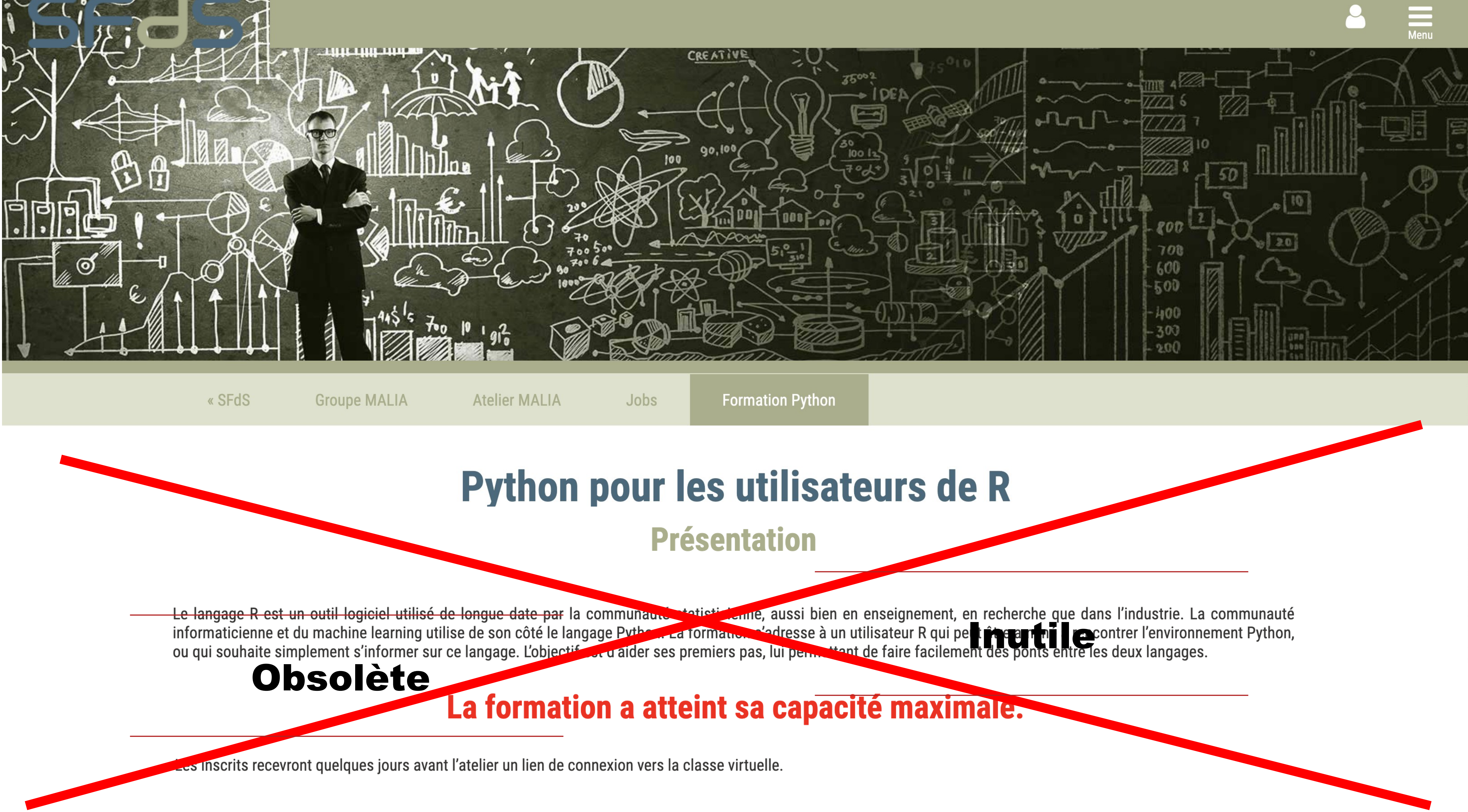
Les packages torch et tabnet !

Plus besoin de python ?

Christophe Regouby

18 octobre 2021



Python pour les utilisateurs de R

Présentation

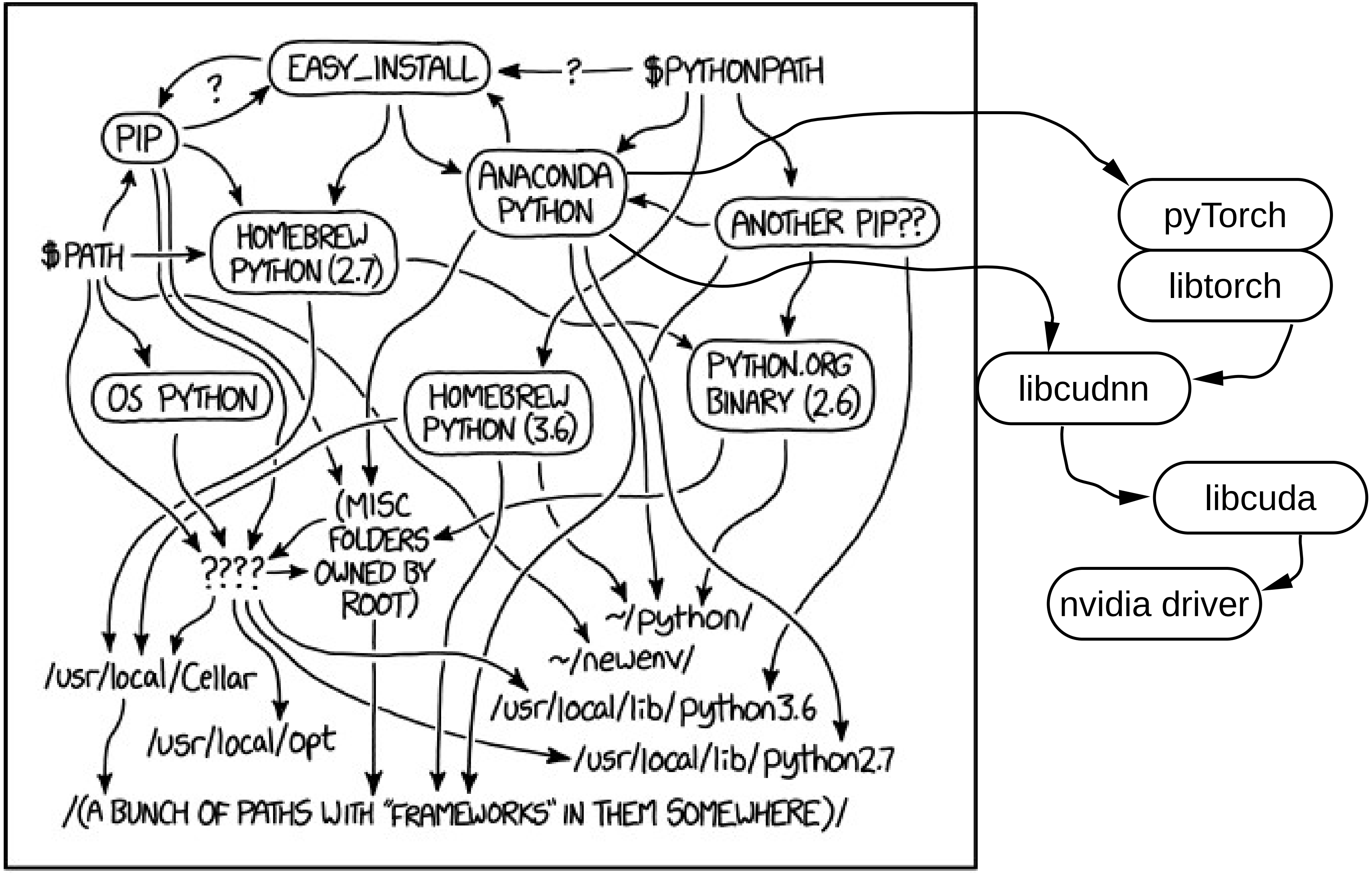
Le langage R est un outil logiciel utilisé de longue date par la communauté statisticienne, aussi bien en enseignement, en recherche que dans l'industrie. La communauté informaticienne et du machine learning utilise de son côté le langage Python. La formation s'adresse à un utilisateur R qui peut être amené à rencontrer l'environnement Python, ou qui souhaite simplement s'informer sur ce langage. L'objectif est d'aider ses premiers pas, lui permettant de faire facilement des ponts entre les deux langages.

Inutile

Obsolète

La formation a atteint sa capacité maximale.

Les inscrits recevront quelques jours avant l'atelier un lien de connexion vers la classe virtuelle.



pyTorch

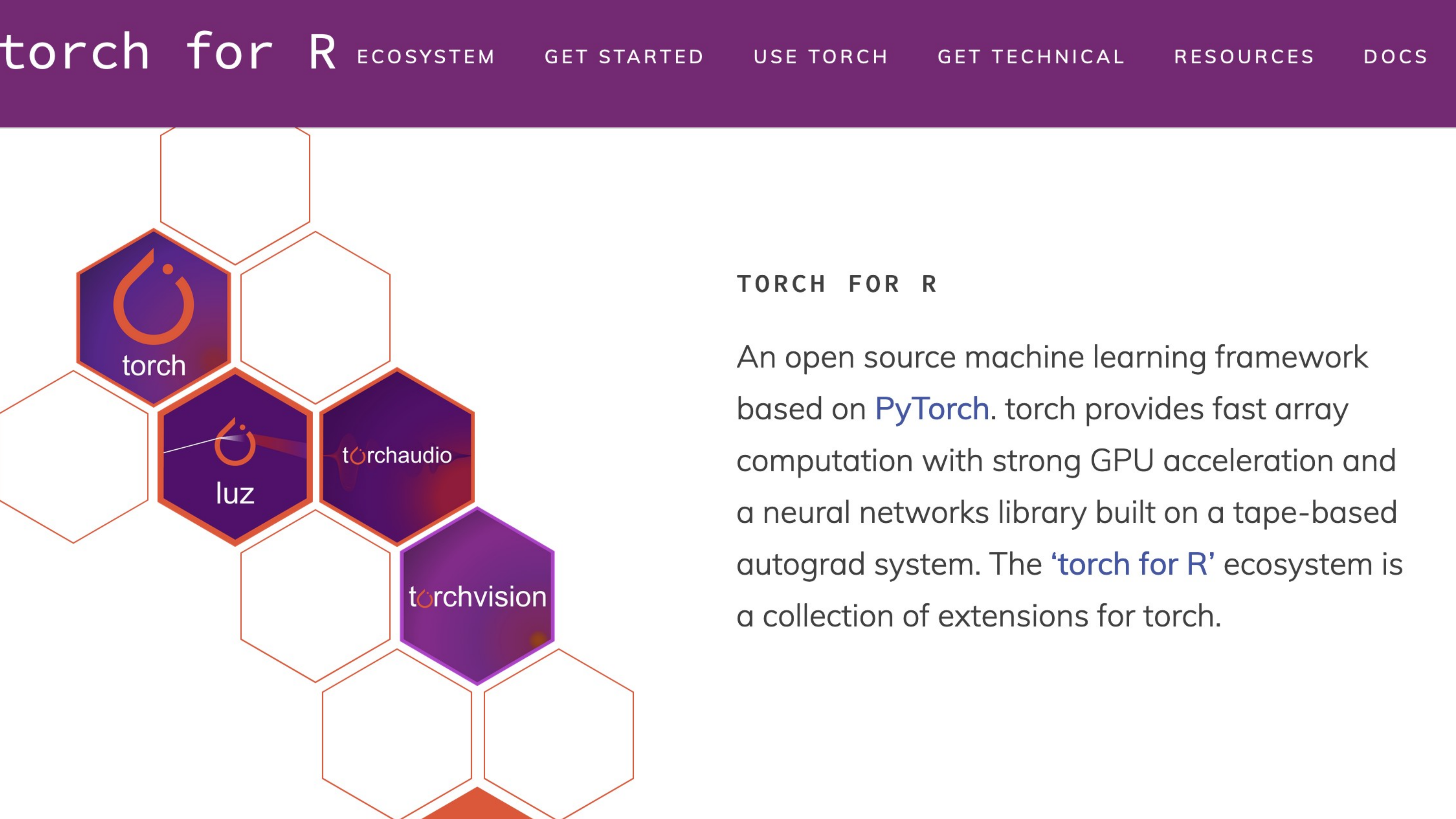
libtorch

libcudnn

libcuda

nvidia driver

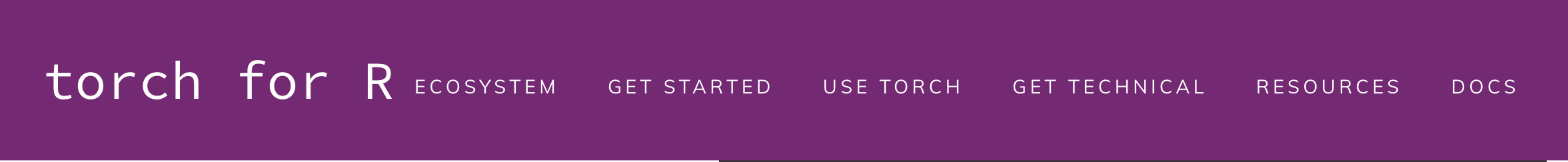
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.



