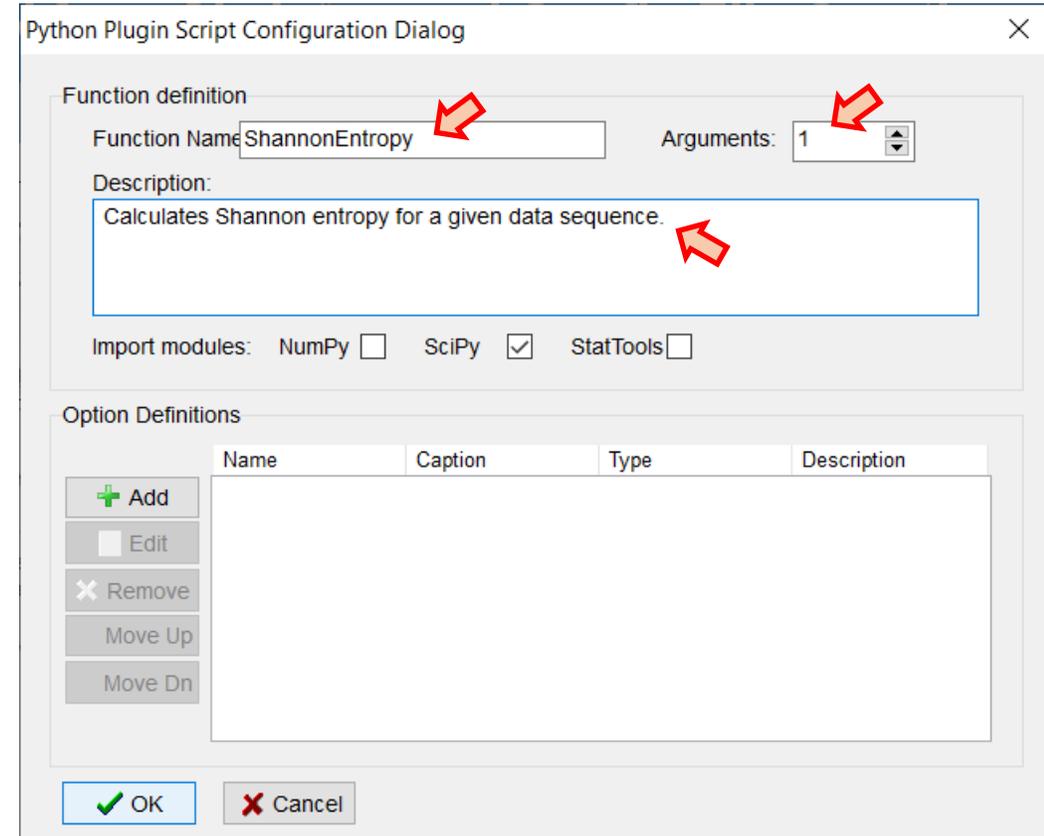
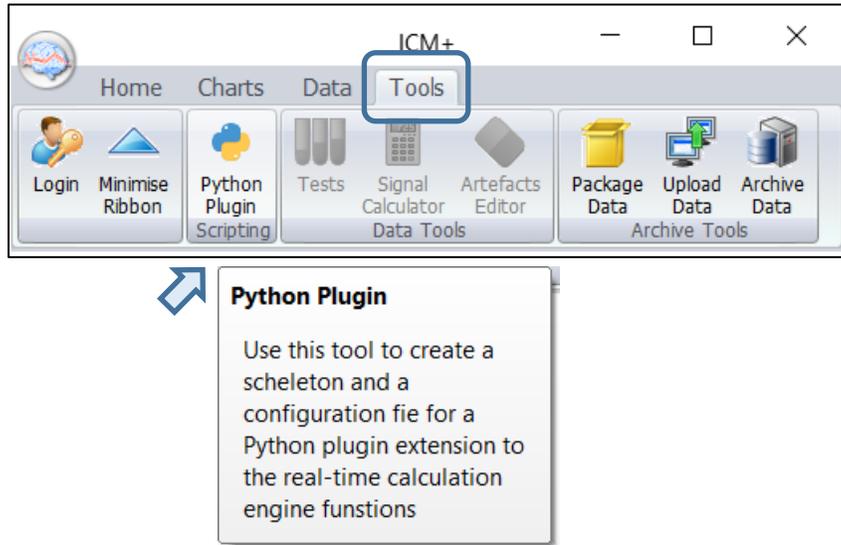
The ICM+ Python plugin folder



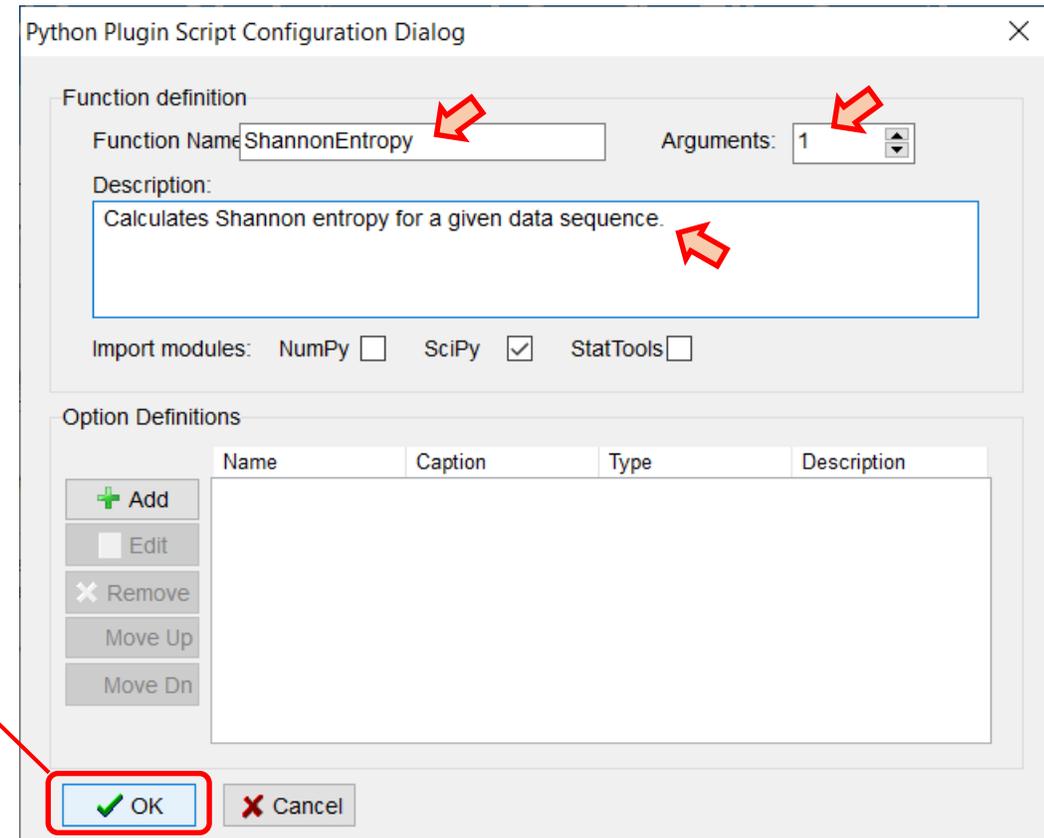
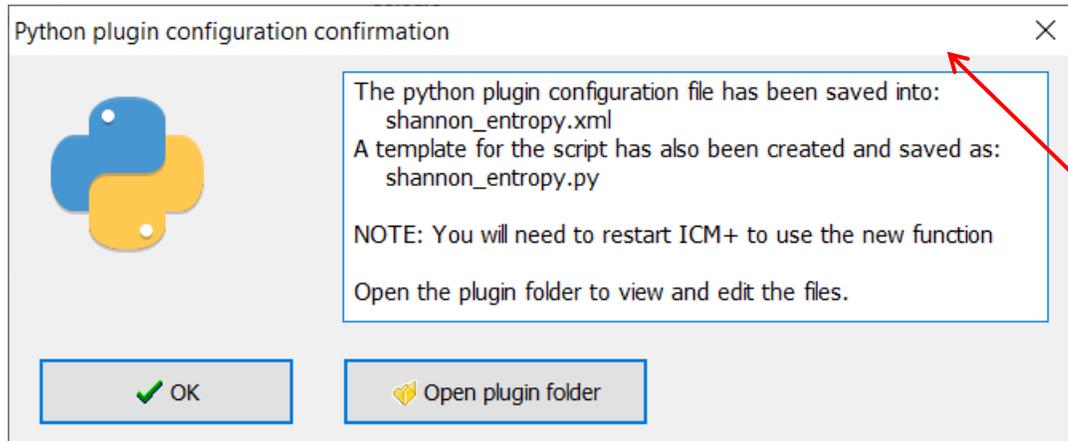
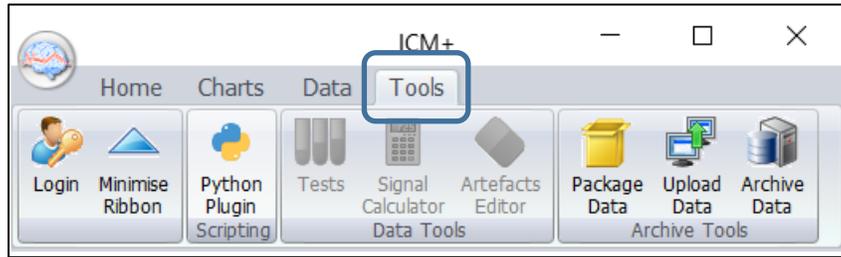
The ICM+ Tool for a Python template and its config file creation



