Point Cloud dataviz and Shiny Apps

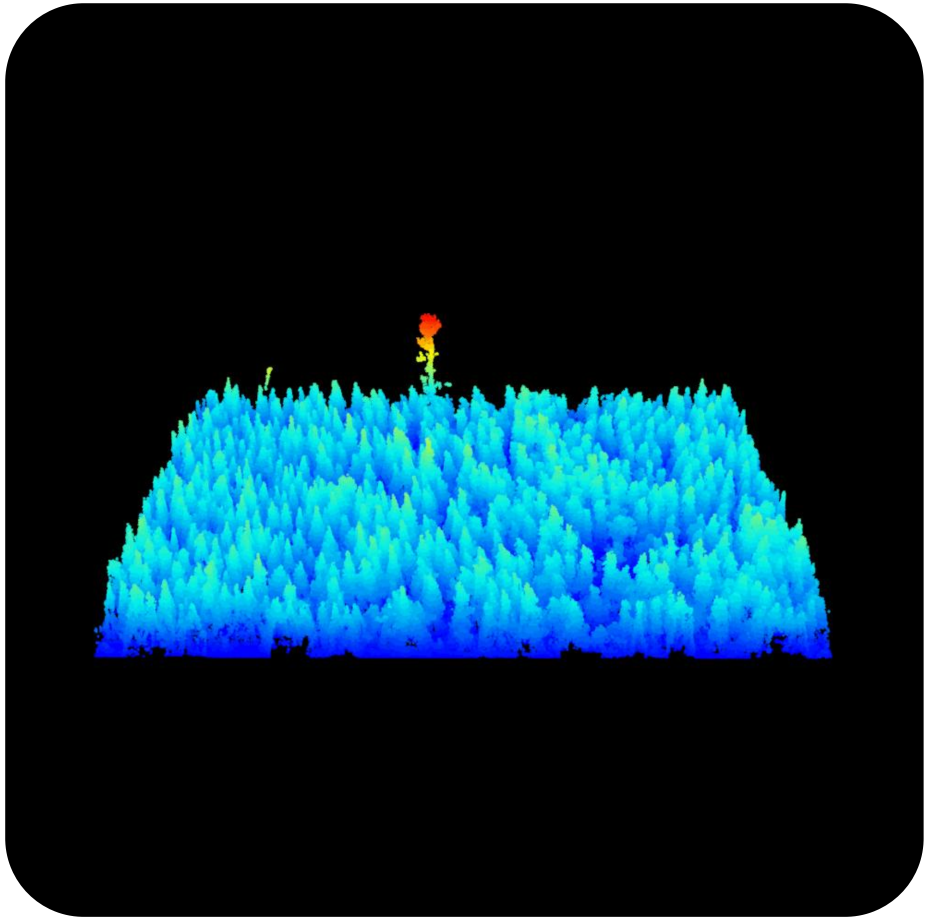
What you need to know

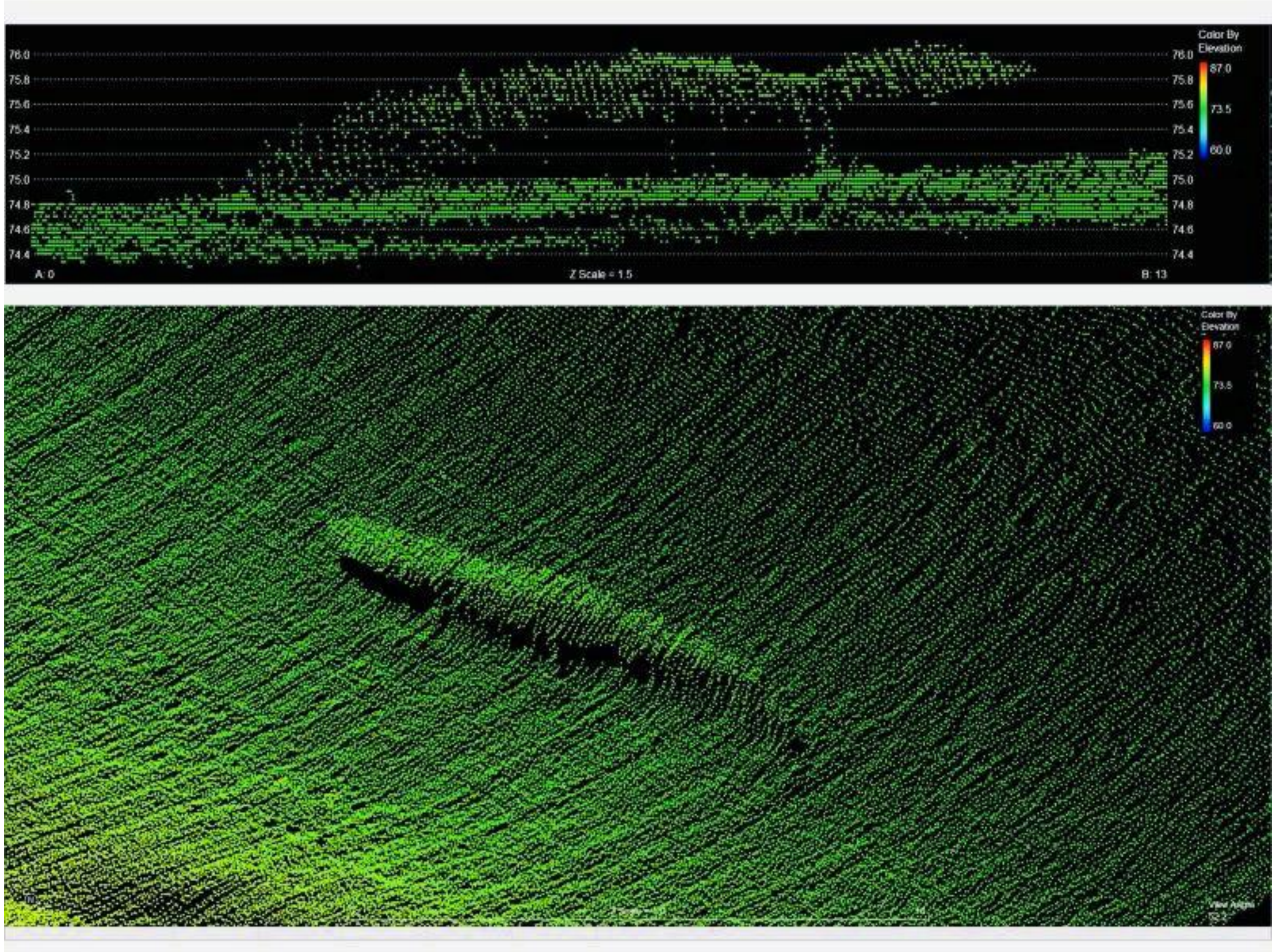
Liam Irwin

FCOR 599

Outline

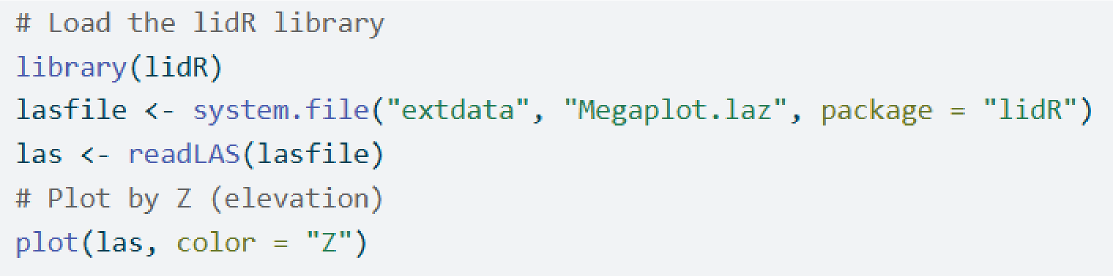
- Making point cloud data beautiful
  - lidR
    - Activity time – make your own GIF! ⏱
  - CloudCompare
  - Potree
    - Activity time – try out Potree! ⏱
- Shiny Apps
  - A brief introduction
  - Where to find help
  - Web-hosting your app
  - Activity time – try out Shiny! ⏱