torch for R ECOSYSTEM GET STARTED USE TORCH GET TECHNICAL RESOURCES DOCS

TORCH FOR R

An open source machine learning framework based on PyTorch. torch provides fast array computation with strong GPU acceleration and a neural networks library built on a tape-based autograd system. The ‘torch for R’ ecosystem is a collection of extensions for torch.



torch for R

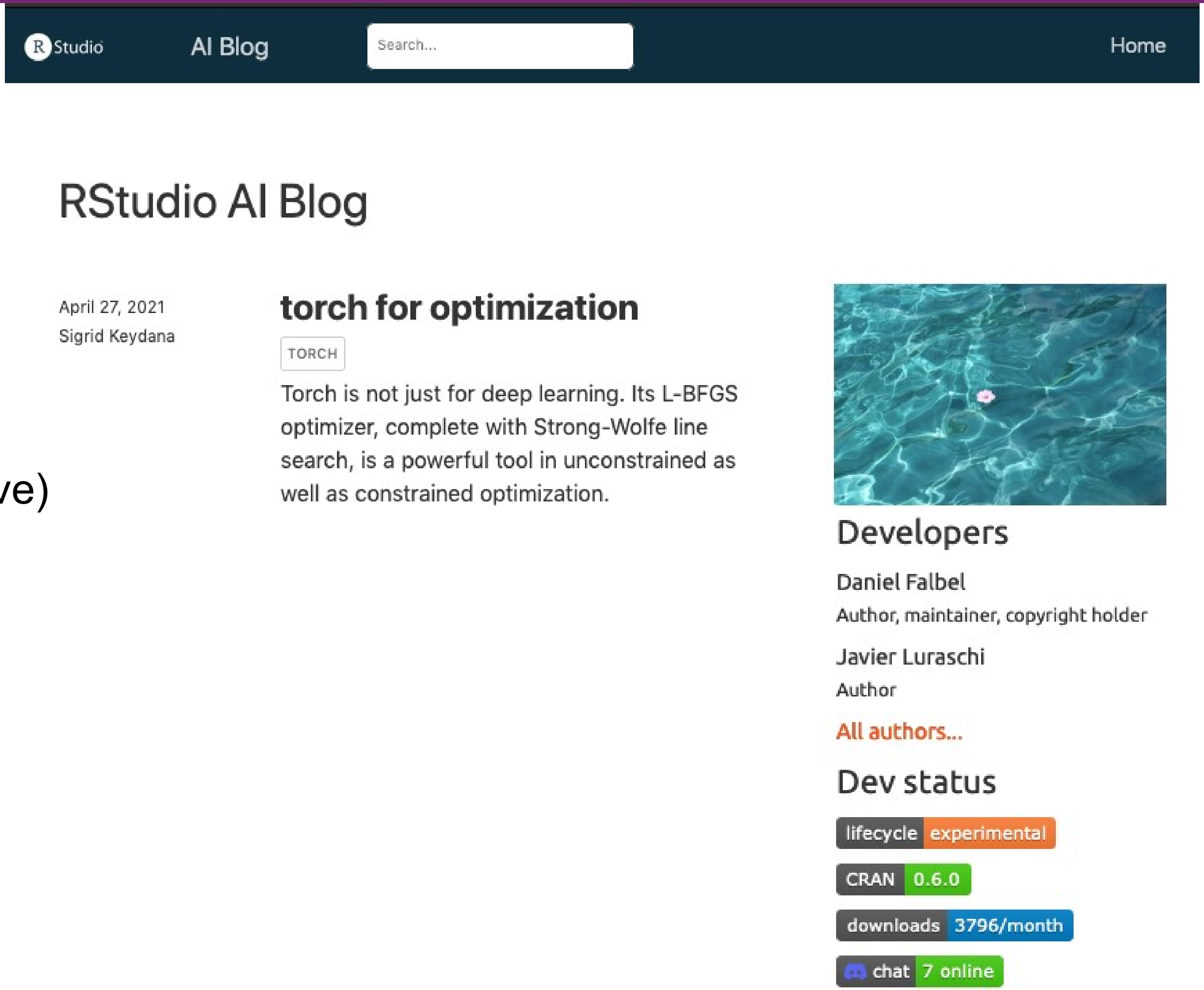
Pourquoi réinventer l’eau chaude ?

- facilité d’installation sur CPU et GPU

- frugalité d’installation

- la qualité des articles de blog de RStudio AI

- l’écosystème de packages (en construction active)



RStudio AI Blog

April 27, 2021
Sigrid Keydana

torch for optimization

TORCH

Torch is not just for deep learning. Its L-BFGS optimizer, complete with Strong-Wolfe line search, is a powerful tool in unconstrained as well as constrained optimization.

Developers

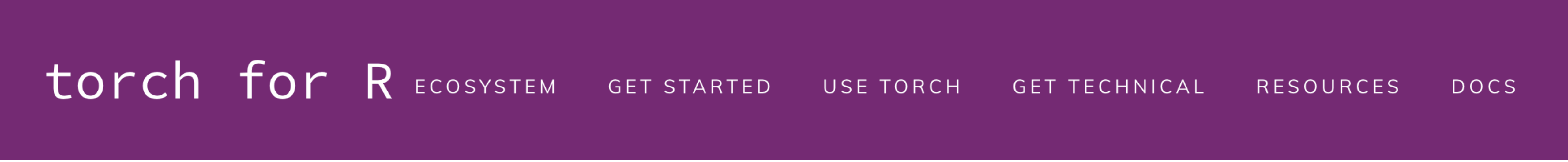
Daniel Falbel
Author, maintainer, copyright holder

Javier Luraschi
Author

All authors...

Dev status

lifecycle	experimental
CRAN	0.6.0
downloads	3796/month
chat	7 online



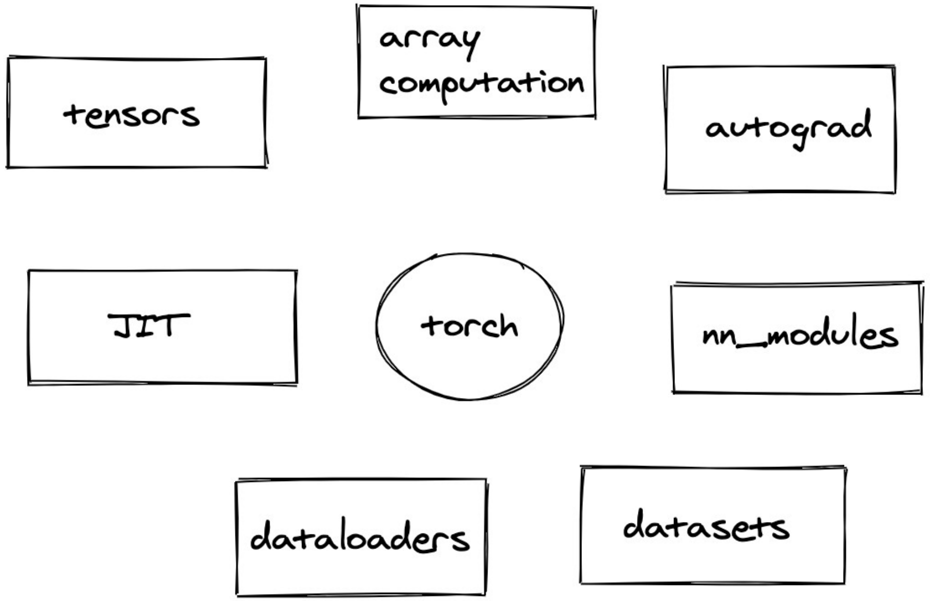
torch for R ECOSYSTEM GET STARTED USE TORCH GET TECHNICAL RESOURCES DOCS

Les qualités de torch

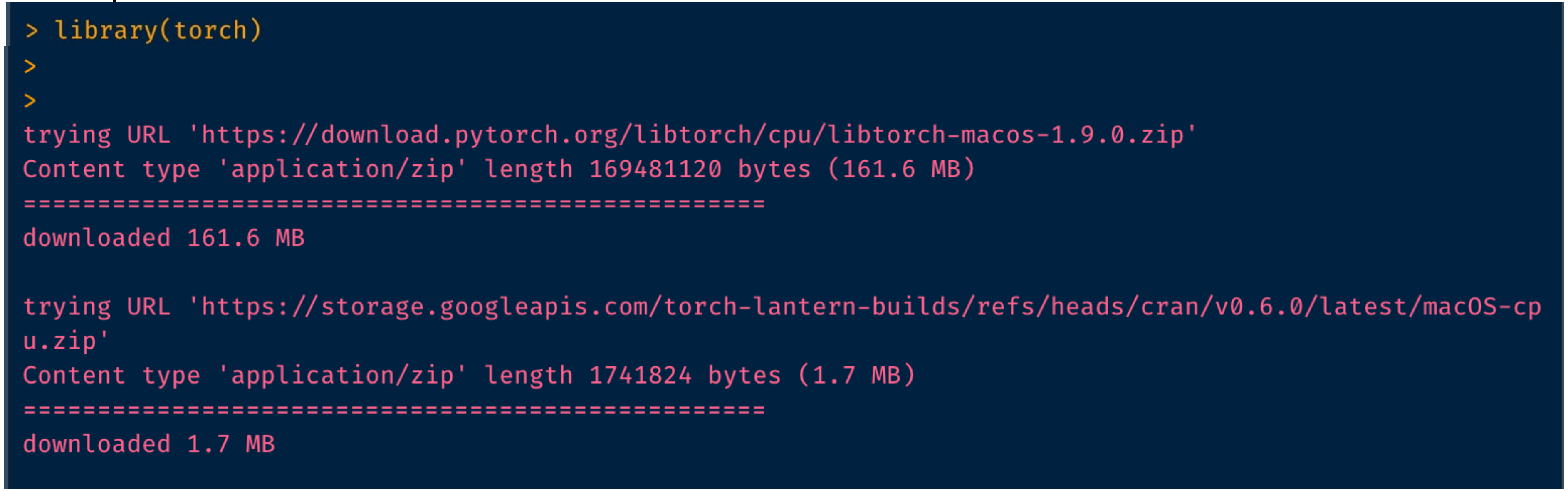
- confort de RStudio pour developper / debugger / visualiser

- le confort de R pour l’indexation à 1

- la différentiation automatique avec autograd

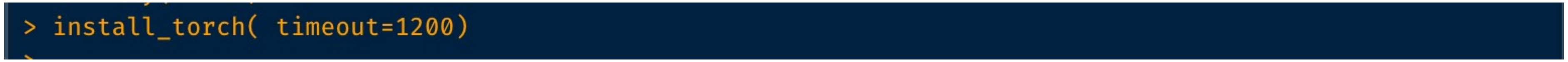


Setup



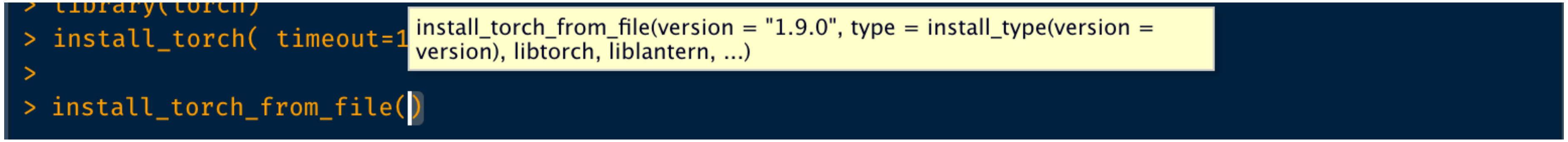
```
> library(torch)
>
>
trying URL 'https://download.pytorch.org/libtorch/cpu/libtorch-macos-1.9.0.zip'
Content type 'application/zip' length 169481120 bytes (161.6 MB)
==================================================
downloaded 161.6 MB

trying URL 'https://storage.googleapis.com/torch-lantern-builds/refs/heads/cran/v0.6.0/latest/macOS-cp
u.zip'
Content type 'application/zip' length 1741824 bytes (1.7 MB)
==================================================
downloaded 1.7 MB
```

