I put in a lot of efforts into bringing practical challenges, examples from real life, and a lot of demonstrations of statistical theory with R. This post is an example for how I’ve been using R code (and specifically Shiny apps) to demonstrate statistical theory, concepts and provide intuition.

**What’s the difference between confidence and prediction intervals?**

Last week I taught multiple linear regression, and I noticed that students have a hard time comprehending the difference between confidence intervals and prediction intervals. The former being an interval for the model (i.e., interval for the underlying model), and the latter being an interval for a noval observation.

As the sample size increases, our uncertainty of the model’s parameters decreases, but the uncertainty in the value of a new observation, \(y\_0\) is associated with variance of \(Y\) (the random variable from which \(y\_0\) is drawn). Hence, it has a lower bound, based on that variance.

In R, you can get a prediction or a confidence interval by using either

predict(object, newdata, interval = "prediction")

Or

predict(object, newdata, interval = "confidence")

For a prediction or for a confidence interval, respectively.

To help me illustrate the differences between the two, I decided to build a small Shiny web app. It shows the differences between confidence intervals, prediction intervals, the regression fit, and the actual (original) model.

With this app you can choose three types of models to demonstrate. Simple linear regression, and regression with a twist (\(\log\) transformation on the \(y\) or \(\sin\) transformation on the \(x\):

* Linear model \(y = a + bx + \epsilon\)
* Log-linear model \(\log(y)=a+bx+\epsilon\)
* Sine \(y = a + b\sin(x) + \epsilon\)

All the models are based on simple linear regression (lm function), for the latter two models with either a log or sin transformation.

The app allows you to play around with various values such as the \(x\) range, the model’s parameters (\(a\) and \(b\)), the error’s standard deviation (\(\epsilon\)), and show or hide any of the following elements, on the chart:

* The original function (i.e., the original model)
* The sampled points
* The confidence interval
* The prediction interval
* The model’s fit

Here’s an example for what the app’s generating code and output looks like, for a model of the type \(\log(y) = 1 + \frac{x}{2} + \epsilon\):

library(dplyr)

library(tidyr)

library(tibble)

library(ggplot2)

sample\_size <- 90

x\_range <- c(0, 1.5)

a <- 1

b <- 1.5

sigma <- 0.3

actual\_function <- tibble(x = seq(x\_range[1], x\_range[2], by = 0.01)) %>%

mutate(actual\_y = exp(a + b\*x))

random\_sample <- tibble(epsilon\_err = rnorm(n = sample\_size,

mean = 0,

sd = sigma),

x = runif(n = sample\_size,

min = x\_range[1],

max = x\_range[2])) %>%

mutate(sampled\_y = exp(a + b\*x + epsilon\_err))

linear\_model <- lm(formula = log(sampled\_y) ~ x, data = random\_sample)

prediction\_i <- predict(object = linear\_model,

newdata = actual\_function,

interval = "prediction") %>%

as\_tibble() %>%

rename\_at(vars(lwr,upr), ~paste0(., "\_pi")) %>%

mutate\_all(exp)

confidence\_i <- predict(object = linear\_model,

newdata = actual\_function,

interval = "confidence") %>%

as\_tibble() %>%

rename\_at(vars(lwr,upr), ~paste0(., "\_ci")) %>%

select(-fit) %>%

mutate\_all(exp)

intervals <- actual\_function %>%

bind\_cols(prediction\_i,

confidence\_i)

ggplot() +

geom\_line(data = actual\_function, aes(x, actual\_y, color = "Original Model"), size = 1) +

geom\_point(data = random\_sample, aes(x, sampled\_y), alpha = 0.5) +

geom\_line(data = intervals,

aes(x, fit, color = "Regression Fit"), size = 1) +

geom\_line(data = intervals,

aes(x, lwr\_pi, color = "Prediction Interval"),

linetype = 2, size = 1) +

geom\_line(data = intervals,

aes(x, upr\_pi, color = "Prediction Interval"),

linetype = 2, size = 1) +

geom\_line(data = intervals,

aes(x, lwr\_ci, color = "Confidence Interval"),

linetype = 2, size = 1) +

geom\_line(data = intervals,

aes(x, upr\_ci, color = "Confidence Interval"),

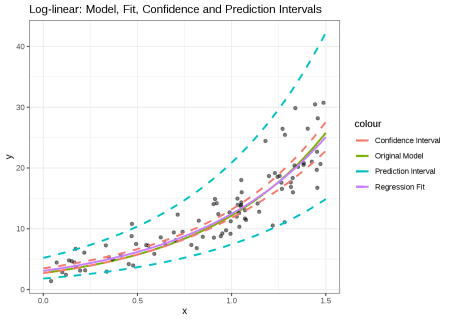
linetype = 2, size = 1) +

theme\_bw() +

xlab("x") +

ylab("y") +

ggtitle("Log-linear: Model, Fit, Confidence and Prediction Intervals")



**Conclusions**

Shiny apps are a great way to illustrate theoretical concepts, to provide intuition, and to let students experiment with parameters and see the outcomes. In this post I demonstrated how a Shiny app can be used to explain the concepts of a regression fit, confidence, and prediction intervals.

