Spectroscopie optique de A à Z sous R

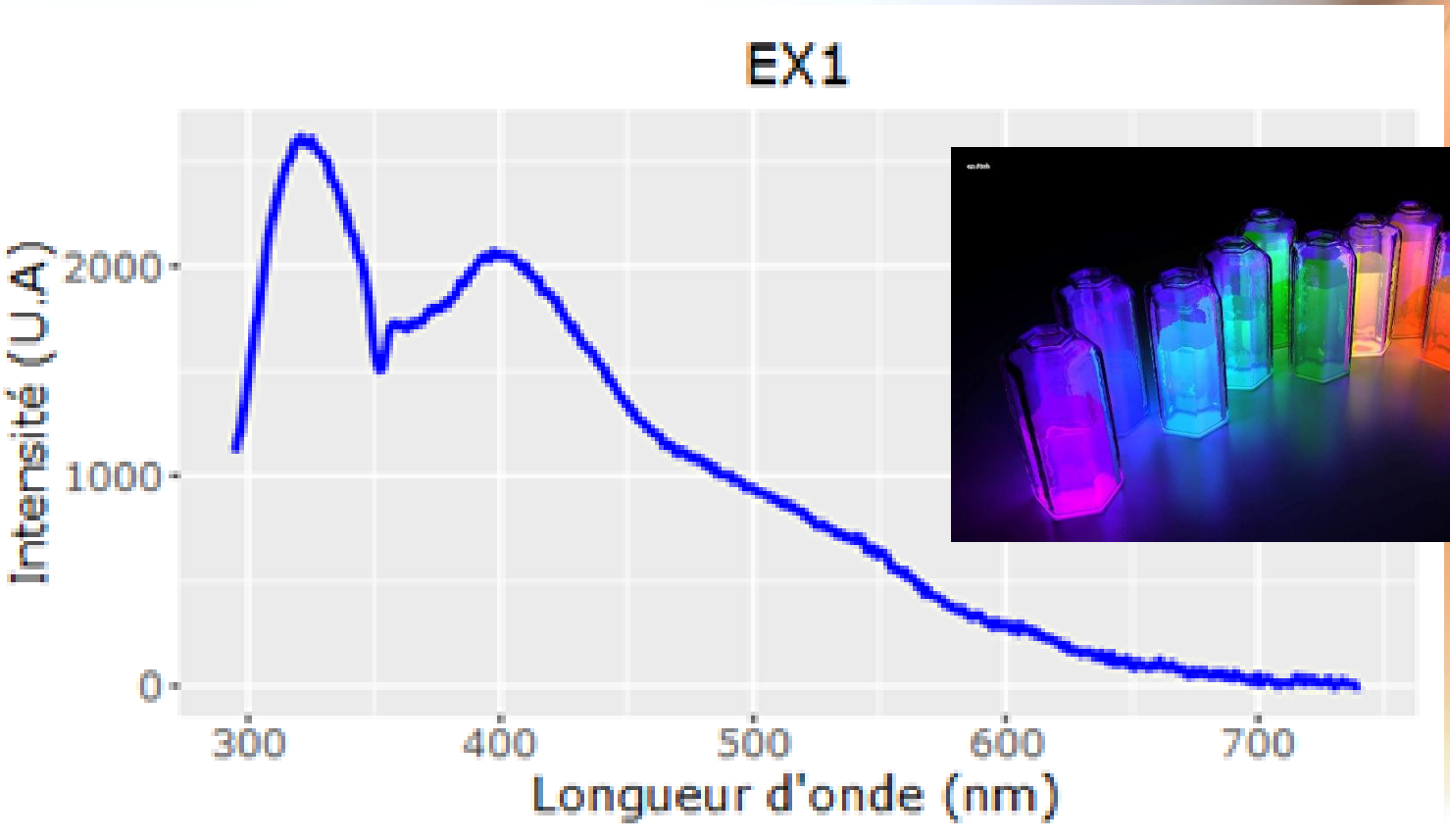
Bernard Panneton, ing. PhD – pannetonb@gmail.com

Alain Clément, PhD – Chercheur
Centre de R&D de St-Hyacinthe, Agriculture et Agroalimentaire Canada
alain.clement@canada.ca

https://github.com/PannetonB/R-a-Quebec

La spectroscopie optique

- On envoie de la lumière vers un échantillon. Cette lumière interagit avec la matière et altère la lumière incidente. On récupère la lumière résultante pour mesurer son spectre avec un spectromètre



Motivation
Nos besoins

- Mettre en place des ***pipelines d'acquisition élaborés***:
  - Plusieurs types de spectre (transmittance, fluorescence, Raman…)
    - Plusieurs spectromètres
    - Plusieurs sources lumineuses
  - Automatisation du positionnement des échantillons
- ***Éliminer la manipulation directe des données brutes***
  - « Copier – Coller » dans un chiffrier est le meilleur chemin pour générer des erreurs!
- ***Environnement flexible de traitement des données*** et pour le développement de modèles.
- Rendre la ***chaîne acquisition – validation – traitement accessible*** à des utilisateurs sans connaissance préalable de la spectroscopie (application sur le terrain)

Motivation
Limitations des solutions commerciales

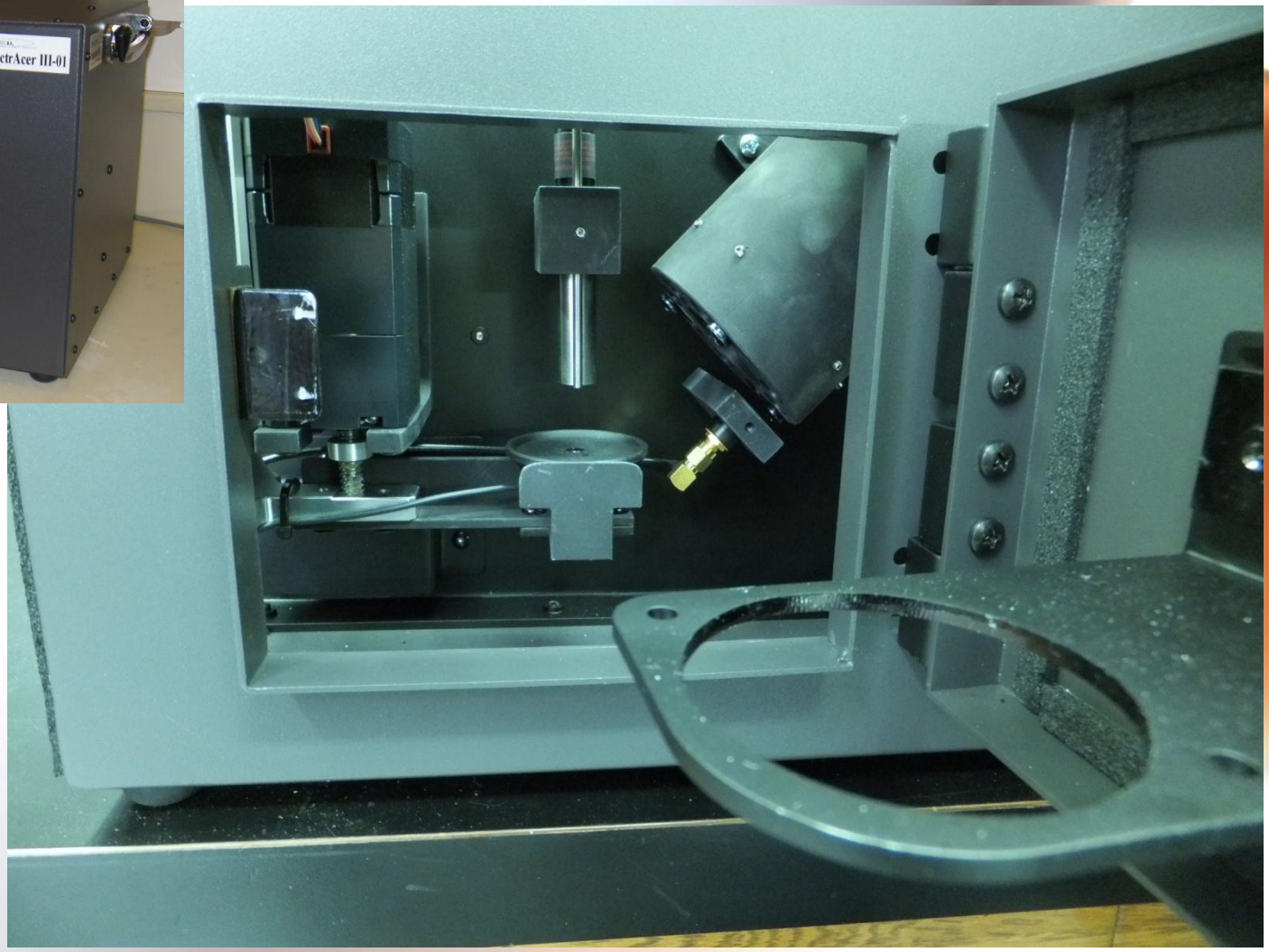
- ***Acquisition de données***
  - Manque de souplesse pour l'acquisition
  - Structure et format de stockage des données uniques au logiciel
    - Souvent 1 spectre par fichier!
  - Difficile de coordonner l'acquisition de spectres sur plus d'un instrument et plus d'une source lumineuse.