Advanced setup



```
> install_torch( timeout=1200)
```

Expert setup

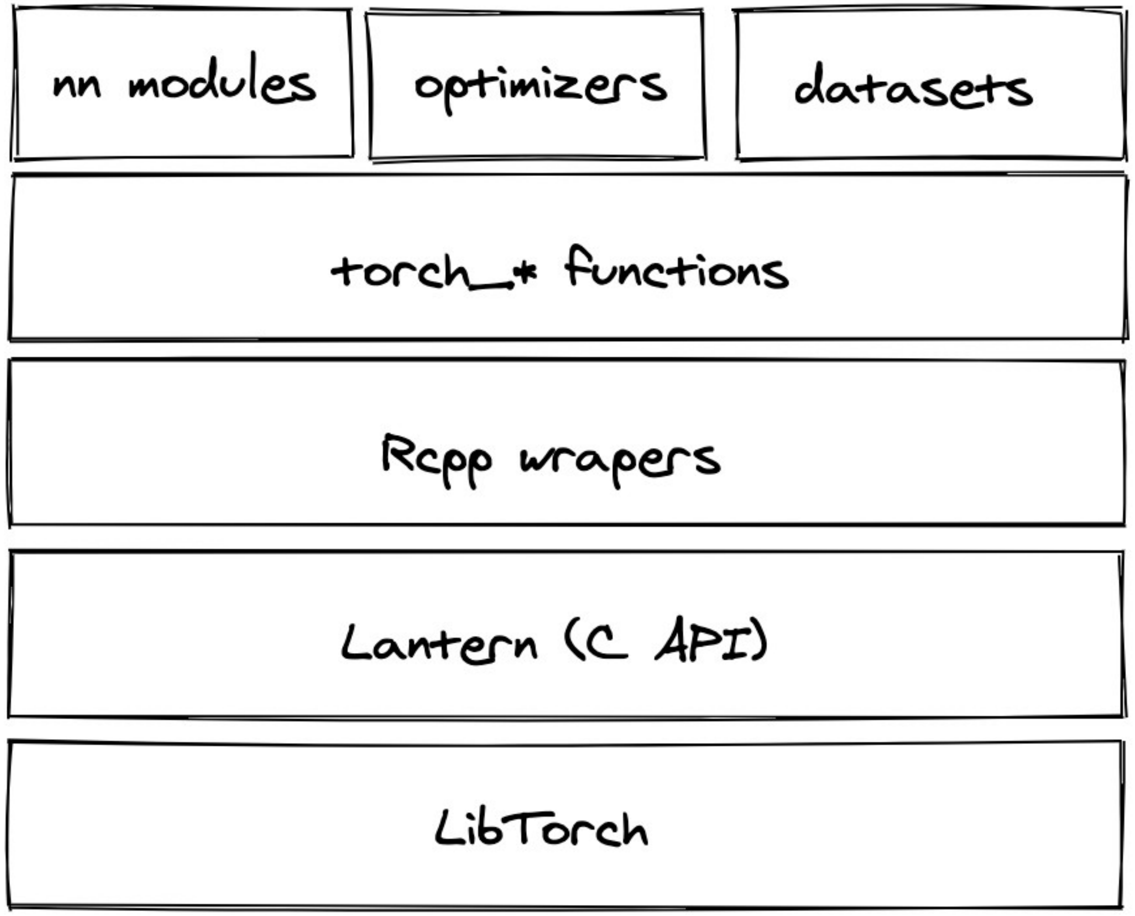


```
> library(torch)
> install_torch( timeout=1
>
> install_torch_from_file()
```

install_torch_from_file(version = "1.9.0", type = install_type(version = version), libtorch, liblantern, ...)

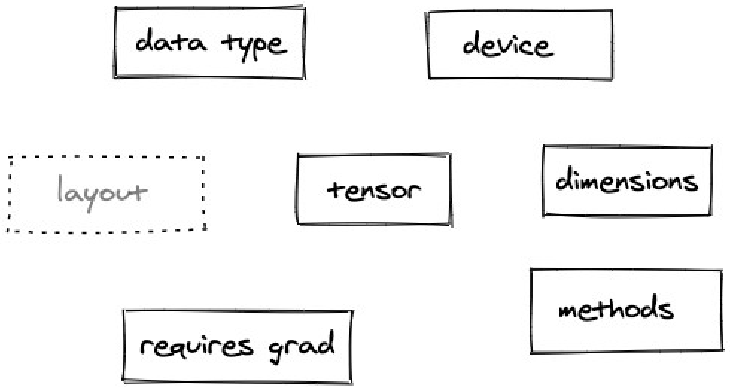
https://torch.mlverse.org/docs/articles/installation.html

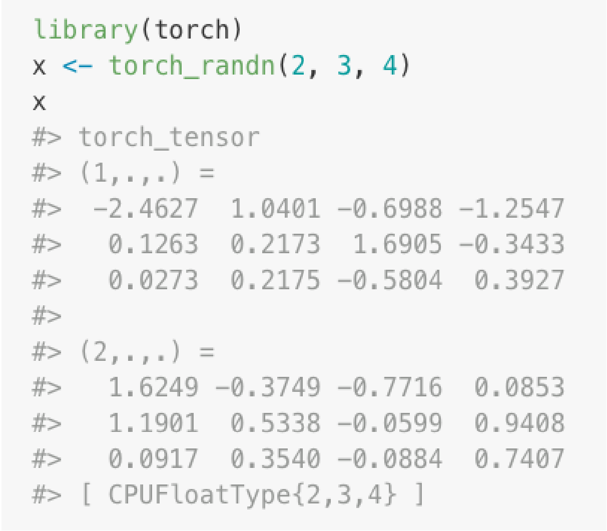
Design logiciel



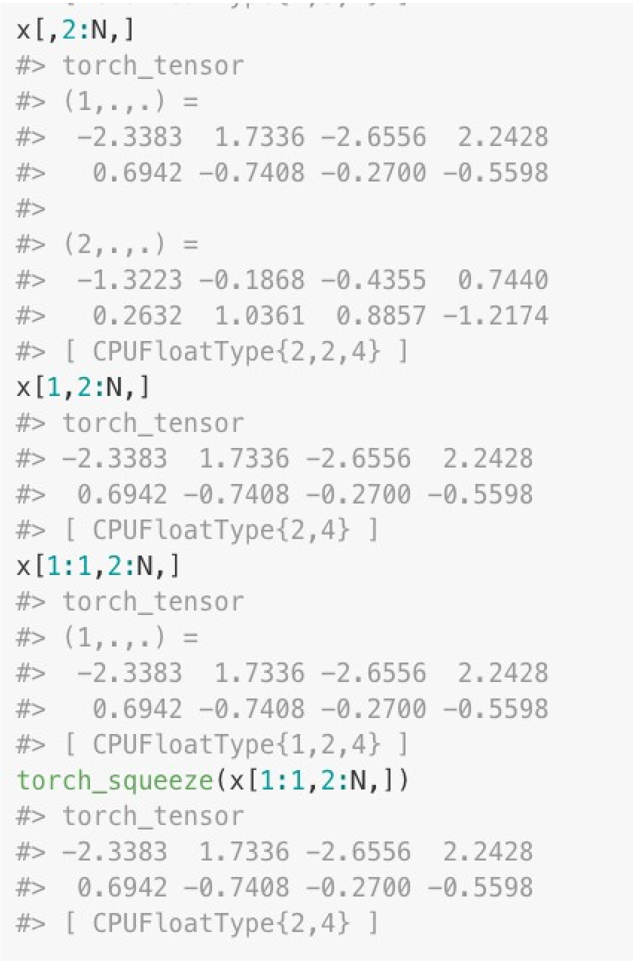
Made with Excalidraw

La manipulation des tensors dans torch





```
library(torch)
x <- torch_randn(2, 3, 4)
x
#> torch_tensor
#> (1,.,.) =
#>  -2.4627  1.0401 -0.6988 -1.2547
#>   0.1263  0.2173  1.6905 -0.3433
#>   0.0273  0.2175 -0.5804  0.3927
#>
#> (2,.,.) =
#>   1.6249 -0.3749 -0.7716  0.0853
#>   1.1901  0.5338 -0.0599  0.9408
#>   0.0917  0.3540 -0.0884  0.7407
#> [ CPUFloatType{2,3,4} ]
```



```
x[,2:N,]
#> torch_tensor
#> (1,.,.) =
#>  -2.3383  1.7336 -2.6556  2.2428
#>   0.6942 -0.7408 -0.2700 -0.5598
#>
#> (2,.,.) =
#>  -1.3223 -0.1868 -0.4355  0.7440
#>   0.2632  1.0361  0.8857 -1.2174
#> [ CPUFloatType{2,2,4} ]
x[1,2:N,]
#> torch_tensor
#> -2.3383  1.7336 -2.6556  2.2428
#>  0.6942 -0.7408 -0.2700 -0.5598
#> [ CPUFloatType{2,4} ]
x[1:1,2:N,]
#> torch_tensor
#> (1,.,.) =
#>  -2.3383  1.7336 -2.6556  2.2428
#>   0.6942 -0.7408 -0.2700 -0.5598
#> [ CPUFloatType{1,2,4} ]
torch_squeeze(x[1:1,2:N,])
#> torch_tensor
#> -2.3383  1.7336 -2.6556  2.2428
#>  0.6942 -0.7408 -0.2700 -0.5598
#> [ CPUFloatType{2,4} ]
```

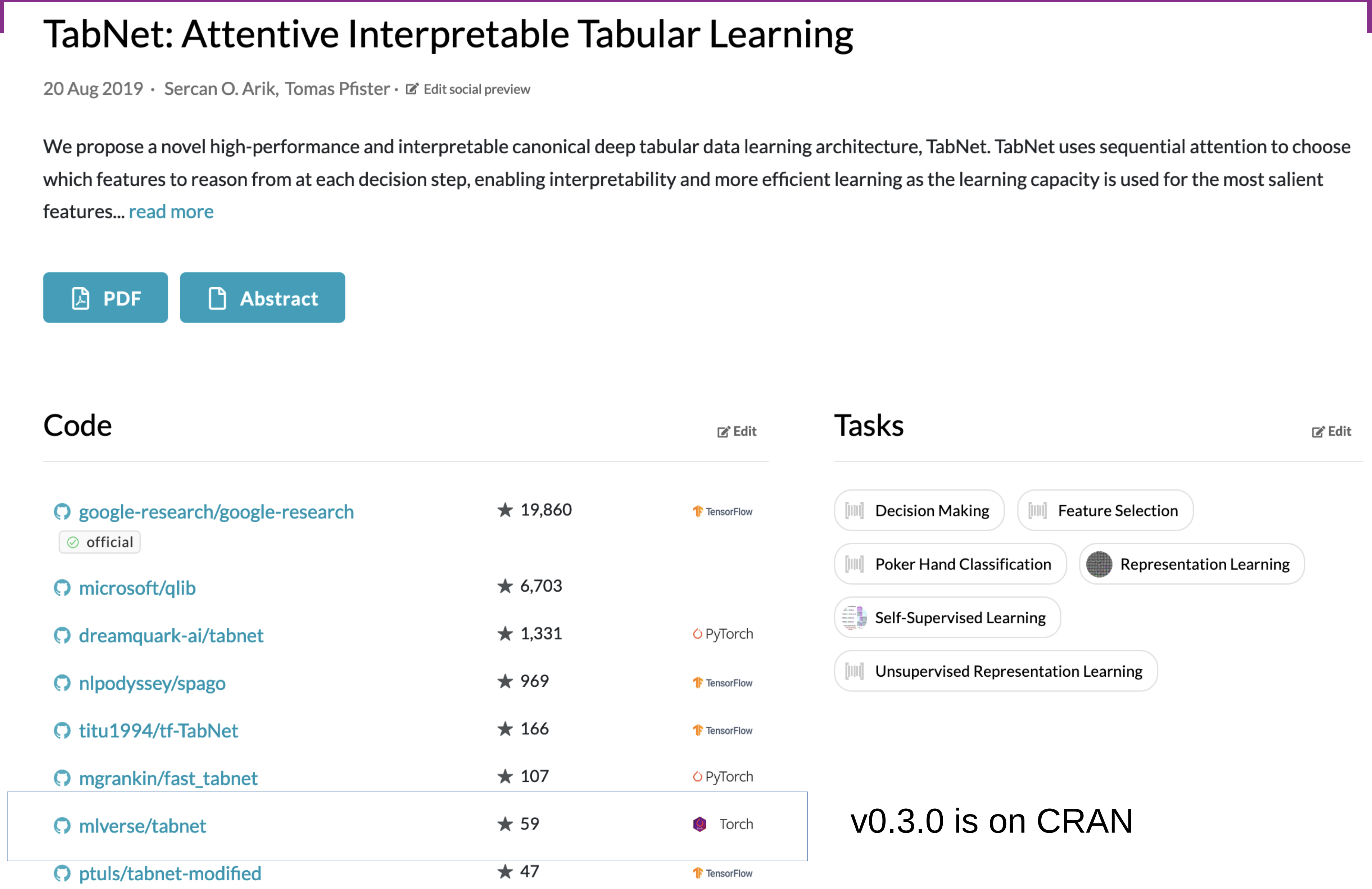
Mon premier module torch :

mlverse.shinyapps.io/torch-tour

Le tutorial Torch de UseR-2021 est en francais !