Source Code

App\_server.R

|  |
| --- |
| #' @import shiny |
|  | app\_server <- function(input, output,session) { |
|  | # List the first level callModules here |
|  | callModule(mod\_create\_intervals\_server, "create\_intervals\_ui\_1") |
|  | } |

App\_ui.R

|  |
| --- |
| #' @import shiny |
|  | app\_ui <- function() { |
|  | tagList( |
|  | # Leave this function for adding external resources |
|  | # golem\_add\_external\_resources(), |
|  | # List the first level UI elements here |
|  | fluidPage(theme = shinythemes::shinytheme("united"), title = "Prediction and Confidence Intervals", |
|  | fluidRow(mod\_create\_intervals\_ui("create\_intervals\_ui\_1")) |
|  | ) |
|  | ) |
|  | } |
|  |  |
|  | #' #' @import shiny |
|  | #' golem\_add\_external\_resources <- function(){ |
|  | #' |
|  | #' # addResourcePath( |
|  | #' # 'www', system.file('app/www', package = 'IntervalsDemo') |
|  | #' # ) |
|  | #' |
|  | #' tags$head( |
|  | #' golem::activate\_js(), |
|  | #' golem::favicon() |
|  | #' # Add here all the external resources |
|  | #' # If you have a custom.css in the inst/app/www |
|  | #' # Or for example, you can add shinyalert::useShinyalert() here |
|  | #' #tags$link(rel="stylesheet", type="text/css", href="www/custom.css") |
|  | #' ) |
|  | #' } |

Golem\_utils\_server.R

|  |
| --- |
| # Inverted versions of in, is.null and is.na |
|  | `%not\_in%` <- Negate(`%in%`) |
|  |  |
|  | not\_null <- Negate(is.null) |
|  |  |
|  | not\_na <- Negate(is.na) |
|  |  |
|  | # Removes the null from a vector |
|  | drop\_nulls <- function(x){ |
|  | x[!sapply(x, is.null)] |
|  | } |
|  |  |
|  | # If x is null, return y, otherwise return x |
|  | "%||%" <- function(x, y){ |
|  | if (is.null(x)) { |
|  | y |
|  | } else { |
|  | x |
|  | } |
|  | } |
|  | # If x is NA, return y, otherwise return x |
|  | "%|NA|%" <- function(x, y){ |
|  | if (is.na(x)) { |
|  | y |
|  | } else { |
|  | x |
|  | } |
|  | } |
|  |  |
|  | # typing reactiveValues is too long |
|  | rv <- shiny::reactiveValues |
|  | rvtl <- shiny::reactiveValuesToList |

