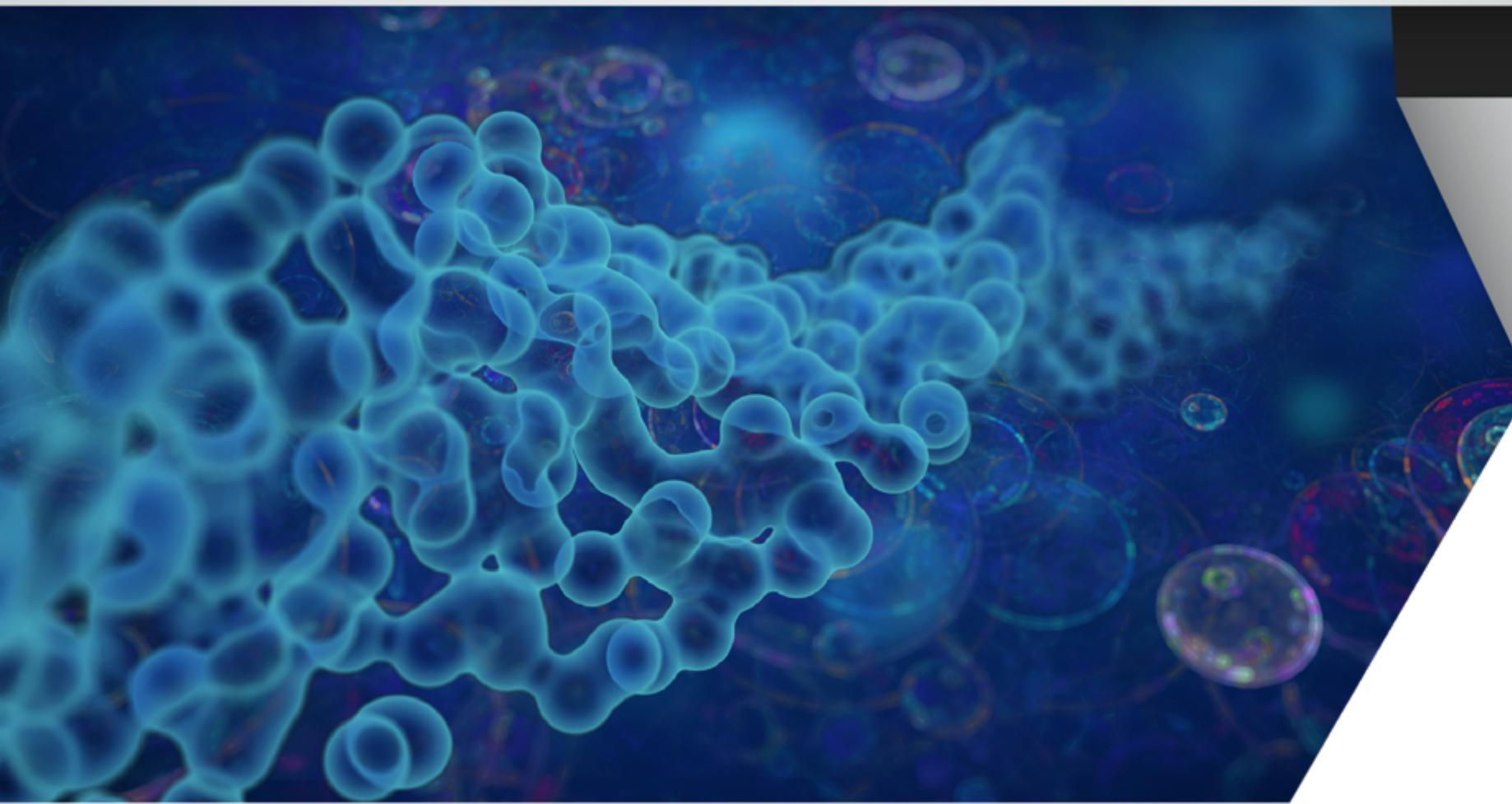


Making Python Objects R-like with pyR



Robert Kirk DeLisle
R/Pharma 2021

somalogic™

Who We Are and What We Do

- The SomaScan Platform detects >7,000 proteins in a small biological sample (e.g., blood).
- Sample collections from different groups can be used to identify biomarkers or develop predictive models of biological endpoints (e.g., disease vs control, treated vs untreated, risk of developing disease, etc.)
- Mostly an R shop expanding usage of Python.
- We value automation, reproducibility, and standardized reporting using a shared code base.



