Bartosz Jabłoński, PhD

- A mathematician working and "playing" with data.
- Seasoned SAS user with experience in: education, telecommunication, clinical trials, and banking.
- As "active-learner" and "hobbyist-teacher" leads SAS classes and courses at the Faculty of Mathematics and Information Science at Warsaw University of Technology.
- During free time coordinates meetups of Polish SAS Users Group (`#PolSUG`) and holds second and a half dan in sasensei (`www.sasensei.com`).
- To turn his internal problem solver mode on just say: "you probably can't do something like this in SAS..."

SAS® Packages

the way to share

Bartosz Jabłoński

Warsaw University of Technology / Citibank Europe PLC

yabwon@gmail.com

November 18th, 2020

BASUG on-line webinar

www.polsug.com

#PolSUG

- Comprehensive TEX Archive Network
  `https://ctan.org/pkg/`
- Comprehensive Я Archive Network
  `https://cran.r-project.org`
- SAS-L discussion list
  `https://listserv.uga.edu`
- and my mates from Citi

SAS package[1]

A **SAS package** is an automatically generated, single, stand alone `zip` file containing organised and ordered code structures, created by the developer and extended with additional automatically generated "driving" files (i.e. description, metadata, load, unload, and help files).

The purpose of a package is to be a simple, and easy to access, code sharing medium, which will allow: on the one hand, to separate the code complex dependencies created by the developer from the user experience with the final product and, on the other hand, reduce developer's and user's unnecessary frustration related to a remote deployment process.

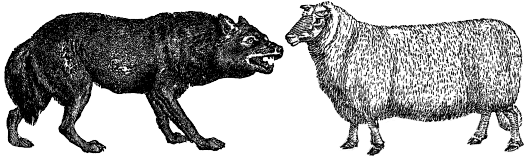
SAS Packages Framework

SAS Packages Framework is a "pack" of macros, which allows to *use* and to *develop* SAS packages.

[1] The idea presented here should not be confused with other occurrences of "package" concept which could be found in the SAS ecosystem, e.g. Proc DS2 packages, SAS/IML packages, SAS ODS packages, SAS Integration Technologies Publishing Framework packages, or a `*.egp` file.

„The wolf is sated and the sheep is whole"

github.com/yabwon/SAS_PACKAGES



sas packages

the way to share

How SPF interacts with your SAS session?

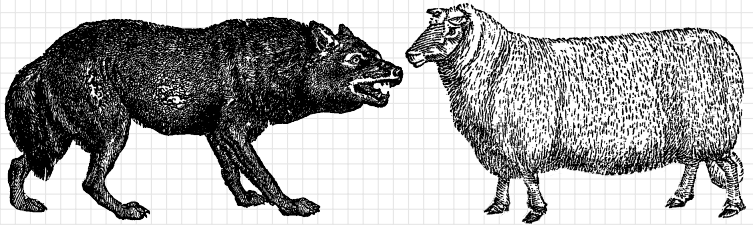
SPF's macros call internally to a folder/directory containing packages. The `packages` fileref is a reserved *key word* for that purpose.

SAS Packages Framework generates the following list of macros:

- `%installPackage()`,
- `%loadPackage()`,
- `%loadPackageS()`,
- `%helpPackage()`,
- `%unloadPackage()`,
- `%verifyPackage()`,
- `%previewPackage()`,
- `%listPackages()`, and
- `%generatePackage()`.

When the first package is loaded into the SAS session the `SYSLoadedPackages` global macrovariable is created and it keeps info about loaded packages and their versions.

The User & The Developer



Files and folders

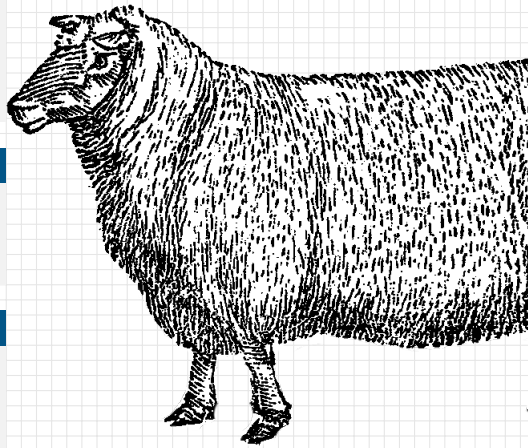
- Download `SPFinit.sas`* file, e.g. into `C:/PCKG`
- Create package's folder, e.g. `C:/PCKG/packageName`
- Prepare package's files

Code

```
filename packages "C:/PCKG";
%include packages(SPFinit.sas);
%generatePackage(filesLocation=C:/PCKG/packagename)
```

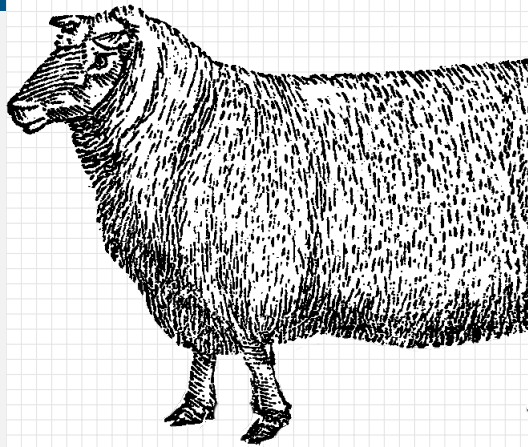
What next?