https://raw.githubusercontent.com/mlverse/torch-learnr/master/tutorial-useR-2021/fr/torch.Rmd

tabnet::



TabNet: Attentive Interpretable Tabular Learning

20 Aug 2019 · Sercan O. Arik, Tomas Pfister · Edit social preview

We propose a novel high-performance and interpretable canonical deep tabular data learning architecture, TabNet. TabNet uses sequential attention to choose which features to reason from at each decision step, enabling interpretability and more efficient learning as the learning capacity is used for the most salient features... read more

Code

google-research/google-research ✓ official	★ 19,860	TensorFlow
microsoft/qlib	★ 6,703	
dreamquark-ai/tabnet	★ 1,331	PyTorch
nlpodyssey/spago	★ 969	TensorFlow
titu1994/tf-TabNet	★ 166	TensorFlow
mgrankin/fast_tabnet	★ 107	PyTorch
mlverse/tabnet	★ 59	Torch
ptuls/tabnet-modified	★ 47	TensorFlow

Tasks

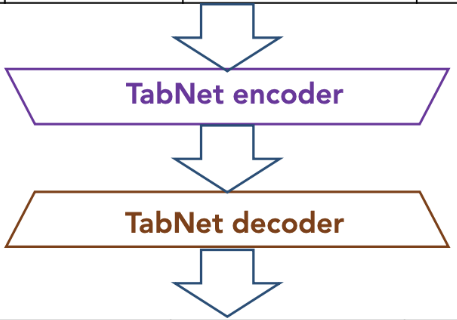
- Decision Making
- Feature Selection
- Poker Hand Classification
- Representation Learning
- Self-Supervised Learning
- Unsupervised Representation Learning

v0.3.0 is on CRAN

tabnet::

Unsupervised pre-training

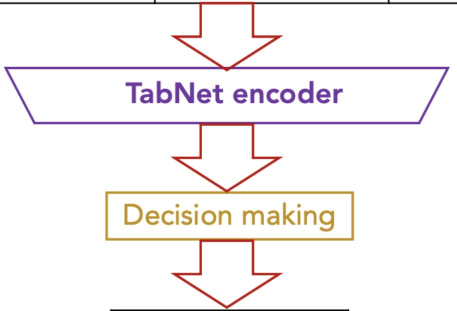
Age	Cap. gain	Education	Occupation	Gender	Relationship
53	200000	?	Exec-managerial	F	Wife
19	0	?	Farming-fishing	M	?
?	5000	Doctorate	Prof-specialty	M	Husband
25	?	?	Handlers-cleaners	F	Wife
59	300000	Bachelors	?	?	Husband
33	0	Bachelors	?	F	?
?	0	High-school	Armed-Forces	?	Husband



Age	Cap. gain	Education	Occupation	Gender	Relationship
		Masters			
		High-school			Unmarried
43					
	0	High-school		F	
			Exec-managerial	M	
			Adm-clerical		Wife
39				M	

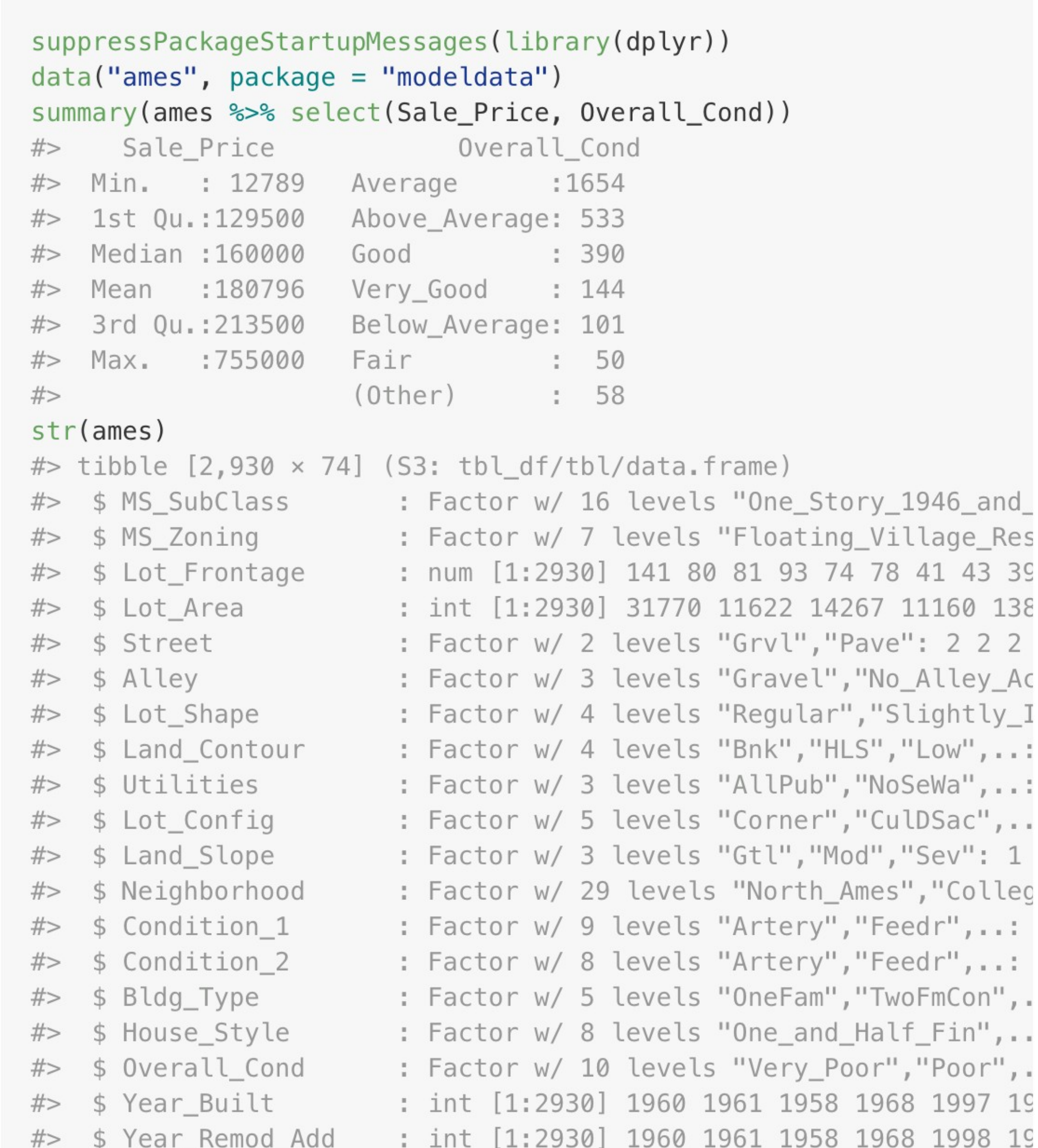
Supervised fine-tuning

Age	Cap. gain	Education	Occupation	Gender	Relationship
60	200000	Bachelors	Exec-managerial	M	Husband
23	0	High-school	Farming-fishing	M	Unmarried
45	5000	Doctorate	Prof-specialty	M	Husband
23	0	High-school	Handlers-cleaners	F	Wife
56	300000	Bachelors	Exec-managerial	M	Husband
38	10000	Bachelors	Prof-specialty	F	Wife
23	0	High-school	Armed-Forces	M	Husband



Income > $50k
True
False
True
False
True
True
False

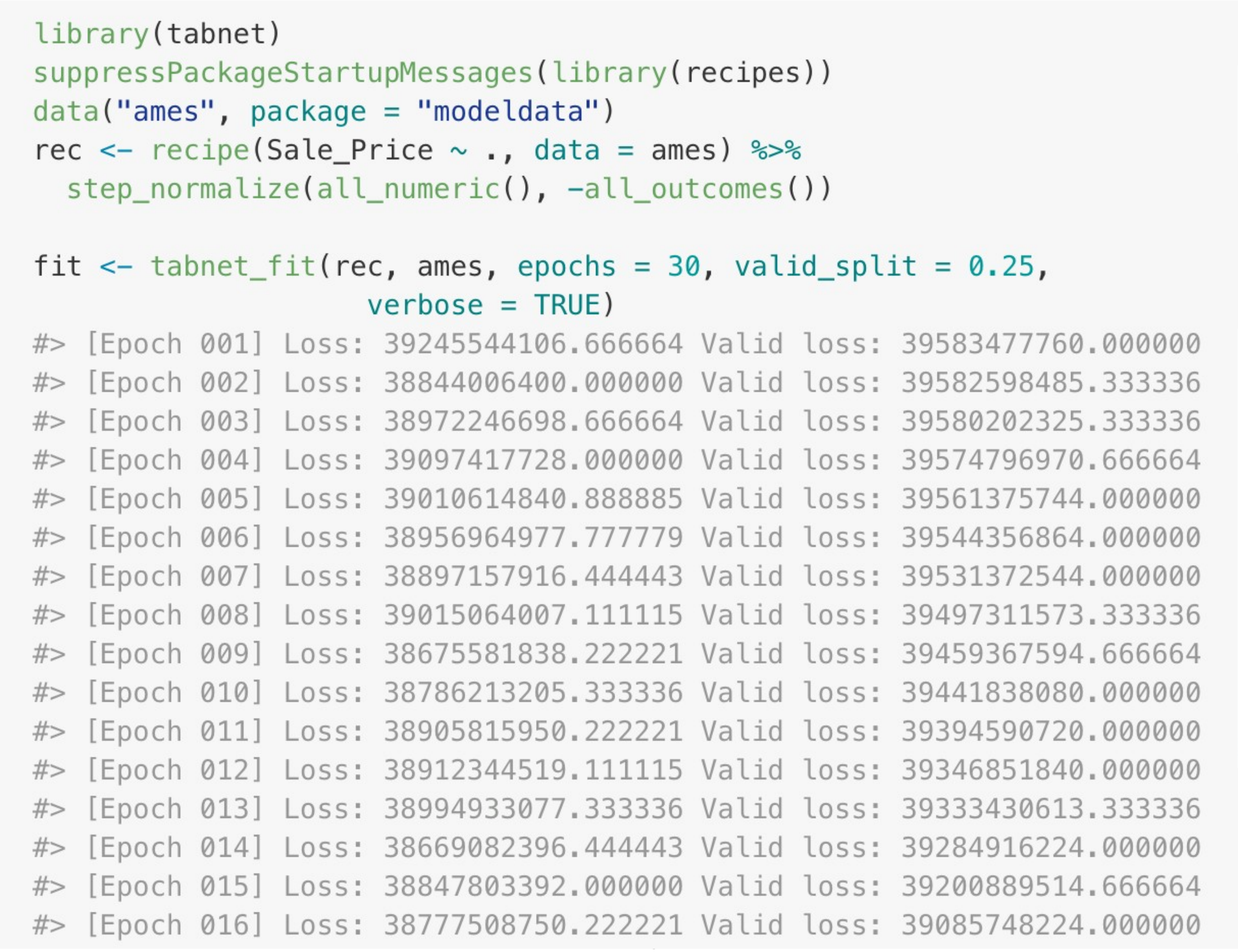
tabnet::   le jeu de données de la ville d’Ames



