

R Package Development at Novo Nordisk (Biostatistics)

Anders Ellern Bilgrau
abiu@novonordisk.com // 2022-11-14

Views and opinions expressed
are those of the speaker and
not necessarily Novo Nordisk



R extensions



- **R** is a language and environment for statistical computing and graphics
- R is easily **extendible** by versioned bundles of code and documentation called **packages**
- Each **package** *can depend* on multiple **other** packages and have system dependencies
- **CRAN** is a network that stores and serves **R**-packages
- **Current packages** on CRAN are checked and tested to ensure **compatibility** and a minimum standard

Packages



R packages

- The organisational unit of **extending** R and bundling functionality
- The **package source** is a collection of .R files, datasets, documentation, needed libraries and/or other source code (C, C++)
- The **package source** can be **bundled** (into a tar.gz) and/or **built** (into a binary) for distribution
- Can **easily** be **tested** and **checked** for standards
- Can **easily** be **shared** with others
 - **CRAN** for public, global sharing

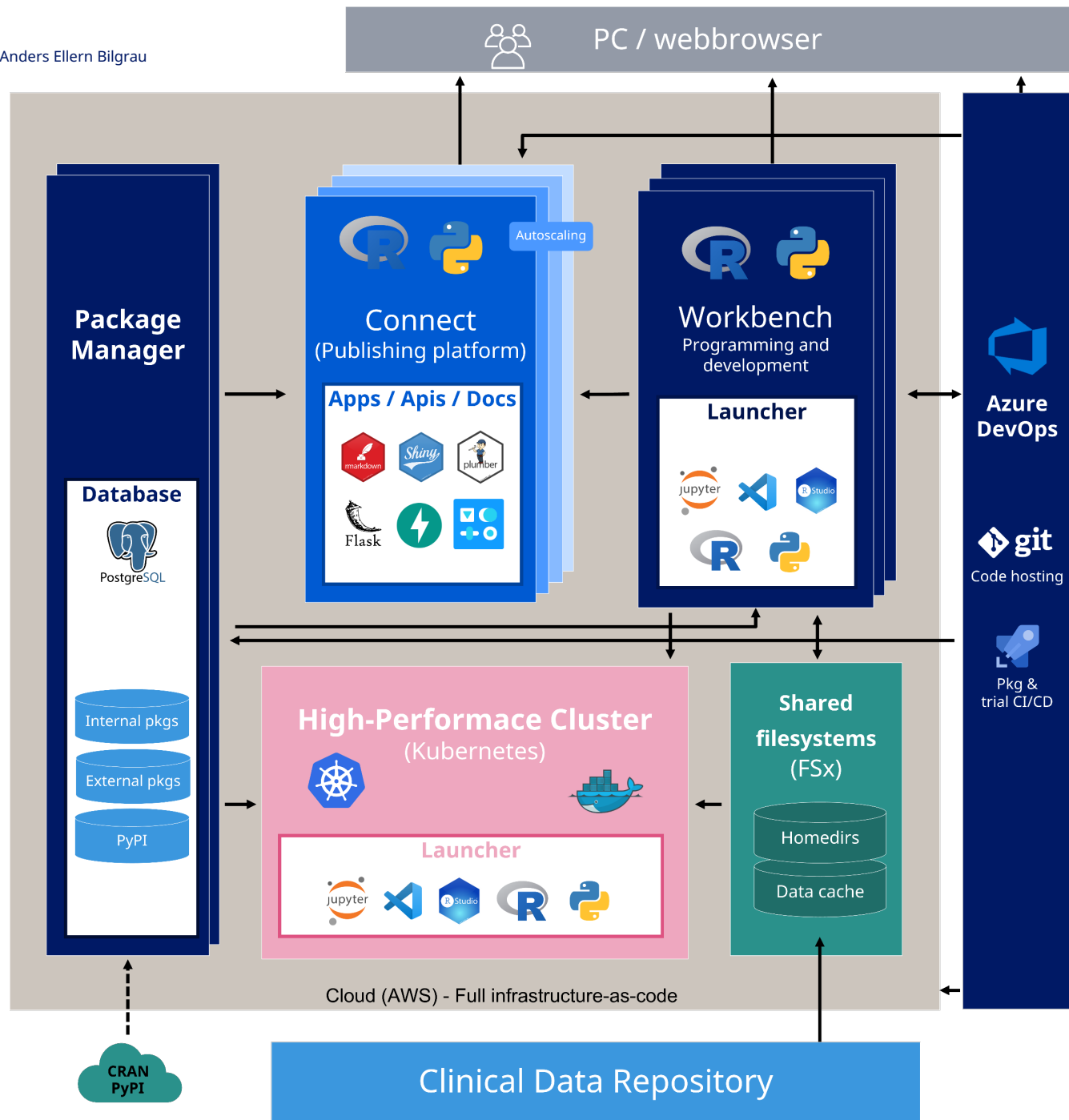
SAS macros

- Wraps functionality into function-like behavior
- Does not extend well to complex functionality
- No common way of testing and documenting

Python packages/modules

- One organisational unit of extending Python and bundling functionality
- Can **easily** be **shared** with others
 - **PyPI** for public, global sharing