SOMASCAN ASSAY



10000-28	10000-7	10000-15	10000-20	10000-43	10001-65	10012-5	10013-34	10014-31	10015-15	10021-1	10022-207	10023-32	10024-1	10030-8	10034-16	10035-1	10036-201	10037-98	
476.5	321.1	100.3	601.8	561.8	6897.1	1798.6	441.9	941.7	728.1	798.1	460.2	922.8	154.8	2803.4	2749.3	309.8			
478.5	234.1	100.3	541.1	541.1	541.1	153.1	531.1	473.1	531.1	473.1	531.1	531.1	531.1	145.1	233.1	250.1	137.1	137.1	
415.6	296.6	3030.1	563.9	423.9	3203.6	2095.6	560.4	103.1	535.1	818	145.5	581.2	800.1	134.5	3560.6	519.6	2633.8	140.7	
442.6	247.9	112.6	503.7	469.8	3140.9	1922.2	582	1207.2	840.0	886	159.8	524.1	797.1	142	3625	561	2805.9	128	
462.7	231.1	75.9	443.1	443.1	443.1	437.1	437.1	529.3	729.3	163.1	453.1	357.1	234.1	203.1	302.1	155.1	155.1		
496.6	669.6	135.5	826	458.7	1705.7	1198	415.7	801.8	793	813.7	178.7	402.8	637.1	162.4	2397.2	678.8	2717.9	169	
693	614.2	100.3	602.7	481.7	6242.4	1675.5	481.1	747.2	149.3	481.3	910.1	142.2	288.4	586.7	2805.5	151.9			
512.3	433.1	100.3	734.1	734.1	734.1	153.1	153.1	798.1	153.1	153.1	153.1	153.1	153.1	147.1	139.1	154.1	154.1	154.1	
452.1	918.3	113.2	905.4	541.2	2674.6	1526.6	747.4	815.5	881.7	609.4	174.8	466.3	704.5	135.9	2598.8	610	2502.1	151.3	
702.1	237.7	126.8	634.2	585	2807.4	1656.8	512.6	869.1	438.8	199.3	490	819.1	202.5	330.4	559.8	2808.8	191.7		
1100.1	139.1	100.3	128.1	128.1	128.1	461.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	
500.9	406.2	99.1	636.1	435.9	2635.4	1820.4	508.3	1611.2	757.5	799.5	159.2	447.2	772.8	139	3623.3	479.5	2884.8	150.8	
45	39.5	34.2	38.2	36.9	74.4	46.1	91.3	73.8	65.8	31.3	37.7	64	40.6	62.4	28.6	22.6			
482.3	239.1	92.1	92.1	92.1	92.1	161.1	161.1	161.1	161.1	161.1	161.1	161.1	161.1	161.1	161.1	161.1	161.1	161.1	
530.6	391.2	138	765.5	670.1	2536.3	2099.2	919	2010.5	1327.1	1801.8	424	815.1	911.5	1644.7	2885.2	674.1	2889.3	366.6	
501.5	207.4	134.9	590.1	510.1	2781.6	1806.1	538.7	1250.9	452.7	826.5	191	542.5	242.2	188.8	290.3	582.1	2844.4	169.2	
452.8	230.1	138.4	437.1	437.1	437.1	527.1	527.1	1155.1	1155.1	1155.1	1155.1	1155.1	1155.1	1155.1	1155.1	1155.1	1155.1	1155.1	
478.8	276.3	97.4	758.2	584.8	2616.4	1551.8	358.4	789.3	828.9	689.5	188.7	442.3	640.6	248.2	2167.9	641.9	1888	152.1	
632.9	299.7	107.7	594.1	471.6	2483.5	1609.4	520.4	941.3	512.4	1082.2	152.3	524.3	1235.6	159.8	2905.5	399.1	2822.8	154.9	