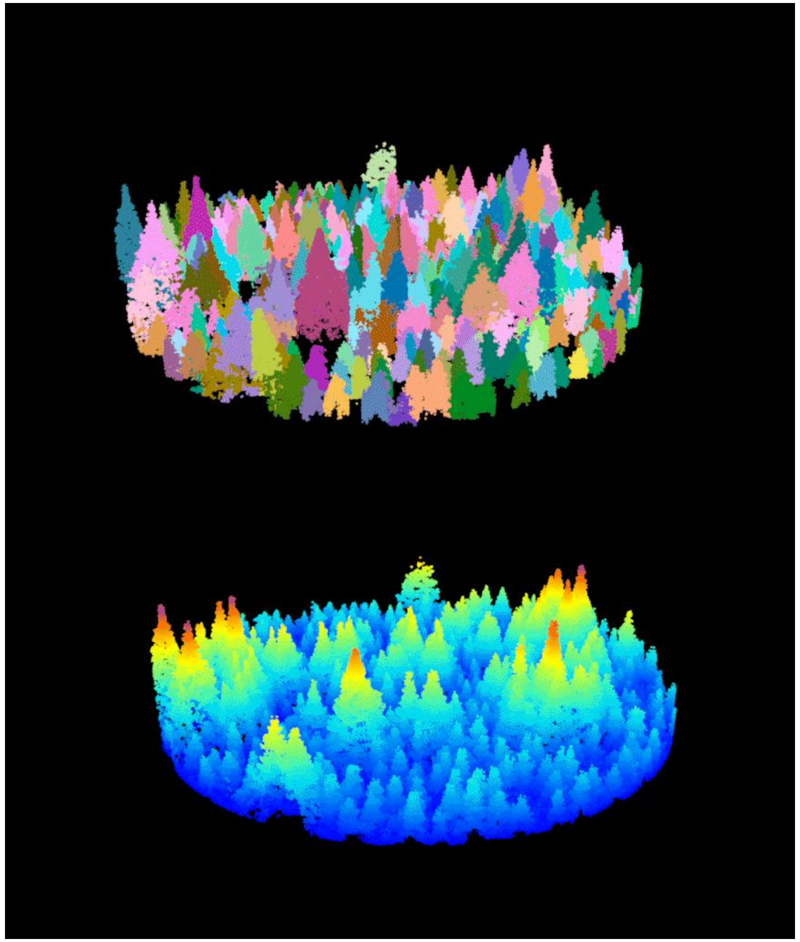

Color By Elevation
87.0
73.5
60.0
76.0
75.8
75.6
75.4
75.2
75.0
74.8
74.6
74.4
A: 0
Z Scale = 1.5
B: 13
Color By Elevation
87.0
73.5
60.0
View Angle

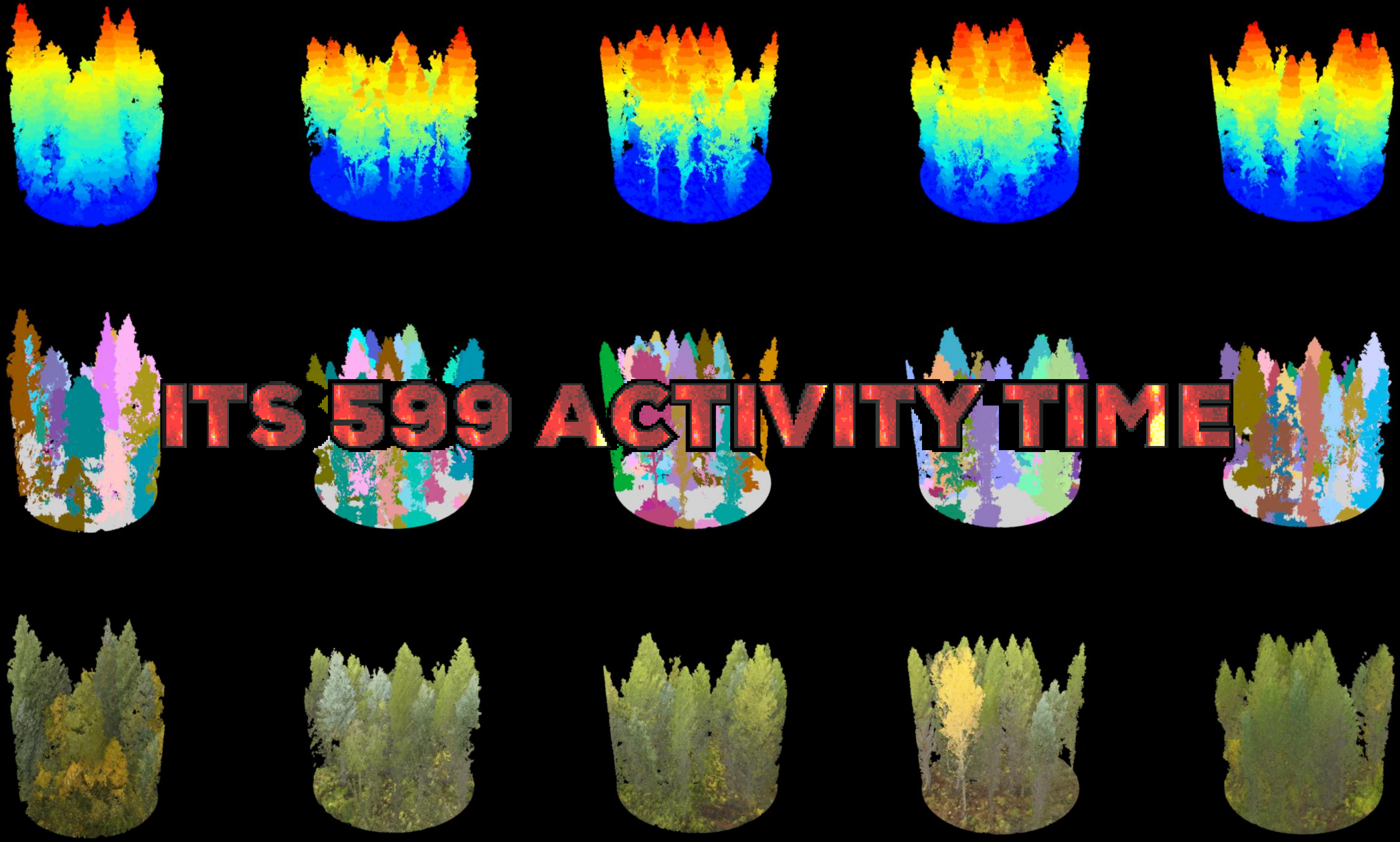
lidR

- Straightforward way to plot
  - Very useful for R workflows
- Normal plot call will plot entire point cloud (not subsampled)
  - Very RAM intensive
  - Should limit to MAX 10m points



```r
# Load the lidR library
library(lidR)
lasfile <- system.file("extdata", "Megaplot.laz", package = "lidR")
las <- readLAS(lasfile)
# Plot by Z (elevation)
plot(las, color = "Z")
```