- ***Traitement des données***
  - Excellents logiciels commerciaux disponibles
  - Coûteux
  - Généraliste donc plus difficile de créer des pipelines de travail (beaucoup de clics!)
  - Importation des données pas toujours facile

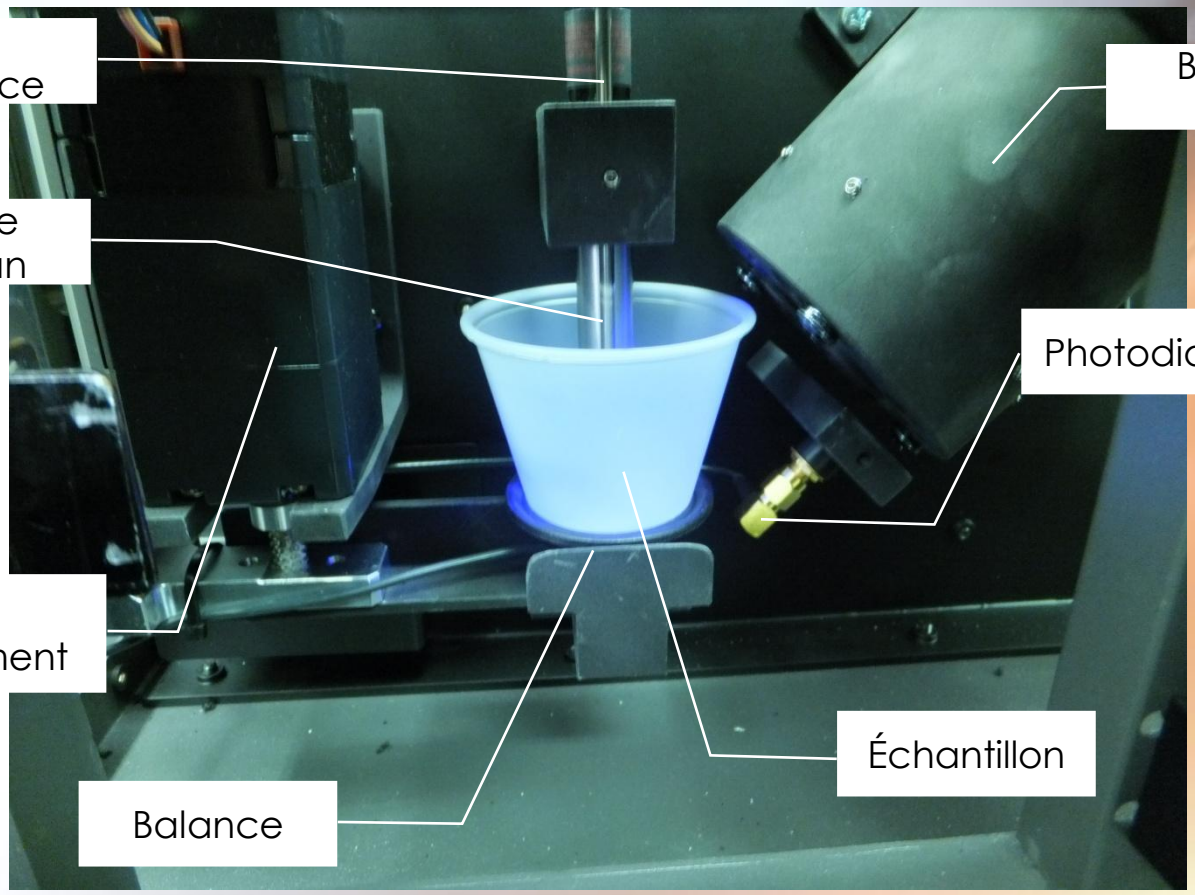
Pourquoi R

- Excellent environnement pour l'analyse des données
  - python est là mais je connaissais R…
- Aucun frais de déploiement sur autant de stations de travail que nécessaire

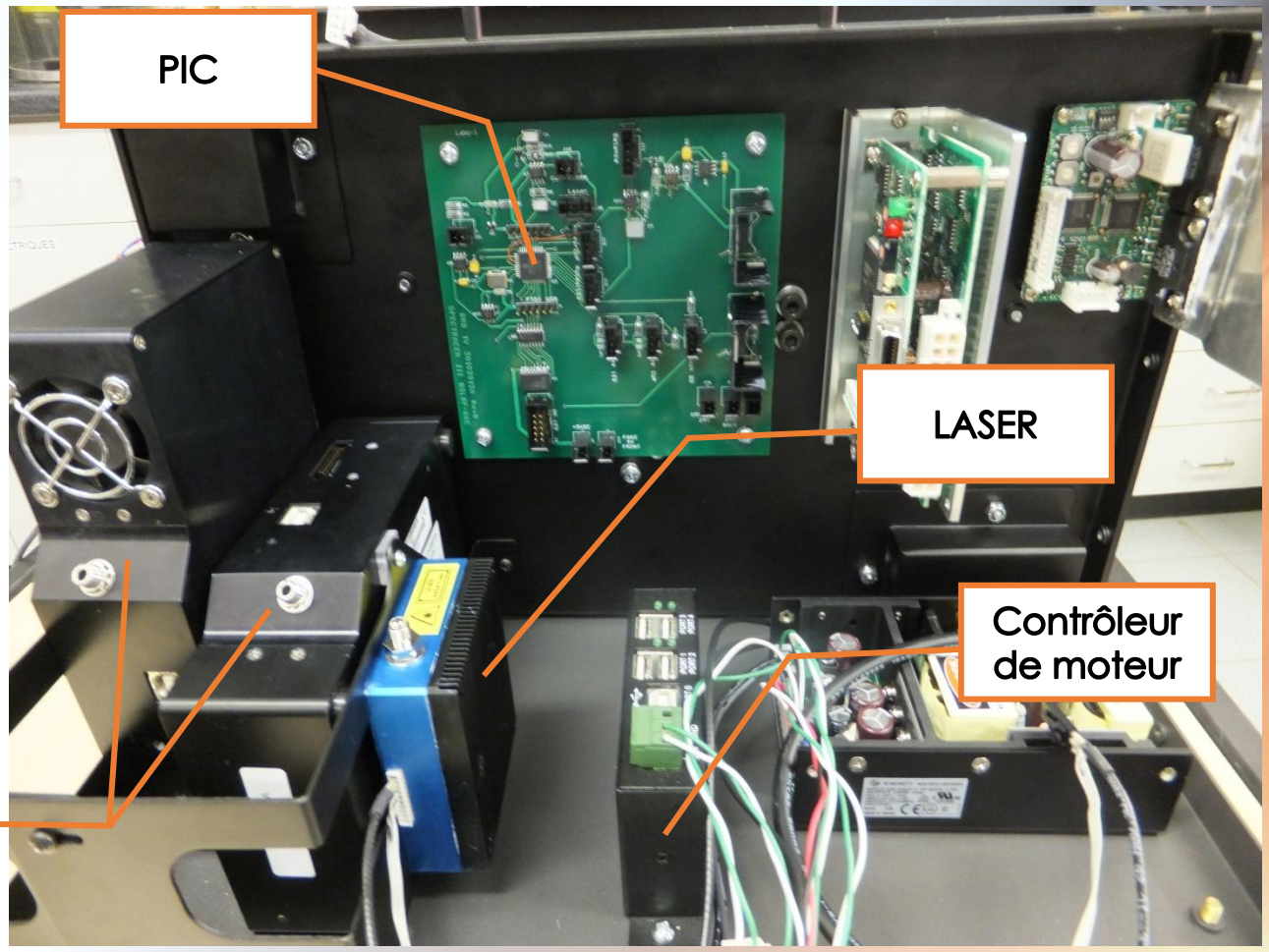
Exemple d'application
SpectrAcer III -Inspection du sirop d'érable



