



Bart Jabłoński, PhD

yabwon (2022)

Warsaw
University
of Technology

SASing since 2009



Experis

Manpower Group
www.experis.com



Takeda

Pharmaceutical
www.takeda.com



Polish SAS

Users Group
www.polsug.com

PHUSE EU Connect, Strasbourg, France, November 11th 2024

SM01

Use of SAS Packages in the Pharma Industry

Opportunities, Possibilities, and Benefits



Bartosz Jabłoński
yabwon

✉ yabwon@gmail.com
[linkedin.com/in/yabwon](https://www.linkedin.com/in/yabwon)

PHUSE EU Connect, Strasbourg, France, November 11th 2024

SM01

Use of **sas** Packages in the **ANY** Industry

Opportunities, Possibilities, and Benefits



Bartosz Jabłoński
yabwon

✉ yabwon@gmail.com
[linkedin.com/in/yabwon](https://www.linkedin.com/in/yabwon)

- Comprehensive T_EX Archive Network
<https://ctan.org/pkg/>
- Comprehensive R Archive Network
<https://cran.r-project.org>
- Python Package Index
<https://pypi.org/>
- Why not in SAS?



SAS package¹

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, etc.)

The purpose of a package is to be a simple, and easy to access, code sharing medium, which allows: 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 developers and users unnecessary frustration related to a remote deployment process.

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

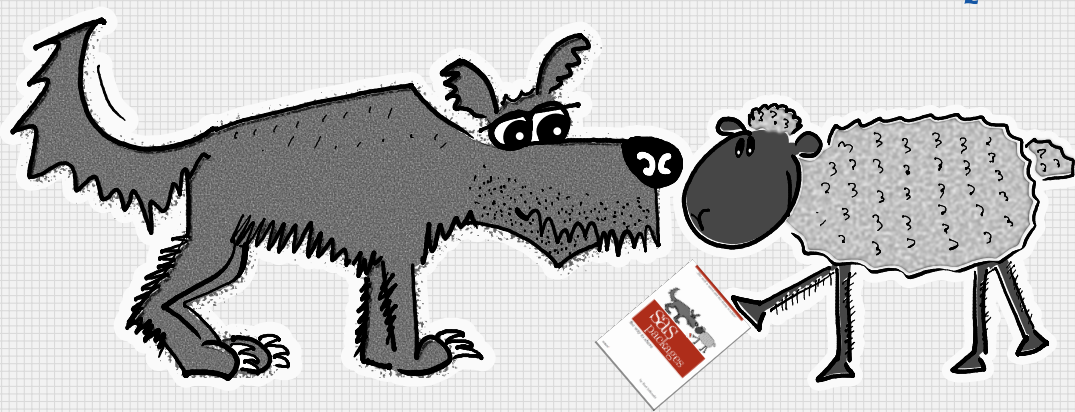
SAS Packages Framework

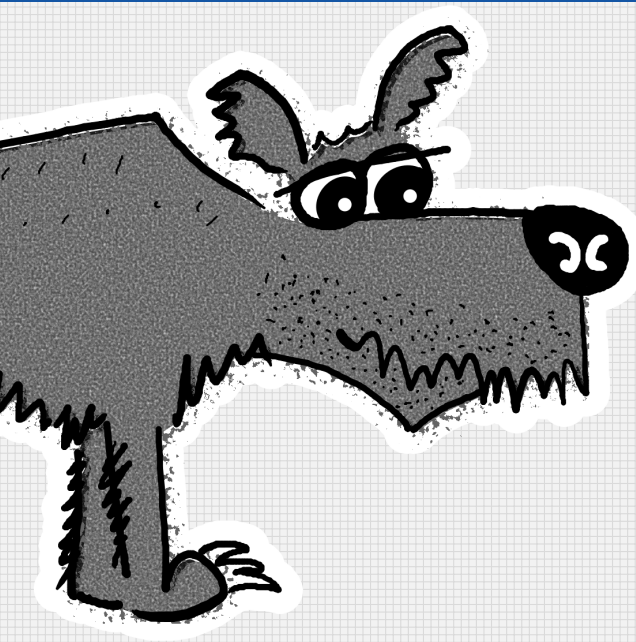
The **SAS Packages Framework** (SPF) is a "pack" of macros, which allows to use and to *develop* SAS packages.

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

100% SAS, open source, MIT licensed code! No "third party" needed to run.

the User & the Developer





Files and folders

- Create a folder for packages, e.g. C:/PCKG
- Install the framework: SPFinit.sas*

Code

```
filename packages "C:/PCKG";  
%include packages(SPFinit.sas);  
  
%installPackage(packageName)  
%<help/verify/preview>Package(packageName)  
  
%loadPackage(packageName)
```

What next?