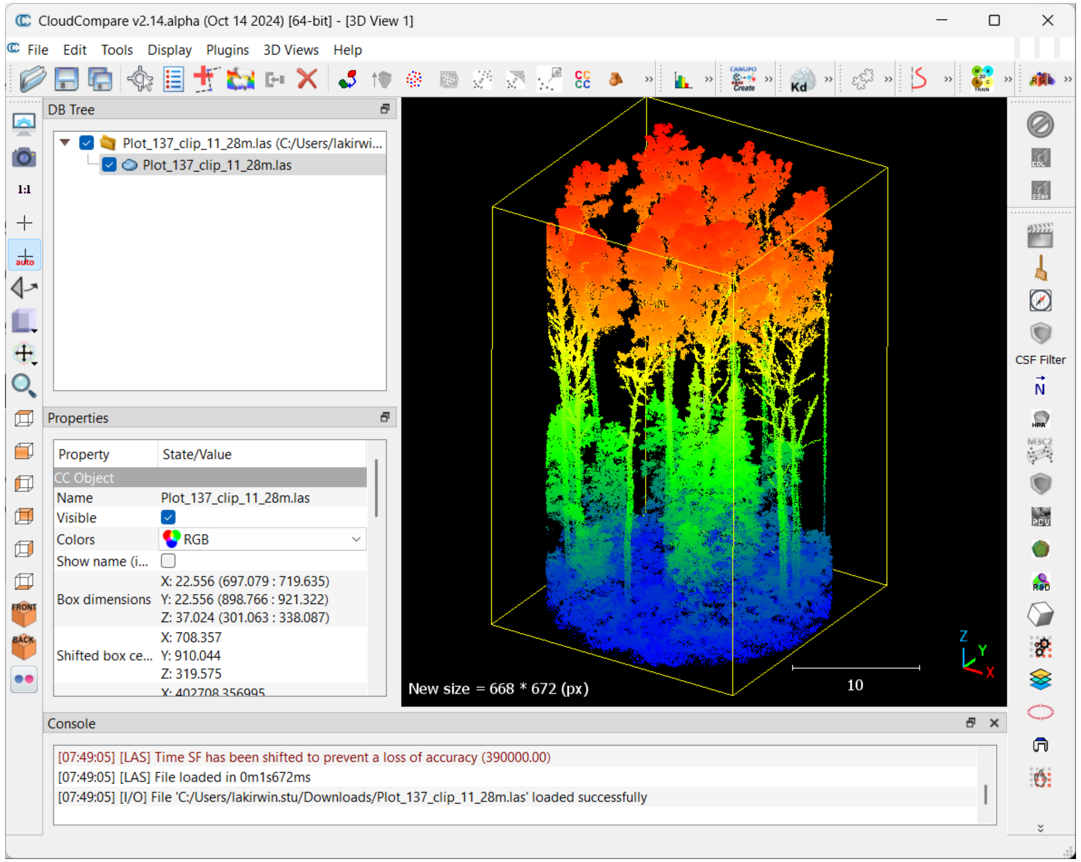


ITS 599 ACTIVITY TIME

CloudCompare

- Really powerful point cloud manipulation software
- Gives user much more power than just visualizing
- Very RAM/compute intensive
- Free and open source
- I use mainly for alignment, data checking, assessing tree segmentation

Pros	Cons
Free and open source Allows manual cleaning/segmentation Many powerful plugins Can call through Python Well documented	Not specifically for lidar Very RAM/CPU intensive Learning curve is steep



CloudCompare v2.14.alpha (Oct 14 2024) [64-bit] - [3D View 1]

Property	State/Value
CC Object	
Name	Plot_137_clip_11_28m.las
Visible	☑
Colors	RGB
Show name (i...	☐
Box dimensions	X: 22.556 (697.079 : 719.635) Y: 22.556 (898.766 : 921.322) Z: 37.024 (301.063 : 338.087)
Shifted box ce...	X: 708.357 Y: 910.044 Z: 319.575
	X: 402708.356995

New size = 668 * 672 (px)

```
[07:49:05] [LAS] Time SF has been shifted to prevent a loss of accuracy (390000.00)
[07:49:05] [LAS] File loaded in 0m1s672ms
[07:49:05] [I/O] File 'C:/Users/lakirwin.stu/Downloads/Plot_137_clip_11_28m.las' loaded successfully
```

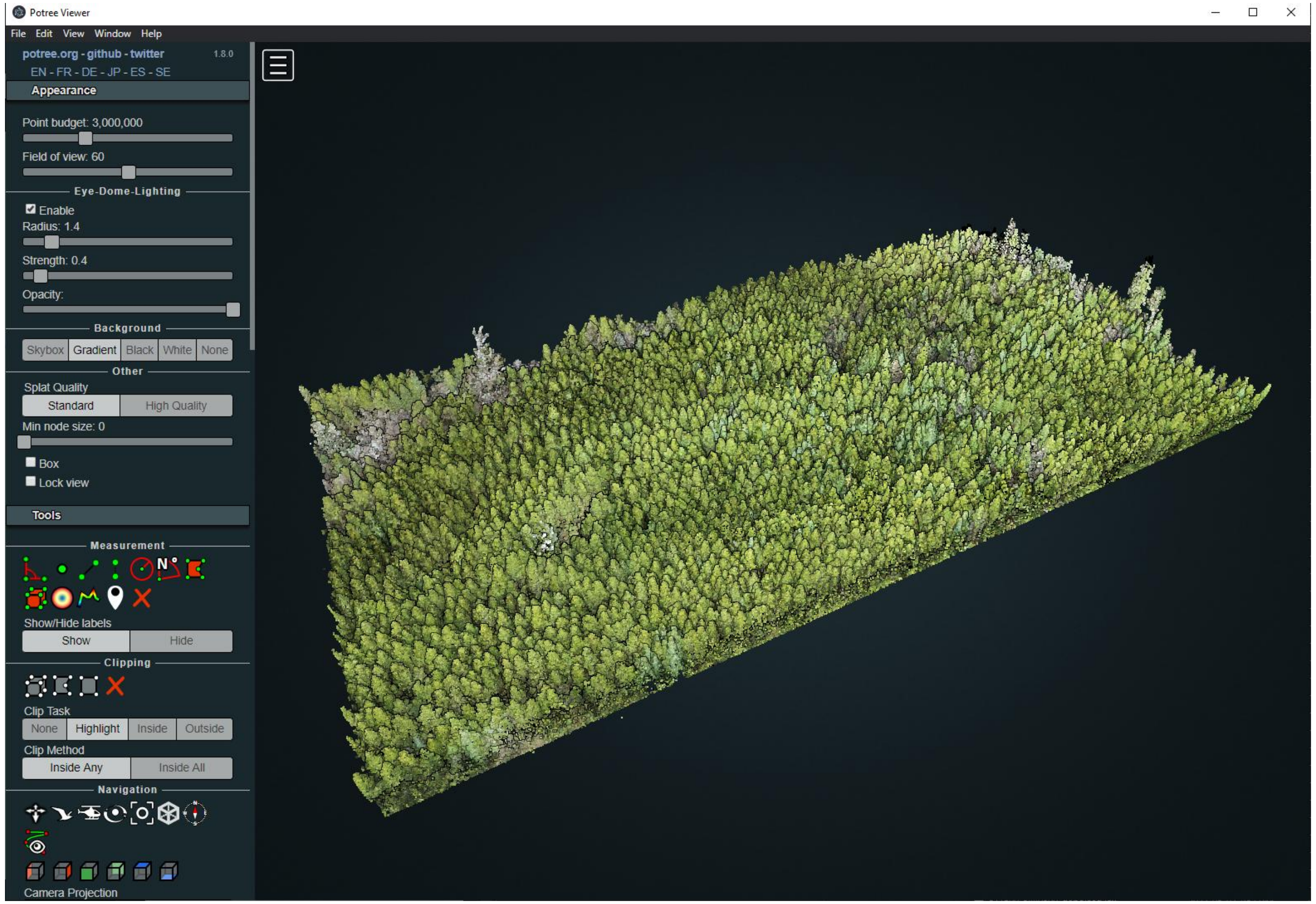
Potree

https://github.com/potree/PotreeDesktop

- Main application is command line/html based

- Very useful for visualization
  - Does the point cloud look "Right"?
  - How does the ground classification look?
  - Getting beautiful screenshots for PowerPoint presentations

- Drag and drop!
  - Will convert .las to Potree format this takes some time, but the converted folder can be dragged in instantly next time