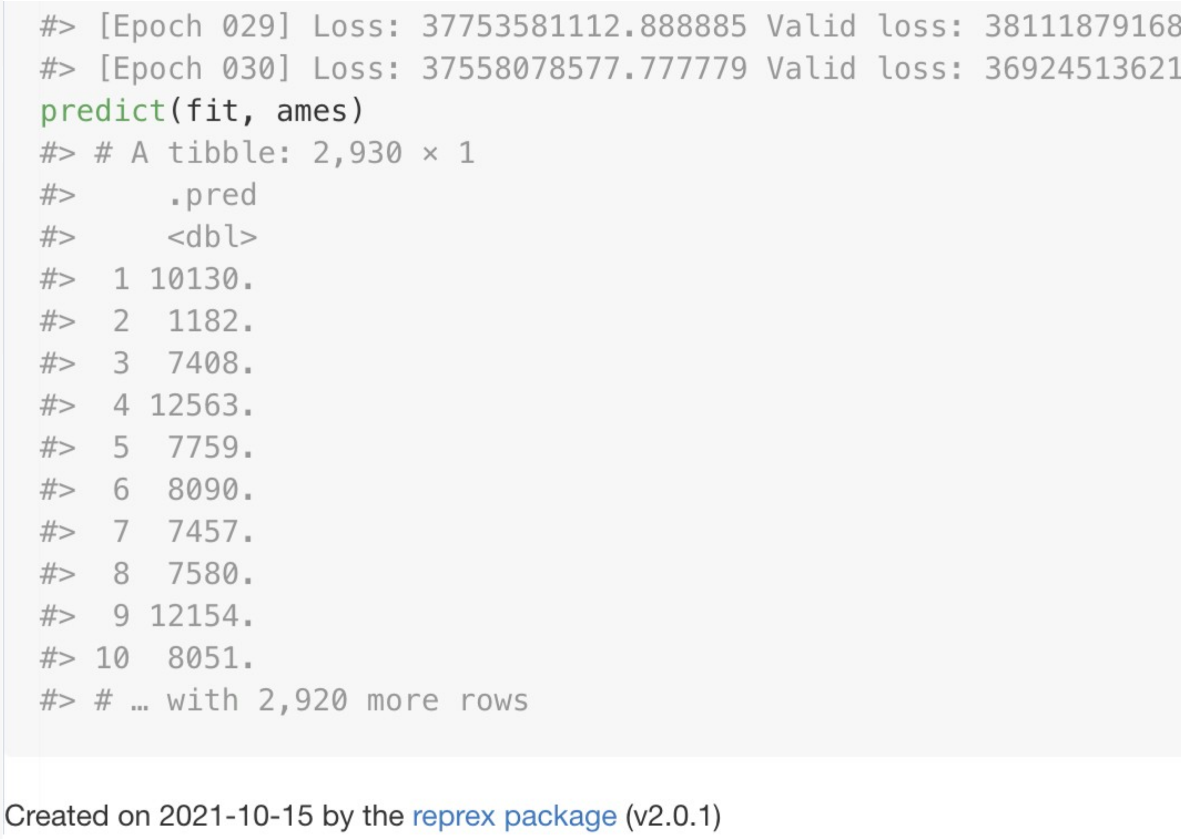
```
suppressPackageStartupMessages(library(dplyr))
data("ames", package = "modeldata")
summary(ames %>% select(Sale_Price, Overall_Cond))
#>    Sale_Price           Overall_Cond
#>  Min.   : 12789   Average      :1654
#>  1st Qu.:129500   Above_Average: 533
#>  Median :160000   Good         : 390
#>  Mean   :180796   Very_Good    : 144
#>  3rd Qu.:213500   Below_Average: 101
#>  Max.   :755000   Fair         :  50
#>                   (Other)      :  58
str(ames)
#> tibble [2,930 × 74] (S3: tbl_df/tbl/data.frame)
#>  $ MS_SubClass        : Factor w/ 16 levels "One_Story_1946_and_
#>  $ MS_Zoning          : Factor w/ 7 levels "Floating_Village_Res
#>  $ Lot_Frontage       : num [1:2930] 141 80 81 93 74 78 41 43 39
#>  $ Lot_Area           : int [1:2930] 31770 11622 14267 11160 138
#>  $ Street             : Factor w/ 2 levels "Grvl","Pave": 2 2 2
#>  $ Alley              : Factor w/ 3 levels "Gravel","No_Alley_Ac
#>  $ Lot_Shape          : Factor w/ 4 levels "Regular","Slightly_I
#>  $ Land_Contour       : Factor w/ 4 levels "Bnk","HLS","Low",..:
#>  $ Utilities          : Factor w/ 3 levels "AllPub","NoSeWa",..:
#>  $ Lot_Config         : Factor w/ 5 levels "Corner","CulDSac",..
#>  $ Land_Slope         : Factor w/ 3 levels "Gtl","Mod","Sev": 1
#>  $ Neighborhood       : Factor w/ 29 levels "North_Ames","Colleg
#>  $ Condition_1        : Factor w/ 9 levels "Artery","Feedr",..:
#>  $ Condition_2        : Factor w/ 8 levels "Artery","Feedr",..:
#>  $ Bldg_Type          : Factor w/ 5 levels "OneFam","TwoFmCon",.
#>  $ House_Style        : Factor w/ 8 levels "One_and_Half_Fin",..
#>  $ Overall_Cond       : Factor w/ 10 levels "Very_Poor","Poor",.
#>  $ Year_Built         : int [1:2930] 1960 1961 1958 1968 1997 19
#>  $ Year_Remod_Add     : int [1:2930] 1960 1961 1958 1968 1998 19
```

tabnet:: intégration dans le flux de modélisation

recipe:: supervised training, regression



```r
library(tabnet)
suppressPackageStartupMessages(library(recipes))
data("ames", package = "modeldata")
rec <- recipe(Sale_Price ~ ., data = ames) %>%
  step_normalize(all_numeric(), -all_outcomes())

fit <- tabnet_fit(rec, ames, epochs = 30, valid_split = 0.25,
                  verbose = TRUE)
#> [Epoch 001] Loss: 39245544106.666664 Valid loss: 39583477760.000000
#> [Epoch 002] Loss: 38844006400.000000 Valid loss: 39582598485.333336
#> [Epoch 003] Loss: 38972246698.666664 Valid loss: 39580202325.333336
#> [Epoch 004] Loss: 39097417728.000000 Valid loss: 39574796970.666664
#> [Epoch 005] Loss: 39010614840.888885 Valid loss: 39561375744.000000
#> [Epoch 006] Loss: 38956964977.777779 Valid loss: 39544356864.000000
#> [Epoch 007] Loss: 38897157916.444443 Valid loss: 39531372544.000000
#> [Epoch 008] Loss: 39015064007.111115 Valid loss: 39497311573.333336
#> [Epoch 009] Loss: 38675581838.222221 Valid loss: 39459367594.666664
#> [Epoch 010] Loss: 38786213205.333336 Valid loss: 39441838080.000000
#> [Epoch 011] Loss: 38905815950.222221 Valid loss: 39394590720.000000
#> [Epoch 012] Loss: 38912344519.111115 Valid loss: 39346851840.000000
#> [Epoch 013] Loss: 38994933077.333336 Valid loss: 39333430613.333336
#> [Epoch 014] Loss: 38669082396.444443 Valid loss: 39284916224.000000
#> [Epoch 015] Loss: 38847803392.000000 Valid loss: 39200889514.666664
#> [Epoch 016] Loss: 38777508750.222221 Valid loss: 39085748224.000000
```

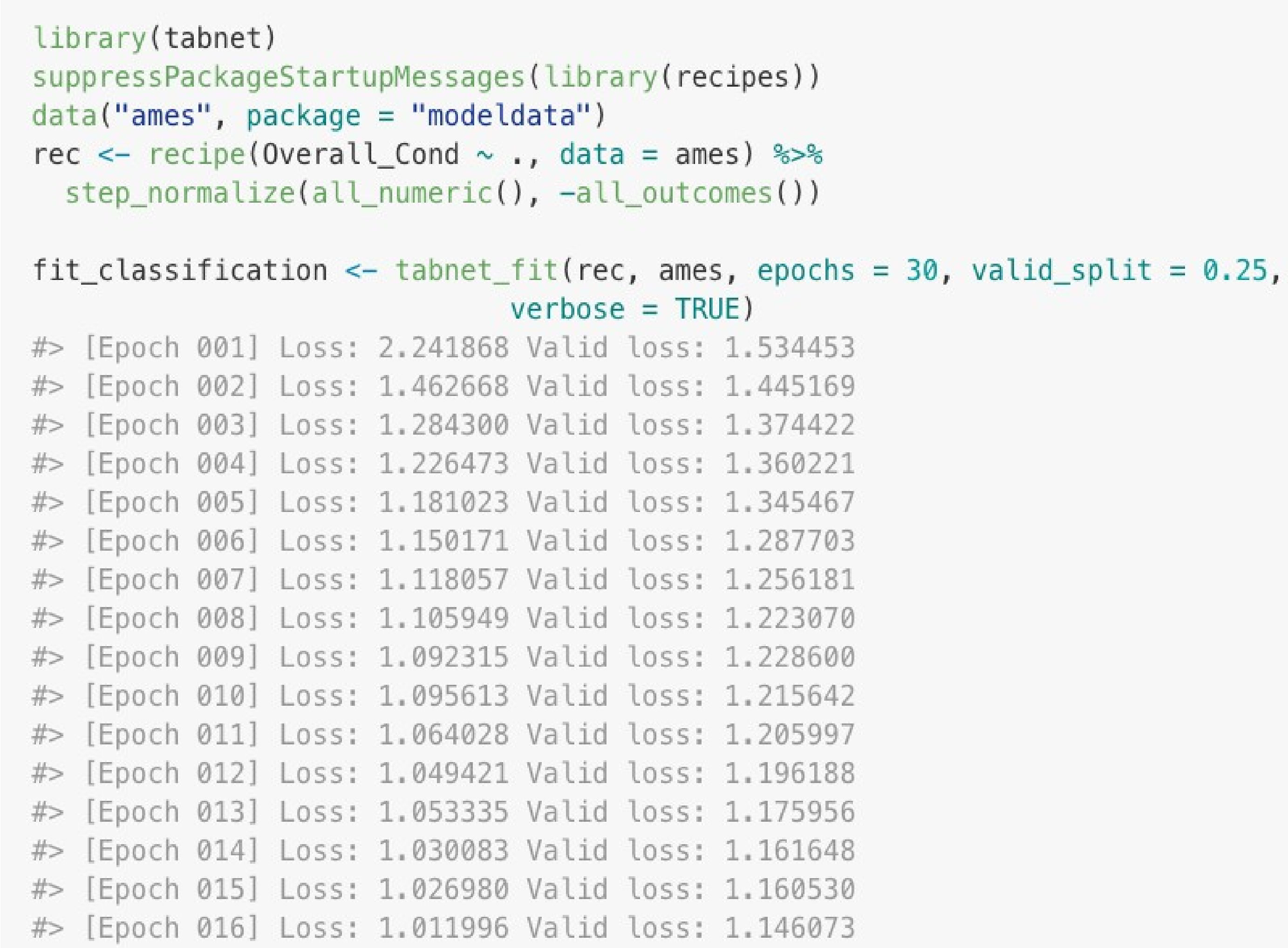


```r
#> [Epoch 029] Loss: 37753581112.888885 Valid loss: 38111879168
#> [Epoch 030] Loss: 37558078577.777779 Valid loss: 36924513621
predict(fit, ames)
#> # A tibble: 2,930 × 1
#>      .pred
#>      <dbl>
#>  1 10130.
#>  2  1182.
#>  3  7408.
#>  4 12563.
#>  5  7759.
#>  6  8090.
#>  7  7457.
#>  8  7580.
#>  9 12154.
#> 10  8051.
#> # … with 2,920 more rows
```