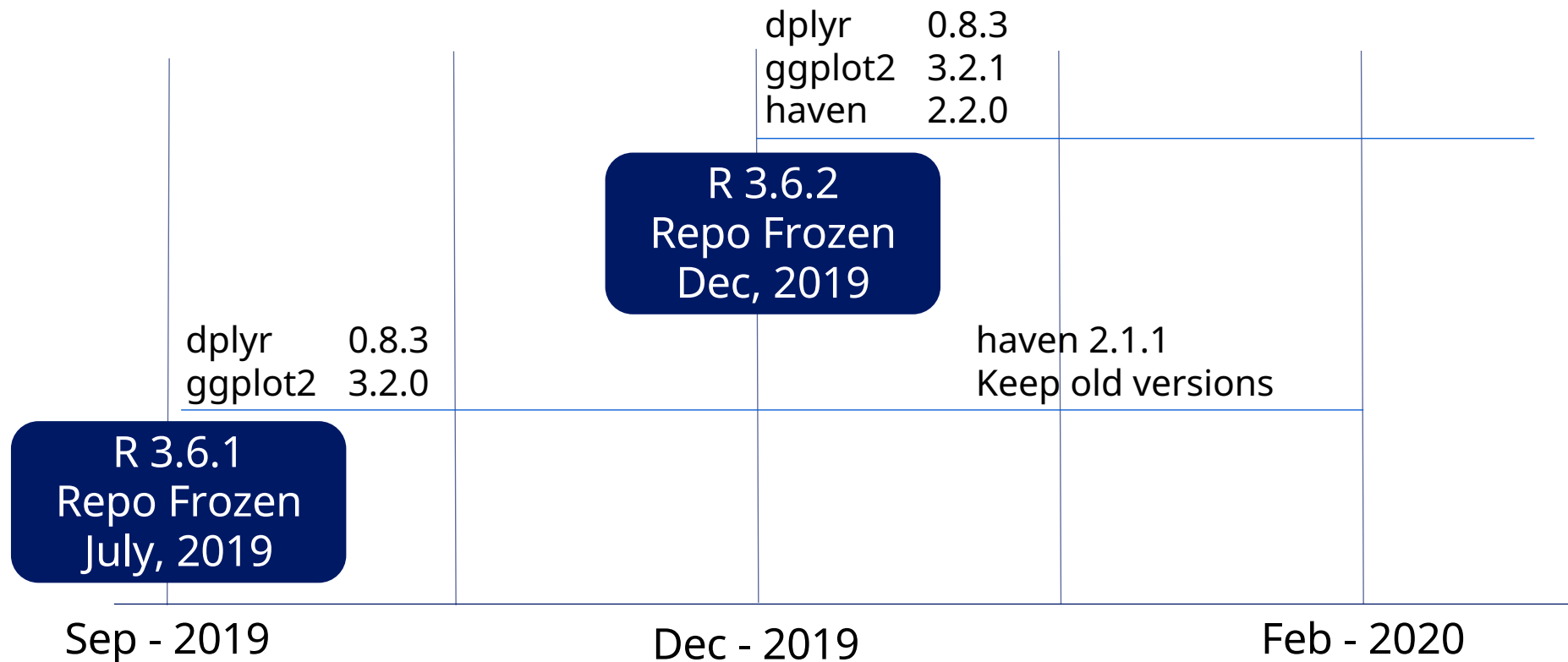
NN SCE-R infrastructure



NN Package Managment

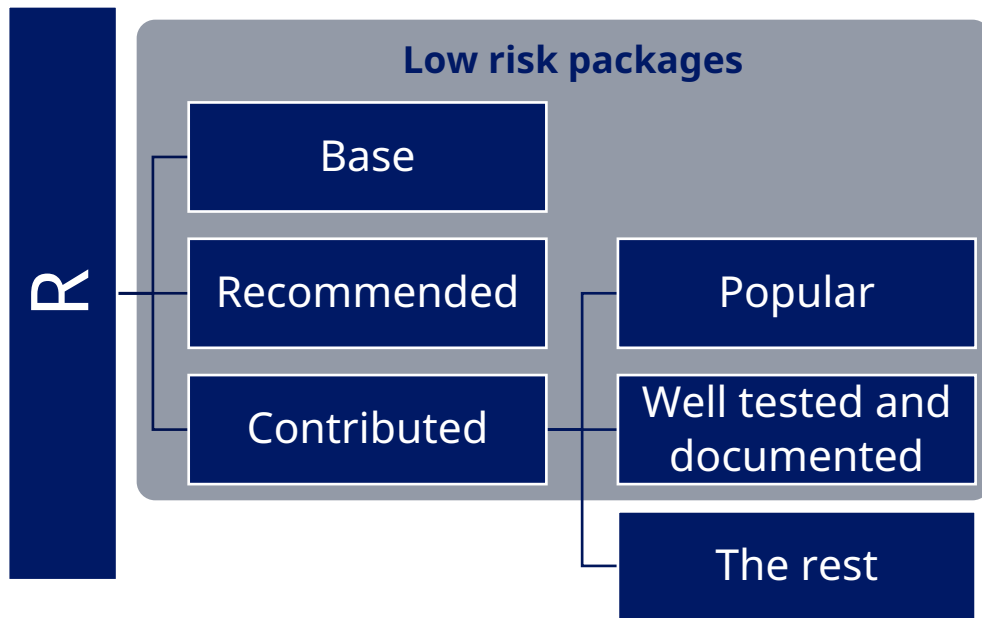
Package strategy in a validated environment

- Create an environment where less experienced users can easily share and re-run work, but restrict **immediate** access to a particular set of packages



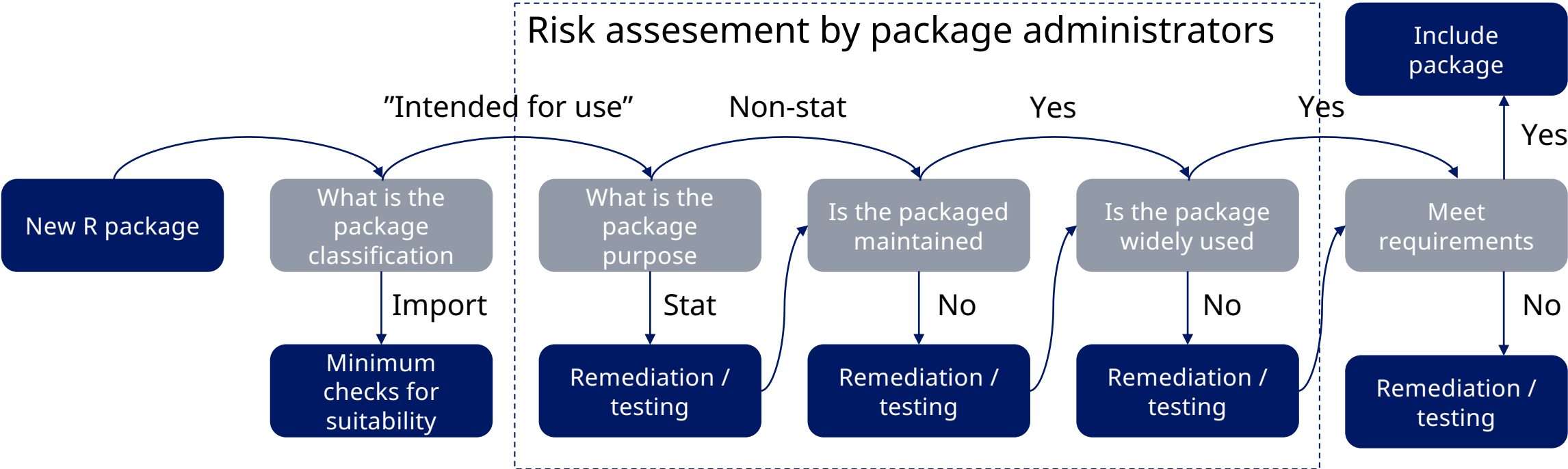
- Teams specify the version of **R** used for a specific trial
- Teams can choose to use specific versions of packages, but this should always be in conjunction with [renv](#)
- People can install packages into their own user library

Which packages can/should we use



- **Base/Recommended:** (Can be trusted)
 - The R Foundation develops both the base and recommended packages, and follows practices that ensures the accuracy of each
- **Contributed:** (Need internal testing)
 - 15000+ **R** packages on CRAN, all tested to some extent, but not all can be considered validated
- **Popular:** (Very low risk)
 - A subset of the contributed packages have an extremely large userbase and extensive test-suites
 - tidyverse, data.table, ...

Risk assessment strategy – R Validation Hub



Strategy suggested by: R-validation hub (<https://www.pharmar.org/blog/2020/01/30/2020-05-07-a-risk-based-approach-for-assessing-r-package-accuracy-within-a-validated-infrastructure-white-paper-summary/>)