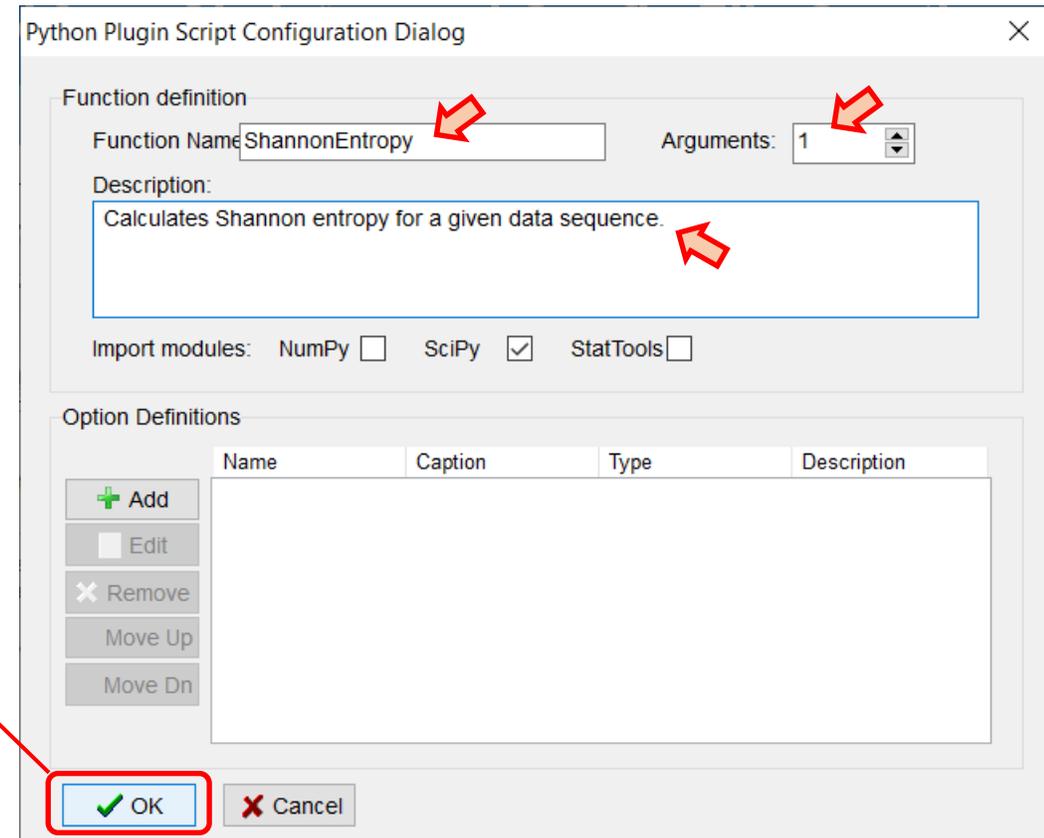
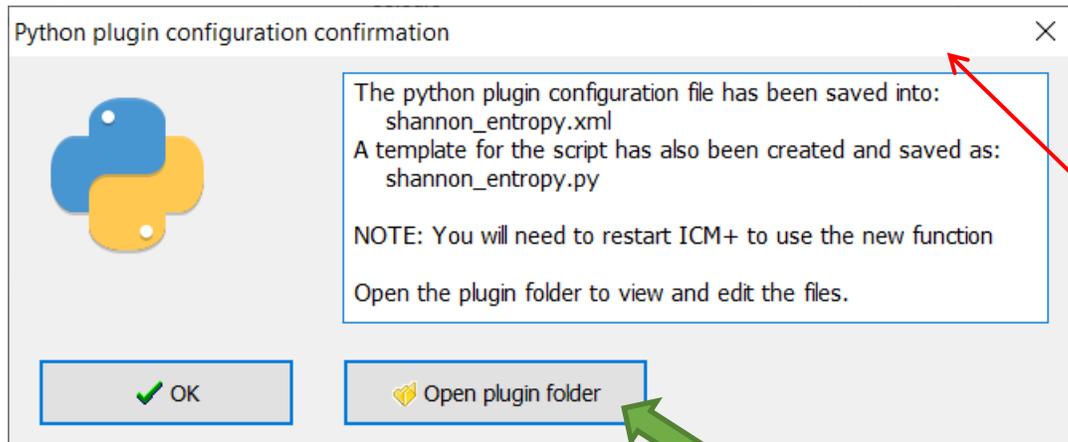
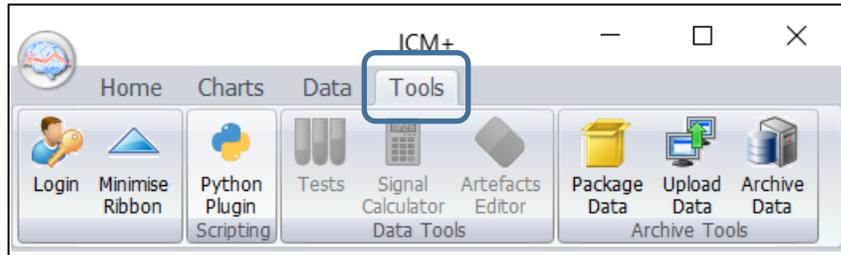
The ICM+ Tool for a Python template and its config file creation



The ICM+ Tool for a Python template and its config file creation



The ICM+ Tool for a Python template and its config file creation



Generated Configuration File

The XML config file generated by ICM+

```
1 <?xml version = "1.0"?>
2
3 <PyToICMPlusConfig>
4   <Function Name="ShannonEntropy" Type="Stats" SignalsCount="1">
5     <GUID>{7A097741-6044-4828-8371-B4BC3E6A1BFE}</GUID>
6     <Description>Calculates Shannon entropy for a given data sequence.</Description>
7   </Function>
8 </PyToICMPlusConfig>
```

Python Plugin Script Configuration Dialog

Function definition

Function Name: ShannonEntropy Arguments: 1

Description:
Calculates Shannon entropy for a given data sequence.

Import modules: NumPy SciPy StatTools

The generated Python script

```
#import ...
import scipy as sp
class ShannonEntropy:

    # DO NOT MODIFY THIS METHOD. It is a part of the ICM+--Python interface.
    def set_parameter(self, param_name, param_value):
        setattr(self, param_name, param_value)

    # You can append your own code to the constructor, if needed.
    # You should not set here values of parameters declared in your XML
    # configuration file because ICM+ will do it for you.
    # You will have to add your own code, only if you need to initialise some
    # extra data structures which were not declared in the XML config file.
    def __init__(self):
        self.sampling_freq = None

    # You can append your own code to the destructor but most likely
    # you will not need it.
    def __del__(self):
        pass

    # 'calculate' is the main work-horse function.
    # It is called with a data buffer (one or more) of size corresponding to the Calculation Window
    # It must return one floating-point number
    # It take the following parameters:
    # sig1 - input variable/signal 1
    # ts_time - part of the data time stamp - number of milliseconds since midnight
    # ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
    # It can also use the data sampling frequency:
    #     self.sampling_freq
    def calculate(self, sig1, ts_time, ts_date):
        # my_own_code_here
        result = 0.0
        return result
```