- Read the log
- Enjoy the package ;-)

Files and folders

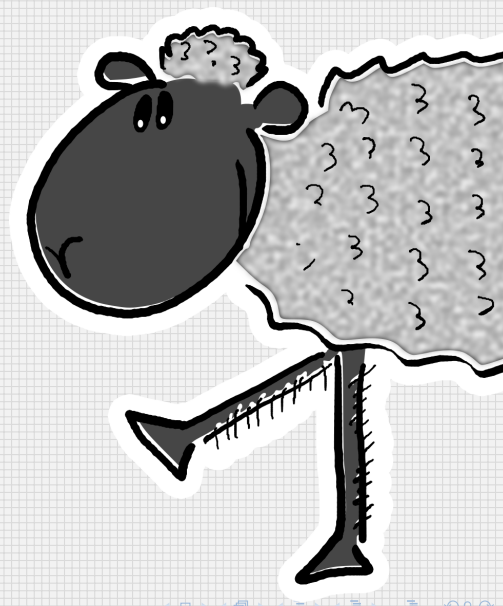
- Install SPFinit.sas* file, e.g. into /pkg
- Create package folder, e.g. /pkg/myPackage
- Prepare package files and the description file.

The Code

```
filename packages "/pkg";  
%include packages(SPFinit.sas);  
  
%generatePackage(filesLocation=/pkg/myPackage)
```

The Result

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



SAS Packages Framework



github.com/yabwon/SAS_PACKAGES

SAS Packages Archive



sas
packages

the way to share

github.com/SASPAC

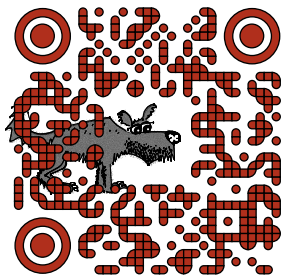
ORLY[®]

by Bart Jablonski

Workshop Materials

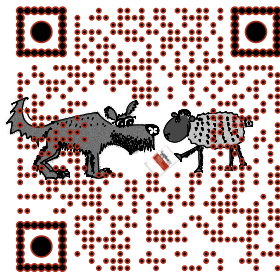


github.com/yabwon/How-SASPackages



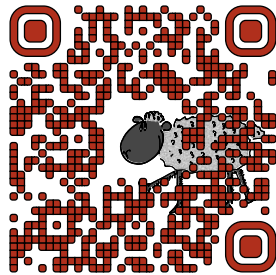
sas
packages

the way to share



sas
packages

the way to share



sas
packages

the way to share

Why is it worth using?

- SAS Packages allow you to **extend SAS capabilities** by referencing reusable, centrally maintained code. You write the code once and you can reuse it many times.
- **Not only macros!** You can use different "types" like: user functions (in FCMP and CASL), IML modules, proc proto C language routines, DS2 methods, formats, libraries, and even data generating code.
- When a package delivers formats or functions **automatic update of the `cmplib=` and the `fmtsearch=` options** is executed. Utility macros for IML modules and CAS-L user defined functions for fast loading (with dependency checks) are automatically generated.
- **Source code** is inside the package so it will **never be lost!**
- Loading **order of the code is organized** the way developer want it to be. This prevents user from loading content in random order what leads to dependency errors.
- It is **all in ONE (1)** file - you will not forget to share "all that is needed" with your peers.
- **"Functionality over complexity"** - share one file and say: "Here is my package with the %ABC() macro, you use the macro like this and that..." and you do not have to say that there are 173 other utility macros working in the "background" on behalf the %ABC().

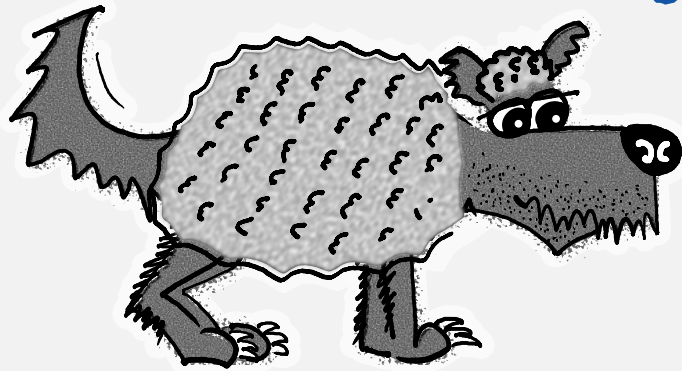
Why is it worth using? Cont.