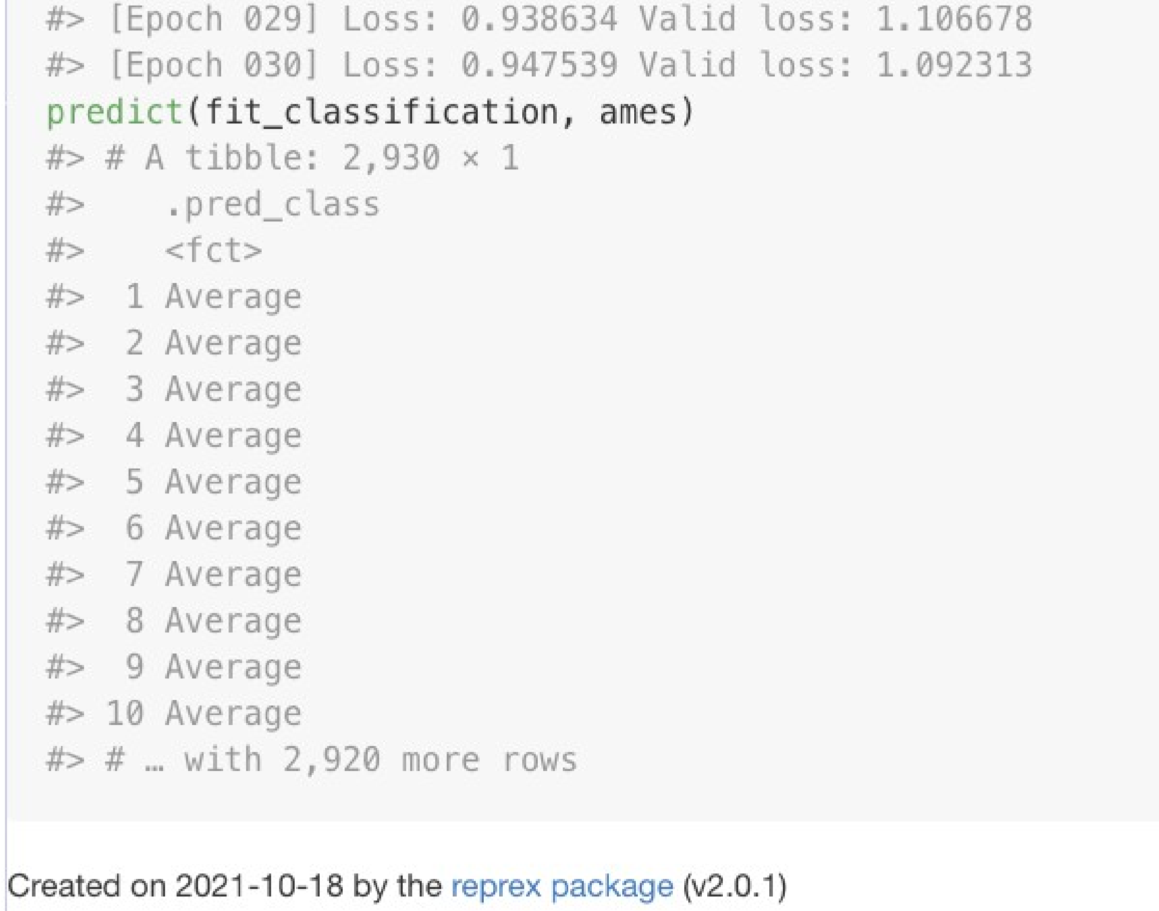
Created on 2021-10-15 by the reprex package (v2.0.1)

tabnet:: intégration dans le flux de modélisation

recipe:: supervised training, classification



```
library(tabnet)
suppressPackageStartupMessages(library(recipes))
data("ames", package = "modeldata")
rec <- recipe(Overall_Cond ~ ., data = ames) %>%
  step_normalize(all_numeric(), -all_outcomes())

fit_classification <- tabnet_fit(rec, ames, epochs = 30, valid_split = 0.25,
                                 verbose = TRUE)
#> [Epoch 001] Loss: 2.241868 Valid loss: 1.534453
#> [Epoch 002] Loss: 1.462668 Valid loss: 1.445169
#> [Epoch 003] Loss: 1.284300 Valid loss: 1.374422
#> [Epoch 004] Loss: 1.226473 Valid loss: 1.360221
#> [Epoch 005] Loss: 1.181023 Valid loss: 1.345467
#> [Epoch 006] Loss: 1.150171 Valid loss: 1.287703
#> [Epoch 007] Loss: 1.118057 Valid loss: 1.256181
#> [Epoch 008] Loss: 1.105949 Valid loss: 1.223070
#> [Epoch 009] Loss: 1.092315 Valid loss: 1.228600
#> [Epoch 010] Loss: 1.095613 Valid loss: 1.215642
#> [Epoch 011] Loss: 1.064028 Valid loss: 1.205997
#> [Epoch 012] Loss: 1.049421 Valid loss: 1.196188
#> [Epoch 013] Loss: 1.053335 Valid loss: 1.175956
#> [Epoch 014] Loss: 1.030083 Valid loss: 1.161648
#> [Epoch 015] Loss: 1.026980 Valid loss: 1.160530
#> [Epoch 016] Loss: 1.011996 Valid loss: 1.146073
```

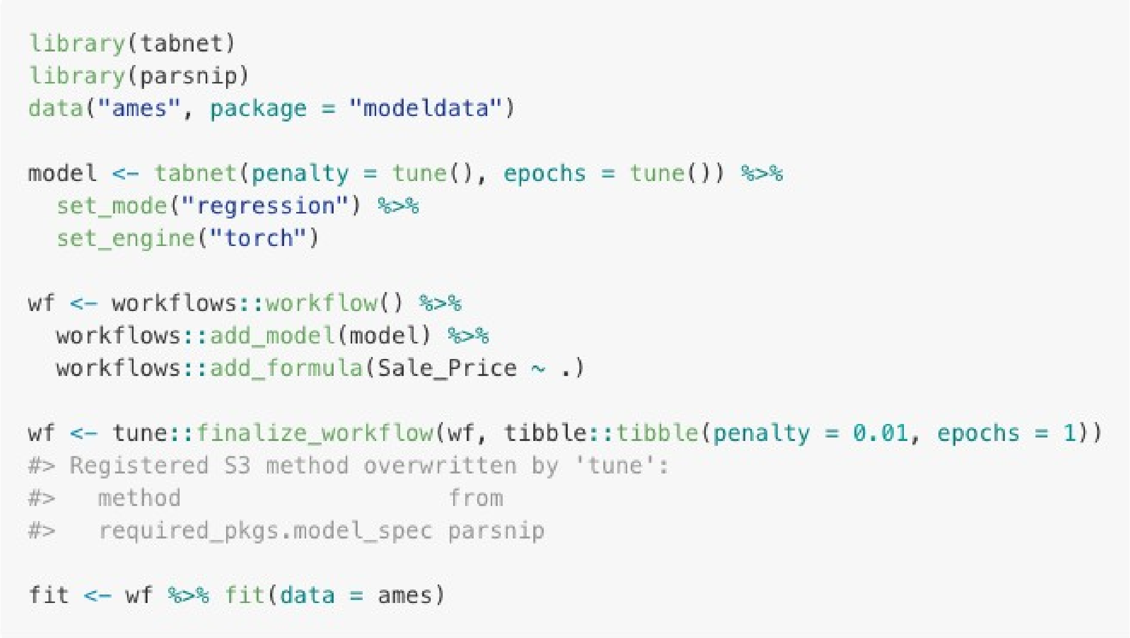


```
#> [Epoch 029] Loss: 0.938634 Valid loss: 1.106678
#> [Epoch 030] Loss: 0.947539 Valid loss: 1.092313
predict(fit_classification, ames)
#> # A tibble: 2,930 × 1
#>    .pred_class
#>    <fct>
#>  1 Average
#>  2 Average
#>  3 Average
#>  4 Average
#>  5 Average
#>  6 Average
#>  7 Average
#>  8 Average
#>  9 Average
#> 10 Average
#> # … with 2,920 more rows
```

Created on 2021-10-18 by the reprex package (v2.0.1)

tabnet:: intégration dans le flux de modélisation

workflow:: training

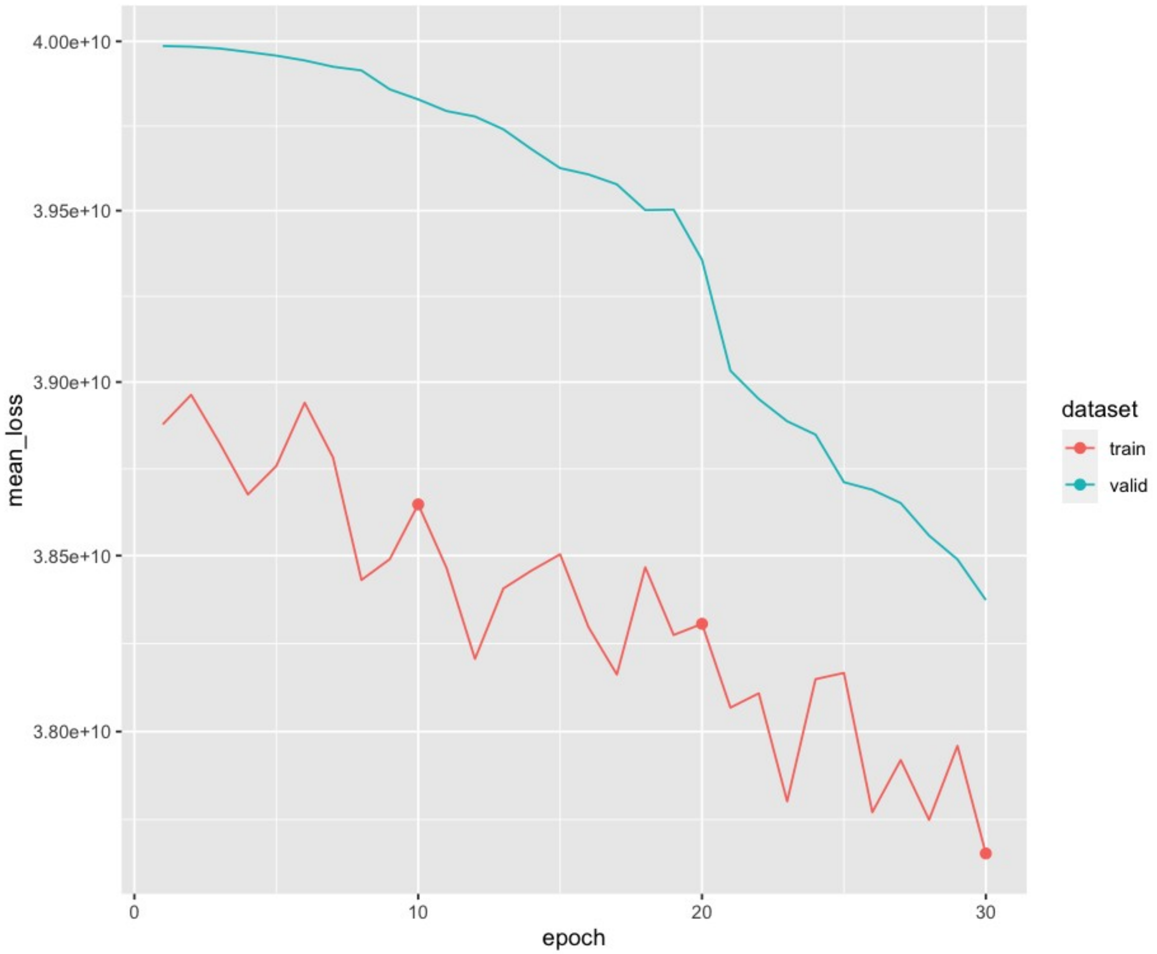


```
library(tabnet)
library(parsnip)
data("ames", package = "modeldata")

model <- tabnet(penalty = tune(), epochs = tune()) %>%
  set_mode("regression") %>%
  set_engine("torch")

wf <- workflows::workflow() %>%
  workflows::add_model(model) %>%
  workflows::add_formula(Sale_Price ~ .)

wf <- tune::finalize_workflow(wf, tibble::tibble(penalty = 0.01, epochs = 1))
#> Registered S3 method overwritten by 'tune':
#>   method                   from
#>   required_pkgs.model_spec parsnip

fit <- wf %>% fit(data = ames)
```