The generated Python script

```
#import ...
import scipy as sp
class ShannonEntropy:

    # DO NOT MODIFY THIS METHOD. It is a part of the ICM+--Python interface.
    def set_parameter(self, param_name, param_value):
        setattr(self, param_name, param_value)

    # You can append your own code to the constructor, if needed.
    # You should not set here values of parameters declared in your XML
    # configuration file because ICM+ will do it for you.
    # You will have to add your own code, only if you need to initialise some
    # extra data structures which were not declared in the XML config file.
    def __init__(self):
        self.sampling_freq = None

    # You can append your own code to the destructor but most likely
    # you will not need it.
    def __del__(self):
        pass

    # 'calculate' is the main work-horse function.
    # It is called with a data buffer (one or more) of size corresponding to the Calculation Window
    # It must return one floating-point number
    # It take the following arguments:
    # sig1 - input signal
    # ts_time - part of the data time stamp - number of milliseconds since midnight
    # ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
    # It can also use the data sampling frequency:
    # self.sampling_freq
    def calculate(self, sig1, ts_time, ts_date):
        # my_own_code_here
        result = 0.0
        return result
```

One input signal

The generated Python script

```
#import ...
import scipy as sp
class ShannonEntropy:

    # DO NOT MODIFY THIS METHOD. It is a part of the ICM+--Python interface.
    def set_parameter(self, param_name, param_value):
        setattr(self, param_name, param_value)

    # You can append your own code to the constructor, if needed.
    # You should not set here values of parameters declared in your XML
    # configuration file because ICM+ will do it for you.
    # You will have to add your own code, only if you need to initialise some
    # extra data structures which were not declared in the XML config file.
    def __init__(self):
        self.sampling_freq = None

    # You can append your own code to the destructor but most likely
    # you will not need it.
    def __del__(self):
        pass

    # 'calculate' is the main work-horse function.
    # It is called with a data buffer (one or more) of size corresponding to the Calculation Window
    # It must return one floating-point number
    # It take the following arguments:
    # sig1 - input signal
    # ts_time - part of the data time stamp - number of milliseconds since midnight
    # ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
    # It can also use the data sampling frequency:
    # self.sampling_freq
    def calculate(self, sig1, ts_time, ts_date):
        # my_own_code_here
        result = 0.0
        return result
```

One input signal

Add your own code to the *calculate* method



The generated Python script

```
#import ...
import scipy as sp
class ShannonEntropy:

    # DO NOT MODIFY THIS METHOD. It is a part of the ICM+--Python interface.
    def set_parameter(self, param_name, param_value):
        setattr(self, param_name, param_value)

    # You can append your own code to the constructor, if needed.
    # You should not set here values of parameters declared in your XML
    # configuration file because ICM+ will do it for you.
    # You will have to add your own code, only if you need to initialise some
    # extra data structures which were not declared in the XML config file.
    def __init__(self):
        self.sampling_freq = None

    # You can append your own code to the destructor but most likely
    # you will not need it.
    def __del__(self):
        pass

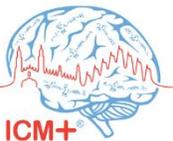
    # 'calculate' is the main work-horse function.
    # It is called with a data buffer (one or more) of size corresponding to the Calculation Window
    # It must return one floating-point number
    # It take the following arguments:
    # sig1 - input signal
    # ts_time - part of the data time stamp - number of milliseconds since midnight
    # ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
    # It can also use the data sampling frequency:
    # self.sampling_freq
    def calculate(self, sig1, ts_time, ts_date):
        # my_own_code_here
        result = 0.0
        return result
```

One input signal

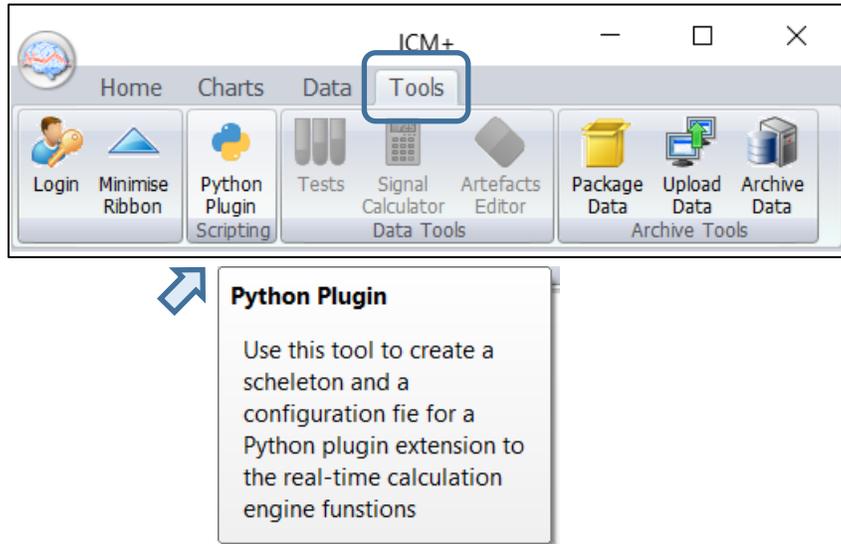
Add your own code to the *calculate* method

```
from scipy import stats
```

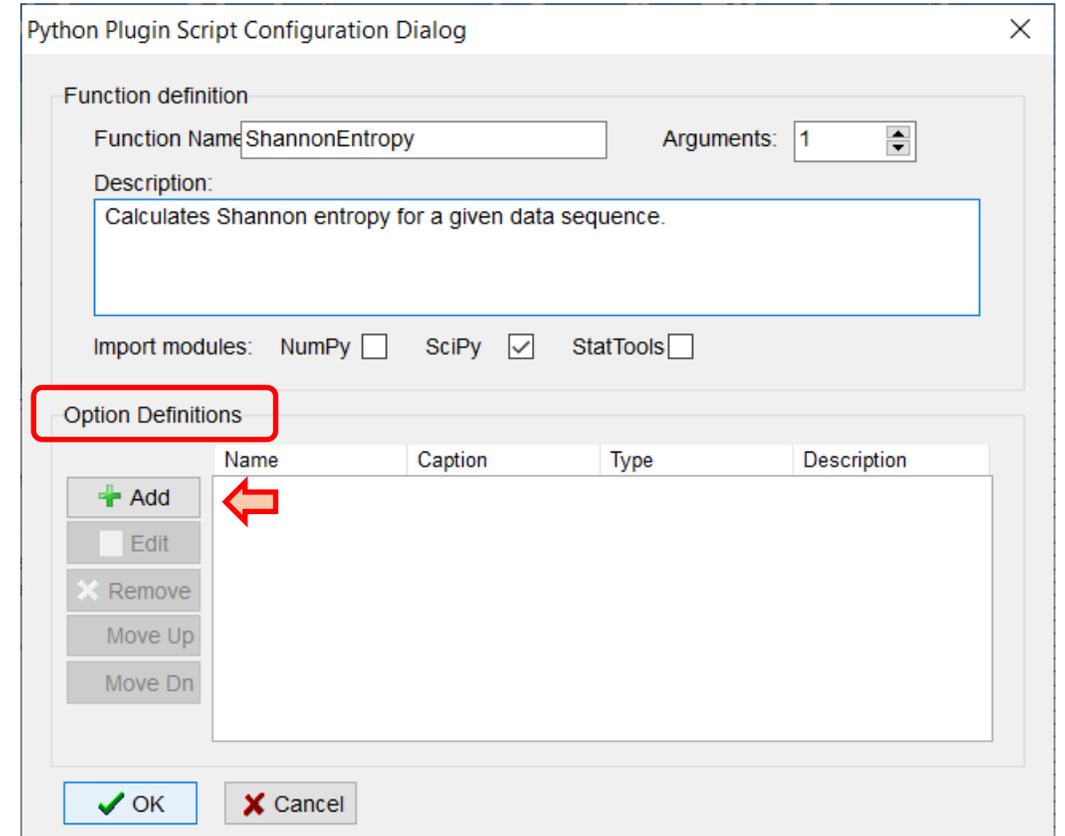
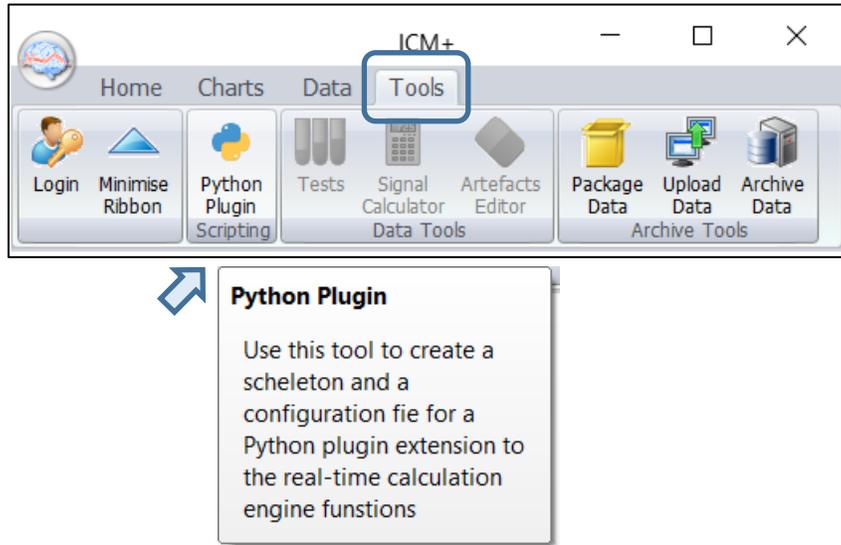
```
def calculate(self, sig1, ts_time, ts_date):
    result = stats.entropy(sig1)
    return result
```