- A package contains **additional metadata** (e.g., version number or generation time-stamp) which supports DevOps approach and makes changes tracking easier. Also, the **versioning** makes it easier when working with historical projects. In such case, a particular version can be saved with the project and there is no fear that so called "future improvements" will break project execution.
- **Help information** provided by the developer can be **printed automatically** in the log what makes packages "independent" from internet access. Furthermore, package does not have to be loaded into the session so the help information could be accessed.
- A "safety vault" called **ICE-loading** allows a package to be loaded into the SAS session even if there is no access to the SPF framework.
- Packages can be **shared between different operating systems** (Windows, Linux and UNIX).
- The SPF supports **dependencies handling** between packages. If package X requires package Y to be loaded for running the framework will verify Y's presence in the packages directory.
- **"Cherry picking"** allows to selectively use content of SAS packages.
- It is not only **BASE SAS**! Packages work with **VIYA** (dedicated macros for Viya are still macros; CAS-L user defined functions - is one of types served) and **WORKBENCH** too.

Why is it worth using? Cont. Cont.

- Code is 100% "inspectable" before even loading it to the SAS session. The %previewPackage() macro provides the functionality allowing to display package source code in SAS session log.
- Package integrity/genuinity can be tested with the %verifyPackage() macro, so the user is assured that no one modified the package on the way between the user and the developer.
- The SPF facilitates ability to store and used packages from multiple directories, e.g., in the statistical computing environment user can work with both centrally maintained packages and "private" ones (located for example in user home directory).
- Packages can be provided with an "additional content", that is, not SAS code. For example additional documents in PDF, figures in PNGs or JPGs, etc.
- "Lazy" data sets loading is supported. Lazy data set is term describing a data set which is not automatically loaded into the SAS session when a package is loaded. To load it such a data set has to be explicitly mentioned in the loading process.
- Cleaning functionality. When a package is no longer needed in the SAS session its content can be unloaded automatically by the framework.
- Since structure of a package has to be organized in separate files in sorted types directories the package development process "enforces" order in code.

Would you like to build a SAS Package?





Bartosz Jabłoński, "SAS Packages: The Way to Share (a How To)", SGF Proceedings, 2020

<https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4725-2020.pdf>

extended version available at: https://github.com/yabwon/SAS_PACKAGES/blob/main/SPF/Documentation



Bartosz Jabłoński, "My First SAS Package - a How To", SGF Proceedings, 2021

https://communities.sas.com/kntur85557/attachments/kntur85557/proceedings-2021/59/1/Paper_1079-2021.pdf

also available at: https://github.com/yabwon/SAS_PACKAGES/tree/main/SPF/Documentation/Paper_1079-2021



Sunil Kumar Pusarla, Kevin Viel, Dante Di Tommaso, "An open-source project to map lab terminology and standardize units for analysis",

PHUSE US 2022 Proceedings, 2022, https://www.lexjansen.com/phuse-us/2022/ds/PAP_DS02.pdf



Eldho Alias, Prasanth Prabhakaran, Rahulraj V R, "SAS Packages for Clinical Programming", PHUSE EU 2023 Proceedings, 2023,

https://phuse.s3.eu-central-1.amazonaws.com/Archive/2023/Connect/EU/Birmingham/PAP_CT04.pdf



Bartosz Jabłoński, "Share your code with SAS Packages a Hands-on-Workshop",

WUSS 2023 Proceedings, 2023, <https://www.lexjansen.com/wuss/2023/WUSS-2023-Paper-208.pdf>



Bartosz Jabłoński, "Integration of SAS GRID environment and SF-36 Health Survey scoring API with SAS Packages",

PharmaSUG 2024 Proceedings, 2024, <https://www.lexjansen.com/pharmasug/2024/SD/PharmaSUG-2024-SD-262.pdf>



Bartosz Jabłoński, "Macro Variable Arrays Made Easy with macroArray SAS package",

PharmaSUG 2024 Proceedings, 2024, <https://www.lexjansen.com/pharmasug/2024/AP/PharmaSUG-2024-AP-108.pdf>



Bartosz Jabłoński, "Here Comes the Rain (Cloud Plot) Again",

WUSS 2024 Proceedings, 2024, https://www.wuss.org/proceedings/2024/164/164_FINAL_paper_pdf.pdf



Eric Ohuma, Linda Vesel, Simon Parker, Bartosz Jablonski, "Guidance for International Growth Standards (GIGS) project",

GIGS SAS Package, 2024, <https://github.com/SASPAC/gigs/>



Kosuke Tsutsugo, "SAS Plotter", SASPlotter SAS Package, 2024,

<https://github.com/SASPAC/sasplotter>