

Getting Started With sas[®] Packages

(how to use them - in seven steps + appendix)

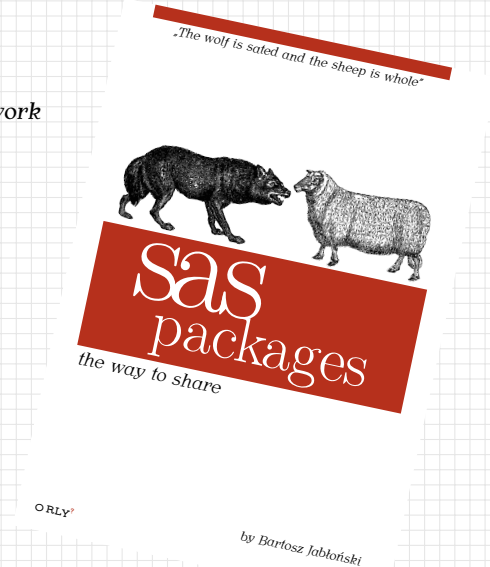
Bartosz Jabłoński
✉ yabwon@gmail.com

August 3rd, 2020

This presentation shows, in few simple steps,
how to:

- start working with *SAS Packages Framework*
- and using SAS Packages.

Shall we?



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).

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 complex code 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

The **SAS Packages Framework** is a set of macros which allow to use and to develop SAS packages.

¹The 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.

files and folders

Both the *SAS Packages Framework* file and *all* packages we want to use have to be stored in the same folder.

The first step is to create such a directory on your computer, e.g.

- under Windows OS family it could be `C:/SAS_PACKAGES` or
- under Linux/UNIX OS family it could be `/home/<username>/SAS_PACKAGES`.

For our convenience let's assume that **the `C:/PCKG` is the one we created.**

step 2

download the framework

The *SAS Packages Framework* is available under the following address:


https://github.com/yabwon/SAS_PACKAGES

Under the `SAS_PACKAGES` repository the `SPF` folder contains the framework. Download the `SPFinit.sas` file into the `C:/PCKG` directory.

 [yabwon](#) / [SAS_PACKAGES](#)

 Code

 Issues

 Pull requests

 Actions

 Projects

 Wiki


 master

[SAS_PACKAGES](#) / [SPF](#) /




yabwon version 20200803

..

 Documentation

 SPFinit.sas

 license.sas



step 3

install the package

SAS packages are available under the following address:

https://github.com/yabwon/SAS_PACKAGES

Under the [SAS_PACKAGES](#) repository the [packages](#) folder contains packages. Download the zip file with the package of choice into the [C:/PKG](#) directory.

 [yabwon / SAS_PACKAGES](#)

[Code](#)

[Issues](#)

[Pull requests](#)

[Actions](#)

[Projects](#)

[Wiki](#)

[Security](#)

 master

[SAS_PACKAGES](#) / packages /



yabwon version 20200730




..

 SQLinDS

 baseplus.zip

 dfa.zip

 dynmacroarray.zip



enable the framework

To enable the SAS Packages Framework we run the following code:

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

The first line assigns the `packages` fileref to the `C:/PCKG` directory. The second includes content of the `SPFinit.sas` file (i.e. the framework content) into the SAS session.

Since the framework's macros rely on the `packages` fileref we have to **remember to keep it assigned through entire SAS session.**

Hint. If we paste the above two lines of code into the `autoexec.sas` file the framework will be automatically available from the beginning of the session.

load the package

Lets assume that the SAS Package we installed (i.e. downloaded into the C:/PCKG directory) was the `baseplus.zip`.

To load the package into SAS session we run the following code:

```
%loadPackage(BasePlus)
```

This line of code loads package content (e.g. macros, functions, data sets, formats, etc.) into the SAS session and prints out in the SAS log a short information about the loading process and the package.

help about the package

Lets assume that the SAS Package we installed (i.e. downloaded into the C:/PCKG directory) was the `baseplus.zip`.

To get help about the package printed in the SAS log we run the following code:

```
%helpPackage(BasePlus)
```

This line of code **prints out in the SAS log** the help information about the package and its elements.

To learn about the package license we run: `%helpPackage(BasePlus, License)`.

To learn about particular element of the package, e.g. a function named `arrFill()`, we run: `%helpPackage(BasePlus, arrFill)`.

To learn about *all* available components of the package we run:
`%helpPackage(BasePlus, *)`.

unload the package

Lets assume that the SAS Package we installed (i.e. downloaded into the C:/PCKG directory) was the `baseplus.zip`.