Adding options to the user-defined Python function



Adding options to the user-defined Python function



Adding options to the user-defined Python function

Data Field Definition Form

Name: Caption:

Description:

Type: **Flag (Y/N)** Is Mandatory

Categories: **Flag (Y/N)**
Category (selection)
Numerical Value
Integer Value (rank)

Min - Max: 0 0

Default: 0

Python Plugin Script Configuration Dialog

Function definition

Function Name: Arguments:

Description:

Import modules: NumPy SciPy StatTools

Option Definitions

Name	Caption	Type	Description
------	---------	------	-------------

Adding an option of the type 'flag'

Data Field Definition Form

Name: zeroOffset Caption: Zero offset

Description: Subtract current minimal value to zero offset

Type: Flag (Y/N) Is Mandatory

Categories:

Value	Caption

Min - Max: 0 0

Default: False 0

Adding an option of the type 'flag'

Data Field Definition Form

Name: zeroOffset Caption: Zero offset

Description: Subtract current minimal value to zero offset

Type: Flag (Y/N) Is Mandatory

Value	Caption

Min - Max: 0 0

Default: False 0

Adding an option of the type 'flag'

Data Field Definition Form

Name: zeroOffset Caption: Zero offset

Description: Subtract current minimal value to zero offset

Type: Flag (Y/N) Is Mandatory

Value	Caption
-------	---------

Min - Max: 0 0

Default: False 0

OK Cancel Keyboard

Python Plugin Script Configuration Dialog

Function definition

Function Name: ShannonEntropy Arguments: 1

Description: Calculates Shannon entropy for a given data sequence.

Import modules: NumPy SciPy StatTools

Option Definitions

Name	Caption	Type	Description
zeroOffset	Zero offset	Flag	Subtract current minimal value

+ Add Edit Remove Move Up Move Dn

OK Cancel

Adding an option of the type 'category'

Data Field Definition Form

Name: Caption:

Description:

Type: Is Mandatory

Categories:

Value	Caption
BIN	binary
NAT	natural
DEC	decimal

Min - Max:

Default:

Adding an option of the type 'category'

Data Field Definition Form

Name: Caption:

Description:

Type: Is Mandatory

Categories:

Value	Caption
BIN	binary
NAT	natural
DEC	decimal

Min - Max:

Default:

Adding an option of the type 'category'

Data Field Definition Form

Name: Caption:

Description:

Type: Is Mandatory

Categories:

Value	Caption
BIN	binary
NAT	natural
DEC	decimal

Min - Max:

Default:

Python Plugin Script Configuration Dialog

Function definition

Function Name: Arguments:

Description:

Import modules: NumPy SciPy StatTools

Option Definitions

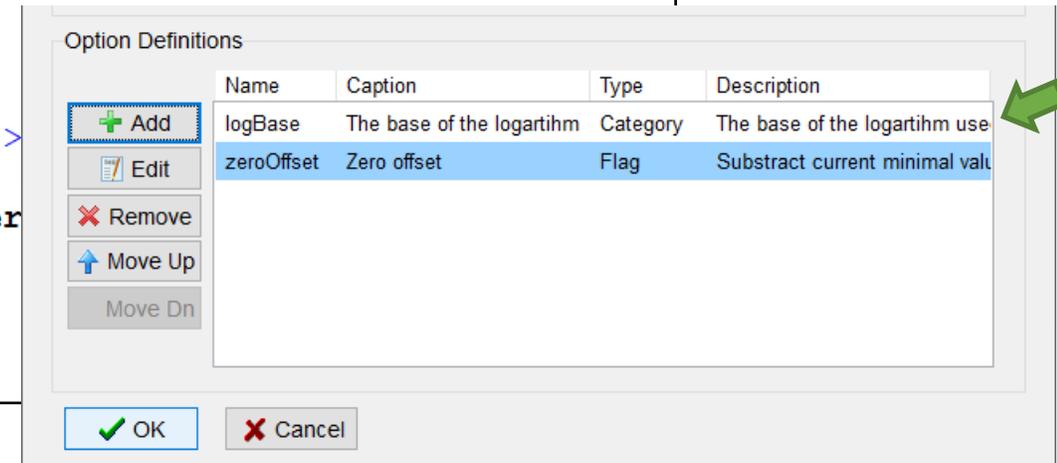
Name	Caption	Type	Description
logBase	The base of the logarithm	Category	The base of the logarithm use
zeroOffset	Zero offset	Flag	Substract current minimal valu

Generated Configuration File

```
1 <?xml version = "1.0"?>
2
3 <PyToICMPlusConfig>
4   <Function Name="ShannonEntropy" Type="Stats" SignalsCount="1">
5     <GUID>{3DE497F8-5AF6-40D5-907E-02B2CCDF19C5}</GUID>
6     <Description>Calculates Shannon entropy for a given data sequence.</Description>
7     <Parameter ShortName="logBase" IsMandatory="False">
8       <Caption>The base of the logartihm</Caption>
9       <Description>The base of the logartihm used to calculate the entropy</Description>
10      <Type Name="StringList">
11        <Item Value="BIN" Caption="binary" IsDefault="True"/>
12        <Item Value="NAT" Caption="natural"/>
13        <Item Value="DEC" Caption="decimal"/>
14      </Type>
15    </Parameter>
16    <Parameter ShortName="zeroOffset" IsMandatory="False">
17      <Caption>Zero offset</Caption>
18      <Description>Substract current minimal value to zero offset</Description>
19      <Type Name="Bool" DefaultValue="False"/>
20    </Parameter>
21  </Function>
22 </PyToICMPlusConfig>
```

Generated Configuration File

```
1 <?xml version = "1.0"?>
2
3 <PyToICMPlusConfig>
4   <Function Name="ShannonEntropy" Type="Stats" SignalsCount="1">
5     <GUID>{3DE497F8-5AF6-40D5-907E-02B2CCDF19C5}</GUID>
6     <Description>Calculates Shannon entropy for a given data sequence.</Description>
7     <Parameter ShortName="logBase" IsMandatory="False">
8       <Caption>The base of the logartihm</Caption>
9       <Description>The base of the logartihm used to calculate the entropy</Description>
10      <Type Name="StringList">
11        <Item Value="BIN" Caption="binary" IsDefault="True"/>
12        <Item Value="NAT" Caption="natural"/>
13        <Item Value="DEC" Caption="decimal"/>
14      </Type>
15    </Parameter>
16    <Parameter ShortName="zeroOffset" IsMandatory="False">
17      <Caption>Zero offset</Caption>
18      <Description>Substract current minimal value to zero</Description>
19      <Type Name="Bool" DefaultValue="False"/>
20    </Parameter>
21  </Function>
22 </PyToICMPlusConfig>
```



Generated Configuration File

```
1 <?xml version = "1.0"?>
2
3 <PyToICMPlusConfig>
4   <Function Name="ShannonEntropy" Type="Stats" SignalsCount="1">
5     <GUID>{3DE497F8-5AF6-40D5-907E-02B2CCDF19C5}</GUID>
6     <Description>Calculates Shannon entropy for a given data sequence.</Description>
7     <Parameter ShortName="logBase" IsMandatory="False">
8       <Caption>The base of the logartihm</Caption>
9       <Description>The base of the logartihm used to calculate the entropy</Description>
10      <Type Name="StringList">
11        <Item Value="BIN" Caption="binary" IsDefault="True"/>
12        <Item Value="NAT" Caption="natural"/>
13        <Item Value="DEC" Caption="decimal"/>
14      </Type>
15    </Parameter>
16    <Parameter ShortName="zeroOffset" IsMandatory="False">
17      <Caption>Zero offset</Caption>
18      <Description>Substract current minimal value to zero</Description>
19      <Type Name="Bool" DefaultValue="False"/>
20    </Parameter>
21  </Function>
22 </PyToICMPlusConfig>
```