459	430.1	111.7	594.1	594.1	594.1	527.1	527.1	527.1	527.1	527.1	527.1	527.1	527.1	527.1	527.1	527.1	527.1	527.1	
447.8	549.8	89.8	1109.4	499	2805.4	1496.8	429.9	1219.2	638.5	678.8	149.1	514.6	1125.3	138.6	2036.7	492.7	2423.2	159	
419.3	216.3	95.3	648.1	591.8	2109.8	1857.8	545.5	795.7	607.5	717.7	155.3	540.1	921	112.4	339.6	566.7	2277.6	388.1	
809	233.1	100.3	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	
516.3	1059.1	117.1	991.2	457.9	2712.7	1749.9	446.5	867.3	1032	722	170.5	482.6	824.3	171.1	3095.4	531.7	2742.7	306.7	
433	254.2	99.2	523.5	434.4	2742.6	1930.8	562.9	841.5	499.2	797.9	159.2	549.2	793.1	331.2	361.1	589.7	2655.4	177.5	
458.2	233.1	111.7	523.1	434.1	2742.1	1749.1	446.1	867.3	1032	722	170.5	482.6	824.3	171.1	3095.4	531.7	2742.7	306.7	
647.4	557.7	119.7	766.1	338.7	1534.1	1068	460.4	1098	1024.5	912.4	234.8	436.5	1262.3	265.9	1911.7	555.8	2706.7	219.1	
543.8	669.7	95	802.4	532.2	1345.1	1122.3	370.9	1008.6	992.6	740.5	158.2	469.2	717.7	136.8	240.2	482.1	2398.9	157.6	
415.2	234.6	114.6	523.1	434.1	2742.1	1749.1	446.1	867.3	1032	722	170.5	482.6	824.3	171.1	3095.4	531.7	2742.7	306.7	
4501.1	526	114.9	701.3	386.1	2623.9	1763.1	651.7	773.2	697.6	856.2	150.1	490.1	735.3	133.4	3216.7	531.5	2928.3	153.8	
474.3	743.8	182.3	833.5	496.6	1784.4	476	754.9	1078.9	992.2	167.9	650.4	784.5	130.8	231.1	484.1	3549.6	154.9		
443.2	244.1	114.1	540.1	540.1	540.1	523.1	523.1	523.1	523.1	523.1	523.1	523.1	523.1	523.1	523.1	523.1	523.1	523.1	
421.4	314	107.7	833	496.3	2681.9	1398.5	746.7	1380.6	482.8	635.8	186.2	425.2	858.6	218.9	2085.5	535.6	2373.6	176.2	
442.1	280	100	855.8	489.2	2529.1	2006	563.3	981.2	546.6	776	281.4	526.5	818.4	130.7	331.7	529.2	3312.1	277.6	
402.9	231	100.3	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	
687.4	228.1	134	654.9	598.4	2886	1623.1	496.6	863.5	816.2	919.9	208.8	489.2	854.8	195.6	3626.6	593.5	2984.8	192.2	
458.8	413	186.2	688.3	445.3	2634.3	1610.2	462.5	911.6	707.2	727.6	179.5	515.4	786.1	205.6	255.9	376.7	2576.5	174.7	
475.2	235	100.3	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	128.1	
8082.7	324.2	100	591.1	517.6	2347.3	1550.9	351.8	784.4	547.6	567.3	181.4	370.6	614.6	162.8	1722.6	498.5	2150.2	147.3	
607.5	438.6	306.4	601.3	472.9	6019.6	2064	528.4	789.8	1477.1	2248	510.4	737.4	177.7	2489.3	955.3	2292.4	200.2		
467.3	252.7	124.1	332	546.7	2751.2	2235.7	643.2	1386.6	991.2	950.2	254.9	643.9	972.1	128.7	3312.3	596.5	2504.3	172.1	