NN package development philosophy

Tools-on-top development

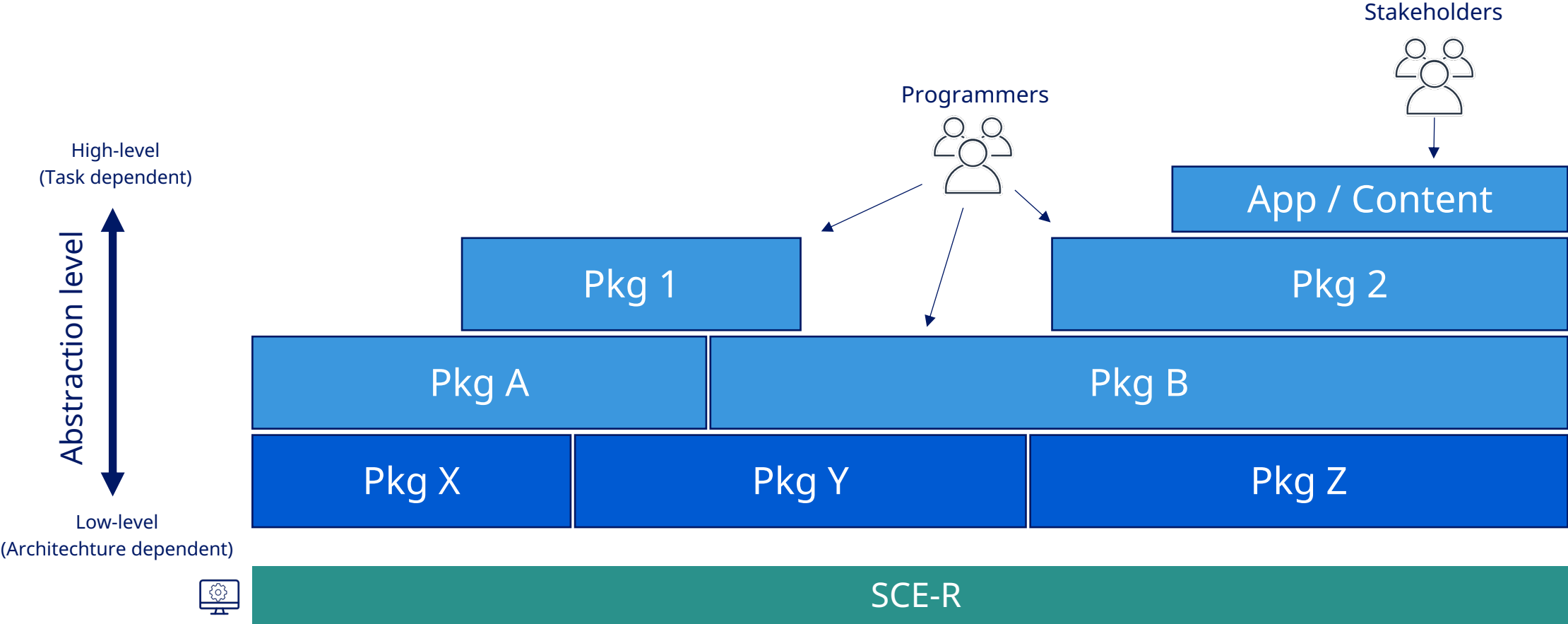
Packages

- Packages builds on top of each other (from low to high level)
- Lego-bricks principle
- Build up more complex functionality toward automation
- Output/app/content programming *all* uses the same packages
- Ability to make APIs to package functions with the connect server