Option Definitions

Name	Caption	Type	Description
logBase	The base of the logartihm	Category	The base of the logartihm use
zeroOffset	Zero offset	Flag	Substract current minimal val

+ Add
Edit
Remove
Move Up
Move Dn

OK Cancel

How to use the configured options in the Python script

```
# 'calculate' is the main work-horse function.
# It is called with a data buffer (one or more) of size corresponding to the Calculation Window
# It must return one floating-point number
# It take the following parameters:
# sig1 - input variable/signal 1
# ts_time - part of the data time stamp - number of milliseconds since midnight
# ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
# It can also use the data sampling frequency:
#     self.sampling_freq
# and the following variables already set at the initialisation time (via function options):
#     self.logBase - The base of the logartihm
#     self.zeroOffset - Substract current minimal value to zero offset

def calculate(self, sig1, ts_time, ts_date):

    if self.zeroOffset == True:
        sig1 = np.array(sig1) - min(sig1)

    if self.logBase == 'BIN':
        base = 2
    elif self.logBase == 'NAT':
        base = math.e
    elif self.logBase == 'DEC':
        base = 10

    result = stats.entropy(sig1, None, base)
    return result
```

How to use the configured options in the Python script

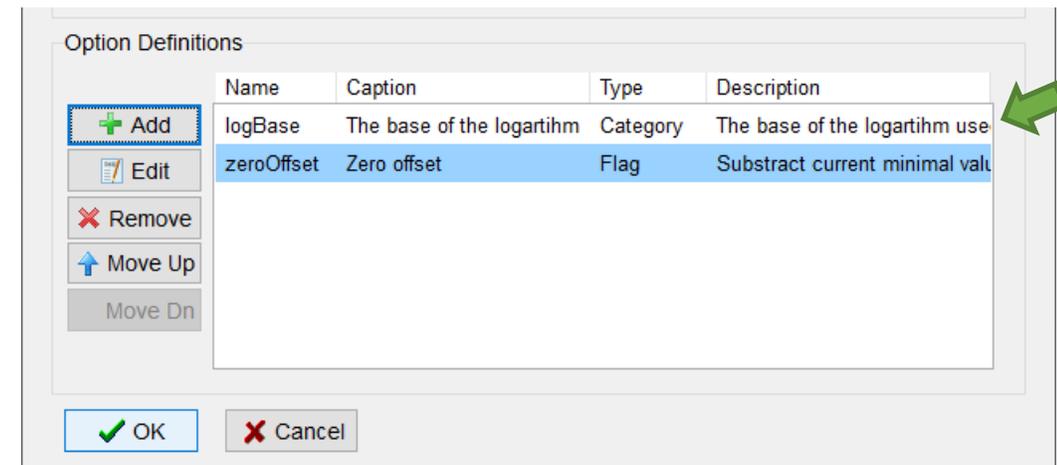
```
# 'calculate' is the main work-horse function.
# It is called with a data buffer (one or more) of size corresponding to the Calculation Window
# It must return one floating-point number
# It take the following parameters:
# sig1 - input variable/signal 1
# ts_time - part of the data time stamp - number of milliseconds since midnight
# ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
# It can also use the data sampling frequency:
# self.sampling_freq
# and the following variables already set at the initialisation time (via function options):
# self.logBase - The base of the logartihm
# self.zeroOffset - Substract current minimal value to zero offset

def calculate(self, sig1, ts_time, ts_date):

    if self.zeroOffset == True:
        sig1 = np.array(sig1) - min(sig1)

    if self.logBase == 'BIN':
        base = 2
    elif self.logBase == 'NAT':
        base = math.e
    elif self.logBase == 'DEC':
        base = 10

    result = stats.entropy(sig1, None, base)
    return result
```



How to use the configured options in the Python script

```
# 'calculate' is the main work-horse function.
# It is called with a data buffer (one or more) of size corresponding to the Calculation
# It must return one floating-point number
# It take the following parameters:
# sig1 - input variable/signal 1
# ts_time - part of the data time stamp - number of milliseconds since midnight
# ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
# It can also use the data sampling frequency:
# self.sampling_freq
# and the following variables already set at the initialisation time (via function optio
# self.logBase - The base of the logartihm
# self.zeroOffset - Substract current minimal value to zero offset
```

```
def calculate(self, sig1, ts_time, ts_date):
```

```
    if self.zeroOffset == True:
        sig1 = np.array(sig1) - min(sig1)
```

```
    if self.logBase == 'BIN':
        base = 2
```

```
    elif self.logBase == 'NAT':
        base = math.e
```

```
    elif self.logBase == 'DEC':
        base = 10
```

```
    result = stats.entropy(sig1, None, base)
    return result
```

Data Field Definition Form

Name: Caption:

Description:

Type: Is Mandatory

Categories:

Value	Caption
BIN	binary
NAT	natural
DEC	decimal

Option Definitions

Name	Caption	Type	Description
logBase	The base of the logartihm	Category	The base of the logartihm use
zeroOffset	Zero offset	Flag	Substract current minimal val

+ Add
Edit
X Remove
↑ Move Up
Move Dn

OK Cancel

How to use the configured options in the Python script

```
# 'calculate' is the main work-horse function.
# It is called with a data buffer (one or more) of size corresponding to the Calculation
# It must return one floating-point number
# It take the following parameters:
# sig1 - input variable/signal 1
# ts_time - part of the data time stamp - number of milliseconds since midnight
# ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
# It can also use the data sampling frequency:
# self.sampling_freq
# and the following variables already set at the initialisation time (via function optio
# self.logBase - The base of the logartihm
# self.zeroOffset - Substract current minimal value to zero offset
```

```
def calculate(self, sig1, ts_time, ts_date):
```

```
    if self.zeroOffset == True:
        sig1 = np.array(sig1) - min(sig1)
```

```
    if self.logBase == 'BIN':
        base = 2
```

```
    elif self.logBase == 'NAT':
        base = math.e
```

```
    elif self.logBase == 'DEC':
        base = 10
```

```
    result = stats.entropy(sig1, None, base)
    return result
```

Data Field Definition Form

Name: Caption:

Description:

Type: Is Mandatory

Categories:

Value	Caption
BIN	binary
NAT	natural
DEC	decimal

Option Definitions

Name	Caption	Type	Description
logBase	The base of the logartihm	Category	The base of the logartihm use
zeroOffset	Zero offset	Flag	Substract current minimal val

+ Add Edit Remove Move Up Move Dn

OK Cancel

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Virtual Signals Primary Analysis Secondary Analysis 1 Final Analysis

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

Modify Add Delete Clear Auto Fill Default Period [s]: 10.0

OK Cancel Save Load Advanced Keyboard

Using user-defined Python function in ICM+

The screenshot shows the 'On Line Analysis Configuration Dialog' with the 'Primary Analysis' tab selected. A table lists analysis configurations for 'ABP', 'FV', and 'etCO2'. The 'Add' button is highlighted with a red box, and a red arrow points from it to the 'Primary Analysis Configuration Editor' on the right.

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

The 'Primary Analysis Configuration Editor' shows settings for an analysis named 'Entr'. The 'Calculation Window Specification' section includes 'Calculation Period' (300 s) and 'Update Period' (60 s). The 'Valid values range' section has 'Max Value' and 'Min Value' both set to 0. The 'Formula' section is empty. The 'Function' list includes 'PyShannonEntropy', which is selected. The 'Arguments' field contains 'ABP'. The 'Options' section includes 'logBase' and 'zeroOffset'. The 'Inputs' list includes 'ABP', 'etCO2', and 'FV'. The 'Function description' is 'Calculates Shannon entropy for a given data sequence.'

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

Buttons: Modify, Add, Delete, Clear, Auto Fill, Default Period [s]: 10.0, OK, Cancel, Save, Load, Advanced, Keyboard

Primary Analysis Configuration Editor

Name : Entr

Enabled

Calculation Window Specification

Calculation Period : 300 s

Update Period : 60 s

Valid values range

Max Value : 0

Min Value : 0

Formula:

abs Insert Function Arguments : ABP

Function :

- PyPartialCorrel
- PyShannonEntropy
- PySpearmanCorrel
- Range
- RankCorrel
- RelCount

Options:

logBase The base of the logarithm (BIN,NAT,DEC)

zeroOffset Zero offset (Y/N)

Inputs:

- ABP
- etCO2
- FV

Function description:

Calculates Shannon entropy for a given data sequence.