To unload the package from SAS session we run the following code:

```
%unloadPackage(BasePlus)
```

This line of code removes all package content (e.g. macros, functions, data sets, formats, etc.) from the SAS session and prints out in the SAS log a short information about unloaded elements.

in "short" words, an example

- Create the `C:/PCKG` directory,
- Go to the https://github.com/yabwon/SAS_PACKAGES
- Download the framework file `SPFinit.sas` and the package of choice (e.g BasePlus) into the `C:/PCKG` directory.
- Run the following code:

```
filename packages "C:/PCKG"; /* set the directory */
%include packages(SPFinit.sas); /* enable the framework */
%loadPackage(BasePlus) /* load the package content */
%helpPackage(BasePlus) /* get help for the package */
%unloadPackage(BasePlus) /* unload the package */
```

the end

appendix

If you would like to learn more about
using packages see upcoming pages.

documentation

The documentation for the SAS Packages Framework is located under the following address:

https://github.com/yabwon/SAS_PACKAGES/tree/master/SPF/Documentation

The current version is in the file:

SAS(r) packages - the way to share (a how to)- Paper 4725-2020 - extended.pdf

how SPF interacts with SAS session

Since the framework's macros rely on the `packages` fileref we have to remember to keep it assigned and unchanged through entire SAS session.

When we enable the framework the following macros are generated:

`%installPackage()`, `%loadPackage()`, `%helpPackage()`, `%unloadPackage()`, `%listPackages()`, and `%generatePackage()`.

When we load the first package, e.g. run the `%loadPackage(somePackage)` macro for the first time, the `SYSLoadedPackages` macrovariable is created and populated. It stores the list of loaded packages in the current SAS session.

help for SPF macros

When we run the framework macros *without* arguments, like:

`%installPackage()`, `%loadPackage()`, `%helpPackage()`, `%unloadPackage()`, and `%generatePackage()` the help information (for five listed macros) is printed out into the SAS log.

Since the `%listPackages()` has no arguments we run `%listPackages(HELP)` to see the help for the macro.

lazy data

Sometimes the developer may provide packages for which data are not automatically loaded into the SAS session, they are so-called *lazydata*.

When we want to load such data the following code has to be executed:

- to see lazy data available we run: `%helpPackage(myPackage)` and read information in the SAS log,
- to load lazy data we copy the data set name and run:
`%loadPackage(myPackage, lazydata=nameOfTheDatasetToBeLoaded)`

We can provide a space separated list of datasets to load (e.g. `lazydata=A B C`) or if we want to load *all* lazy data we can use an asterisk (i.e. `lazydata=*`)

alternative to steps 2 & 4

If the SAS session allows us to use URL file reference we can enable the SAS Packages Framework with the following code:

```
filename packages "C:/PCKG";  
filename SPFinit URL  
"https://raw.githubusercontent.com/yabwon/SAS_PACKAGES/master/SPF/SPFinit.sas";  
%include SPFinit;
```

But we have to remember that the code above **does not create a copy of the `SPFinit.sas` file** in the `C:/PCKG` directory. It only enables the framework for the current session.

alternative to step 3

If the SAS session allows us to use URL file reference we can , *after* enabling the SAS Packages Framework, download packages with use of the following code:

```
%installPackage(BasePlus);
```

During the installation a copy of the `baseplus.zip` file is created in the `C:/PCKG` directory so it allows us to use the package in future (even if we don't have access to the network).

in "short" words, an example no. two

- Create the `C:/PCKG` directory,
- Go to the https://github.com/yabwon/SAS_PACKAGES
- Download the framework file `SPFinit.sas` into the `C:/PCKG` directory.
- Run the following code:

```
filename packages "C:/PCKG"; /* set the directory */  
  
%include packages(SPFinit.sas); /* enable the framework */  
  
%installPackage(BasePlus) /* download the package zip from the github */  
%loadPackage(BasePlus) /* load the package content */  
%helpPackage(BasePlus) /* get help for the package */  
%unloadPackage(BasePlus) /* unload the package */
```

in "short" words, an example no. three

Run the following code:

```
filename packages "%sysfunc(pathname(WORK))"; /* set the directory */  
filename SPFininit URL  
    "https://raw.githubusercontent.com/yabwon/SAS_PACKAGES/master/SPF/SPFininit.sas";  
%include SPFininit; /* enable the framework */  
  
%installPackage(BasePlus) /* download the package zip from the github */  
%loadPackage(BasePlus) /* load the package content */  
%helpPackage(BasePlus) /* get help for the package */  
%unloadPackage(BasePlus) /* unload the package */
```

use packages!