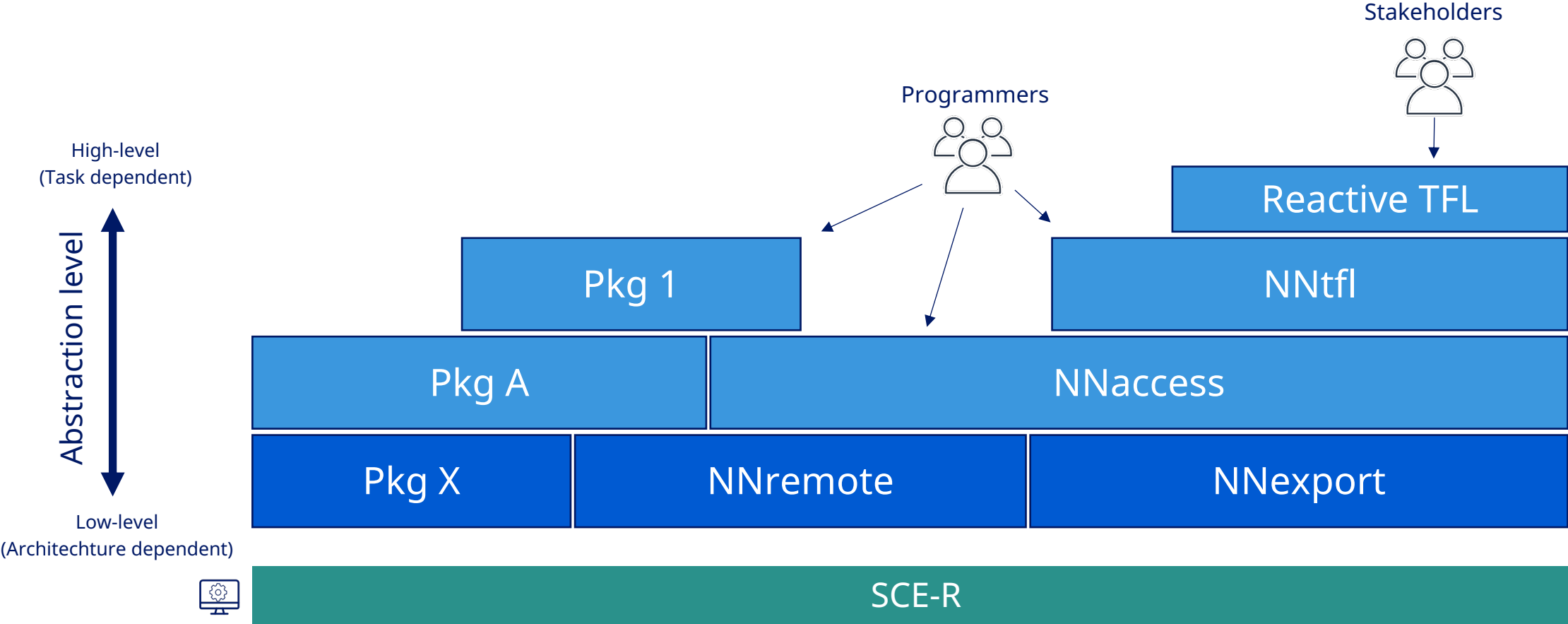
Tools-on-top development

Reused, modular, and replacable code



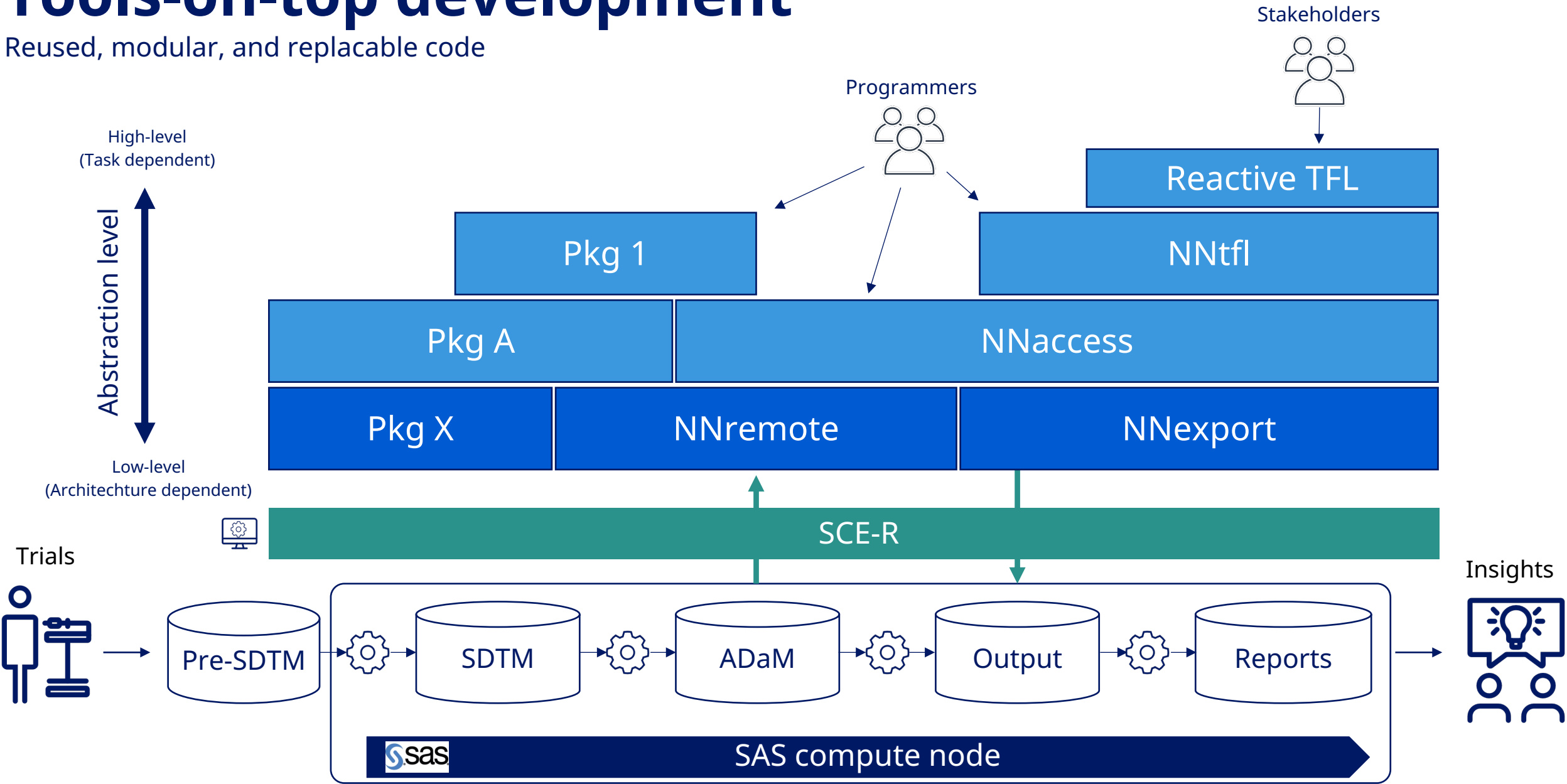
Tools-on-top development

Reused, modular, and replacable code



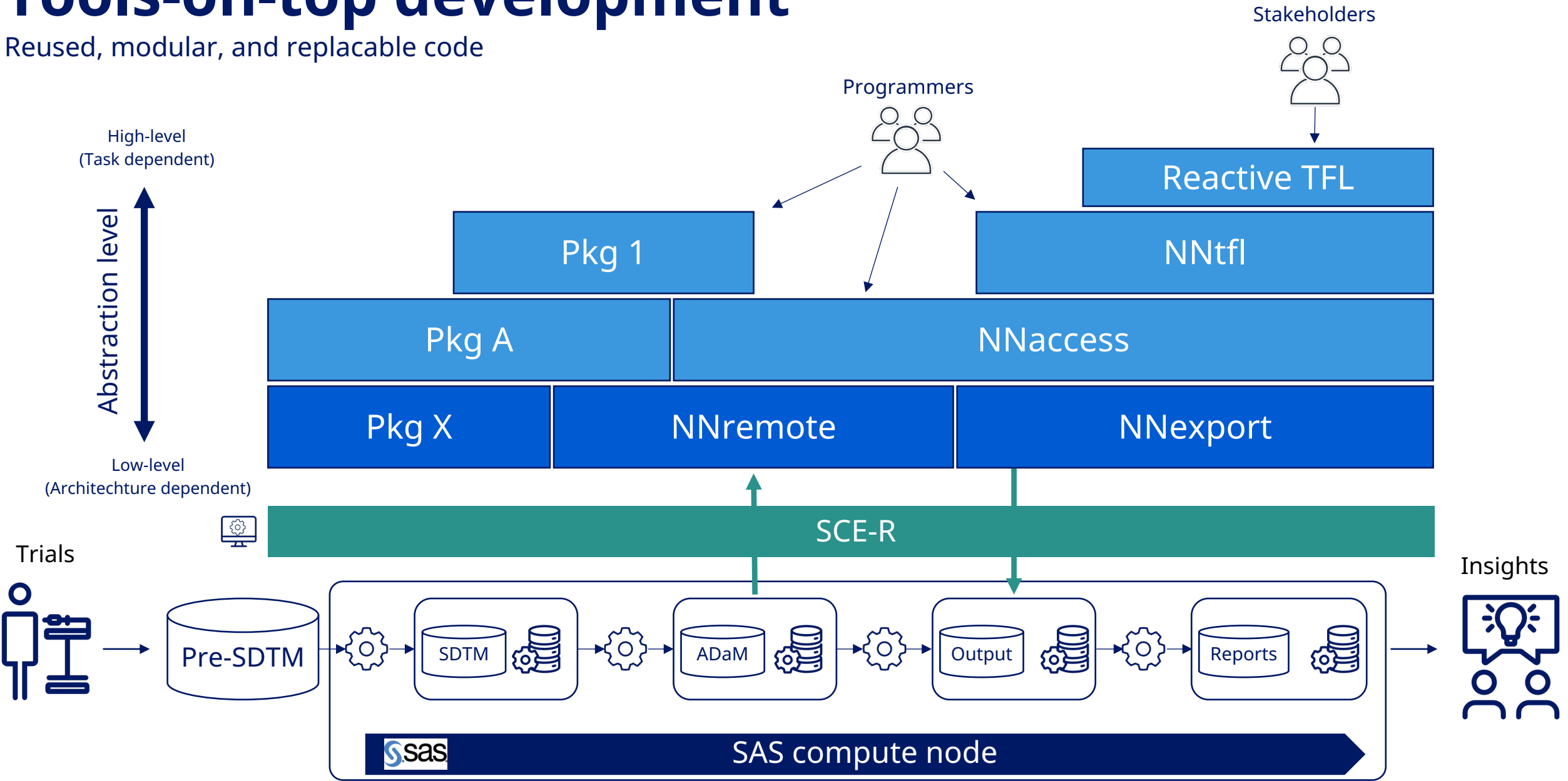
Tools-on-top development

Reused, modular, and replacable code



Tools-on-top development

Reused, modular, and replacable code



R Package development in NN

Internal packages

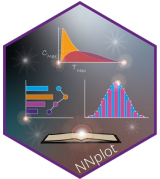


- Started with the package "NNR" developed from local laptops (no SCE supporting R)
- A "NNBiostat" package defined to be "accepted packages"
- NNR redefined to umbrella of logically related packages

Internal packages producing TFL



- Supplies easy access to read and write data
- Easy to direct outputs to wanted folders



- Plots with titles, footnotes, and NN colours
- Plots With colours and symbols as defined by MDPARAM



- Tables similar to current layout from SAS
- Designed in a flexible format so that we can add stuff to it

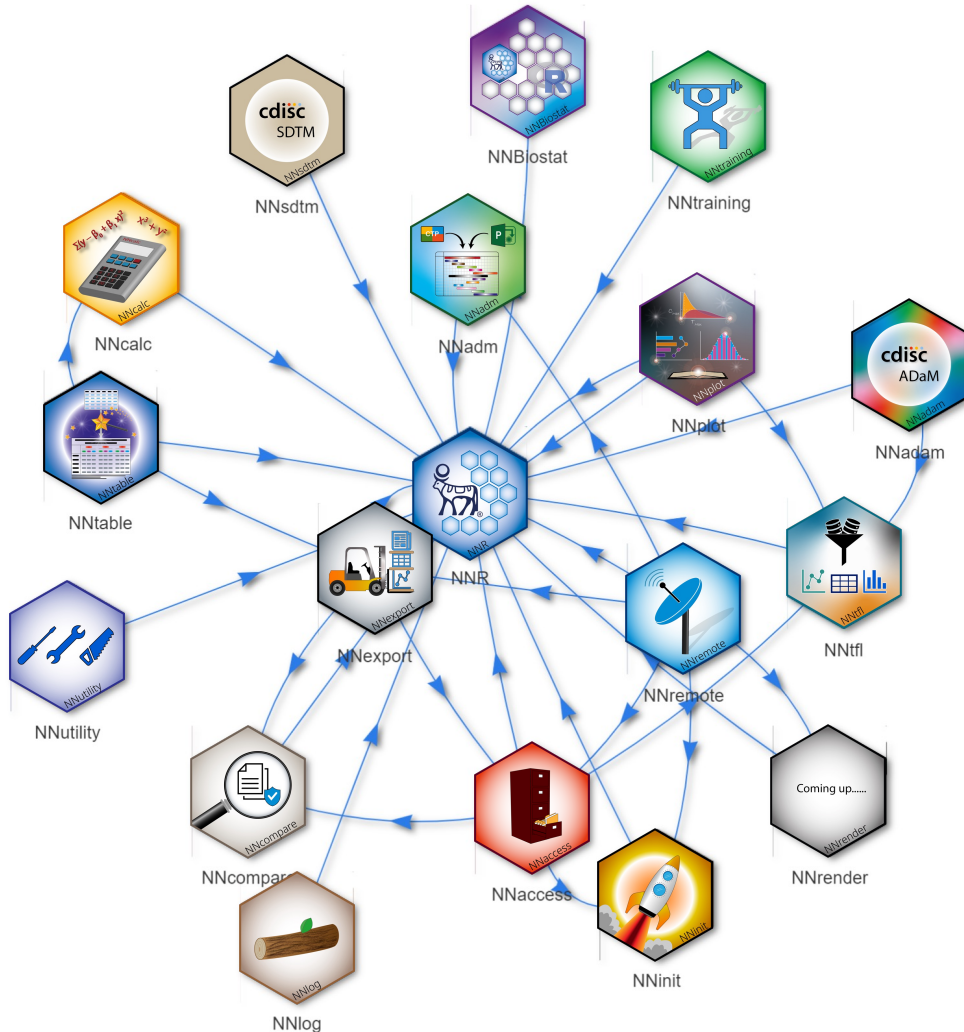


- Export tables and figures in a format that can be used in our current TFL pipeline
- Generate associated xml files automatically