Plasma protein patterns as comprehensive indicators of health

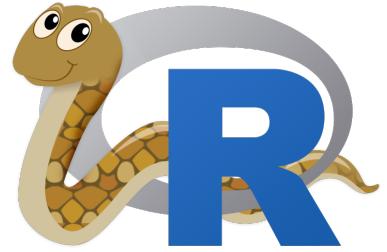
Stephen A. Williams^{1,12*}, Mika Kivimaki², Claudia Langenberg³, Aroon D. Hingorani^{4,5,6}, J. P. Casas⁷, Claude Bouchard⁸, Christian Jonasson⁹, Mark A. Sarzynski¹⁰, Martin J. Shipley², Leigh Alexander¹, Jessica Ash¹, Tim Bauer¹, Jessica Chadwick¹, Gargi Datta¹⁰, Robert Kirk DeLisle¹, Yolanda Hagar¹, Michael Hinterberg¹, Rachel Ostroff¹, Sophie Weiss¹, Peter Ganz^{11,12} and Nicholas J. Wareham^{3,12}



11/29/21

CONFIDENTIAL & PROPRIETARY

Integrating Python with R using *reticulate*



```
library(reticulate)

# set my conda environment
reticulate::use_condaenv('scikitlearn', required = TRUE)

# load the 'linear_model' module of SciKit-Learn
sklearn <- import('sklearn.linear_model')

# Create a logistic regression object
py_linreg <- sklearn$LinearRegression()

# fit the model and get some predictions
py_linreg$fit(x = training_X, y = training_Y)

py_preds <- py_linreg$predict(test_X)

#####
# compared to R-based linear regression
r_linreg <- lm(Y ~ ., data = training)

r_preds <- predict(r_linreg, newdata = test_X)
```

- Very direct and concise code to make Python modules available to R
 - Direct access to Python object methods
 - Code integrates seamlessly
-
- Requires use of *import()* and creation of Python object within code block
 - Access to Python object methods goes through the \$ operator – this makes the code inconsistent with typical R usage

It would be nice if Python portions behaved more like R code for consistency and to enable automation across R and Python objects.

Model Class Hierarchy

Model Abstract Base Class

Defines the functions (interface) that every model must support, regardless of type

Does not provide function implementations



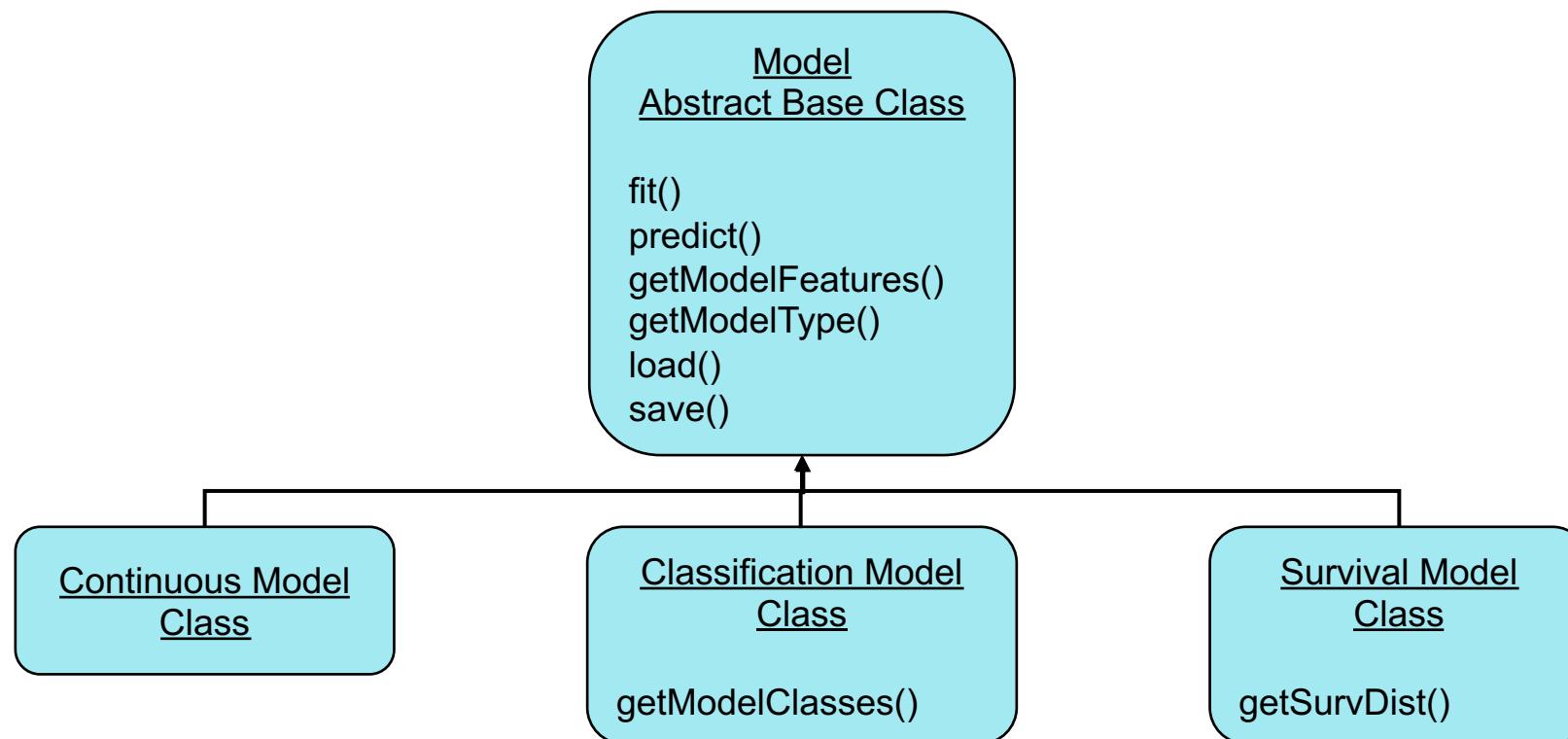
Derived Classes

encapsulate specific models (from R or Python)