Buttons: OK, Cancel, Keyboard

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

Buttons: Modify, Add, Delete, Clear, Auto Fill, Default Period [s]: 10.0, OK, Cancel, Save, Load, Advanced, Keyboard

Primary Analysis Configuration Editor

Name : Entr

Enabled

Calculation Window Specification

Calculation Period : 300 s

Update Period : 60 s

Valid values range

Max Value : 0

Min Value : 0

Formula:

abs Insert Function Arguments : ABP

Function :

- PyPartialCorrel
- PyShannonEntropy
- PySpearmanCorrel
- Range
- RankCorrel
- RelCount

Options:

logBase The base of the logarithm (BIN,NAT,DEC)

zeroOffset Zero offset (Y/N)

Inputs:

- ABP
- etCO2
- FV

Function description:

Calculates Shannon entropy for a given data sequence.

Buttons: OK, Cancel, Keyboard

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

Buttons: Modify, Add, Delete, Clear, Auto Fill, Default Period [s]: 10.0, OK, Cancel, Save, Load, Advanced, Keyboard

Primary Analysis Configuration Editor

Name: Entr

Enabled:

Calculation Window Specification

Calculation Period: 300 s

Update Period: 60 s

Valid values range

Max Value: 0

Min Value: 0

Formula:

abs

Insert Function

Arguments: ABP

Function:

- PyPartialCorrel
- PyShannonEntropy**
- PySpearmanCorrel
- Range
- RankCorrel
- RelCount

Options:

logBase: The base of the logarithm (BIN, NAT, DEC)

zeroOffset: Zero offset (Y/N)

Inputs:

- ABP
- etCO2
- FV

Function description:

Calculates Shannon entropy for a given data sequence.

Buttons: OK, Cancel, Keyboard

Function options

Function: **PyShannonEntropy**

The base of the logarithm: natural

Zero offset:

Buttons: OK, Cancel, Keyboard

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Virtual Signals Primary Analysis Secondary Analysis 1 Final Analysis

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

Modify Add Delete Clear Auto Fill Default Period [s]: 10.0

OK Cancel Save Load Advanced Keyboard

Function options

Function: **PyShannonEntropy**

The base of the logarithm: natural

Zero offset:

OK Cancel Keyboard

Primary Analysis Configuration Editor

Name : Entr

Calculation Window Specification

Calculation Period : 300 s

Update Period : 60 s

Valid values range

Max Value : 0

Min Value : 0

Enabled

Formula:

abs Insert Function Arguments : ABP

Function : PyPartialCorrel PyShannonEntropy PySpearmanCorrel Range RankCorrel RelCount

Options: logBase The base of the logarithm (BIN,NAT,DEC) zeroOffset Zero offset (Y/N)

Inputs: ABP etCO2 FV

Function description: Calculates Shannon entropy for a given data sequence.

OK Cancel Keyboard

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Virtual Signals Primary Analysis Secondary Analysis 1 Final Analysis

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

Modify Add Delete Clear Auto Fill Default Period [s]: 10.0

OK Cancel Save Load Advanced Keyboard

Function options

Function: **PyShannonEntropy**

The base of the logarithm: natural

Zero offset:

OK Cancel Keyboard

Primary Analysis Configuration Editor

Name: Entr

Calculation Window Specification Valid values range

Enabled

Formula:

abs

7 8 9 +
4 5 6 -
1 2 3 *
0 . /
Delete ()

OK X

Python Plugin Script Configuration Dialog

Function definition

Function Name: ShannonEntropy Arguments: 1

Description:
Calculates Shannon entropy for a given data sequence.

Import modules: NumPy SciPy StatTools

Option Definitions

Name	Caption	Type	Description
logBase	The base of the logarithm	Category	The base of the logarithm use
zeroOffset	Zero offset	Flag	Subtract current minimal val

+ Add Edit Remove Move Up Move Dn

OK Cancel

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Virtual Signals Primary Analysis Secondary Analysis 1 Final Analysis

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

Modify Add Delete Clear Auto Fill Default Period [s]: 10.0

OK Cancel Save Load Advanced Keyboard

Function options

Function: PyShannonEntropy

The base of the logarithm: natural

Zero offset:

OK Cancel Keyboard

Primary Analysis Configuration Editor

Name: Entr

Calculation Window Specification: Calculation Period: 300 s, Update Period: 60 s

Valid values range: Max Value: 0, Min Value: 0

Enabled

Formula:

abs Insert Function Arguments: ABP

Function: PyShannonEntropy

Options: logBase (BIN, NAT, DEC), zeroOffset (Y/N)

Inputs: ABP, etCO2, FV

Function description: Calculates Shannon entropy for a given data sequence.

OK Cancel Keyboard

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Virtual Signals Primary Analysis Secondary Analysis 1 Final Analysis

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y

Modify Add Delete Clear Auto Fill Default Period [s]: 10.0

OK Cancel Save Load Advanced Keyboard

Primary Analysis Configuration Editor

Name : Entr

Calculation Window Specification

Calculation Period : 300 s

Update Period : 60 s

Valid values range

Max Value : 0

Min Value : 0

Enabled

Formula:

PyShannonEntropy(ABP, 'logBase=NAT&zeroOffset')

abs Insert Function Arguments : ABP

Function :

- PyPartialCorrel
- PyShannonEntropy**
- PySpearmanCorrel
- Range
- RankCorrel
- RelCount

Options:

logBase The base of the logarithm (BIN,NAT,DEC)

zeroOffset Zero offset (Y/N)

Inputs:

- ABP
- etCO2
- FV

Function description:

Calculates Shannon entropy for a given data sequence.

OK Cancel Keyboard

Function options

Function: **PyShannonEntropy**

The base of the logarithm natural

Zero offset

OK Cancel Keyboard

Using user-defined Python function in ICM+

On Line Analysis Configuration Dialog

Virtual Signals Primary Analysis Secondary Analysis 1 Final Analysis

Name	Formula	Calc. Window [s]	Updated [s]	Min	Max	En
ABP	Mean(ABP)	300	60	0	0	Y
FV	Mean(FV)	300	60	0	0	Y
etCO2	Mean(etCO2)	300	60	0	0	Y
Entr	PyShannonEntropy(ABP, 'logBase=NAT&zeroOffset')	300	60	0	0	Y

Modify Add Delete Clear Auto Fill Default Period [s]: 60.0

OK Cancel Save Load Advanced Keyboard

Primary Analysis Configuration Editor

Name : Entr

Calculation Window Specification

Calculation Period : 300 s

Update Period : 60 s

Valid values range

Max Value : 0

Min Value : 0

Enabled

Formula:

PyShannonEntropy(ABP, 'logBase=NAT&zeroOffset')

abs Arguments : ABP

Function :

- PyPartialCorrel
- PyShannonEntropy**
- PySpearmanCorrel
- Range
- RankCorrel
- RelCount

Options:

logBase The base of the logarithm (BIN,NAT,DEC)

zeroOffset Zero offset (Y/N)

Inputs:

- ABP
- etCO2
- FV

Function description:

Calculates Shannon entropy for a given data sequence.

OK Cancel Keyboard

Function options

Function: **PyShannonEntropy**

The base of the logarithm natural

Zero offset

OK Cancel Keyboard

Using user-defined Python function in ICM+

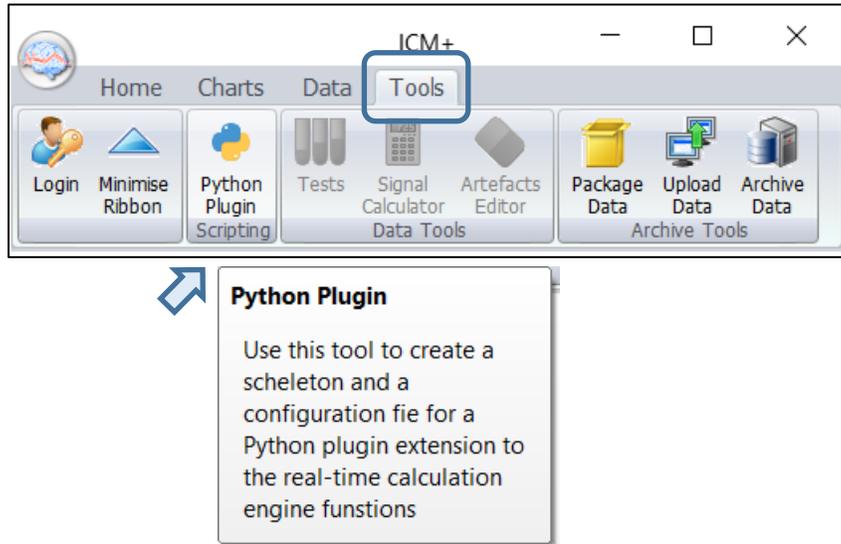
The screenshot displays the ICM+ software interface. On the left, the 'On Line Analysis Configuration Dialog' is open, showing a table of virtual signals. A red arrow points to the 'Entr' signal, which is defined by the formula 'PyShannonEntropy'. Below the table are 'Modify', 'Add', 'OK', and 'Cancel' buttons.

