When submitting one R package to **CRAN**, you must ensure that no errors, no warnings and no notes appears in the check process, enforcing use of **’–as.cran’** option.

devtools::check(document = FALSE, args = c('--as-cran'))

Sometimes a note or a warning appears because of a change you made, and then everything seems to go weird. I faced such a case recently.

**The most annoying warning**

The most annoying warning I got so far, is tied to procurement of code sample files as a folder hierarchy under the **inst** folder.

This works well up to the moment when the folder path exceeds 100 characters. Here the **devtools::check** function complains and emits following warning.

Warning in utils::tar(filepath, pkgname, compression = compression, compression\_level = 9L, :

storing paths of more than 100 bytes is not portable:

...

Tarballs are only required to store paths of up to 100 bytes and cannot

store those of more than 256 bytes, with restrictions including to 100

bytes for the final component.

See section 'Package structure' in the 'Writing R Extensions' manual.

To submit this package to **CRAN**, I must get rid of the warning.

**Resolution**

Resolution is of course quite simple. You must change the path names in order to comply with this weird 100 bytes length. Up to now quite easy.

Aftermath as indeed much more difficult to handle. You must find back, all the packages that use one or another of the original path, locate this use, and replace the incriminated paths with new ones. This could be tackled in minutes or hours depending on the length of the package hieracht that make use of those code sample files.

**Unforeseen aftermath**

Another aftermath is faced when submitting these new aligned/corrected packages on CRAN. I am in a case where a package – let’s name it symbolically **A** – exposes such code sample tree hierarchy. Two others packages – **B** and **C** – depends on **A** and use code samples according to their needs. Many submissions where achieved on **CRAN** without any issue, using this scheme.

Now, having changed the path names, here is what happens when submitting the new versions of packages **A**, **B** and **C** (in this order) to **CRAN**.

First, **A** is being checked and compiles. So far, so good. But now **CRAN** process knows by historical information that there is a reverse dependency from **B** and **C** to **A** and checks and compiles **B** and **C** from old versions, with **A** that is now under new version. Very big mess!

**Troubleshooting the issues …**

I received emails stating that **B** and **C** does not pass **CRAN** checks anymore. The error message was

cannot coerce type 'closure' to vector of type 'character'

1: sapply(source\_files, function(e) {

source(system.file(e, package = source\_package))

}) at ...

I figured out quite quickly a highly proable version incompatibility, but as there is no statement of package versions on **CRAN** emails, I cannot check deeper. I suggested to **CRAN** professionnals following process that should solve the faced issues

1. check and build package **A** new version
2. Do not trigger any reverse dependency check, as **A** is a root package
3. check and build package **B** new version
4. check and build package **C** new version

Currently, packages **A**, **B** and **C** are under inspection by **CRAN**, and I am quite confident about the output. Just have to be patient and to let **CRAN** professionnals do their job.

**Will this impact CRAN processes?**

Indeed, still get a fuzzy feeling about **CRAN** package submissions and their aftermath.

I understand that they automated and industrialized the check and build processes and that’s a good thing. Indeed, this is done on historical information, that is the only available at **CRAN** level, I guess.

I do not understand how they believe to be able to handle dynamically a changing context, that remains unknown to them, without considering deprecation and obsolescence information from the package submitter.

In the presented case, it is absolutely clear from the start, that the new package releases have to go together, and that old releases are now obsoletes. I see no way to mention this in the **CRAN** submission process, except as a comment, and this will apparently not prevent current **CRAN** processes from wrongly triggering reverse dependencies check and build processes.

Ultimately, this is the story of the 100 bytes limits that turns out to trigger seismic shocks on **CRAN** processes.