- Run R and Rmd scripts in batch mode
- Produce nicely formatted html and markdown logs – integrated with Azure DevOps

Internal packages – Network



- The internally developed packages are dependent on each other
 - No cyclic dependencies
- Each package is hosted version controlled on **Azure DevOps** and has:
 - Version number
 - Maintainer group
 - Repository
 - Pipeline
 - Board for bugs, questions, feature requests

Internal package / tool chain

```
|— .git
|— azure-pipelines.yml
|— DESCRIPTION
|— docs
|   |— *.html
|— inst
|   |— extdata
|— man
|   |— *.Rd
|— NAMESPACE
|— NEWS.md
|— pkgdown
|   |— ...
|— _pkgdown.yml
|— R
|   |— *.R
|— README.md
|— tests
|   |— testthat
|   |   |— *.R
|   |— testthat.R
|— vignettes
|   |— *.Rmd
```


Internal package / tool chain

```
|— .git .....  
|— azure-pipelines.yml  
|— DESCRIPTION  
|— docs  
|   |— *.html  
|— inst  
|   |— extdata  
|— man  
|   |— *.Rd  
|— NAMESPACE  
|— NEWS.md  
|— pkgdown  
|   |— ...  
|— _pkgdown.yml  
|— R  
|   |— *.R  
|— README.md  
|— tests  
|   |— testthat  
|       |— *.R  
|       |— testthat.R  
|— vignettes  
|   |— *.Rmd
```

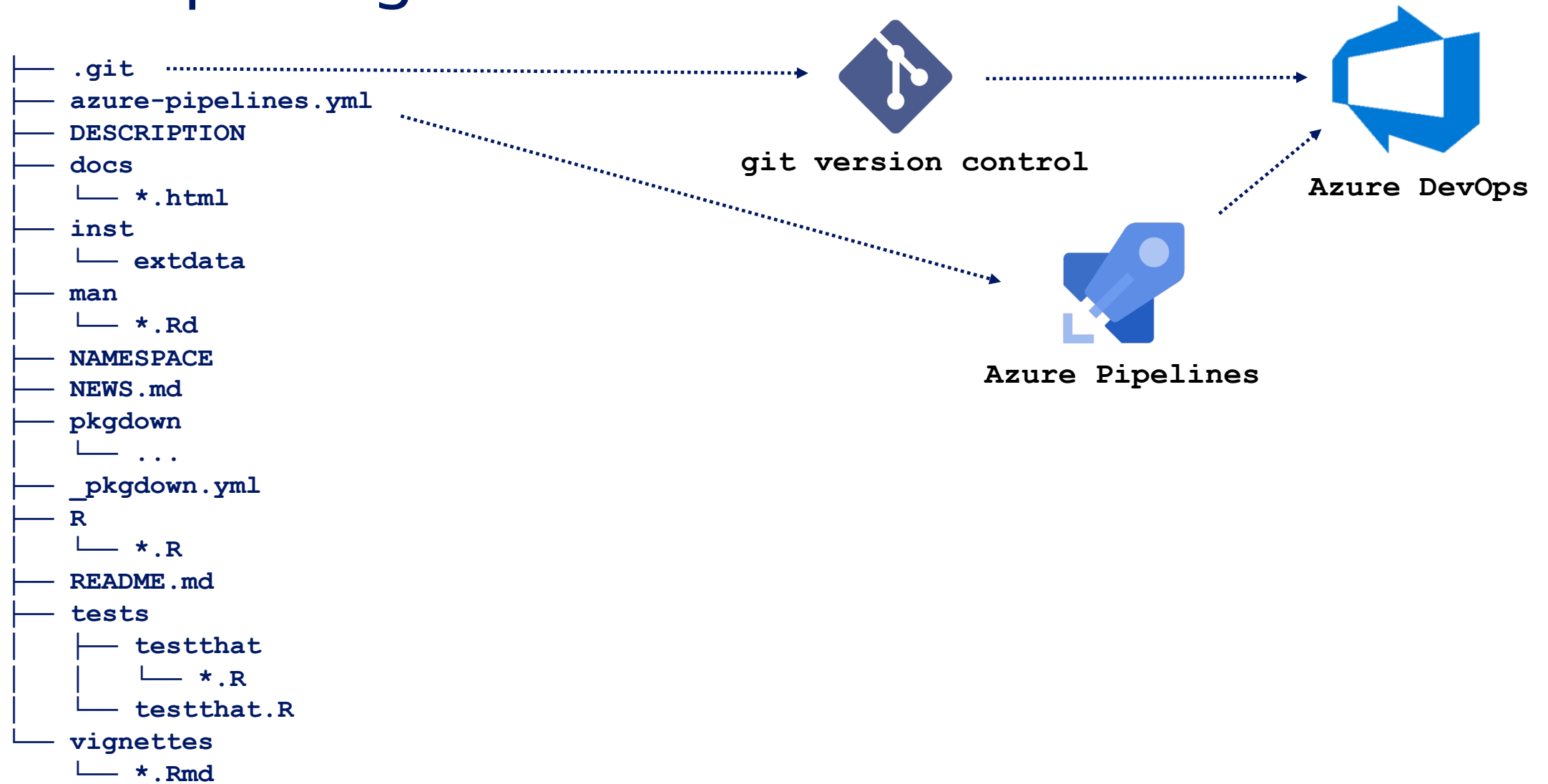


git version control

Internal package / tool chain

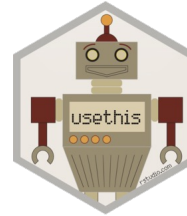


Internal package / tool chain



Internal package / tool chain

```
.git
azure-pipelines.yml
DESCRIPTION
docs
├── *.html
inst
├── extdata
man
├── *.Rd
NAMESPACE
NEWS.md
pkgdown
├── ...
_pkgdown.yml
R
├── *.R
README.md
tests
├── testthat
│   └── *.R
└── testthat.R
vignettes
├── *.Rmd
```



git version control



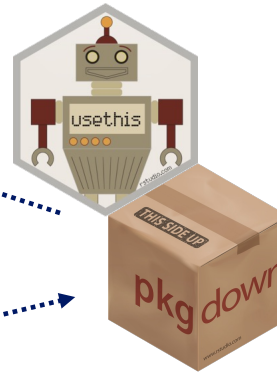
Azure DevOps



Azure Pipelines

Internal package / tool chain

```
.git
azure-pipelines.yml
DESCRIPTION
docs
├── *.html
inst
├── extdata
man
├── *.Rd
NAMESPACE
NEWS.md
pkgdown
├── ...
└── _pkgdown.yml
R
├── *.R
README.md
tests
├── testthat
│   └── *.R
└── testthat.R
vignettes
├── *.Rmd
```



git version control



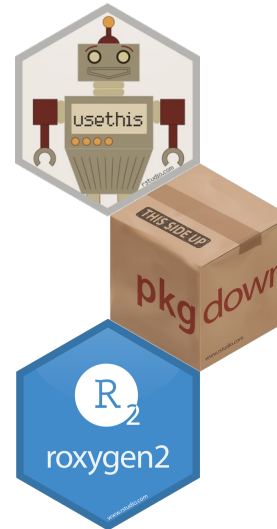
Azure DevOps



Azure Pipelines

Internal package / tool chain

```
.git
azure-pipelines.yml
DESCRIPTION
docs
├── *.html
inst
├── extdata
man
├── *.Rd
NAMESPACE
NEWS.md
pkgdown
├── ...
└── _pkgdown.yml
R
├── *.R
README.md
tests
├── testthat
│   └── *.R
└── testthat.R
vignettes
├── *.Rmd
```