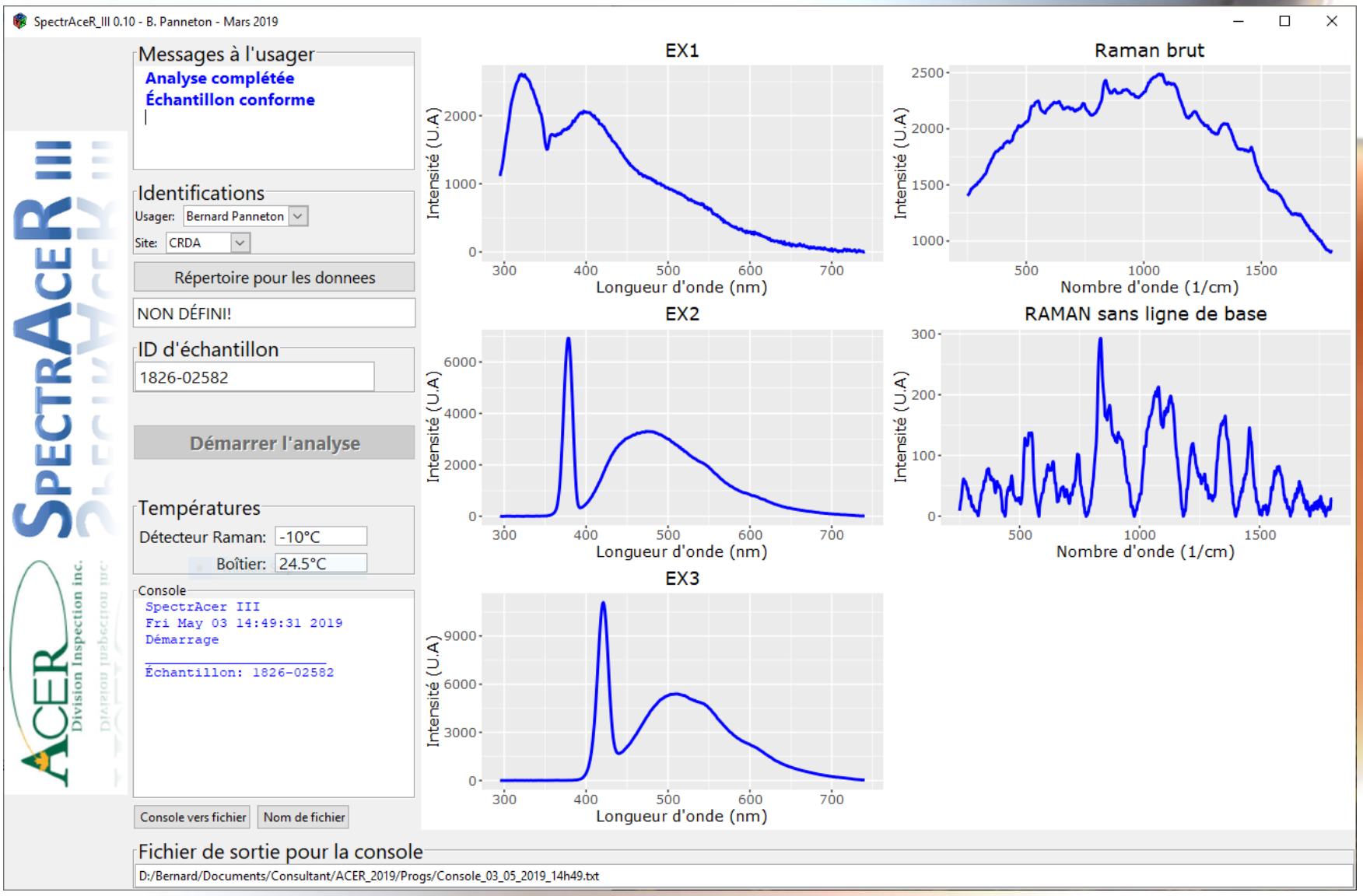
SpectrAcer III
Inspection du sirop d'érable



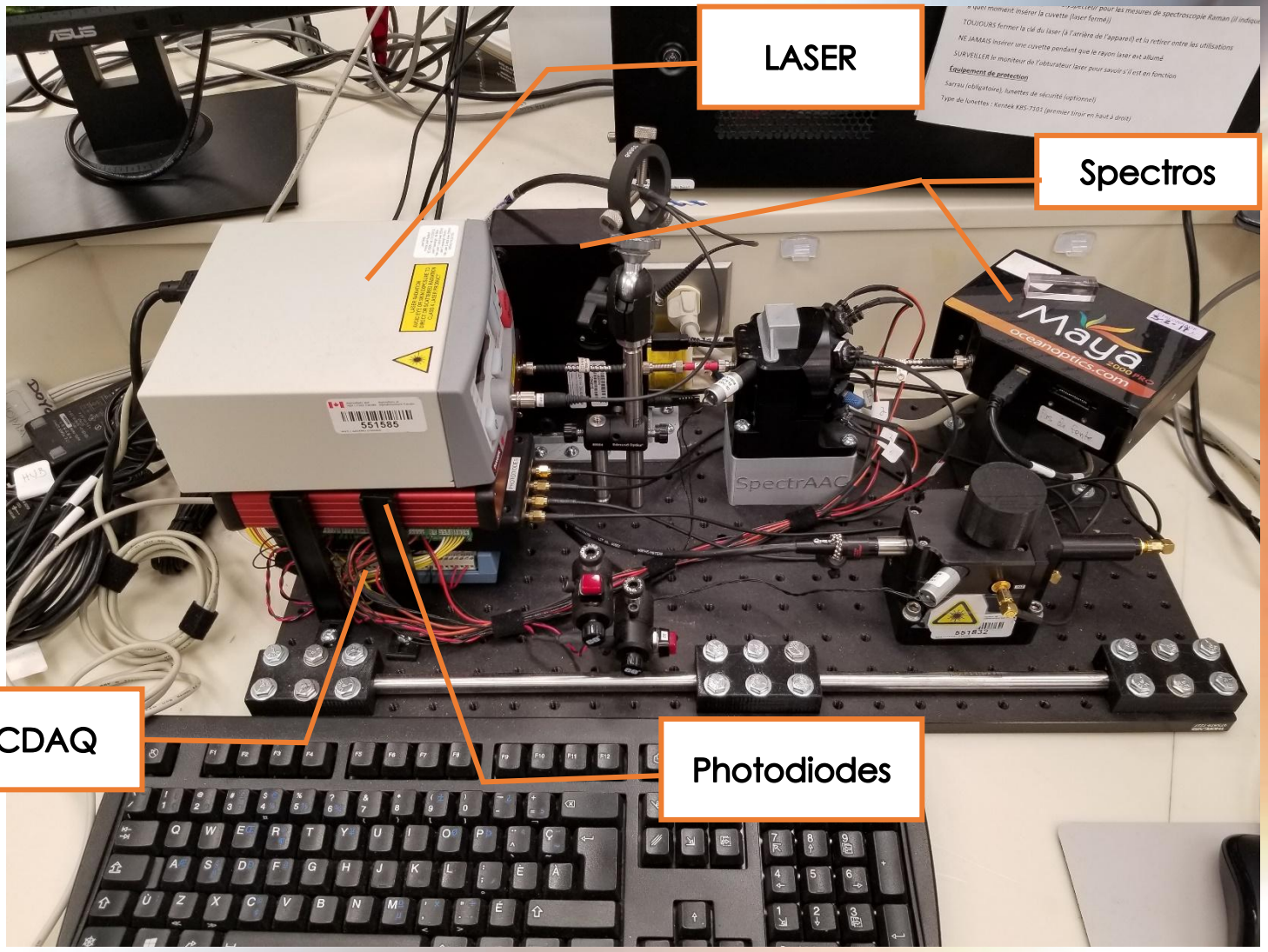
SpectrAcer III
Inspection du sirop d'érable