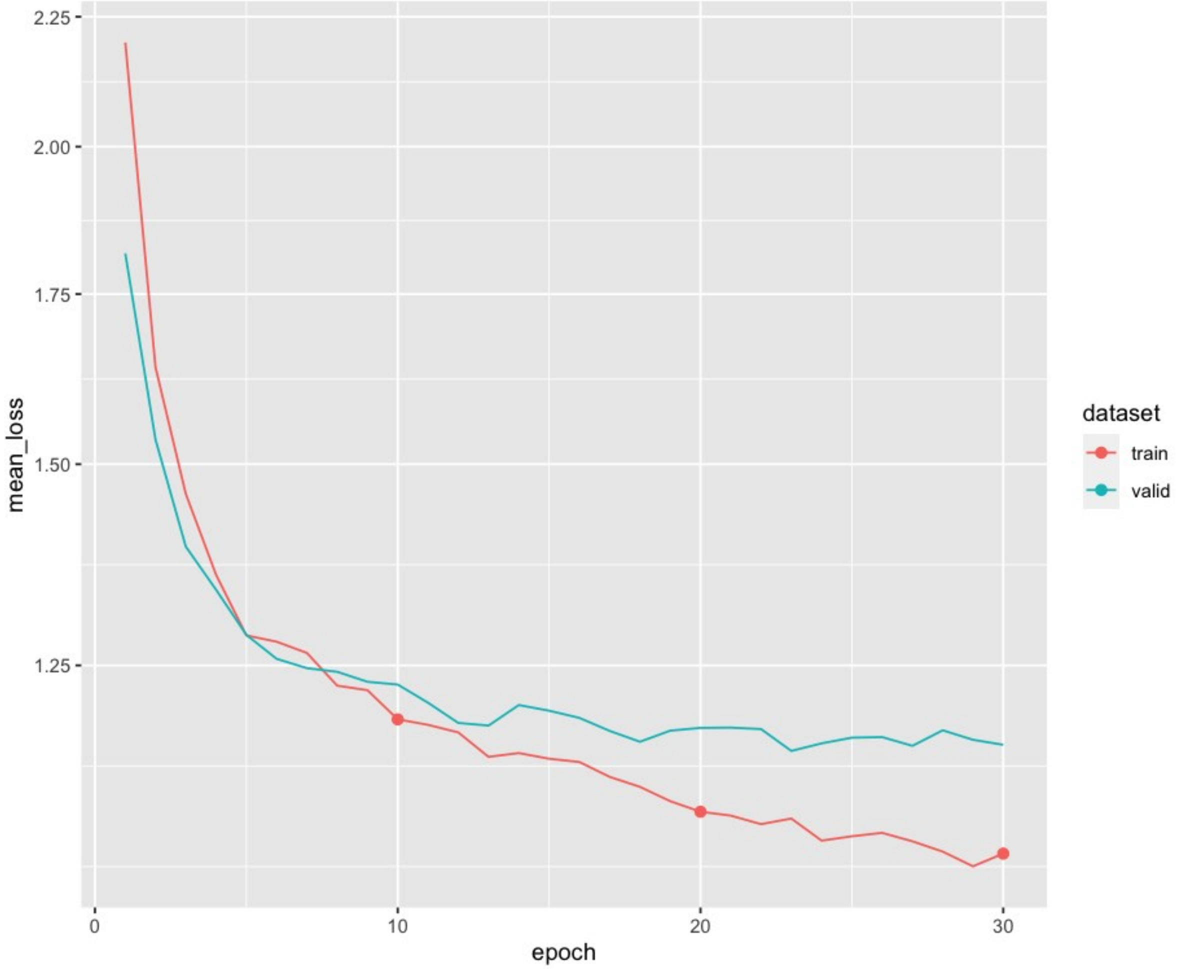
tabnet:: diagnostique du modèle

`ggplot2::autoplot()` view the training loss evolution

```
ggplot2::autoplot(fit_regression)
```

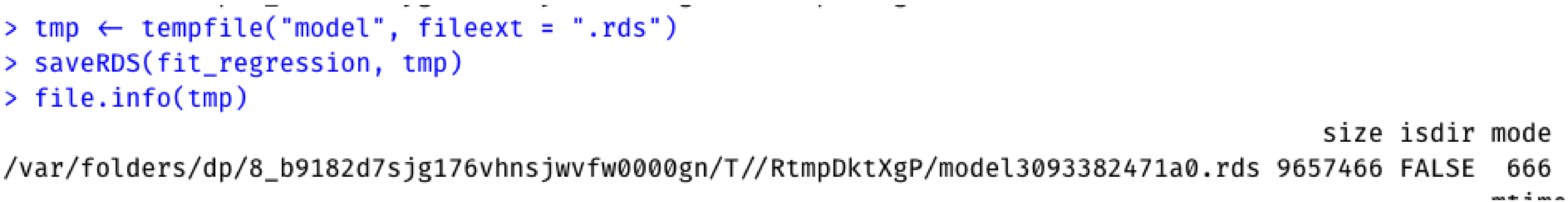


```
ggplot2::autoplot(fit_classification)
```



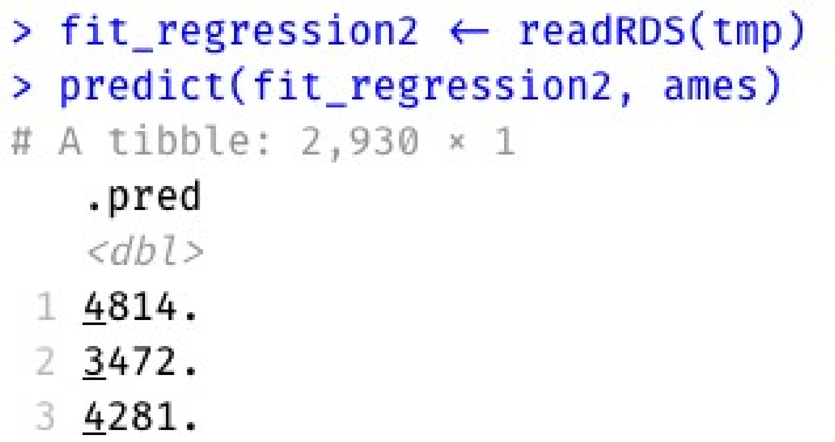
tabnet:: sauvegarde et chargement d’un modèle sur disque

saveRDS(tabnet_model)



```
> tmp ← tempfile("model", fileext = ".rds")
> saveRDS(fit_regression, tmp)
> file.info(tmp)
                                                                                  size isdir mode
/var/folders/dp/8_b9182d7sjg176vhnsjwvfw0000gn/T//RtmpDktXgP/model3093382471a0.rds 9657466 FALSE  666
```

readRDS(file.Rds)

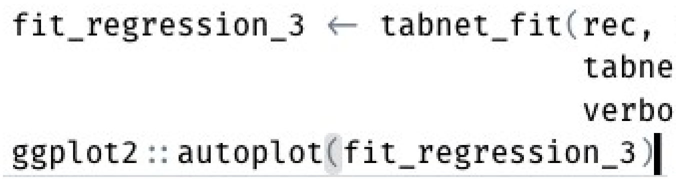


```
> fit_regression2 ← readRDS(tmp)
> predict(fit_regression2, ames)
# A tibble: 2,930 × 1
   .pred
   <dbl>
 1 4814.
 2 3472.
 3 4281.
```

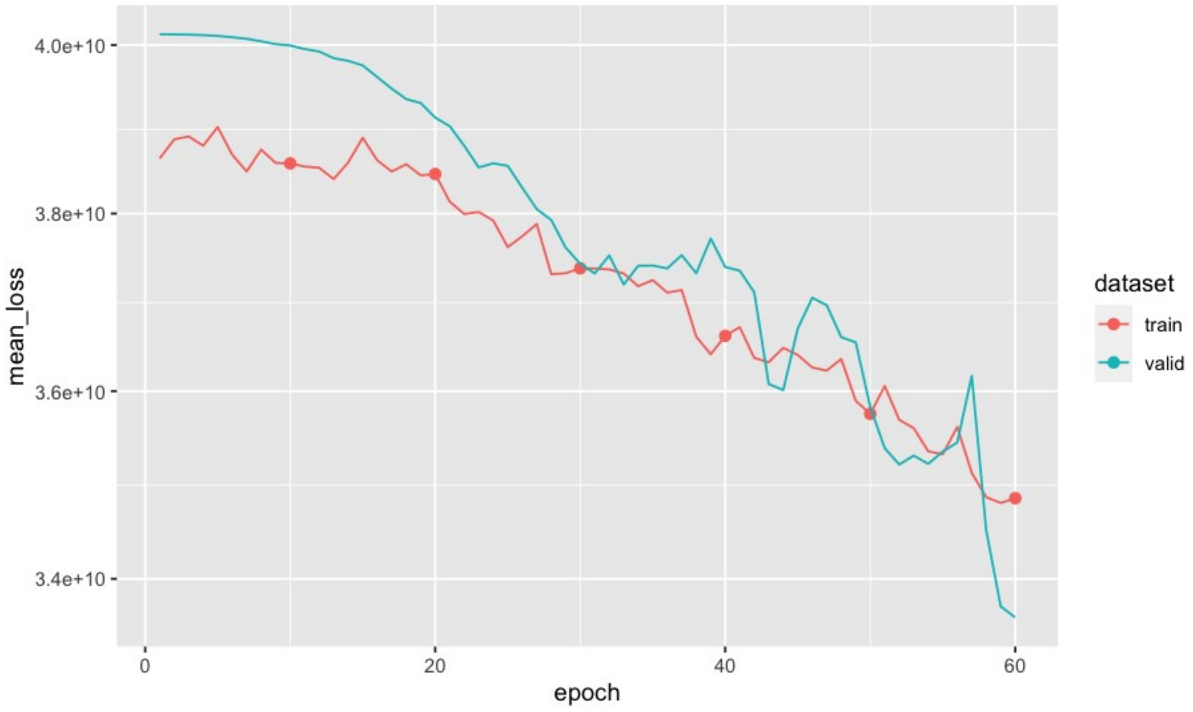
tabnet:: reprise sur entrainement d’un modèle

tabnet_fit(…, tabnet_model = <previous model> , from_epoch = 17)

- Depuis un modèle en mémoire



```
fit_regression_3 <- tabnet_fit(rec, ames, epochs = 30, valid_split = 0.25,
                               tabnet_model = fit_regression, from_epoch=30,
                               verbose = TRUE)
ggplot2::autoplot(fit_regression_3)
```



- Depuis un modèle sur disque

Idem, mais il faut que from_epoch soit un checkpoint …

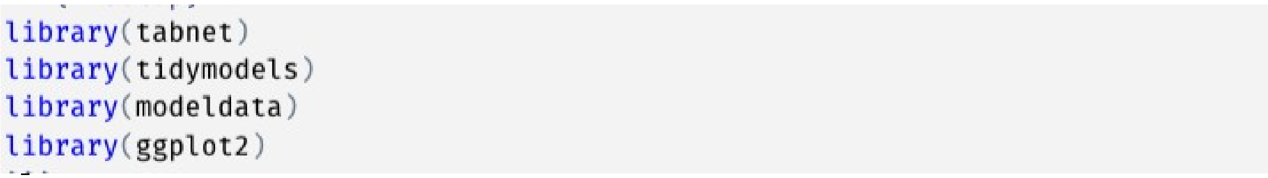
- En changeant les paramètres d’entrainement (lr, batch size, ...)

Mais pas le design du modèle !

tabnet:: intégration dans le flux de modélisation

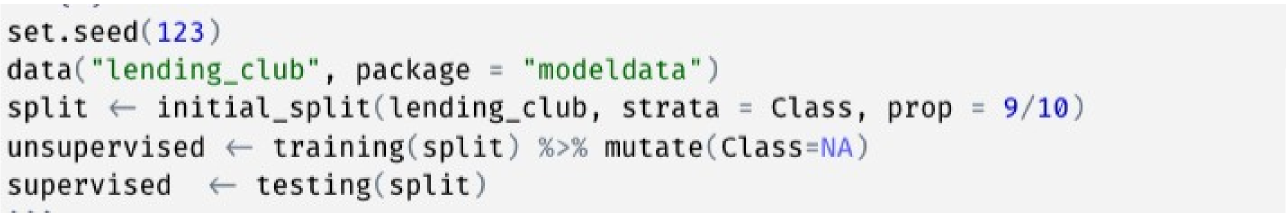
tabnet_pretrain() unsupervised training

- les librairies nécessaires



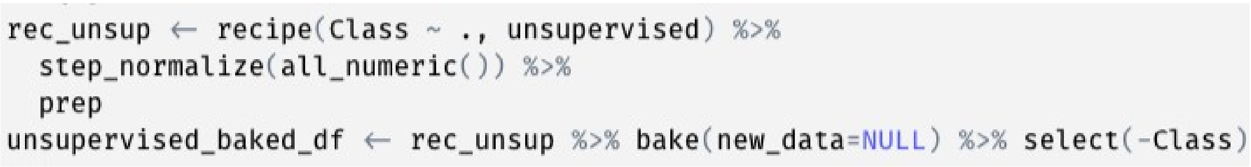
```
library(tabnet)
library(tidymodels)
library(modeldata)
library(ggplot2)
```

le jeu de données non-supervisé



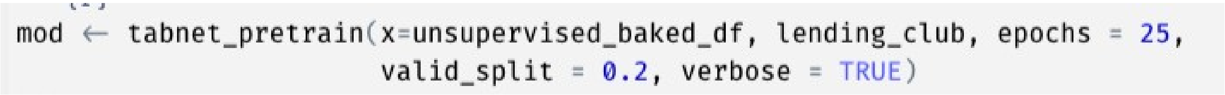
```
set.seed(123)
data("lending_club", package = "modeldata")
split ← initial_split(lending_club, strata = Class, prop = 9/10)
unsupervised ← training(split) %>% mutate(Class=NA)
supervised   ← testing(split)
```