git version control



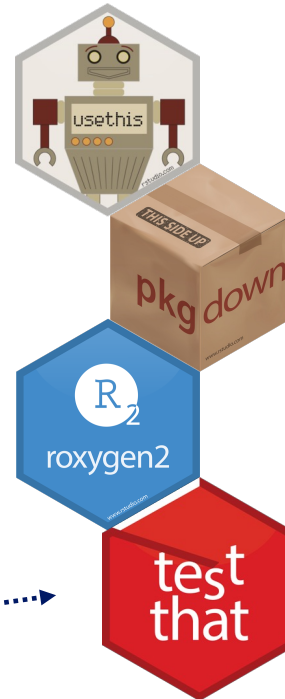
Azure DevOps



Azure Pipelines

Internal package / tool chain

```
.git
azure-pipelines.yml
DESCRIPTION
docs
├── *.html
inst
├── extdata
man
├── *.Rd
NAMESPACE
NEWS.md
pkgdown
├── ...
_pkgdown.yml
R
├── *.R
README.md
tests
├── testthat
│   └── *.R
├── testthat.R
vignettes
├── *.Rmd
```



git version control



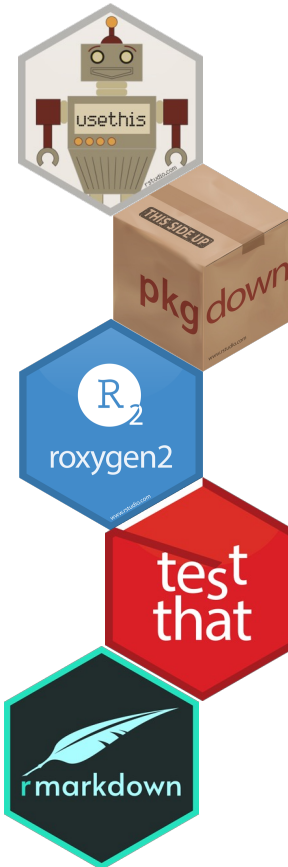
Azure DevOps



Azure Pipelines

Internal package / tool chain

```
.git
azure-pipelines.yml
DESCRIPTION
docs
├── *.html
inst
├── extdata
man
├── *.Rd
NAMESPACE
NEWS.md
pkgdown
├── ...
_pkgdown.yml
R
├── *.R
README.md
tests
├── testthat
│   └── *.R
└── testthat.R
vignettes
├── *.Rmd
```



git version control



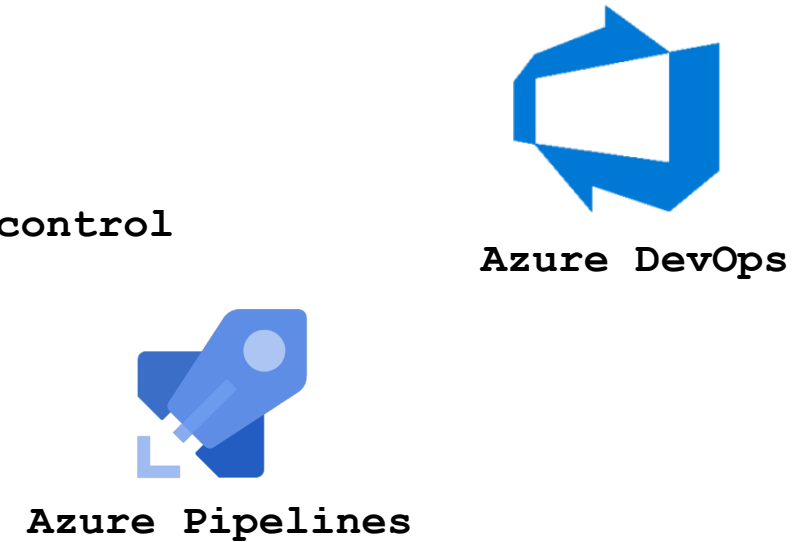
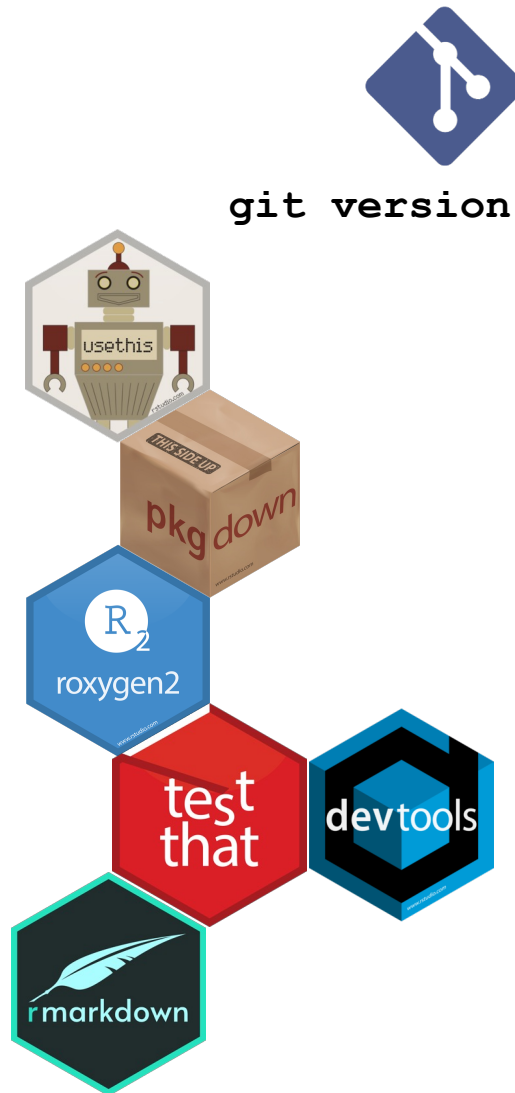
Azure DevOps



Azure Pipelines

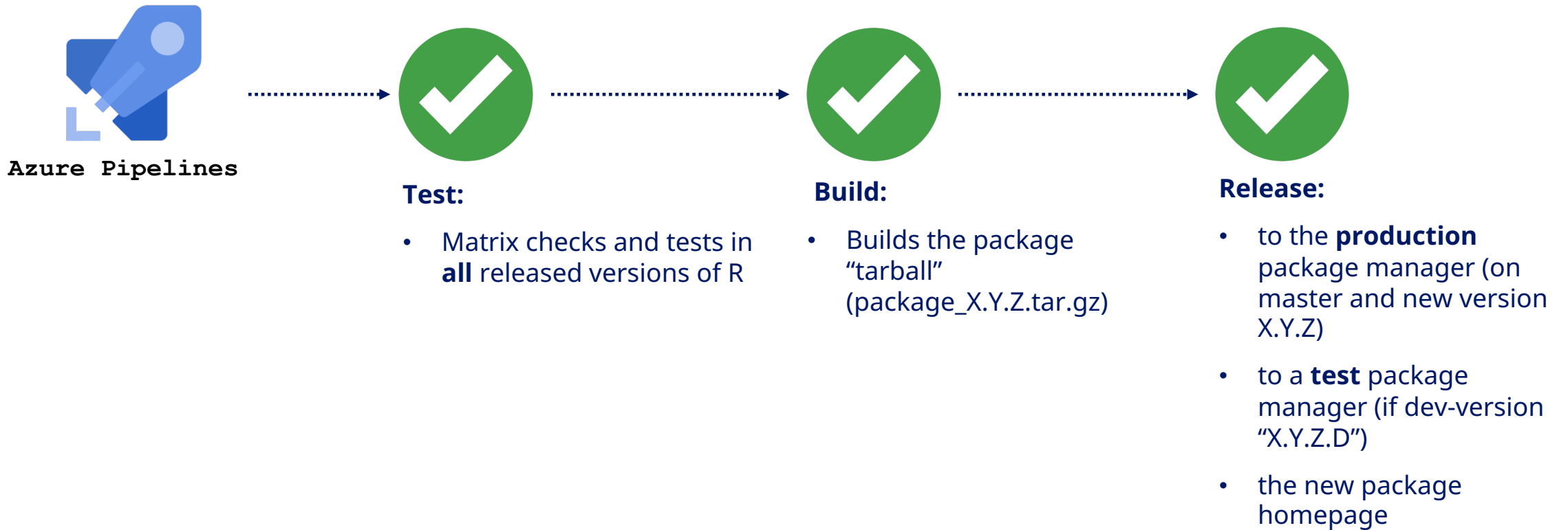
Internal package / tool chain

```
.git
azure-pipelines.yml
DESCRIPTION
docs
├── *.html
inst
├── extdata
man
├── *.Rd
NAMESPACE
NEWS.md
pkgdown
├── ...
_pkgdown.yml
R
├── *.R
README.md
tests
├── testthat
│   └── *.R
└── testthat.R
vignettes
├── *.Rmd
```