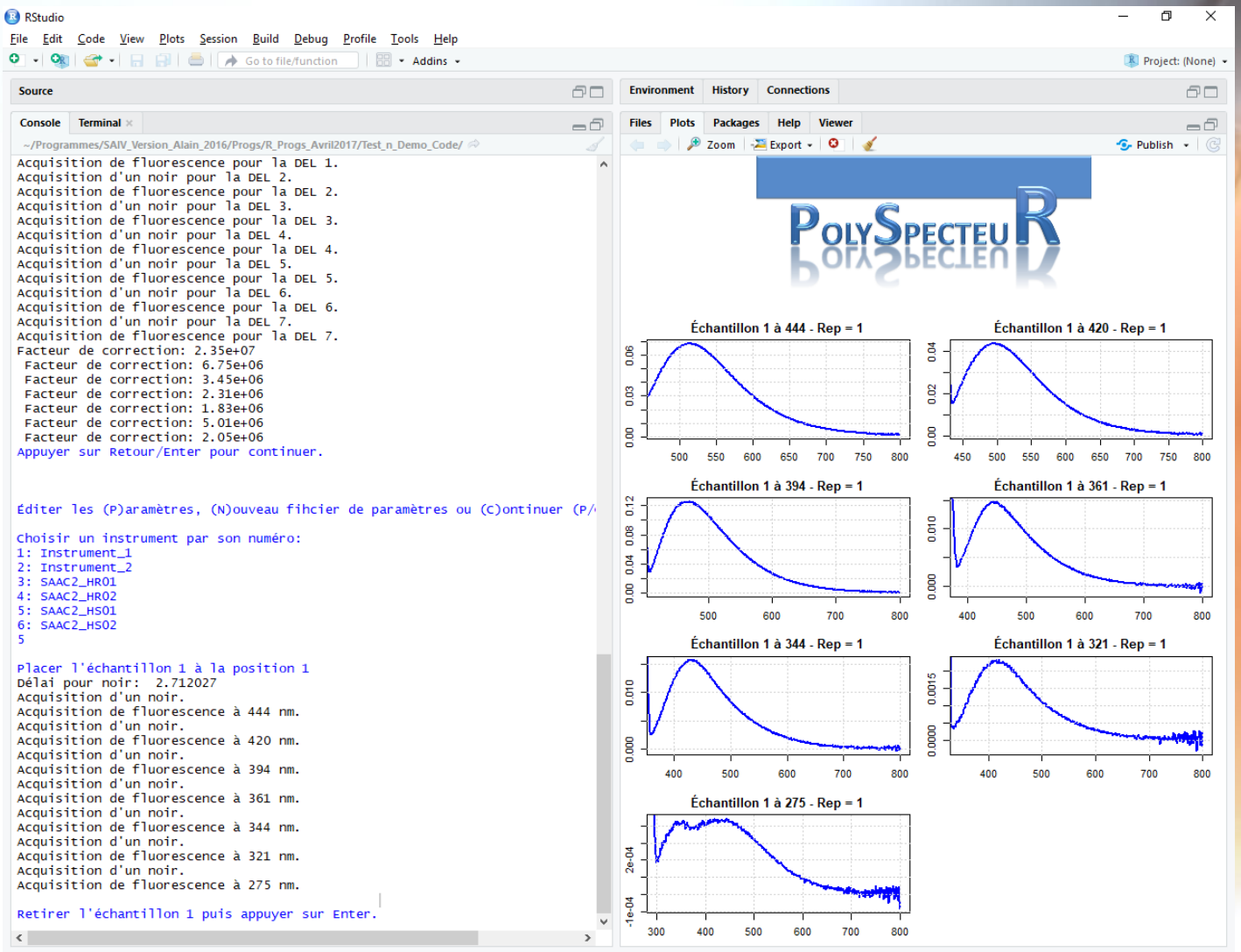
SpectrAcer III – Interface usager



Exemple d'application - SpectrAAC



PolySpecteur – Interface usager



Écosystème de spectroscopie

Données

- Interface usager
  - Rstudio
  - GUI sous gWidgets2 et RGtk2
- OOInterface.r
  - Java library
  - Package rJava
- MCDAQ.r
  - A C++ dll
  - .C du package base
- RS232
  - Commande de moteurs
  - Commande d'un laser
  - Package serial ou tcltk

- GUI
  - Packages gWidgets2 et RGtk2
- Packages de chimiométrie
  - ChemoSpec,
  - chemometrics,
  - prospectr,
  - caret,
  - stats

Écosystème de spectroscopie

Données

- Interface usager
  - Rstudio
  - GUI sous gWidgets2 et RGtk2
- OOInterface.r
  - Java library
  - Package rJava
- MCDAQ.r
  - A C++ dll
  - .C du package base
- RS232
  - Commande de moteurs
  - Commande d'un laser
  - Package serial ou tcltk

- GUI
  - Packages gWidgets2 et RGtk2
- Packages de chimiométrie
  - ChemoSpec,
  - chemometrics,
  - prospectr,
  - caret,
  - stats

rJava

JAVA – Méthodes de la classe Wrapper

openAllSpectrometers

public int openAllSpectrometers()

Returns:

int the number of spectrometers found.
Returns -1 if an I/O error occurred. In this case,
call getLastException() to determine the nature of the error.

Accéder à Java depuis R

```
library(rJava)
```

Initialiser un objet de la classe Wrapper

```
ooi_home=Sys.getenv("OOI_HOME")
mypath=file.path(ooi_home,"OmniDriver.jar")
.jinit()
.jaddClassPath(mypath)
mywrap<-.jnew("com.oceanoptics.omnidriver.api.wrapper.Wrapper")
```

Utiliser les méthodes de la classe Wrapper

```
nbspectro <- mywrap$openAllSpectrometers()
xaxis=mywrap$getWavelengths(as.integer(leSpectro$number))
```

rJava

JAVA – Méthodes de la classe Wrapper

openAllSpectrometers

public int openAllSpectrometers()

Returns:

int the number of spectrometers found.
Returns -1 if an I/O error occurred. In this case,
call getLastException() to determine the nature of the error.

Accéder à Java depuis R

```
library(rJava)
```

Initialiser un objet de la classe Wrapper

```
ooi_home=Sys.getenv("OOI_HOME")
mypath=file.path(ooi_home,"OmniDriver.jar")
.jinit()
.jaddClassPath(mypath)
mywrap<-.jnew("com.oceanoptics.omnidriver.api.wrapper.Wrapper")
```

Utiliser les méthodes de la classe Wrapper

```
nbspectro <- mywrap$openAllSpectrometers()
xaxis=mywrap$getWavelengths(as.integer(leSpectro$number))
```

rJava

JAVA – Méthodes de la classe Wrapper

openAllSpectrometers

public int openAllSpectrometers()

Returns:

int the number of spectrometers found.
Returns -1 if an I/O error occurred. In this case,
call getLastException() to determine the nature of the error.

Accéder à Java depuis R

```
library(rJava)
```

Initialiser un objet de la classe Wrapper

```
ooi_home=Sys.getenv("OOI_HOME")
mypath=file.path(ooi_home,"OmniDriver.jar")
.jinit()
.jaddClassPath(mypath)
mywrap<-.jnew("com.oceanoptics.omnidriver.api.wrapper.Wrapper")
```

Utiliser les méthodes de la classe Wrapper

```
nbspectro <- mywrap$openAllSpectrometers()
xaxis=mywrap$getWavelengths(as.integer(leSpectro$number))
```

rJava

JAVA – Méthodes de la classe Wrapper

openAllSpectrometers

public int openAllSpectrometers()

Returns:

int the number of spectrometers found.
Returns -1 if an I/O error occurred. In this case,
call getLastException() to determine the nature of the error.