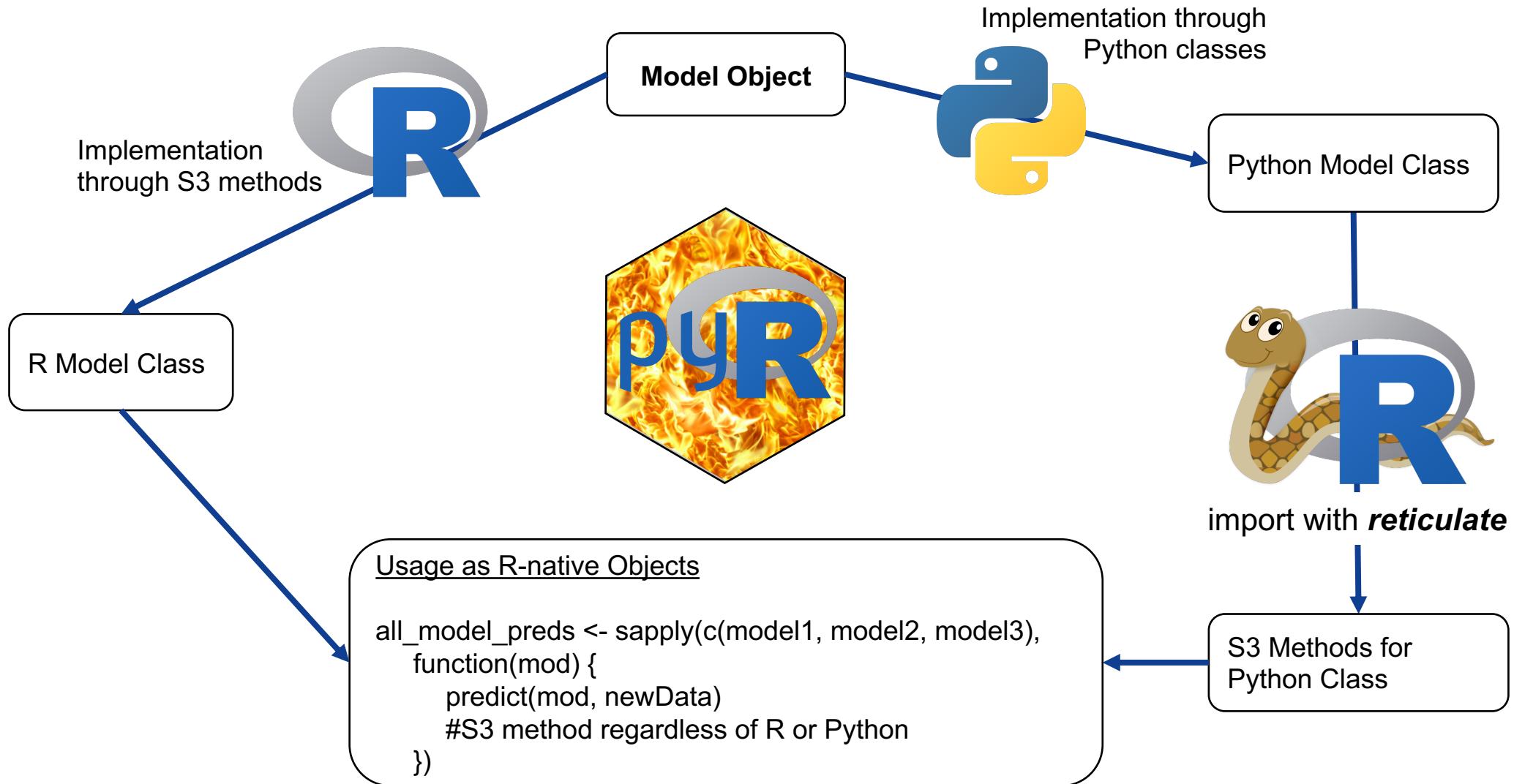
Inherit interface from Model Abstract Base Class

Defines the interface for that specific model type by providing additional specialization functions, if needed