Continuous integration and deployment

On each code-change/commit, the package pipeline is triggered and runs:



Continuous integration and deployment

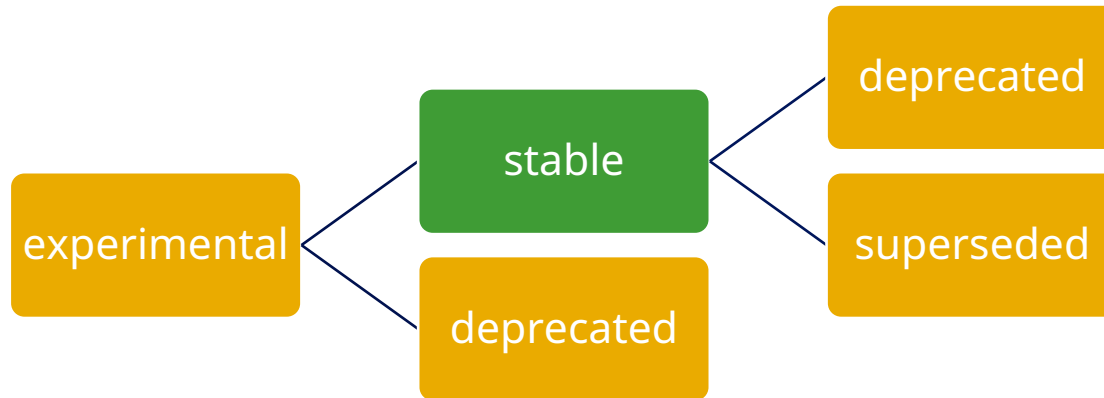
- A git master-branch **policy** is enforced:
 - Only via pull-requests
 - A person in the maintainer group should approve the code changes
 - Pipeline runs with **success**
 - A feature/bug is linked



Ensures validation of internal packages

Package and function lifecycle

lifecycle **stable**



- Functions are marked using the lifecycle package to communicate to users the risk level of functions within a package
- **Future:**
 - Stable functions must be covered thoroughly by tests
 - Demands on experimental functions are relaxed

User support & education

User support & education


Support

- **One-entry point documentation**
- **Establishing a dedicated support team**
- **Establishing a StackOverflow enviroment**

Education

1. Git is a prerequisite
 - All trial/exploratory tasks that use R should use git
2. Conduct courses in how to get started with R in NN
 - Will include some training in git
3. R self-learning and learning-by-doing
 - Material for self-learning is provided via degreed/r-doc

Homepage


[Home](#)
[Get started](#)
[Connect](#)
[R packages +](#)
[pRodcast](#)
[Working with R +](#)

NNPackages

- Package installation
- Feature requests and bug reports
- Output Catalogue

SCE-R

The Department of Biostatistics have setup an R and python environment that is GxP approved for production of tables, figures, and listings. The environment currently consists of three main elements:

- The **Workbench-servers** for creating and executing R-scripts directly or via Azure Pipelines
- The **Connect-servers** for publishing content and output.
- Two **Package management** servers for hosting R-package repositories:
 - **Production** for production packages. The packages made available on this server can be installed from the Workbench by `install.packages` without specifying repos.
 - **Bleeding edge** for bleeding edge packages. The packages made available on this server can be installed from the Workbench by the `install.packages`.

All servers use the same image and should therefore behave similarly when running code. The packages should preferably be part of BOS.

Several popular packages come pre-installed on the system and is ready to use. This includes several packages developed internally in Novo Nordisks Biostatistics Department (see below).

NNPackages

The **NNPackages** is a collection of R-packages designed for working with R within the department of biostatistics. Each package covers a distinct set of tools needed to solve everyday tasks. The hex stickers below links to help pages for each specific package. It is also possible to go to [packages](#) which contain a small description of each package.

- Everything needed to use R within Biostatistics should be available at [r-doc](#)
- Go to the homepage by typing r-doc in the browser



Vignettes

NNcompare0.1.13

ReferenceArticlesChangelog

at Biostat

Output parallel programming

Mette Dahl Bendtsen

2021-01-21

Setup training data base

In case you have not already setup a training database please run

```
NNtraining::createTrainingDB()
#> [1] TRUE TRUE TRUE
```

Assuming that you work on the R-server, this will copy the training data sets from the NNtraining package into your home directory in a sub folder called "training". If the files already exists, nothing will happen. The training folder can then be used to read/write from, as if it was a real trial folder on the p-drive.

Introduction

The NNcompare package can support the parallel review process when the parallel program is written in R. The `comparedf()` function from the arsenal package can be used to compare two data frames, and the `export()` function in the NNcompare package can be used to export a summary report of the comparison in various formats (HTML, Word, PDF, and github).

This vignette gives examples of how to use the NNcompare package in a parallel review process.

Example 1: Adverse events by SOC/PT

A snippet of the table that should be parallel reviewed is shown here - it is a more or less standard adverse events table showing AEs by treatment, system organ class and preferred term:

1: Adverse events by system organ class and preferred term - safety analysis set

MedDRA system organ class Preferred term	new drug N (%)	E	R	old drug N (%)	E	R	total N (%)	E	R
Subjects Exposed	381			420			801		
Total patient years	166.2			190.9			356.2		
All adverse events	357 (93.7)	1330	809.7	406 (96.7)	1620	848.6	763 (95.3)	2950	830.6
Infections and infestations	153 (40.2)	200	121.8	192 (45.7)	259	135.7	345 (43.1)	459	129.2
Round infection	27 (7.1)	27	16.4	35 (7.9)	56	19.3	62 (7.5)	62	17.7
Staphylococcal infection	21 (5.6)	21	12.8	19 (4.6)	20	10.8	40 (5.0)	41	11.6
Urinary tract infection	18 (4.9)	19	9.7	25 (6.0)	24	12.4	38 (4.7)	39	13.0
Tracheitis	12 (3.1)	13	7.9	24 (5.7)	25	13.1	36 (4.5)	38	10.7
Viral upper respiratory tract infection	16 (4.3)	16	9.7	20 (4.8)	20	10.8	36 (4.4)	36	10.1
Knotorrhitis	17 (4.8)	17	10.4	16 (3.8)	17	8.9	33 (4.1)	36	9.4
Gastroenteritis	14 (3.7)	15	9.1	13 (3.1)	20	10.5	27 (3.3)	35	9.9
Cytitis	19 (4.7)	19	11.6	12 (2.9)	13	6.3	30 (3.7)	31	8.7
Pharyngotonsillitis	11 (2.9)	11	6.3	19 (4.3)	18	9.4	29 (3.6)	29	8.2
Viral infection	8 (2.1)	8	4.9	20 (4.8)	20	10.5	28 (3.5)	28	7.9
Conjunctivitis	13 (3.4)	13	7.9	9 (2.1)	9	4.7	22 (2.7)	22	6.2
Ear infection	9 (2.4)	9	5.4	13 (3.1)	13	6.8	22 (2.7)	22	6.2
Dermatocellitis	5 (1.3)	5	3.0	12 (2.9)	12	6.3	17 (2.1)	17	4.8