Accéder à Java depuis R

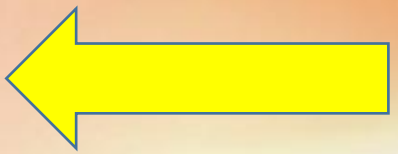
```
library(rJava)
```

Initialiser un objet de la classe Wrapper

```
ooi_home=Sys.getenv("OOI_HOME")
mypath=file.path(ooi_home,"OmniDriver.jar")
.jinit()
.jaddClassPath(mypath)
mywrap<-.jnew("com.oceanoptics.omnidriver.api.wrapper.Wrapper")
```

Utiliser les méthodes de la classe Wrapper

```
nbspectro <- mywrap$openAllSpectrometers()
xaxis=mywrap$getWavelengths(as.integer(leSpectro$number))
```



Écosystème de spectroscopie

Données

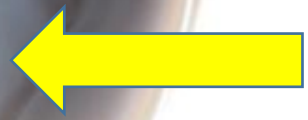
- Interface usager
  - Rstudio
  - GUI sous gWidgets2 et RGtk2
- OOInterface.r
  - Java library
  - Package rJava
- MCDAQ.r
  - A C++ dll
  - .C du package base
- RS232
  - Commande de moteurs
  - Commande d'un laser
  - Package serial ou tcltk

- GUI
  - Packages gWidgets2 et RGtk2
- Packages de chimiométrie
  - ChemoSpec,
  - chemometrics,
  - prospectr,
  - caret,
  - stats

Interface .C (base package)

.C() **is simpler** than .Call() and can be **useful if you already have standard C code.**[1]

- **Fonction de la librairie**
  - `int cbDConfigBit(int BoardNum, int PortType, int BitNum, int Direction)`



- **Fonction d'interface cbw64.dll -> MC_cbw64_CWrapper.c**

```
void BP_cbDConfigBit( int *BoardNum, int *PortType, int *BitNum, int *Direction, int *out)
// Direction = 1 pour out; = 2 pour in
{
   int y;
   y=cbDConfigBit(*BoardNum, *PortType, *BitNum, *Direction);
   *out=y;
}
```

- **Compiler**
  - **system("R CMD SHLIB MC_cbw64_CWrapper.c -L. cbw64.dll")**

- **Librairie R**
  - dyn.load(« …//Progs//C//MC_cbw64_CWrapper.dll »)

```
Config_IO_Bit<-function(bnum, type, bit, direct){
   err=.C("BP_cbDConfigBit",BoardNum=as.integer(bnum), PortType=as.integer(type),
                            BitNum = as.integer(bit), Direction=as.integer(direct), out=as.integer(0))$out
   if (err!=0) return(paste("Échec de la configuration du bit ", as.character(bit),
                            " sur le module ", as.character(bnum),sep=""))
   else return("O.K")
}
```

[1] http://r-pkgs.had.co.nz/src.html

Interface .C (base package)

- **Fonction de la librairie**
  - `int cbDConfigBit(int BoardNum, int PortType, int BitNum, int Direction)`
- **Fonction d'interface cbw64.dll -> MC_cbw64_CWrapper.c**
  - `void BP_cbDConfigBit( int *BoardNum, int *PortType, int *BitNum, int *Direction, int *out)`

```
// Direction = 1 pour out; = 2 pour in
{
   int y;
   y=cbDConfigBit(*BoardNum, *PortType, *BitNum, *Direction);
   *out=y;
}
```

- **Compiler**
  - **system("R CMD SHLIB MC_cbw64_CWrapper.c -L. cbw64.dll")**
- **Librairie R**
  - dyn.load(« …//Progs//C//MC_cbw64_CWrapper.dll »)
  -

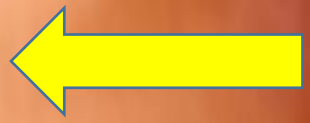
```
Config_IO_Bit<-function(bnum, type, bit, direct){
   err=.C("BP_cbDConfigBit",BoardNum=as.integer(bnum), PortType=as.integer(type),
                            BitNum = as.integer(bit), Direction=as.integer(direct), out=as.integer(0))$out
   if (err!=0) return(paste("Échec de la configuration du bit ", as.character(bit),
                            " sur le module ", as.character(bnum),sep=""))
   else return("O.K")
}
```

Interface .C (base package)

- **Fonction de la librairie**
  - `int cbDConfigBit(int BoardNum, int PortType, int BitNum, int Direction)`
- **Fonction d'interface cbw64.dll -> MC_cbw64_CWrapper.c**
  - `void BP_cbDConfigBit( int *BoardNum, int *PortType, int *BitNum, int *Direction, int *out)`

```
// Direction = 1 pour out; = 2 pour in
{
   int y;
   y=cbDConfigBit(*BoardNum, *PortType, *BitNum, *Direction);
   *out=y;
}
```

- **Compiler**
  - **system("R CMD SHLIB MC_cbw64_CWrapper.c -L. cbw64.dll")**



- **Librairie R**
  - dyn.load(« …//Progs//C//MC_cbw64_CWrapper.dll »)
  -

```
Config_IO_Bit<-function(bnum, type, bit, direct){
   err=.C("BP_cbDConfigBit",BoardNum=as.integer(bnum), PortType=as.integer(type),
                            BitNum = as.integer(bit), Direction=as.integer(direct), out=as.integer(0))$out
   if (err!=0) return(paste("Échec de la configuration du bit ", as.character(bit),
                            " sur le module ", as.character(bnum),sep=""))
   else return("O.K")
}
```

Interface .C (base package)

- **Fonction de la librairie**
  - **int** cbDConfigBit(int BoardNum, int PortType, int BitNum, int Direction)
- **Fonction d'interface cbw64.dll -> MC_cbw64_CWrapper.c**
  - **void** BP_cbDConfigBit(int *BoardNum, int *PortType, int *BitNum, int *Direction, **int *out**)

```
// Direction = 1 pour out; = 2 pour in
{
   int y;
   y=cbDConfigBit(*BoardNum, *PortType, *BitNum, *Direction);
   *out=y;
}
```

- **Compiler**
  - **system("R CMD SHLIB MC_cbw64_CWrapper.c -L. cbw64.dll")**
- **Librairie R**
  - dyn.load(« …//Progs//C//MC_cbw64_CWrapper.dll »)
  -

```
Config_IO_Bit<-function(bnum, type, bit, direct){
   err=.C("BP_cbDConfigBit",BoardNum=as.integer(bnum), PortType=as.integer(type),
                            BitNum = as.integer(bit), Direction=as.integer(direct), out=as.integer(0))$out
   if (err!=0) return(paste("Échec de la configuration du bit ", as.character(bit),
                            " sur le module ", as.character(bnum),sep=""))
   else return("O.K")
}
```

Écosystème de spectroscopie

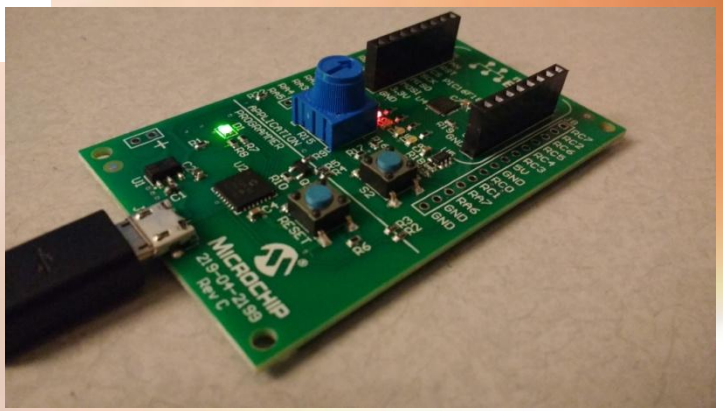
Données

- Interface usager
  - Rstudio
  - GUI sous gWidgets2 et RGtk2
- OOInterface.r
  - Java library
  - Package rJava
- MCDAQ.r
  - A C++ dll
  - .C du package base
- RS232
  - Commande de moteurs
  - Commande d'un laser
  - Package serial ou tcltk

- GUI
  - Packages gWidgets2 et RGtk2
- Packages de chimiométrie
  - ChemoSpec,
  - chemometrics,
  - prospectr,
  - caret,
  - stats

Communication série – RS232
Package serial

TX : “CMDL00100”



RX : “Pos. Del - 00100”

Voir https://fr.wikipedia.org/wiki/RS-232
pour le paramétrage.