Point budget: 3,000,000

Field of view: 60

☑ Enable

Radius: 1.4

Strength: 0.4

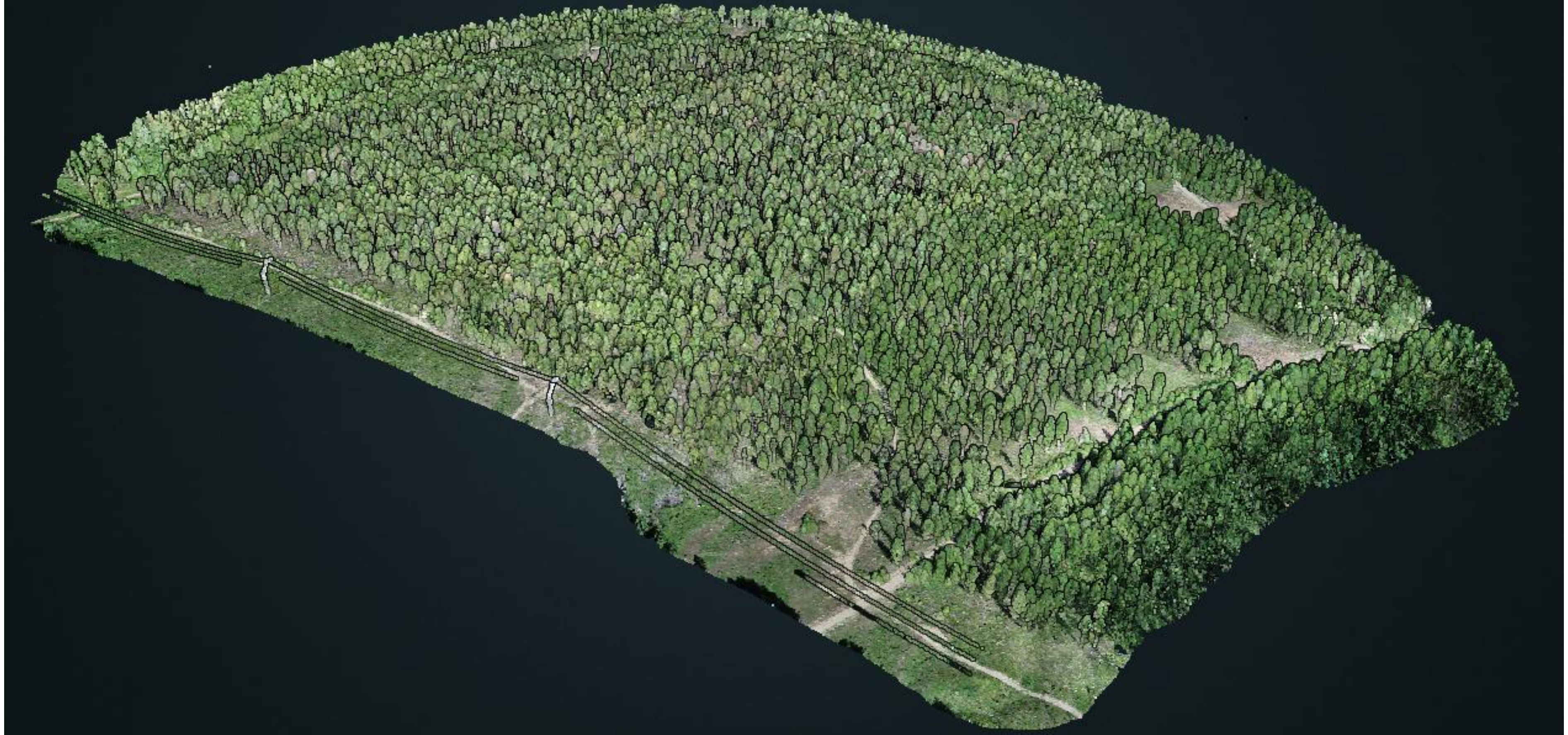
Opacity:

