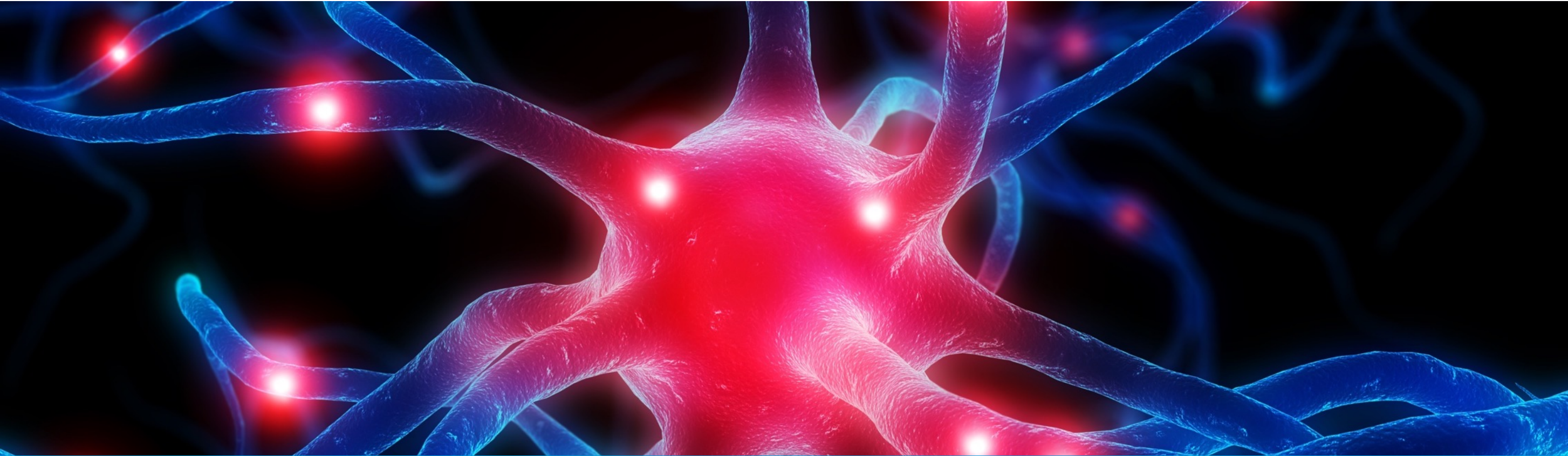




AD02 - Introducing raccess

An R package that provides a generic framework for implementing sophisticated access controls in Shiny applications

Peyman Eshghi
PEshghi@its.jnj.uk

Janssen UK

Active neuron

Agenda

- 1 **What is raccess**

- 2 **Access Management, without raccess**

- 3 **Access Management, with raccess**

- 4 **Demo/ Examples**

- 5 **Q&A**

What is raccess

raccess is an **R package**,
with **Shiny module** functions,
enabling **multiple-layer access management**
in shiny applications.



<https://jnj.sharepoint.com/sites/PHM-GCSP/JIL>

What is raccess

- It provides a framework to facilitate **within-the-app** access control mechanism with the ability to store, organize, and retrieve access lists locally and/ or remotely.
- It includes a **pluggable shiny module** to create and maintain access lists for individual Shiny applications.
- It sits on top of the RStudio Connect Access Management and therefore no credentials will be collected or stored.

Access Management, without raccess

- RStudio Connect has made it simple to publish and maintain apps in organisations.
- Access to individual apps can be easily managed through Access tab within the RStudio Connect.

Access Management, without racess

- You need to have a **publish** account in order to grant access to others
 - Organisations may want to restrict publish accounts to app developers only
 - App developers/ publishers do not want to be in charge of access control
- RStudio Connect access management only provides a **single layer** binary control
 - i.e. users either have access to an app or they don't

Access Management, with raccess

raccess terminology

- **Access Unit**
 - the smallest unit that you want to control access to
 - it can be
 - a specific module in the app
 - a single UI element
 - a collection of UI elements
 - etc...
- **Access Panel**
 - a container for a collection of **Access Units**
 - can be used to implement deep layers of access control

Access Management, with raccess

Workflow

- **App owner** can identify multiple **Access Panels** and a number of **Access Units** within each **Access Panel**
- **App developer** lists the **Access Panels** and **Access units** in a config Yaml file
- **App developer** will implement the required server side controlling logic.
- **App publisher** will publish the app with the sharing settings of:
 - All users - login required
- **App users/ admins** can control and maintains access lists within the app. A nice shiny GUI is provided to facilitate this.

Access Management, with raccess

Behind the scene

ADMIN Panel	Access Panel 1	Access Panel 2	Access Panel 3	Etc.
admin_list [1]	admin_list [2]	admin_list [3]	admin_list [4]	...

	Access Unit 1	Access Unit 2	Access Unit 3	Access Unit 4	Access Unit 5
Access Panel 1	user_list [1,1]	user_list [1,2]	user_list [1,3]	user_list [1,4]	user_list [1,5]
Access Panel 2	user_list [2,1]	user_list [2,2]	user_list [2,3]	user_list [2,4]	user_list [2,5]
Access Panel 3	user_list [3,1]	user_list [3,2]	user_list [3,3]	user_list [3,4]	user_list [3,5]
Etc.	...	...	...	...	...