Communication série – RS232
Package serial

```
baud_rate = c(9600, 4800, 14400)  #valeurs possibles de baudrate.
tout=2   #toute les commandes devraient s'exécuter en dedans de 2 secondes

#Obtenir une liste des ports
listports=listPorts()

#Avec deux boucles (ports com puis baudrate), on ouvre un port et on teste des
#baudrates jusqu'à ce qu'on obtienne une réponse 'PIC' à la commande
#'C$ID00000'. On balaye les ports dans l'ordre inverse de 'listports' car
#souvent le bon port est le dernier attribué pour passer le caractère '$' dans
#la commande: '\\x24' Le double \ est nécessaire pour éviter que R interprète
#'\x24' comme '$' avant de passer la commande.
commande='C\\x24ID00000' #


for (cp in rev(listports)){
  nom_port<-"SAIII_PIC"  #nom
  # On teste les baudrate
  for (br in baud_rate){

      #Configuration du port
      le_com <- serialConnection(name=nom_port, port=cp,
                      mode=paste(br,',n,8,1',sep=''),
                      newline = TRUE,
                      translation = 'crlf',
                      handshake='none')
      open(le_com)
      Sys.sleep(0.1)
```

```
#Envoie la commande
#Besoin d'ajouter crlf?
dum <- write.serialConnection(le_com,commande)
```

```
      #Read
      le_t=0
      dum=NULL
      while (is.null(dum) & le_t<tout){
        dum <- read.serialConnection(le_com)
        if (debug_me) cat('Réponse à ',commande,': ',dum,'\n')
        Sys.sleep(0.1)
        le_t=le_t+0.1
      }
```

Communication série – RS232
Package serial

```
baud_rate = c(9600, 4800, 14400)  #valeurs possibles de baudrate.
tout=2   #toute les commandes devraient s'exécuter en dedans de 2 secondes

#Obtenir une liste des ports
listports=listPorts()

#Avec deux boucles (ports com puis baudrate), on ouvre un port et on teste des
#baudrates jusqu'à ce qu'on obtienne une réponse 'PIC' à la commande
#'C$ID00000'. On balaye les ports dans l'ordre inverse de 'listports' car
#souvent le bon port est le dernier attribué pour passer le caractère '$' dans
#la commande: '\\x24' Le double \ est nécessaire pour éviter que R interprète
#'\x24' comme '$' avant de passer la commande.
commande='C\\x24ID00000' #


for (cp in rev(listports)){
  nom_port<-"SAIII_PIC"  #nom
  # On teste les baudrate
  for (br in baud_rate){

      #Configuration du port
      le_com <- serialConnection(name=nom_port, port=cp,
                      mode=paste(br,',n,8,1',sep=''),
                      newline = TRUE,
                      translation = 'crlf',
                      handshake='none')
      open(le_com)
      Sys.sleep(0.1)
```

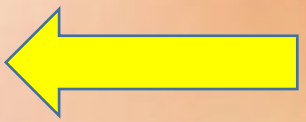
```
#Envoie la commande
#Besoin d'ajouter crlf?
dum <- write.serialConnection(le_com,commande)
```

```
      #Read
      le_t=0
      dum=NULL
      while (is.null(dum) & le_t<tout){
        dum <- read.serialConnection(le_com)
        if (debug_me) cat('Réponse à ',commande,': ',dum,'\n')
        Sys.sleep(0.1)
        le_t=le_t+0.1
      }
```

Communication série – RS232
Package serial

```
baud_rate = c(9600, 4800, 14400)  #valeurs possibles de baudrate.
tout=2   #toute les commandes devraient s'exécuter en dedans de 2 secondes

#Obtenir une liste des ports
listports=listPorts()

#Avec deux boucles (ports com puis baudrate), on ouvre un port et on teste des
#baudrates jusqu'à ce qu'on obtienne une réponse 'PIC' à la commande
#'C$ID00000'. On balaye les ports dans l'ordre inverse de 'listports' car
#souvent le bon port est le dernier attribué pour passer le caractère '$' dans
#la commande: '\\x24' Le double \ est nécessaire pour éviter que R interprète
#'\x24' comme '$' avant de passer la commande.
commande='C\\x24ID00000' #


for (cp in rev(listports)){
  nom_port<-"SAIII_PIC"  #nom
  # On teste les baudrate
  for (br in baud_rate){

      #Configuration du port
      le_com <- serialConnection(name=nom_port, port=cp,
                      mode=paste(br,',n,8,1',sep=''),
                      newline = TRUE,
                      translation = 'crlf',
                      handshake='none')
      open(le_com)
      Sys.sleep(0.1)
```

```
#Envoie la commande
#Besoin d'ajouter crlf?
dum <- write.serialConnection(le_com,commande)
```



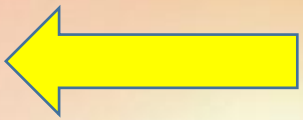
```
      #Read
      le_t=0
      dum=NULL
      while (is.null(dum) & le_t<tout){
        dum <- read.serialConnection(le_com)
        if (debug_me) cat('Réponse à ',commande,': ',dum,'\n')
        Sys.sleep(0.1)
        le_t=le_t+0.1
      }
```

Communication série – RS232
Package serial

```
baud_rate = c(9600, 4800, 14400)  #valeurs possibles de baudrate.
tout=2   #toute les commandes devraient s'exécuter en dedans de 2 secondes

#Obtenir une liste des ports
listports=listPorts()

#Avec deux boucles (ports com puis baudrate), on ouvre un port et on teste des
#baudrates jusqu'à ce qu'on obtienne une réponse 'PIC' à la commande
#'C$ID00000'. On balaye les ports dans l'ordre inverse de 'listports' car
#souvent le bon port est le dernier attribué pour passer le caractère '$' dans
#la commande: '\\x24' Le double \ est nécessaire pour éviter que R interprète
#'\x24' comme '$' avant de passer la commande.
commande='C\\x24ID00000' #


for (cp in rev(listports)){
  nom_port<-"SAIII_PIC"  #nom
  # On teste les baudrate
  for (br in baud_rate){

      #Configuration du port
      le_com <- serialConnection(name=nom_port, port=cp,
                      mode=paste(br,',n,8,1',sep=''),
                      newline = TRUE,
                      translation = 'crlf',
                      handshake='none')
      open(le_com)
      Sys.sleep(0.1)
```

```
#Envoie la commande
#Besoin d'ajouter crlf?
dum <- write.serialConnection(le_com,commande)
```

```
      #Read
      le_t=0
      dum=NULL
      while (is.null(dum) & le_t<tout){
        dum <- read.serialConnection(le_com)
        if (debug_me) cat('Réponse à ',commande,': ',dum,'\n')
        Sys.sleep(0.1)
        le_t=le_t+0.1
      }
```



Écosystème de spectroscopie

Données

- Interface usager
  - Rstudio
  - GUI sous gWidgets2 et RGtk2
- OOInterface.r
  - Java library
  - Package rJava
- MCDAQ.r
  - A C++ dll
  - .C du package base
- RS232
  - Commande de moteurs
  - Commande d'un laser
  - Package serial ou tcltk

- GUI
  - Packages gWidgets2 et RGtk2
- Packages de chimiométrie
  - ChemoSpec,
  - chemometrics,
  - prospectr,
  - caret,
  - stats

InSpectoR 0.1.0 - March 2019

Data selection | Apply models | PrePro | PCA | PLSDA | PLS

D:\Bernard\mesProgs\R\InSpectoR\Data\Insectes_MAF_Avril2018\Y_Test_Maf.txt

OPEN a Y file

Spectral data files - Select

X Data Files

EX277_Test_Maf_B.txt
EX277_Test_Maf_I.txt
EX297_Test_Maf_B.txt
EX297_Test_Maf_I.txt
EX321_Test_Maf_B.txt
EX321_Test_Maf_I.txt

Define new factor
Aggregate spectra
Define subset
Find duplicate samples from Col. 1
Remove selected samples
Merge Y data files