Min node size: 0

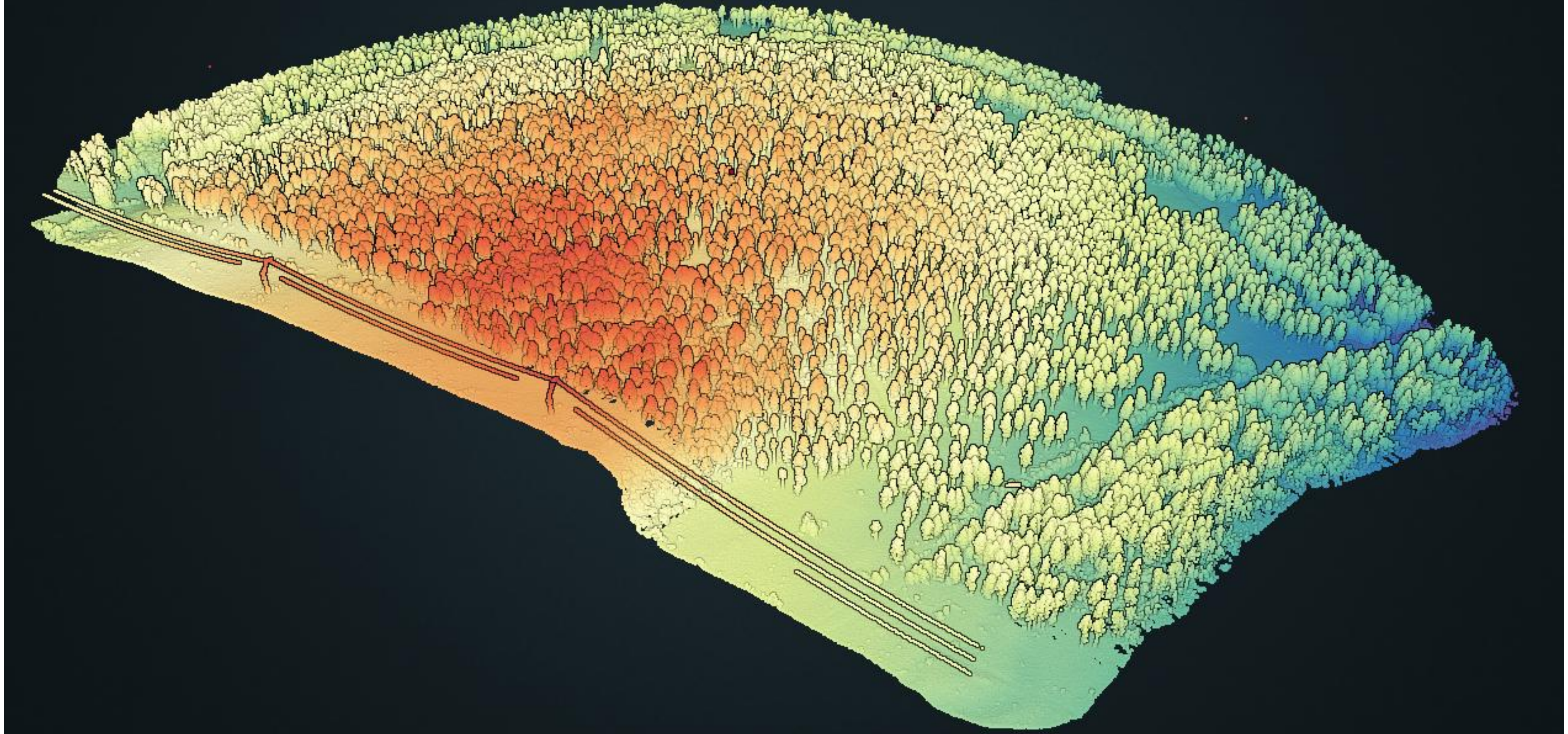
□ Box

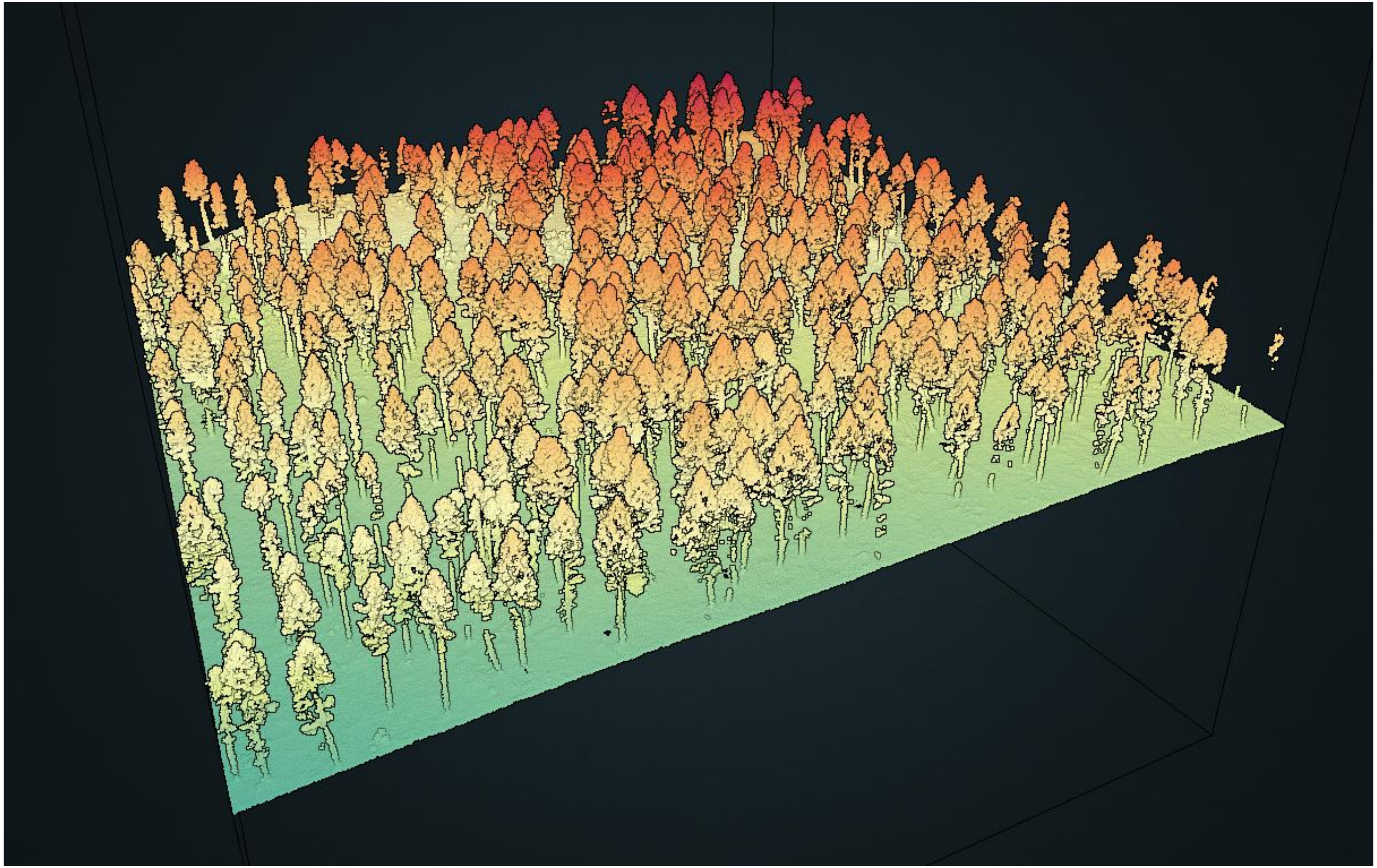
□ Lock view

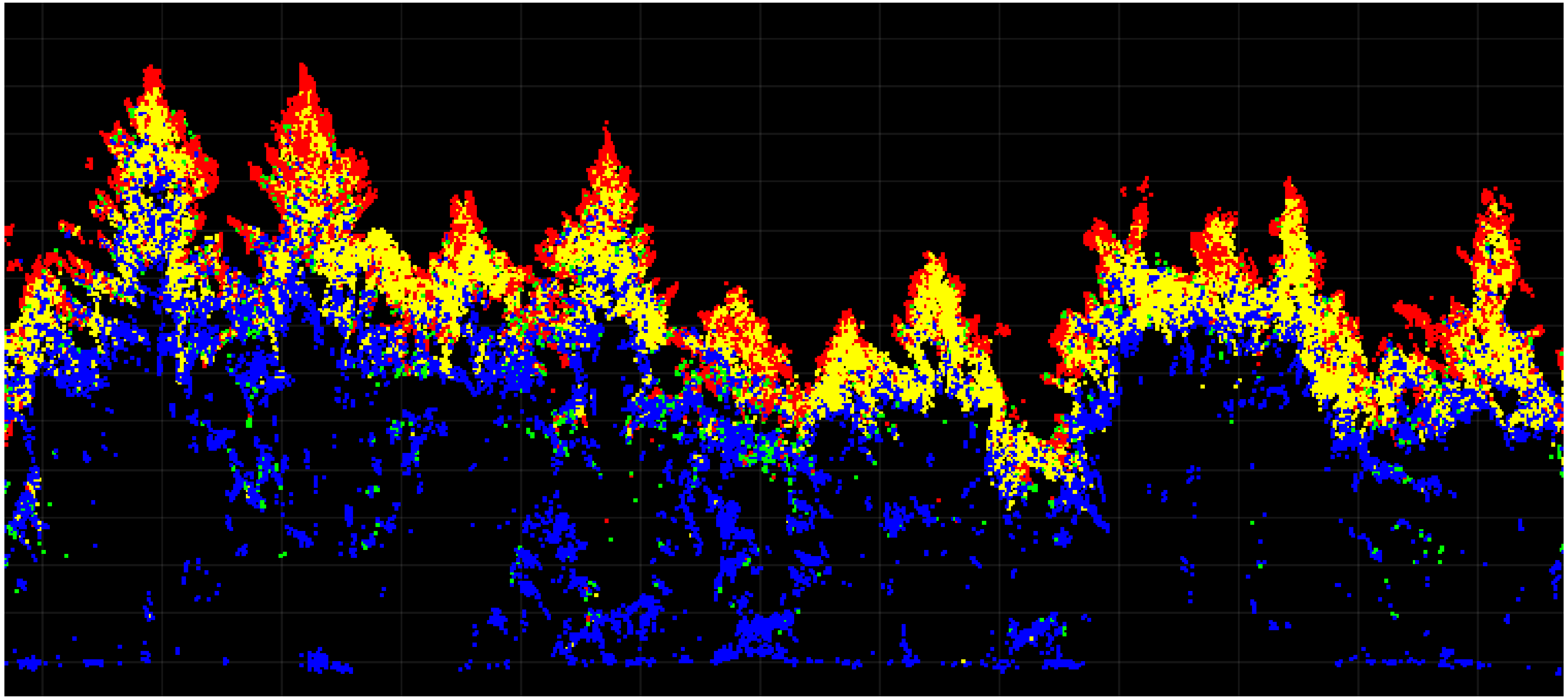
Drone Lidar – RGB Colourized



Drone Lidar – Coloured by Elevation



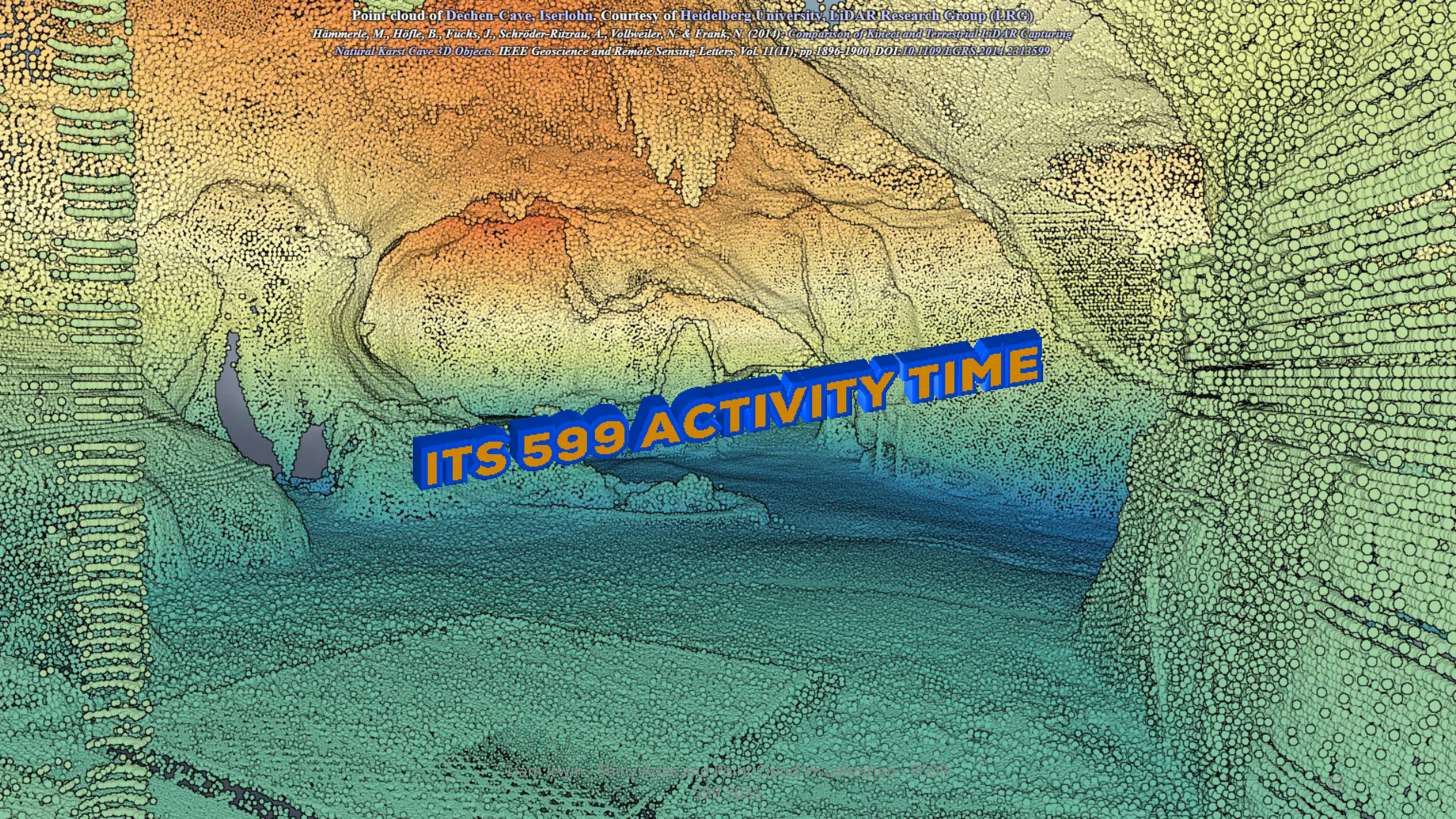




First Returns

Second Returns

Third Returns



ITS 599 ACTIVITY TIME

Point cloud of Dechen-Cave, Iserlohn. Courtesy of Heidelberg University, LiDAR Research Group (LRG)