- Read the summary and the log
- Check tests results
- Share the package ;-)



Available types

`libname`,
`macro`,
`function`,
`functions`,
`format`,
`imlmodule`,
`proto`,
`data`,
`lazydata`,
`exec`,
`clean`,
(`test`).

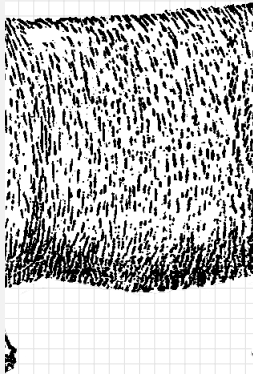


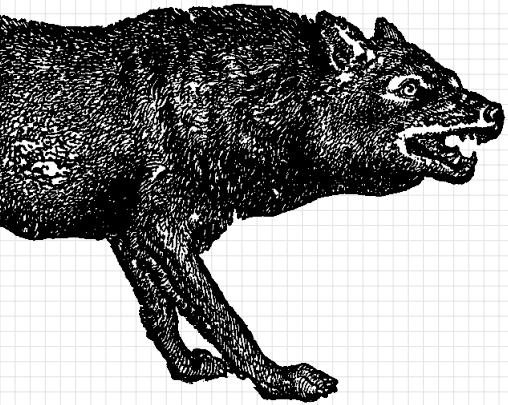
description.sas

```
Type: Package
Package: ThePackageName
Title: Short description, single sentence.
Version: x.y
Author: Fname1 Lname1 (xxx1@yyy.zz), Fname2 Lname2 (xxx2@yyy.zz)
Maintainer: Fname3 Lname3 (xxx3@yyy.zz)
License: XYZ17
Encoding: UTF8

Required: "Base SAS Software", "SAS/ACCESS Interface to ABC"
ReqPackages: "somePackage (3.14)", "otherPackage (42)"

DESCRIPTION START:
 Lorem ipsum dolor sit amet, ThePackageName consec tetur
 adipis cingelit. Nullamdapibus lacus a elit congue
 elementum. Suspendisse iaculis ipsum nec ante luctus
 volutpat. Donec iaculis laoreet tristique.
DESCRIPTION END:
```





Files and folders

- Create a folder for packages, e.g. `C:/PCKG`
- Download the framework: `SPFinit.sas`*
- Download the package: `packagename.zip`

Code

```
filename packages "C:/PCKG";
%include packages(SPFinit.sas);
%loadPackage(packageName)
```

What next?

- Read the log
- Run `%helpPackage(packageName)`
- Use the package ;-)

OK, but why...?

- Not only macros. You can use functions, IML modules, proc proto C routines, formats, and even data generating code in a package.
- Automatic update of the `cmplib=` and the `fmtsearch=` options for functions and formats.
- Loading order of the code is organised the way you want it to be.
- It is all in 1 (one) file - you won't forget to share "all that is needed" with your peers.
- Functionality over complexity - share one file and say, e.g. "Here is the macro `%ABC()`, you use it like this and that" and you don't have to say that there are 73 other utility macros working in the "background".
- A package contains additional metadata (e.g. version number or generation timestamp).
- Help info is printed automatically in the log.
- A package can be loaded into the SAS session even if you don't have access to the SPF macros.
- Shareable between different OS.
- Cleaning functionality.
- Supports dependencies between packages.

dziękuję

thank you

- Bartosz Jabłoński, "SAS packages - the way to share (a how to) - extended", SAS GF 2020 Proceedings (stan z października 2020), `https://github.com/yabwon/SAS_PACKAGES/tree/master/SPF/Documentation`
- Mike Rhoads, "Use the Full Power of SAS in Your Function-Style Macros", SAS GF 2012 Proceedings, `https://support.sas.com/resources/papers/proceedings12/004-2012.pdf`
- The wolf and sheep are from *ClipArt ETC* archive, `https://etc.usf.edu/clipart/`

I would like to acknowledge a few people for their contribution.

The people are:

Filip Kulon,
Krzysztof Socki,
Allan Bowe,
Quentin McMullen,
Piotr Wójcik,
Michał Wojtasiewicz,
Richard DeVenezia, and
Christian Graffeuille.