Access Management, with raccess

raccess package key functions

[module_iam_server\(\)](#)
[module_iam_ui\(\)](#)

module functions to plug to the app

check_access(a_p, a_u, user)

check user's access to Access Units of a given Access Panel

is_admin(user)

check if the user is an admin

no_admin()

check if the app has no admins (i.e. open access)

access_list(a_p)

return the current access list for a given Access Panel

access_lists()

return all access lists for all Access Panels in the app

Demo/ Examples (1)

Both dataset A and dataset B are fed to the app

App owner wants to control who should have access to what dataset

Developer sets up the config file

```
# config.yml
app_name: "a_demo_app"
access_panels: !expr c("ADMIN", "DATA")
access_units: !expr c("Dataset_A", "Dataset_B")
```

Developer adds controls (server logic)

```
# in serve.R
myAccess <- check_access("DATA") # check if logged in user has access to
If(myAccess$'Dataset_A') {
  # consequent statements
}
```

App user/ admin can add or edit users (GUI)

ADMIN		DATA	
Add User		Edit User Access	
Show 10 entries		Search:	
UserName	UserID	Dataset A	Dataset B
Peyman Eshghi	PEshghi	✓	

Demo/ Examples (2)

Study data A and B are fed, and the app generates a **plot** and a **table** for each study

App owner wants all users to see the **plot** but only certain users to see the **table**

Developer sets up the config file

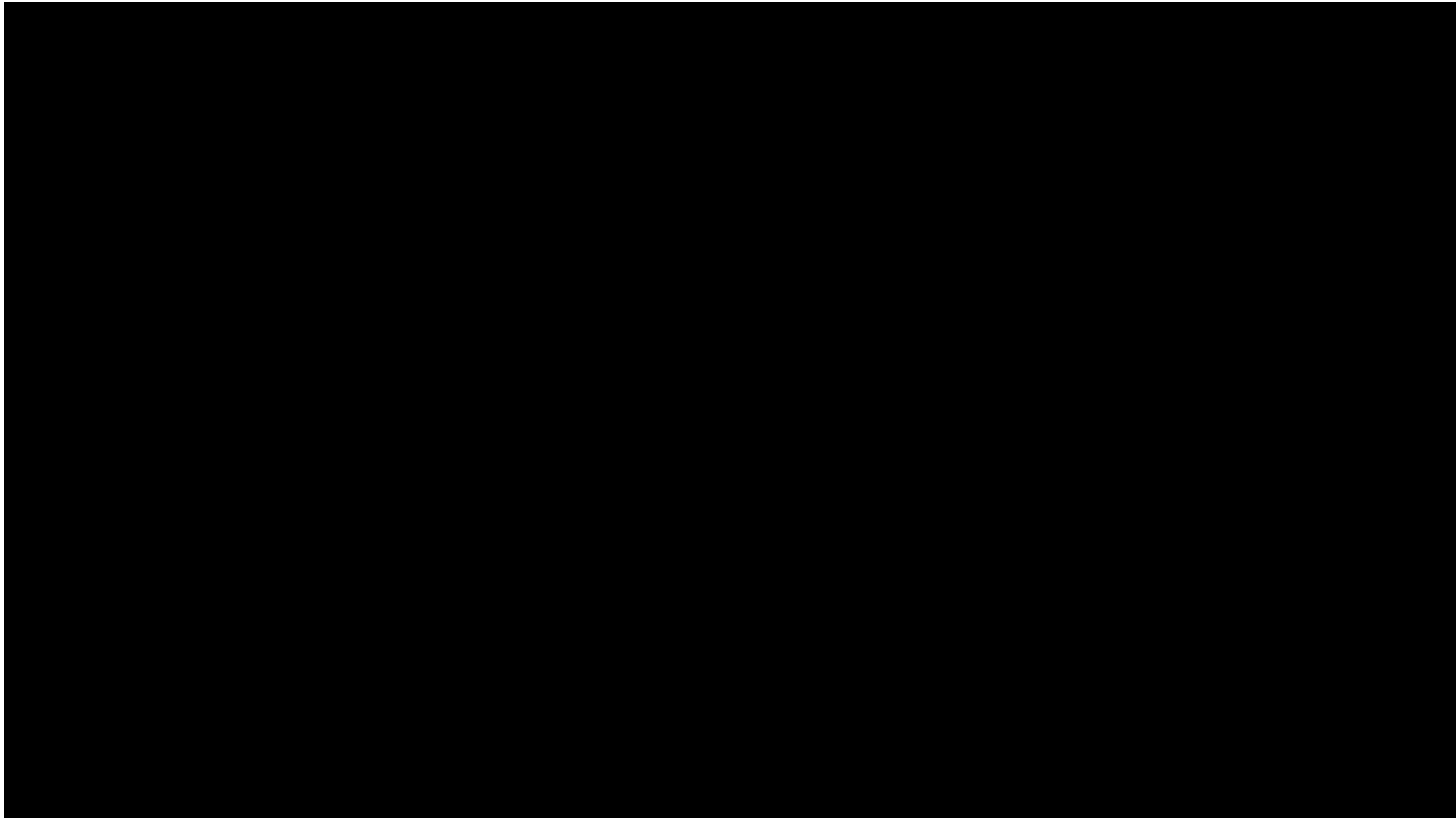
```
# config.yml
app_name: "a_demo_app"
access_panels: !expr c("ADMIN", "STUDY_A", "STUDY_B")
access_units: !expr c("PLOT", "TABLE")
```

Developer adds controls (server logic)

```
myAccess <- check_access("STUDY_A") # check what units does logged-in
user have access to
If(myAccess$TABLE) {
  # show the study A related tables....
}
```

App user/ admin can add or edit users (GUI)

ADMIN		DATA A	DATA B
Add User		Edit User Access	
Show 10 entries		Search:	
UserName	UserID	PLOT	TABLE
Peyman Eshghi	PEshghi	✓	
NICHOLAS MASEL	nmasel		✓



Q&A

Skin cells at 20x magnification