Hämmerle, M., Höfle, B., Fuchs, J., Schröder-Ritzrau, A., Vollweiler, N. & Frank, N. (2014): Comparison of Kinect and Terrestrial LiDAR Capturing Natural Karst Cave 3D Objects. IEEE Geoscience and Remote Sensing Letters, Vol. 11(11), pp 1896-1900, DOI:10.1109/LGRS.2014.2313599

What is Shiny?

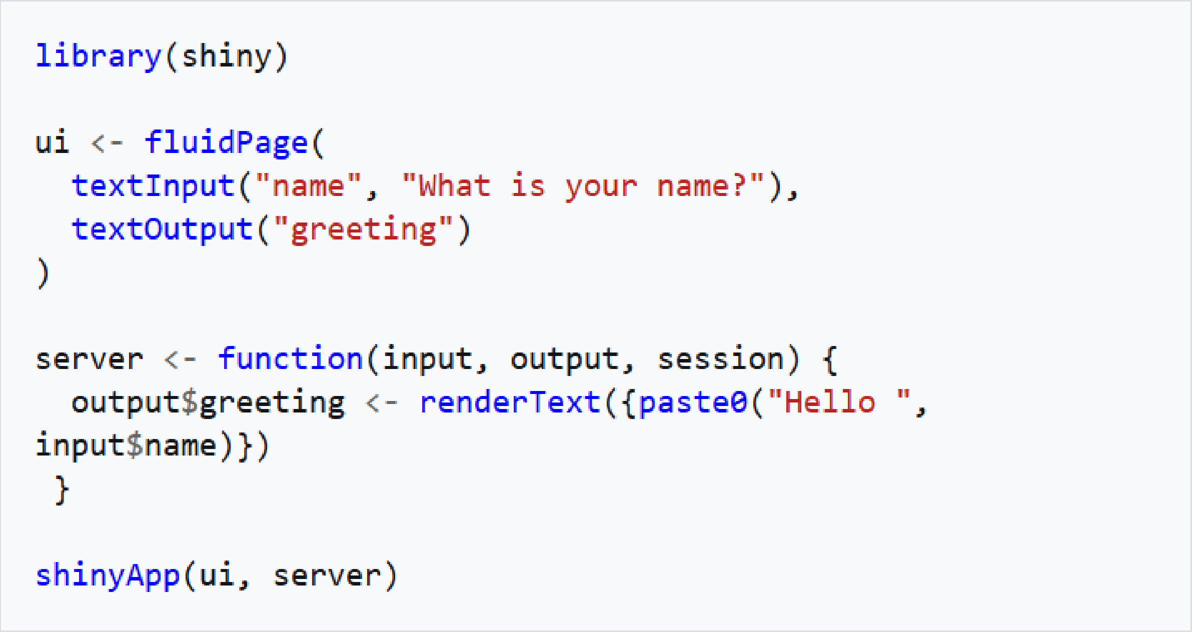
- Workflow to build interactive web applications directly in R (or now Python)
- Reactive programming
    - Shiny apps automatically update when users interact with **inputs**
- Powered by R (or python) in the background
- Goal: “Democratize web app development for data scientists and analysts comfortable in R (or python)”
- Developed by RStudio Inc (now Posit)
- Initial release 2012

Shiny App Components

- Server

- User Interface (UI)

- Both are just functions, when combine they make a Shiny App



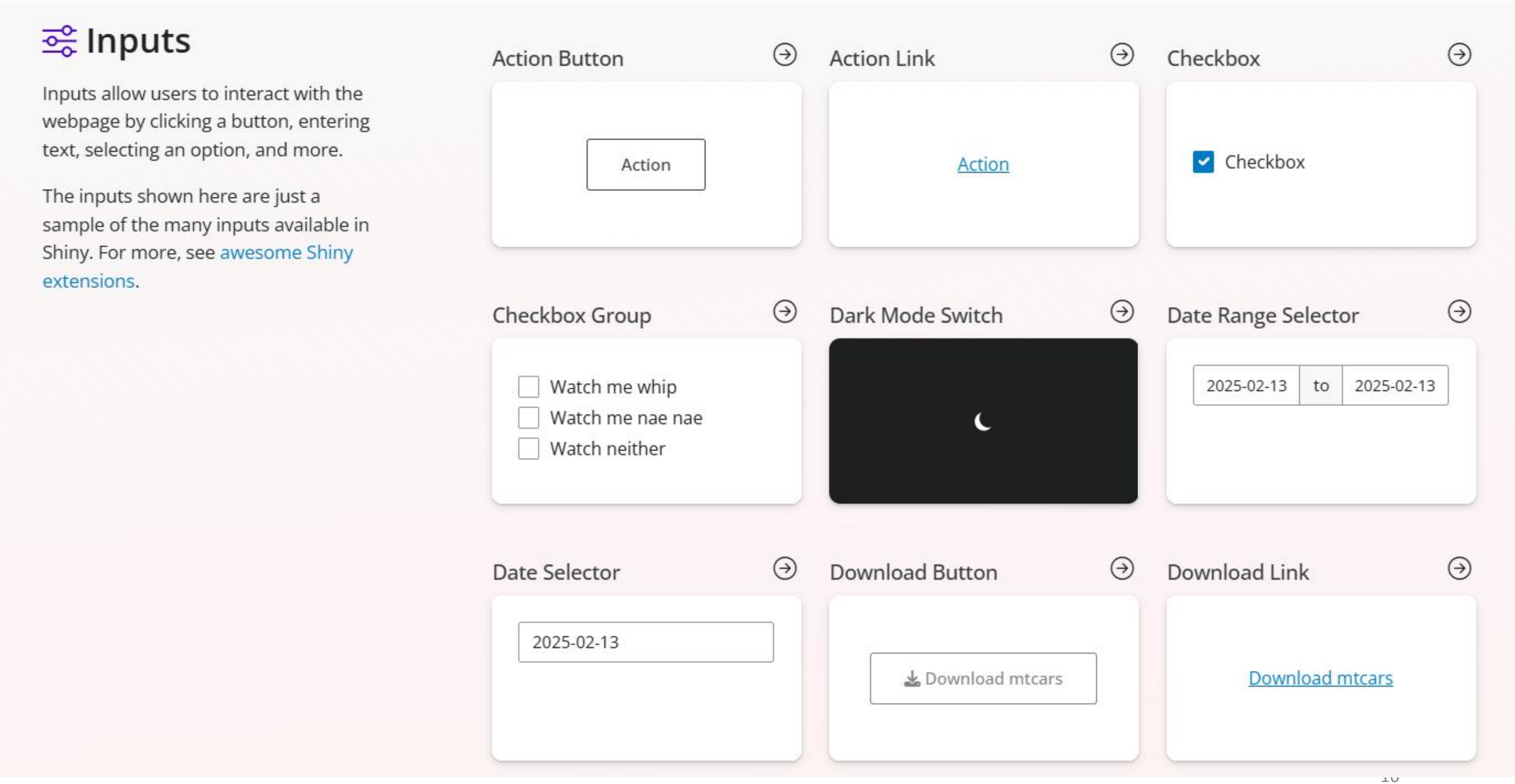
```r
library(shiny)

ui <- fluidPage(
  textInput("name", "What is your name?"),
  textOutput("greeting")
)

server <- function(input, output, session) {
  output$greeting <- renderText({paste0("Hello ",
input$name)})
 }

shinyApp(ui, server)
```