Properties
X-axis min 0
X-axis max 0
Y-axis min 0.0
Y-axis max 0.0

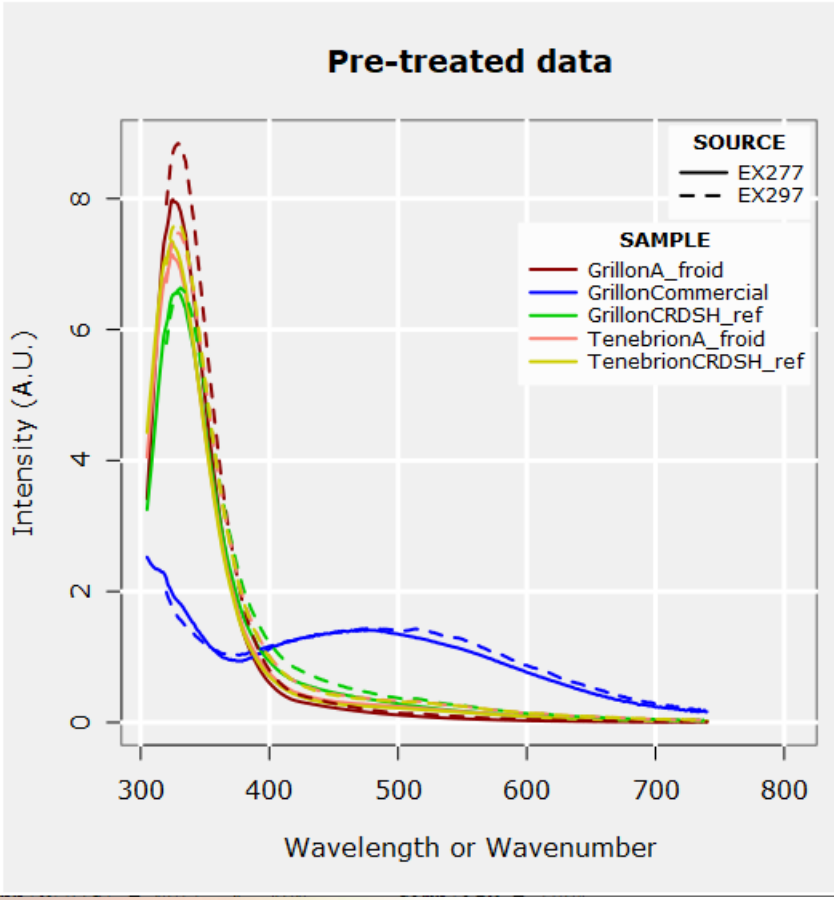
Hints

----- PLOT AREA -----
Right click to select 'Select', 'Zoom' modes and 'Copy' or 'Save'
Select mode: click for nearest or drag rectangle.
Zoom mode: click to zoom around or drag rectangle.
Right click and select Zoom all.
Right click to Copy/Save

----- TABLE AREA -----
Click/CtrlClick/ShiftClick in table to select samples
Click on column for sorting and plot by factor. If column

Y data

nt.1	PosRep	Classe1	Classe2	NoSeqFile	NoSeq
:nt_1	1	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	129	129
:nt_1	2	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	130	130
:nt_1	3	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	131	131
:nt_1	4	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	132	132
:nt_1	1	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	133	133
:nt_1	2	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	134	134
:nt_1	3	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	135	135
:nt_1	4	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	136	136
:nt_1	1	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	137	137
:nt_1	2	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	138	138
:nt_1	3	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	139	139
:nt_1	4	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	140	140
:nt_1	1	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	141	141
:nt_1	2	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	142	142
:nt_1	3	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	143	143
:nt_1	4	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	144	144
:nt_2	1	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	145	145
:nt_2	2	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	146	146
:nt_2	3	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	147	147
:nt_2	4	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	148	148
:nt_2	1	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	149	149
:nt_2	2	TenebrionCRDSH_refS46	TenebrionCRDSH_ref	150	150



InSpectoR 0.1.0 - March 2019

Data selection | Apply models | PrePro | PCA | PLSDA | PLS

Modeling definitions

Spectral data set(s) for PLSDA

X Data Files

EX277

EX297

Factor to predict

Pick a factor: Classe_2

Train control options

Prop. of data for training: 0.6

Resampling method: cv

Number of folds: 5

Number of repetitions: 1

Training options

Nb of LVs (max): 4

PreProcessing: None

Prediction method: softmax

Performance metric: Kappa

Aggregation options

Aggregation operator: concatenate

Compute model

Save model

Output area

Output controls

Accuracy - VL plot

Confusion matrix

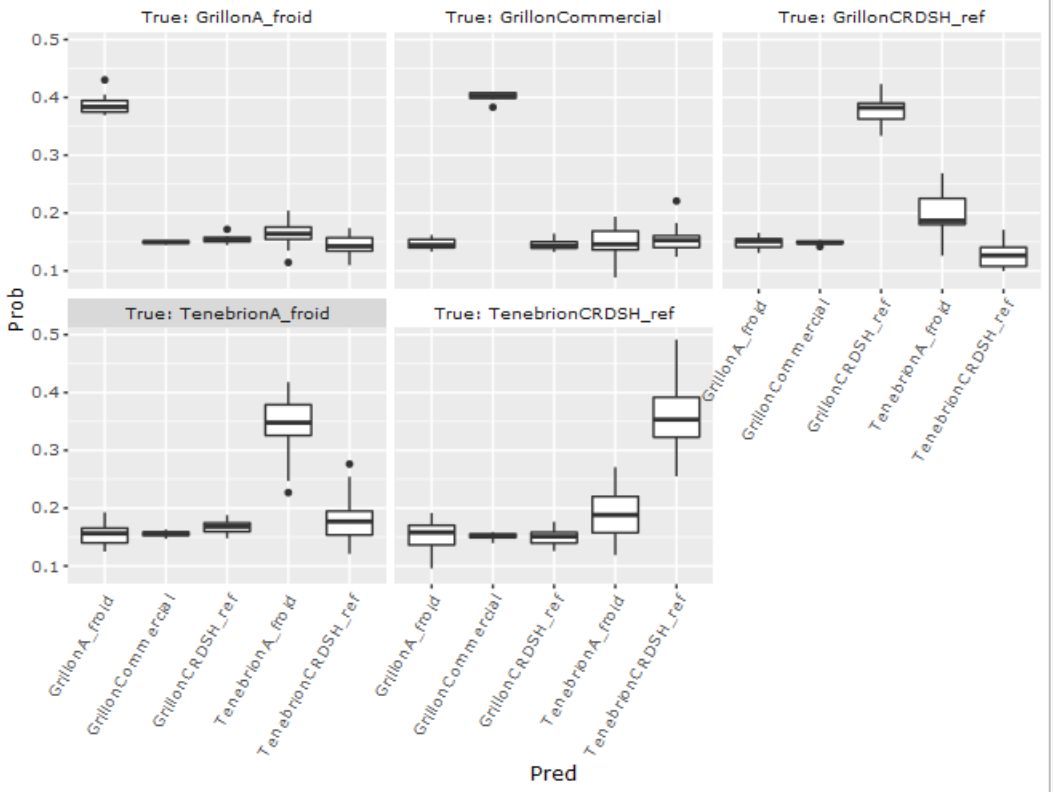
Validation / Test — Plot

Probs boxplot

Validation / Test — Plot

Show probs

Plot B-coeff



Console output

```
Specificity                 0.9730
Pos Pred Value              0.9231
Neg Pred Value              0.9863
Prevalence                  0.2525
Detection Rate              0.2424
Detection Prevalence        0.2626
Balanced Accuracy           0.9665
```

Console to txt | Pick an output file name | Output file name: D:/Bernard/Documents/Default_PLSDA_Ouptut.txt