Contents

Setup training data base

Introduction

Example 1: Adverse events by SOC/PT

Example 2: Most frequent (>=5%) adverse events by PT/SOC

Summary of comparisons

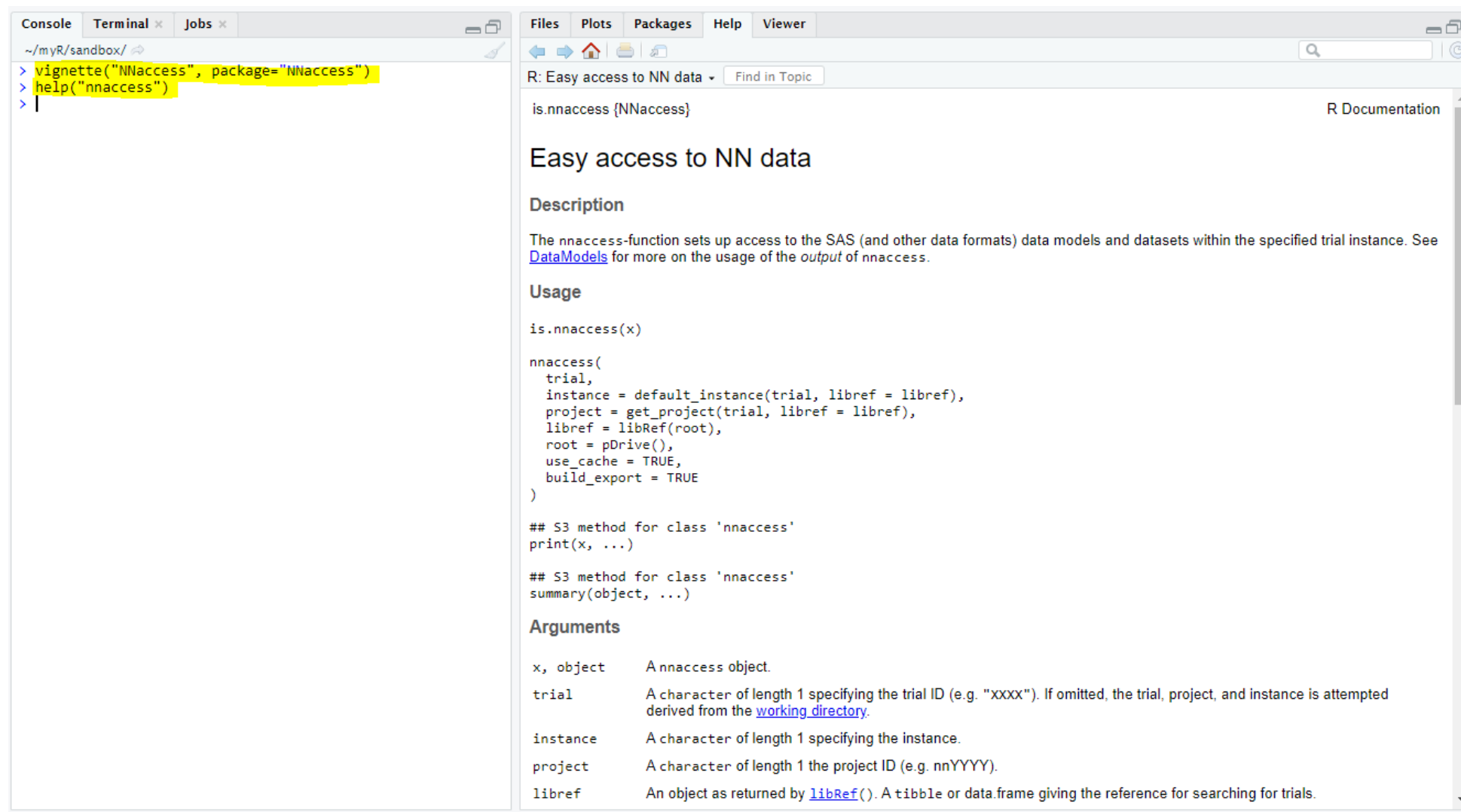
- Directly available from training platform (and from within R)
- Re-run when package is built (to ensure it runs)
- Anybody can contribute
- Written in R-markdown
- Gradually being made

pRodcast

Episode	Level	Released	Description
1	1	05-01-2021	Getting started with the SCE-R (R-server) in Biostatistics (https://rstudio1.bifrost-prd.corp.aws.novonordisk.com/). Make a sandbox project. A quick tour of Rstudio (editor, console, terminal, file explorer, environment). Where are the files located on the R-server. Using pmount and hmount in the terminal to get access to P-drive and H-drive. Make an R script that loads NNBiostat. Install the NNtraining package and call createTrainingDB. Make an R script that loads the packages in NNBiostat and loads trial datasets.
2	2	07-01-2021	A closer look at accessing data using tidyverse (https://www.tidyverse.org/) functions %>%, filter, mutate, arrange, select, group_by, summarize. Read the book "R for datascience" (http://r4ds.had.co.nz/).
3	1	12-01-2021	Making Gantt diagrams from impact data using the getCTP function. Using tidyverse functions to select trials. Use a reference date and window. Add events by editing an excel sheet.
4	3	14-01-2021	Create an AE table using tidyverse commands and make it output-ready using NNtable. Export the table.
5	2	19-01-2021	Working in Rstudio projects and sessions. Consider to quit sessions. Note you can "restart R" within a session using CTRL-SHIFT-F10 (or menu "Session / Restart R").
6	3	21-01-2021	Follow these instructions: http://10.59.86.7/getstarted.html . Connecting with Azure DevOps and using Git. You need to do the connection once. Use the functions in NNinit.
7	1	26-01-2021	Having problems with the R-server. Where to report issues. Loss of connection to p-drive and h-drive. Problems logging in.
8	1	28-01-2021	Feedback to this podcast. Go to NNtraining board (https://novonordiskit.visualstudio.com/BOS/_boards/board/t/NNtraining/Stories) and pose Questions, Feature requests (i.e. ideas for new episodes), Bug reports (eg if something has changed since the episode was recorded). Note that you can view the pRodcast on your iphone via the Stream app available in the NN App store (potentially via airplay/chromecast). You are welcome to 'Like' an episode if you think it was helpful to you - this could guide others.

- 2 episodes released per week
- 10-15 minutes 'how-to'
- All code examples available in vignettes
- DevOps system for getting feedback
- Anybody can contribute with episodes

Help at your fingertips (within R)



The screenshot displays an R IDE interface with two main panels. The left panel, titled 'Console', shows the following commands being executed:

```
> vignette("NNaccess", package="NNaccess")
> help("nnaccess")
> |
```

The right panel, titled 'R Documentation', shows the help page for the 'nnaccess' package. The page title is 'Easy access to NN data'. The description states: 'The nnaccess-function sets up access to the SAS (and other data formats) data models and datasets within the specified trial instance. See [DataModels](#) for more on the usage of the output of nnaccess.'

The usage section shows the function signature:

```
is.nnaccess(x)

nnaccess(
  trial,
  instance = default_instance(trial, libref = libref),
  project = get_project(trial, libref = libref),
  libref = libRef(root),
  root = pDrive(),
  use_cache = TRUE,
  build_export = TRUE
)
```

Below the usage section, there are two S3 methods for the class 'nnaccess':

```
## S3 method for class 'nnaccess'
print(x, ...)

## S3 method for class 'nnaccess'
summary(object, ...)
```

The arguments section lists the parameters:

Argument	Description
x, object	Annnaccess object.
trial	A character of length 1 specifying the trial ID (e.g. "xxxx"). If omitted, the trial, project, and instance is attempted derived from the working directory .
instance	A character of length 1 specifying the instance.
project	A character of length 1 the project ID (e.g. nnYYYY).
libref	An object as returned by libRef() . A tibble or data.frame giving the reference for searching for trials.

StackOverflow

- We have no good place for Q&A
 - ***Tried:*** MS SharePoint, MS Teams, Azure DevOps, Service Now, Wiki
- Avoid answering the same question 100 times
- Important as it alleviates support burden
 - R support team should check questions here
- StackOverflow is familiar for programmers
- Not *just* for code