https://shiny.posit.co/r/components/

https://shiny.posit.co/py/components/

Shiny Examples

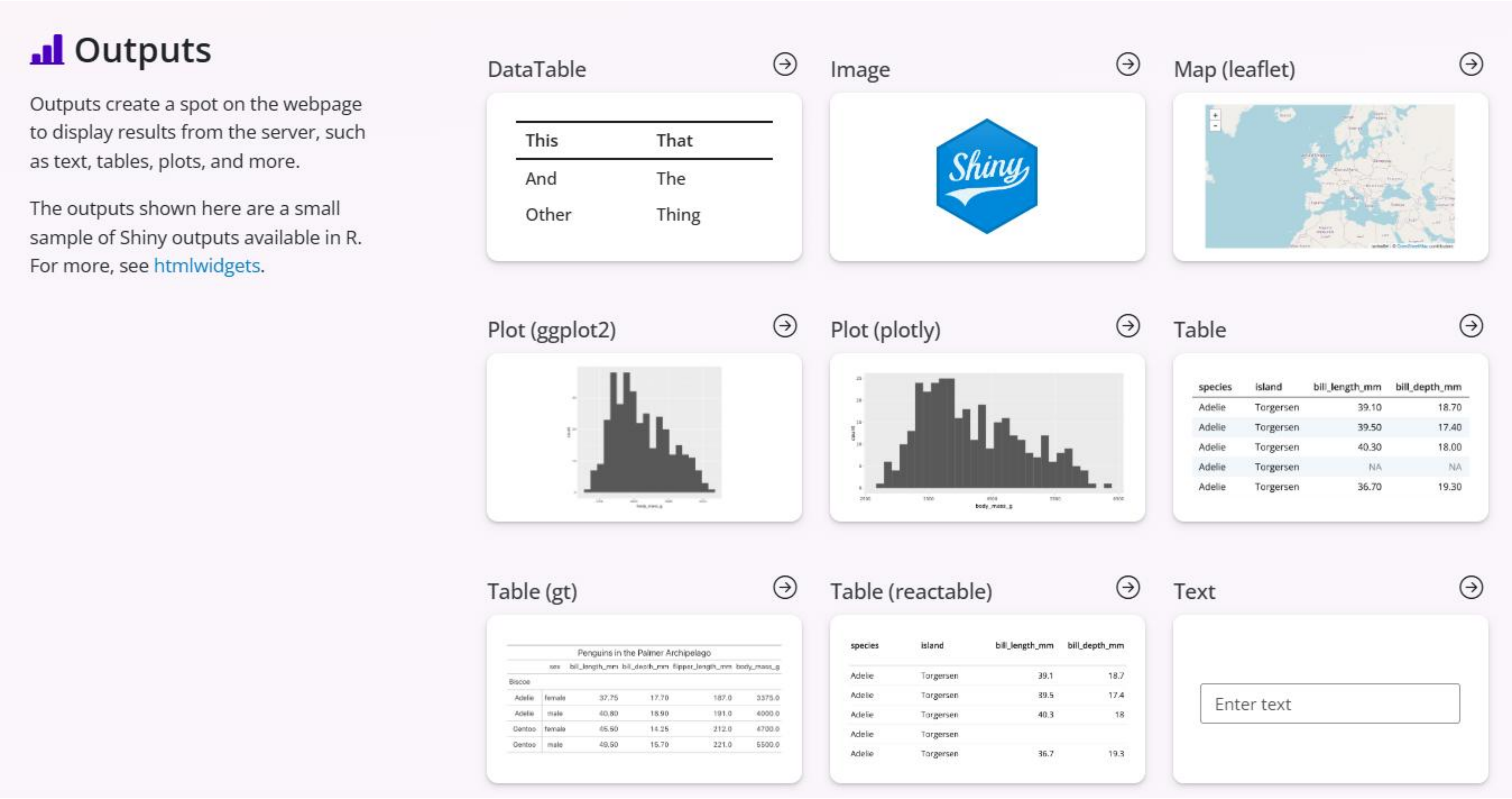
- https://shinylive.io/
- https://shinylive.io/r/examples/
- https://shinylive.io/py/examples/



Inputs

Inputs allow users to interact with the webpage by clicking a button, entering text, selecting an option, and more.

The inputs shown here are just a sample of the many inputs available in Shiny. For more, see awesome Shiny extensions.



Outputs

Outputs create a spot on the webpage to display results from the server, such as text, tables, plots, and more.

The outputs shown here are a small sample of Shiny outputs available in R. For more, see htmlwidgets.

DataTable

This	That
And	The
Other	Thing

Image

Map (leaflet)

Plot (ggplot2)

Plot (plotly)

Table

species	island	bill_length_mm	bill_depth_mm
Adelie	Torgersen	39.10	18.70
Adelie	Torgersen	39.50	17.40
Adelie	Torgersen	40.30	18.00
Adelie	Torgersen	NA	NA
Adelie	Torgersen	36.70	19.30

Table (gt)

Penguins in the Palmer Archipelago

	sex	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
Biscoe					
Adelie	female	37.75	17.70	187.0	3375.0
Adelie	male	40.80	18.90	191.0	4000.0
Gentoo	female	45.50	14.25	212.0	4700.0
Gentoo	male	49.50	15.70	221.0	5500.0

Table (reactable)

species	island	bill_length_mm	bill_depth_mm
Adelie	Torgersen	39.1	18.7
Adelie	Torgersen	39.5	17.4
Adelie	Torgersen	40.3	18
Adelie	Torgersen		
Adelie	Torgersen	36.7	19.3

Text

Shiny for R : : CHEATSHEET

Building an App

A **Shiny** app is a web page (**ui**) connected to a computer running a live R session (**server**).

Users can manipulate the UI, which will cause the server to update the UI's displays (by running R code).

Save your template as **app.R**. Keep your app in a directory along with optional extra files.

- ●●● app-name ← The directory name is the app name
- app.R
- DESCRIPTION ← (optional) used in showcase mode
- README
- R/ ← (optional) directory of supplemental .R files that are sourced automatically, must be named "**R**"
- www/ ← (optional) directory of files to share with web browsers (images, CSS, .js, etc.), must be named "**www**"

Launch apps stored in a directory with **runApp(**<path to directory>**)**.

To generate the template, type **shinyapp** and press **Tab** in the RStudio IDE or go to **File > New Project > New Directory > Shiny Application**

```r
# app.R
library(shiny)

ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)

server <- function(input, output, session) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}

shinyApp(ui = ui, server = server)
```

In **ui** nest R functions to build an HTML interface

Customize the UI with **Layout Functions**

Add Inputs with ***Input() functions**

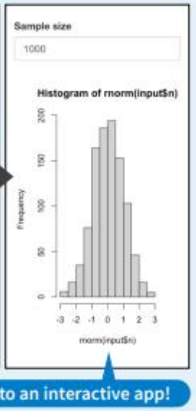
Add Outputs with ***Output() functions**

Tell the **server** how to render outputs and respond to inputs with R

Wrap code in **render*() functions** before saving to output

Refer to UI inputs with **input$<id>** and outputs with **output$<id>**

Call **shinyApp()** to combine **ui** and **server** into an interactive app!



See annotated examples of Shiny apps by running **runExample(**<example name>**)**. Run **runExample()** with no arguments for a list of example names.

Share

Share your app in three ways:

1. **Host it on shinyapps.io**, a cloud based service from Posit. To deploy Shiny apps:
   - Create a free or professional account at **shinyapps.io**
   - Click the Publish icon in RStudio IDE, or run: **rsconnect::deployApp(**"<path to directory>"**)**
2. **Purchase Posit Connect**, a publishing platform for R and Python. **posit.co/products/enterprise/connect/**
3. **Build your own Shiny Server** **posit.co/products/open-source/shinyserver/**

Outputs

render*() and *Output() functions work together to add R output to the UI.

DT::**renderDataTable(**expr, options, searchDelay, callback, escape, env, quoted, outputArgs**)**	**dataTableOutput(**outputId**)**
renderImage(expr, env, quoted, deleteFile, outputArgs**)**	**imageOutput(**outputId, width, height, click, dblclick, hover, brush, inline**)**
renderPlot(expr, width, height, res, …, alt, env, quoted, execOnResize, outputArgs**)**	**plotOutput(**outputId, width, height, click, dblclick, hover, brush, inline**)**
renderPrint(expr, env, quoted, width, outputArgs**)**	**verbatimTextOutput(**outputId, placeholder**)**
renderTable(expr, striped, hover, bordered, spacing, width, align, rownames, colnames, digits, na, …, env, quoted, outputArgs**)**	**tableOutput(**outputId**)**
foo **renderText(**expr, env, quoted, outputArgs, sep**)**	**textOutput(**outputId, container, inline**)**

renderUI(expr, env, quoted, outputArgs**)**	**uiOutput(**outputId, inline, container, …**)**
	htmlOutput(outputId, inline, container, …**)**

These are the core output types. See **htmlwidgets.org** for many more options.