Provide implementations of all functions for that model type (e.g., classification, regression, survival)



pyR Architecture Concept



Any code can expect the Model Object Interface to be present – this enables all models to be used consistently, and *non-R objects to be used as if they were native R objects*.

pyR Object Factory

Saved pyR objects

pyR_GPFlow_Class.pyr

pyR_TPOT_Class.pyr

New pyR objects

"pyR_logreg"

pyR ObjectFactory

knows which class to build from saved objects, or how to create new objects from scratch.



pyR S3 Objects

pyR GPFlow_Class Object

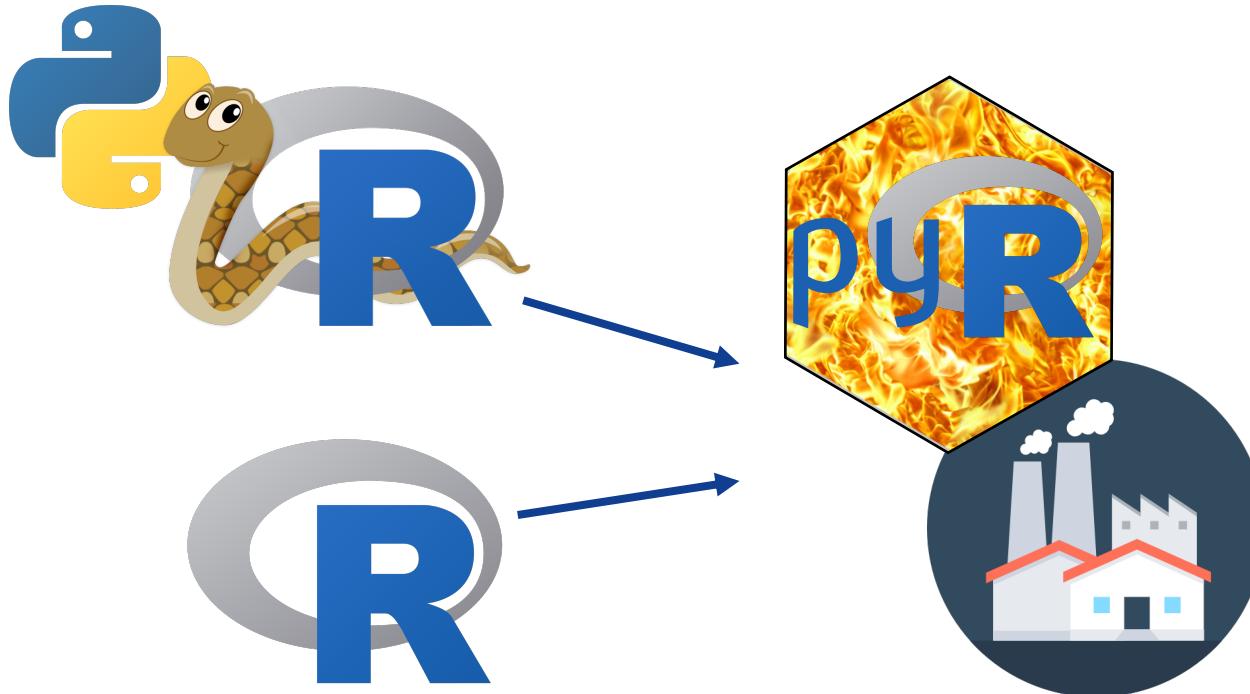
pyR TPOT_Class Object

pyR Logistic Regression Object

Common usage as native R objects

```
# get saved classifier models  
model_gp <-  
pyR_ObjectFactory.loadObject(  
  'pyR_GPFlow_Class.pyr')  
  
model_tpot <-  
pyR_ObjectFactory.loadObject(  
  'pyR_TPOT_Class.pyr')  
  
# create a new classifier model  
model_logreg <-  
pyR_ObjectFactory.newObject(  
  'pyR_logreg')  
fit(model_logreg, training_data)  
  
# generate predictions for all  
preds <- sapply(model_gp,  
  model_tpot, model_logreg),  
  function(fn) {  
    predict(mod, newData)  
  })
```

Unification of R and Python with pyR



R-native S3 Objects

Consistency
Shareability
Reproducibility
Standardization

```
all_model_preds <-  
  sapply(c(model1, model2, model3),  
         function(mod) {  
           predict(mod, newData)  
           #S3 method regardless of R or Python  
         })
```

Thanks to my colleagues!

Laura Sampson

Leigh Alexander

Stu Field

Robert Kirk DeLisle
kdelisle@somalogic.com
KirkD_CO