The main window, titled 'Primary Analysis Configuration Editor', features a toolbar with icons for Login, Minimise, Signals, Calculations, Connections, Change Project, Save Profile, Load Profile, Start, Stop, Monitor, Data Snapshot, and New Event Annotations. The 'Series' tab is active, showing 'Page 1' with two data series:

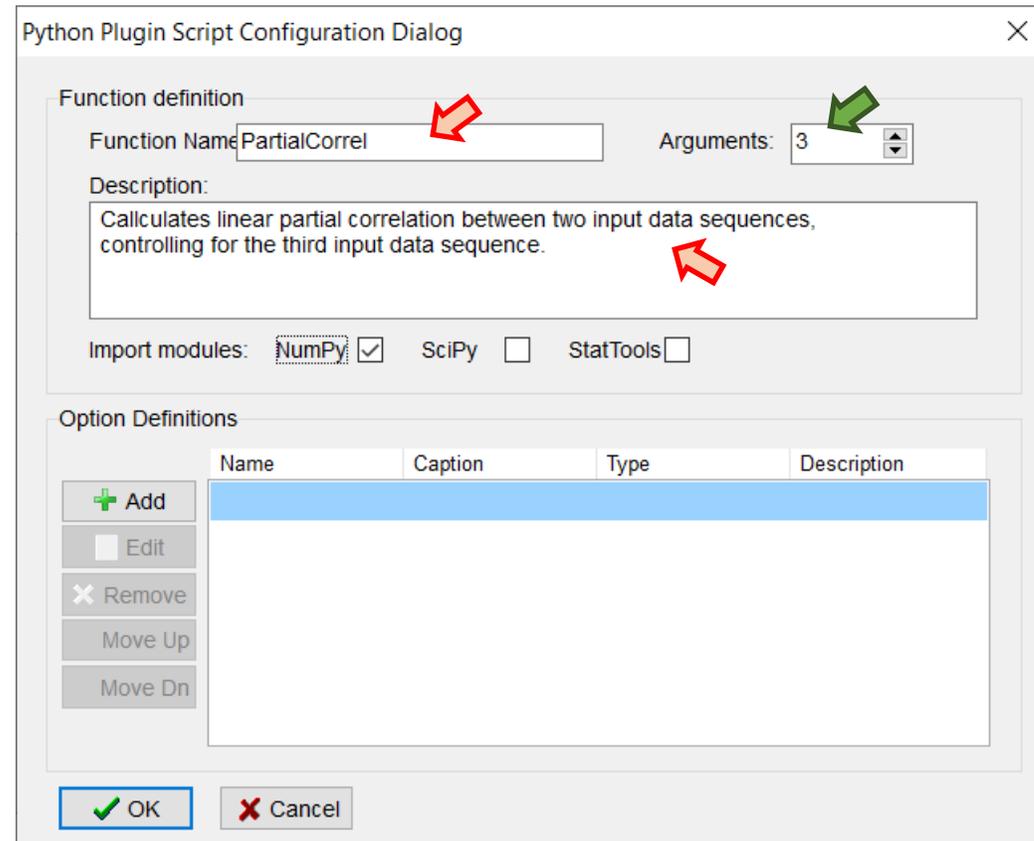
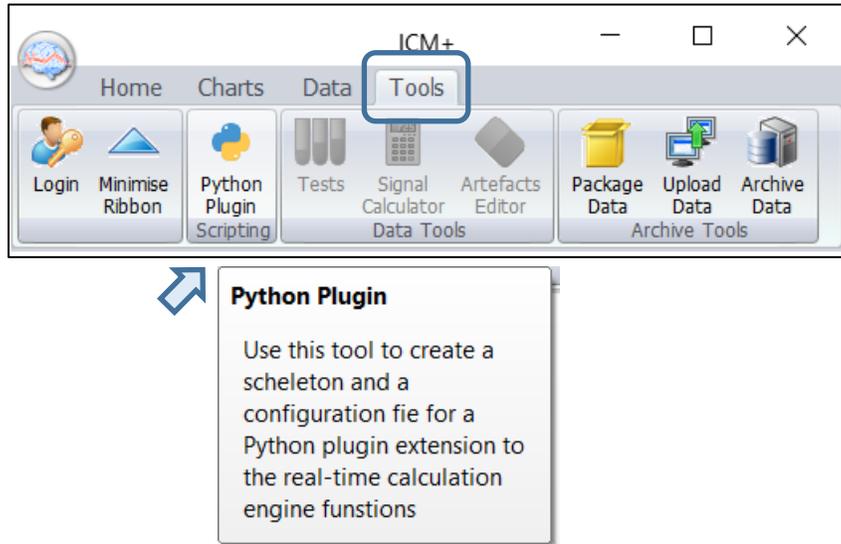
- ABP** (red area plot): Y-axis is [mmHg], ranging from 90 to 95. The plot shows a fluctuating signal over time.
- Entr** (green area plot): Y-axis is [au], ranging from 1.5 to 3. The plot shows a fluctuating signal over time.

The x-axis for both plots is time, with labels at 5/12 12:28, 5/12 12:32, 5/12 12:36, 5/12 12:40, and 5/12 12:44. The time scale is indicated as '< 21 minutes > 2014/12/05 12:24:20 - 12:45:20'. On the right side, there are input fields for 'values range' (both set to 0) and an 'Inputs' section listing 'ABP', 'etCO2', and 'FV'.

Partial correlation – example of a function with three inputs



Partial correlation – example of a function with three inputs



The generated Python script

```
import numpy as np

class PartialCorrel:

    # DO NOT MODIFY THIS METHOD. It is a part of the ICM+--Python interface.
    def set_parameter(self, param_name, param_value):
        setattr(self, param_name, param_value)

    # ...
    # ...

    # 'calculate' is the main work-horse function.
    # It is called with a data buffer (one or more) of size corresponding to the Calculation Window
    # It must return one floating-point number
    # It take the following parameters:
    # sig1 - input variable/signal 1
    # sig2 - input variable/signal 2
    # sig3 - input variable/signal 3
    # ts_time - part of the data time stamp - number of milliseconds since midnight
    # ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
    # It can also use the data sampling frequency:
    #     self.sampling_freq
    def calculate(self, sig1, sig2, sig3, ts_time, ts_date):
        # my own code here
        result = 0.0
        return result
```

The generated Python script

```
import numpy as np

class PartialCorrel:

    # DO NOT MODIFY THIS METHOD. It is a part of the ICM+--Python interface.
    def set_parameter(self, param_name, param_value):
        setattr(self, param_name, param_value)

    # ...
    # ...

    # 'calculate' is the main work-horse function.
    # It is called with a data buffer (one or more) of size corresponding to the Calculation Window
    # It must return one floating-point number
    # It take the following parameters:
    # sig1 - input variable/signal 1
    # sig2 - input variable/signal 2
    # sig3 - input variable/signal 3
    # ts_time - part of the data time stamp - number of milliseconds since midnight
    # ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
    # It can also use the data sampling frequency:
    # self.sampling_freq
    def calculate(self, sig1, sig2, sig3, ts_time, ts_date):
        # my own code here
        result = 0.0
        return result
```

Three input signals



The generated Python script

```
import numpy as np
from par_corr_module import partial_corr ← Importing function from another module

class PartialCorrel:

    # DO NOT MODIFY THIS METHOD. It is a part of the ICM+--Python interface.
    def set_parameter(self, param_name, param_value):
        setattr(self, param_name, param_value)

    # ...
    # ...

    # 'calculate' is the main work-horse function.
    # It is called with a data buffer (one or more) of size corresponding to the Calculation Window
    # It must return one floating-point number
    # It take the following parameters:
    # sig1 - input variable/signal 1
    # sig2 - input variable/signal 2
    # sig3 - input variable/signal 3
    # ts_time - part of the data time stamp - number of milliseconds since midnight
    # ts_date - Part of the data time stamp - One plus number of days since 1/1/0001
    # It can also use the data sampling frequency:
    # self.sampling_freq
    def calculate(self, sig1, sig2, sig3, ts_time, ts_date):

        A = np.array([sig1, sig2, sig3]).transpose()
        R_coefficients = partial_corr(A) ← My own code added to the calculate method
        return R_coefficients[0,1]
```

Three input signals

Using user-defined Python function in ICM+

The image shows two overlapping windows from the ICM+ software. The background window is the 'On Line Analysis Configuration Dialog' with tabs for 'Virtual Signals', 'Primary Analysis', 'Secondary Analysis 1', and 'Final Analysis'. The 'Primary Analysis' tab is active, showing a table of analysis configurations:

Name	Formula	Calc. Window [s]	Updated [s]
ABP	Mean(ABP)	300	60
FV	Mean(FV)	300	60
etCO2	Mean(etCO2)	300	60
	PyPartialCorrel(ABP, FV, etCO2)	300	60

The foreground window is the 'Primary Analysis Configuration Editor' for the 'parCorr' analysis. It includes the following sections:

- Name:** parCorr
- Enabled:**
- Calculation Window Specification:** Calculation Period: 300 s, Update Period: 60 s
- Valid values range:** Max Value: 0, Min Value: 0
- Formula:** PyPartialCorrel(ABP, FV, etCO2)
- Arguments:** ABP, FV, etCO2
- Function List:** PyNanStd, PyPartialCorrel (selected), PyShannonEntropy, PySpearmanCorrel, Range, RankCorrel, RelCount
- Inputs:** ABP, etCO2, FV
- Function description:** Calculates linear partial correlation between two input data sequences, controlling for the third input data sequence.