- la recette, préparation, et thermisation des données

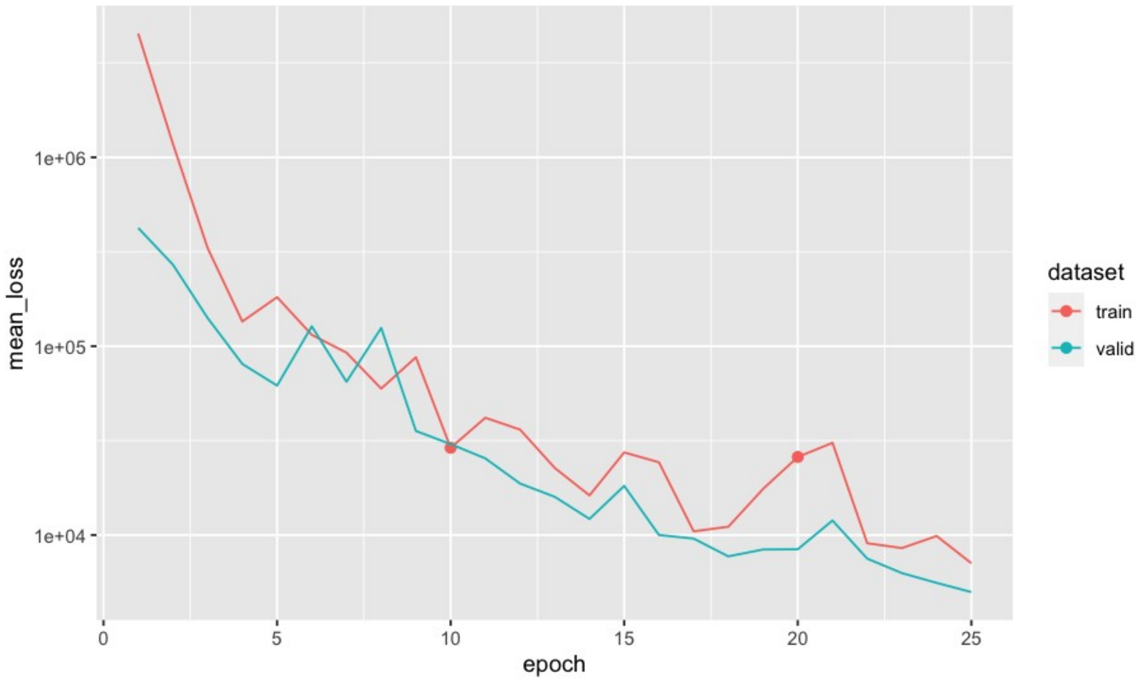


```
rec_unsup ← recipe(Class ~ ., unsupervised) %>%
  step_normalize(all_numeric()) %>%
  prep
unsupervised_baked_df ← rec_unsup %>% bake(new_data=NULL) %>% select(-Class)
```

- l'entrainement non-supervisé



```
mod ← tabnet_pretrain(x=unsupervised_baked_df, lending_club, epochs = 25,
                      valid_split = 0.2, verbose = TRUE)
```



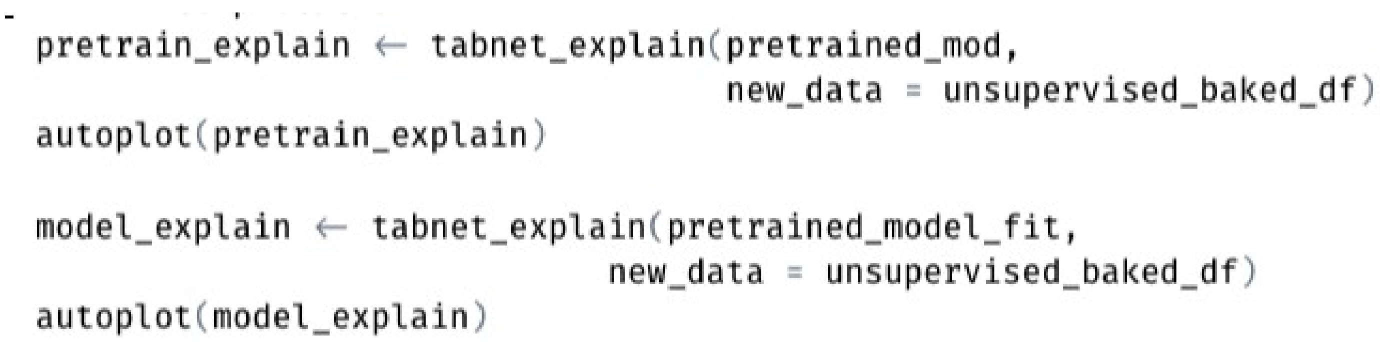
tabnet:: reprise sur entrainement d’un modèle

```
tabnet_fit(…, tabnet_model = <unsupervised model> )
```

- Depuis un modèle non supervisé, la fonction de coût est modifiée

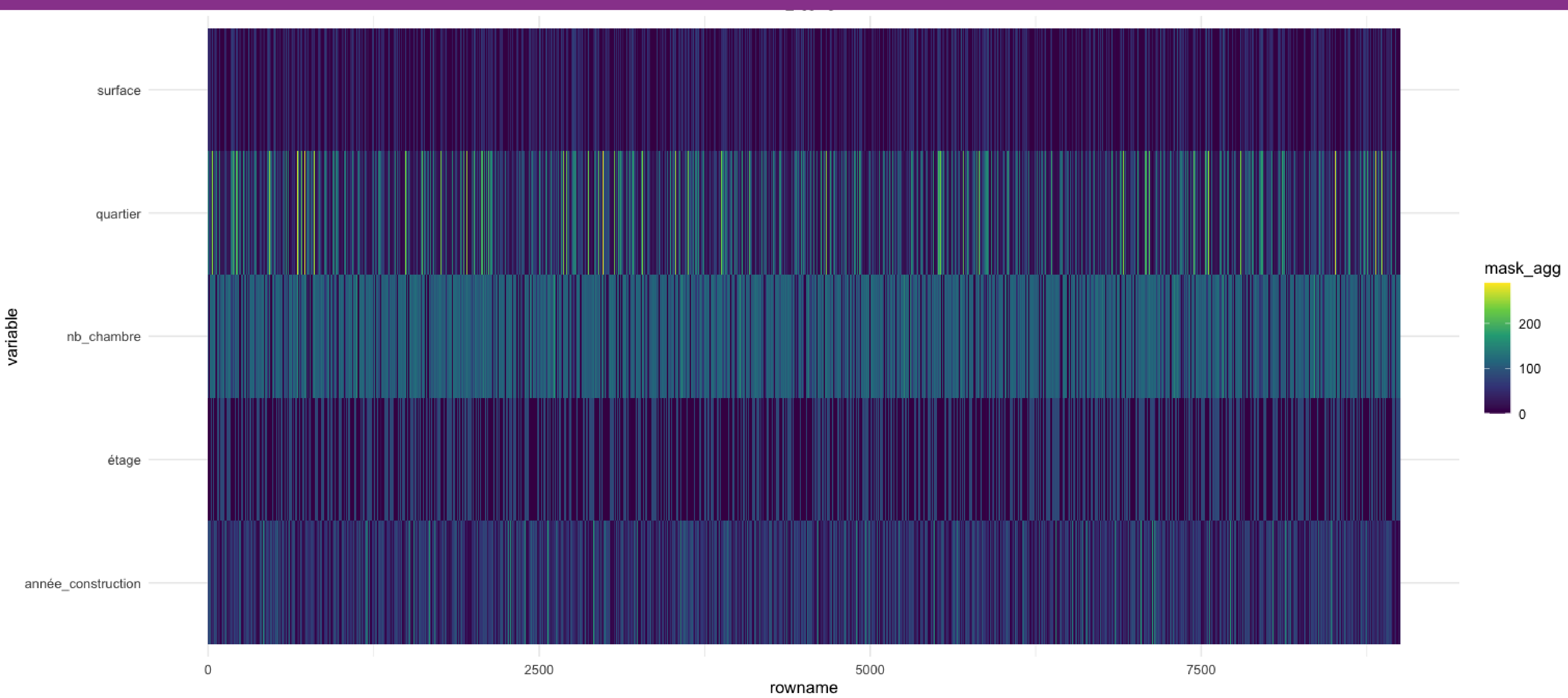
tabnet:: interprétation du modèle

tabnet_explain() extraction du masque agrégé

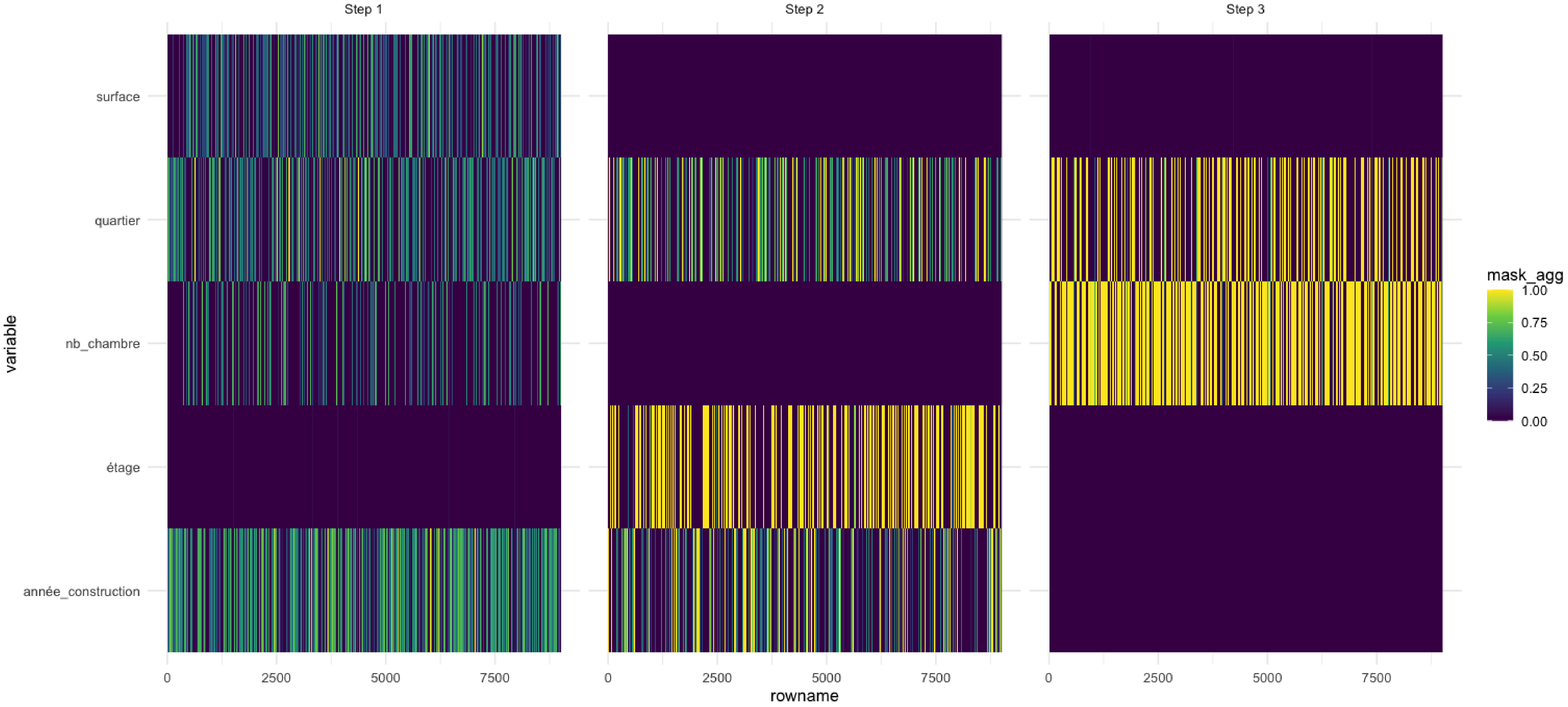


```
pretrain_explain ← tabnet_explain(pretrained_mod,
                                  new_data = unsupervised_baked_df)
autoplot(pretrain_explain)

model_explain ← tabnet_explain(pretrained_model_fit,
                               new_data = unsupervised_baked_df)
autoplot(model_explain)
```

tabnet::


surface
quartier
nb_chambre
étage
année_construction
variable
0
2500
5000
7500
rowname
mask_agg
200
100
0

tabnet::


Step 1
Step 2
Step 3
variable
surface
quartier
nb_chambre
étage
année_construction
0
2500
5000
7500
rowname
mask_agg
1.00
0.75
0.50
0.25
0.00