InSpectoR 0.1.0 - March 2019

Data selection | Apply models | PrePro | PCA | PLSDA | PLS

Spectral data set for PCA
EX277

Estimated number of significant PCs
4

Choose plot type
Scores

Pick PC1 (X-axis for score biplot or first loading)
1

Pick PC2 (Y-axis for score biplot or last loading)
2

Point coloring options
Grouping for score plot: Classe_2

Point labeling options
Labels for score plot: None
Criteria: All

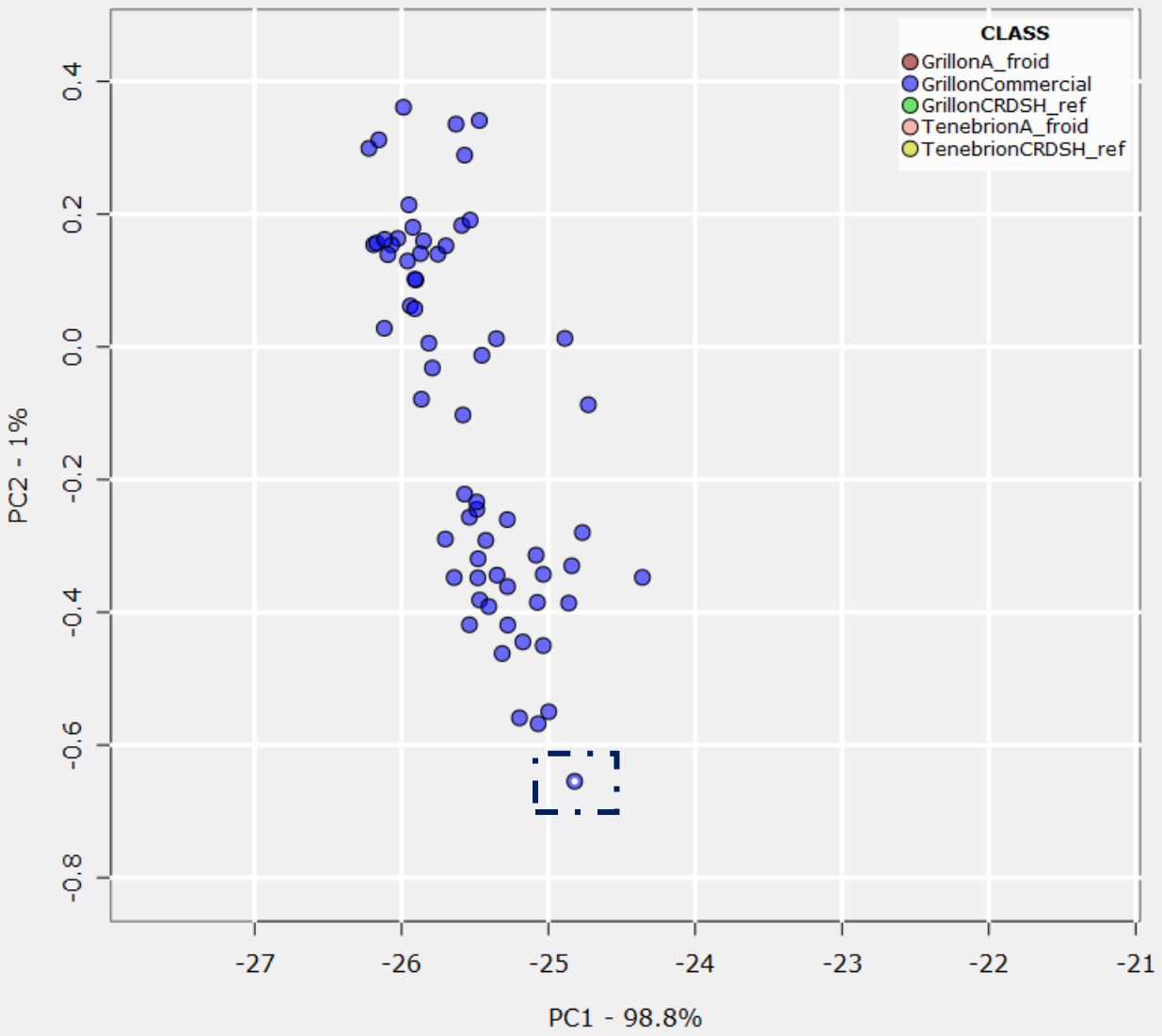
Data ellipse plotting
Pick option: None

Update plot

Remove selected from subset

Save scores

Save PCA model



Interaction de base avec graphiques

```
ggacp <- gWidgets2::ggraphics(cont=pca_tab,
                              width=100, height=100,
                              expand=TRUE,no_popup=TRUE)
#Handle to find sample associated with mouse click on graph
gWidgets2::addHandlerChanged(ggacp,interact_w_pca_plot)
```

```
#*********************************************************************************
interact_w_pca_plot <- function(h,...)
  # Manage user interaction with graphics
  # IF isSelectMode is TRUE (isZoomMode and IsZoomAll are false):
  #   This finds samples isolated by dragging a rectangle with mouse.
  #   Just clicking cancel selection.
```

```
> h
$`obj`
Object of class GGraphics

$action
NULL

$x
[1] -0.3218085  1.3297872

$y
[1]  6.886076 12.126582
```

```
}else   #this is a rectangle
{
  gWidgets2::enabled(subset_from_ACPPlotbut) <- TRUE
  #Find points in rectangle
  with (Sc_plot_params, {
          indices <- which(Xsc>h$x[1] & Xsc<h$x[2]
                           & Ysc>h$y[1] & Ysc<h$y[2])
          #redraw the points with a small black dot
          gWidgets2::visible(ggacp) <- TRUE
          points(Xsc[indices],Ysc[indices],pch=20, cex=0.75, col="white")
          ISR_env$SelectedScores <- c(indices,ISR_env$SelectedScores)
          ISR_env$SelectedScores=sort(ISR_env$SelectedScores[!duplicated(ISR_env$SelectedScores)])

        }
  )
```

Interaction de base avec graphiques

```
ggacp <- gWidgets2::ggraphics(cont=pca_tab,
                    width=100, height=100,
                    expand=TRUE,no_popup=TRUE)
#Handle to find sample associated with mouse click on graph
gWidgets2::addHandlerChanged(ggacp,interact_w_pca_plot)
```

```
#*************************************************************************
interact_w_pca_plot <- function(h,...)
  # Manage user interaction with graphics
  # IF isSelectMode is TRUE (isZoomMode and IsZoomAll are false):
  #   This finds samples isolated by dragging a rectangle with mouse.
  #   Just clicking cancel selection.
```

```
> h
$`obj`
Object of class GGraphics

$action
NULL

$x
[1] -0.3218085  1.3297872

$y
[1]  6.886076 12.126582
```

```
}else   #this is a rectangle
{
  gWidgets2::enabled(subset_from_ACPPlotbut) <- TRUE
  #Find points in rectangle
  with (Sc_plot_params, {
          indices <- which(Xsc>h$x[1] & Xsc<h$x[2]
                           & Ysc>h$y[1] & Ysc<h$y[2])
          #redraw the points with a small black dot
          gWidgets2::visible(ggacp) <- TRUE
          points(Xsc[indices],Ysc[indices],pch=20, cex=0.75, col="white")
          ISR_env$SelectedScores <- c(indices,ISR_env$SelectedScores)
          ISR_env$SelectedScores=sort(ISR_env$SelectedScores[!duplicated(ISR_env$SelectedScores)])

        }
  )
```

Interaction de base avec graphiques

```
ggacp <- gWidgets2::ggraphics(cont=pca_tab,
                    width=100, height=100,
                    expand=TRUE,no_popup=TRUE)
#Handle to find sample associated with mouse click on graph
gWidgets2::addHandlerChanged(ggacp,interact_w_pca_plot)
```

```
#*************************************************************************
interact_w_pca_plot <- function(h,...)
  # Manage user interaction with graphics
  # IF isSelectMode is TRUE (isZoomMode and IsZoomAll are false):
  #   This finds samples isolated by dragging a rectangle with mouse.
  #   Just clicking cancel selection.
```

```
> h
$`obj`
Object of class GGraphics

$action
NULL

$x
[1] -0.3218085  1.3297872

$y
[1]  6.886076 12.126582
```

```
}else   #this is a rectangle
{
  gWidgets2::enabled(subset_from_ACPPlotbut) <- TRUE
  #Find points in rectangle
  with (Sc_plot_params, {
          indices <- which(Xsc>h$x[1] & Xsc<h$x[2]
                           & Ysc>h$y[1] & Ysc<h$y[2])
          #redraw the points with a small black dot
          gWidgets2::visible(ggacp) <- TRUE
          points(Xsc[indices],Ysc[indices],pch=20, cex=0.75, col="white")
          ISR_env$SelectedScores <- c(indices,ISR_env$SelectedScores)
          ISR_env$SelectedScores=sort(ISR_env$SelectedScores[!duplicated(ISR_env$SelectedScores)])

        }
  )
```

Package inspectrar

https://github.com/PannetonB/inspectrar

Using InSpectoR

A GUI for working with spectral data

Bernard Panneton

2019-05-01

pannetonb@gmail.com

https://github.com/PannetonB/inspectrar