Inputs

Collect values from the user.

Access the current value of an input object with **input$<inputId>**. Input values are **reactive**.

Action — **actionButton(**inputId, label, icon, width, …**)**

Link — **actionLink(**inputId, label, icon, …**)**

☑ Choice 1 ☑ Choice 2 □ Choice 3 — **checkboxGroupInput(**inputId, label, choices, selected, inline, width, choiceNames, choiceValues**)**

☑ Check me — **checkboxInput(**inputId, label, value, width**)**

dateInput(inputId, label, value, min, max, format, startview, weekstart, language, width, autoclose, datesdisabled, daysofweekdisabled**)**

dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator, width, autoclose**)**

Choose File — **fileInput(**inputId, label, multiple, accept, width, buttonLabel, placeholder**)**

1 — **numericInput(**inputId, label, value, min, max, step, width**)**

•••••••• — **passwordInput(**inputId, label, value, width, placeholder**)**

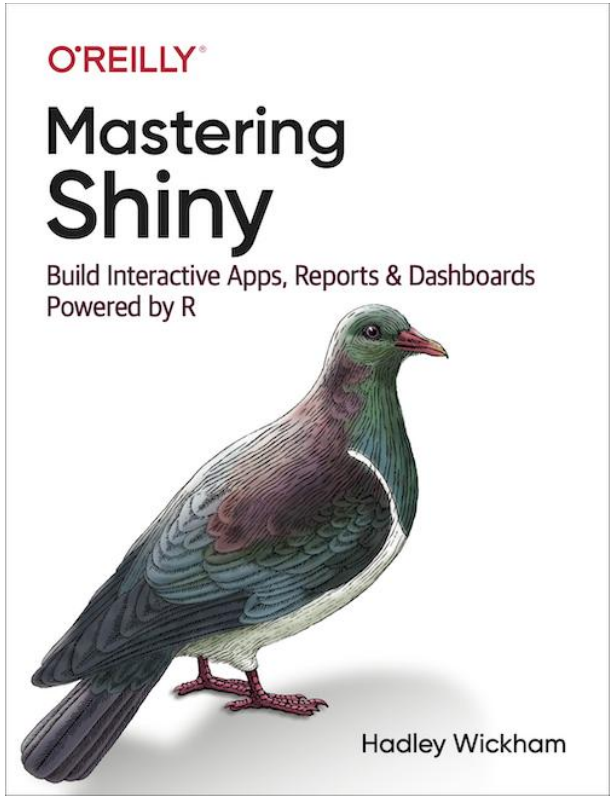
● Choice A ○ Choice B ○ Choice C — **radioButtons(**inputId, label, choices, selected, inline, width, choiceNames, choiceValues**)**

Choice 1 / Choice 1 / Choice 2 — **selectInput(**inputId, label, choices, selected, multiple, selectize, width, size**)** Also **selectizeInput()**

sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post, timeFormat, timezone, dragRange**)**

Enter text — **textInput(**inputId, label, value, width, placeholder**)** Also **textAreaInput()**

posit



https://mastering-shiny.org/

Hosting your Shiny App

https://liam-irwin.shinyapps.io/itd_webapp/

Shinyapps.io offers hosting by Posit

- Free hosting up to 25 hours of usage
- Very easy, if your ShinyApp runs locally you can get it uploaded in a few lines of code.

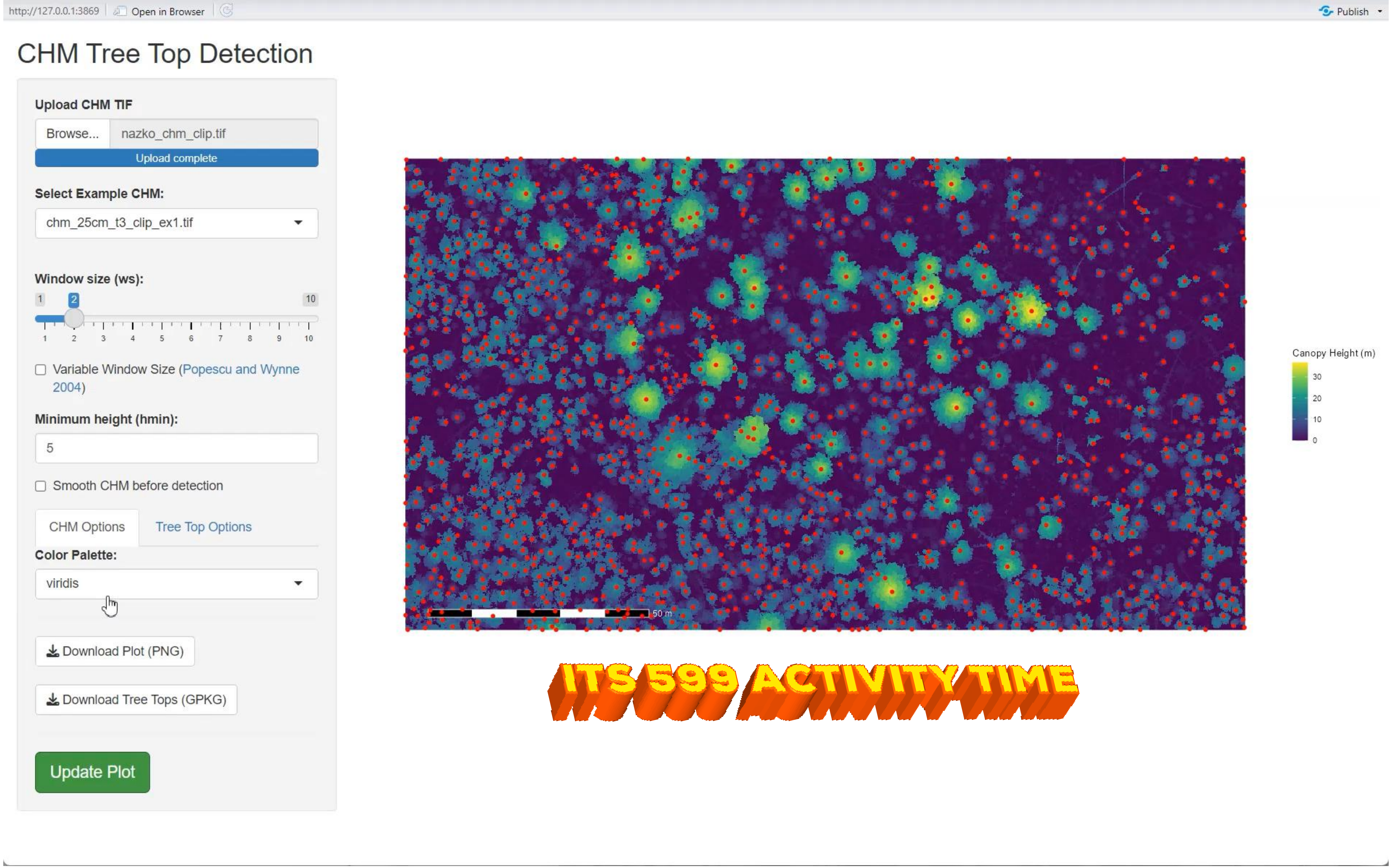
Cost/Month	Active Hours/ month
Free	25
$13	100
$49	500
$119	2000
$349	10,000

Your turn!

- Download the ShinyApp code, try to run it locally
  - Export three customized tree top visualizations using the Download Plot (PNG) button

- Make your own 3D rotating GIF of a point cloud!

- Upload these to Canvas for participation credit for this workshop



CHM Tree Top Detection

Upload CHM TIF

Browse... nazko_chm_clip.tif

Upload complete

Select Example CHM: chm_25cm_t3_clip_ex1.tif

Window size (ws): 2

☐ Variable Window Size (Popescu and Wynne 2004)

Minimum height (hmin): 5

☐ Smooth CHM before detection

Color Palette: viridis

ITS 599 ACTIVITY TIME