Both windows have 'OK', 'Cancel', 'Save', 'Load', and 'Advanced' buttons at the bottom.

Using user-defined Python function in ICM+

The image shows two overlapping windows from the ICM+ software. The background window is the 'On Line Analysis Configuration Dialog' with the 'Primary Analysis' tab selected. It contains a table of analysis configurations:

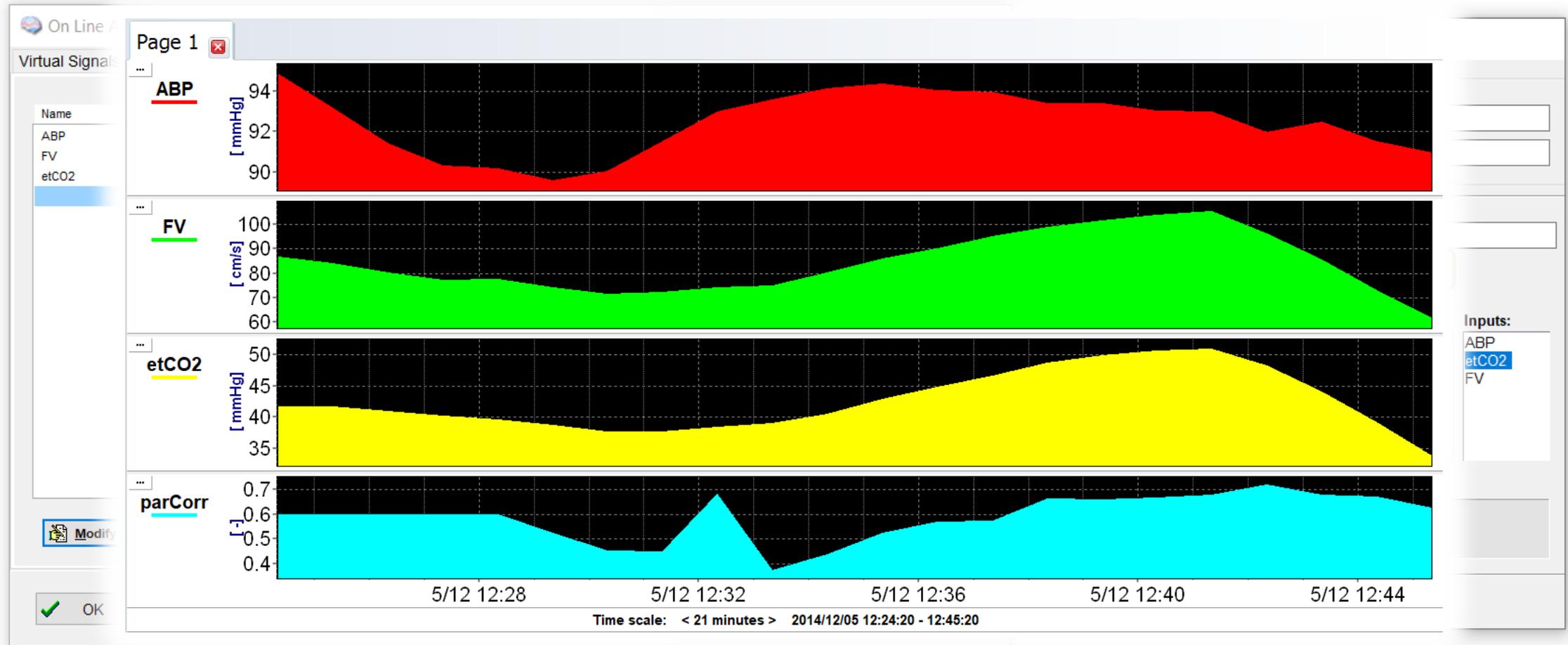
Name	Formula	Calc. Window [s]	Updated [s]
ABP	Mean(ABP)	300	60
FV	Mean(FV)	300	60
etCO2	Mean(etCO2)	300	60
	PyPartialCorrel(ABP, FV, etCO2)	300	60

A green arrow points to the 'PyPartialCorrel(ABP, FV, etCO2)' row. The foreground window is the 'Primary Analysis Configuration Editor' for the 'parCorr' analysis. It shows the following settings:

- Name: parCorr
- Enabled:
- Calculation Window Specification: Calculation Period: 300 s, Update Period: 60 s
- Valid values range: Max Value: 0, Min Value: 0
- Formula: PyPartialCorrel(ABP, FV, etCO2)
- Arguments: ABP, FV, etCO2 (highlighted with a green box and an arrow pointing to the 'Inputs' list)
- Function: PyPartialCorrel (selected in a dropdown)
- Function description: Calculates linear partial correlation between two input data sequences, controlling for the third input data sequence.

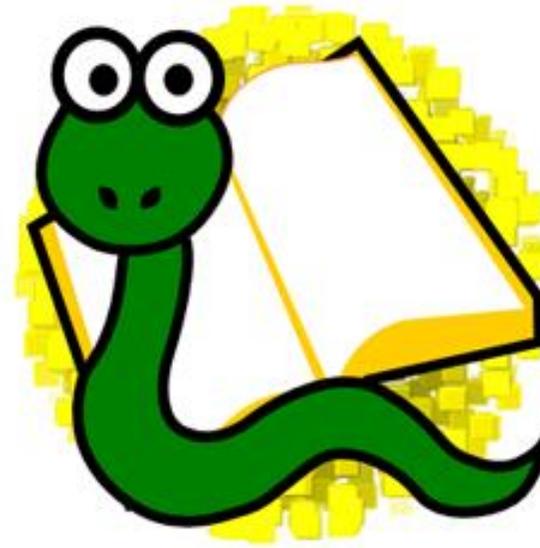
At the bottom of the foreground window, a green box labeled 'Three input signals' has an arrow pointing to the 'ABP', 'FV', and 'etCO2' arguments.

Using user-defined Python function in ICM+



Happy pythoning!


**KEEP
CALM
AND
CODE
PYTHON**



Based on image by Jeffrey Elkner



UNIVERSITY OF
CAMBRIDGE



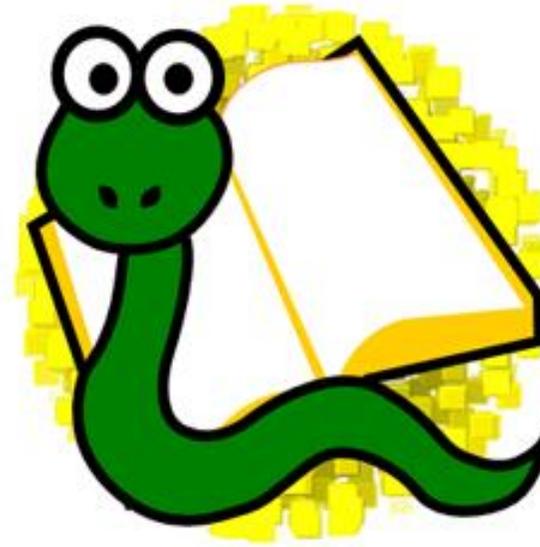
Wrocław University
of Science and Technology

Brain Physics Lab



Happy pythoning!


**KEEP
CALM
AND
CODE
PYTHON**



Based on image by Jeffrey Elkner

I invite you to see my poster (**#321**, Monday, 12:00–13:00)



UNIVERSITY OF
CAMBRIDGE



Wrocław University
of Science and Technology

Brain Physics Lab