Golem\_utils\_ui.R

|  |
| --- |
| # Turn an R list into an HTML list |
|  | # |
|  | # @param list An R list |
|  | # @param class a class for the list |
|  | # @return an HTML list |
|  | # @examples |
|  | # list\_to\_li(c("a","b")) |
|  | # |
|  | #' @importFrom htmltools tags tagAppendAttributes tagList |
|  | list\_to\_li <- function(list, class = NULL){ |
|  | if (is.null(class)){ |
|  | tagList(lapply(list, tags$li)) |
|  | } else { |
|  | res <- lapply(list, tags$li) |
|  | res <- lapply(res, function(x) tagAppendAttributes(x, class = class)) |
|  | tagList(res) |
|  | } |
|  |  |
|  | } |
|  |  |
|  | #' @importFrom htmltools tags tagAppendAttributes tagList |
|  | list\_to\_p <- function(list, class = NULL){ |
|  | if (is.null(class)){ |
|  | tagList(lapply(list, tags$p)) |
|  | } else { |
|  | res <- lapply(list, tags$p) |
|  | res <- lapply(res, function(x) tagAppendAttributes(x, class = class)) |
|  | tagList(res) |
|  | } |
|  |  |
|  | } |
|  |  |
|  | #' @importFrom glue glue |
|  | #' @importFrom htmltools tags tagAppendAttributes tagList |
|  | named\_to\_li <- function(list, class = NULL){ |
|  | if(is.null(class)){ |
|  | res <- mapply( |
|  | function(x, y){ |
|  | tags$li(HTML(glue("<b>{y}:</b> {x}"))) |
|  | }, |
|  | list, names(list), SIMPLIFY = FALSE) |
|  | #res <- lapply(res, HTML) |
|  | tagList(res) |
|  | } else { |
|  | res <- mapply( |
|  | function(x, y){ |
|  | tags$li(HTML(glue("<b>{y}:</b> {x}"))) |
|  | }, |
|  | list, names(list), SIMPLIFY = FALSE) |
|  | res <- lapply(res, function(x) tagAppendAttributes(x, class = class)) |
|  | tagList(res) |
|  | } |
|  | } |
|  |  |
|  | # Remove a tag attribute |
|  | # |
|  | # @param tag the tag |
|  | # @param ... the attributes to remove |
|  | # |
|  | # @return a new tag |
|  | # @export |
|  | # |
|  | # @examples |
|  | # a <- shiny::tags$p(src = "plop", "pouet") |
|  | # tagRemoveAttributes(a, "src") |
|  | tagRemoveAttributes <- function(tag, ...) { |
|  | attrs <- as.character(list(...)) |
|  | for (i in seq\_along(attrs)) { |
|  | tag$attribs[[ attrs[i] ]] <- NULL |
|  | } |
|  | tag |
|  | } |
|  |  |
|  | # Hide or display a tag |
|  | # @param tag the tag |
|  | # @return a tag |
|  | # @examples |
|  | # ## Hide |
|  | # a <- shiny::tags$p(src = "plop", "pouet") |
|  | # undisplay(a) |
|  | # b <- shiny::actionButton("go\_filter", "go") |
|  | # undisplay(b) |
|  |  |
|  | #' @importFrom htmltools tagList |
|  | undisplay <- function(tag) { |
|  | # if not already hidden |
|  | if (!is.null(tag$attribs$style) && !grepl("display:\\s+none", tag$attribs$style)) { |
|  | tag$attribs$style <- paste("display: none;", tag$attribs$style) |
|  | } else { |
|  | tag$attribs$style <- "display: none;" |
|  | } |
|  | tag |
|  | } |
|  |  |
|  | #' @importFrom htmltools tagList |
|  | display <- function(tag) { |
|  | if (!is.null(tag$attribs$style) && grepl("display:\\s+none", tag$attribs$style)) { |
|  | tag$attribs$style <- gsub("(\\s)\*display:(\\s)\*none(\\s)\*(;)\*(\\s)\*", "", tag$attribs$style) |
|  | } |
|  | tag |
|  | } |
|  |  |
|  | # Hide an elements by calling jquery hide on it |
|  | #' @importFrom htmltools tags |
|  | jq\_hide <- function(id) { |
|  | tags$script(sprintf("$('#%s').hide()", id)) |
|  | } |
|  |  |
|  | # Add a red star at the end of the text |
|  | # |
|  | # Adds a red star at the end of the text |
|  | # (for example for indicating mandatory fields). |
|  | # |
|  | # @param text the HTLM text to put before the red star |
|  | # |
|  | # @return an html element |
|  | # |
|  | # @examples |
|  | # with\_red\_star("Enter your name here") |
|  | # |
|  | #' @importFrom htmltools tags HTML |
|  | with\_red\_star <- function(text) { |
|  | htmltools::tags$span( |
|  | HTML( |
|  | paste0( |
|  | text, |
|  | htmltools::tags$span( |
|  | style = "color:red", "\*" |
|  | ) |
|  | ) |
|  | ) |
|  | ) |
|  | } |
|  |  |
|  |  |
|  |  |
|  | # Repeat tags$br |
|  | # |
|  | # @param times the number of br to return |
|  | # |
|  | # @return the number of br specified in times |
|  | # @export |
|  | # |
|  | # @examples |
|  | # rep\_br(5) |
|  | # |
|  | #' @importFrom htmltools HTML |
|  | rep\_br <- function(times = 1) { |
|  | HTML(rep("<br/>", times = times)) |
|  | } |
|  |  |
|  | # Create an url |
|  | # |
|  | # @param url the URL |
|  | # @param text the text to display |
|  | # |
|  | # @return an a tag |
|  | # @export |
|  | # |
|  | # @examples |
|  | # enurl("https://www.thinkr.fr", "ThinkR") |
|  | enurl <- function(url, text){ |
|  | tags$a(href = url, text) |
|  | } |
|  |  |
|  |  |
|  | # Columns 12, 6 and 4 |
|  | # |
|  | # Most shiny columns are 12, 6 or 4 of width. |
|  | # These are convenient wrappers around |
|  | # `column(12, ...)`, `column(6, ...)` and `column(4, ...)`. |
|  | # |
|  | # @export |
|  | # @rdname columns |
|  | #' @importFrom shiny column |
|  | col\_12 <- function(...){ |
|  | column(12, ...) |
|  | } |
|  |  |
|  | #' @importFrom shiny column |
|  | col\_6 <- function(...){ |
|  | column(6, ...) |
|  | } |
|  |  |
|  | #' @importFrom shiny column |
|  | col\_4 <- function(...){ |
|  | column(4, ...) |
|  | } |

Mod\_Create\_Intervals.R

|  |
| --- |
| # Module UI |
|  |  |
|  | #' @title mod\_create\_intervals\_ui and mod\_create\_intervals\_server |
|  | #' @description A shiny Module. |
|  | #' |
|  | #' @param id shiny id |
|  | #' @param input internal |
|  | #' @param output internal |
|  | #' @param session internal |
|  | #' |
|  | #' @rdname mod\_create\_intervals |
|  | #' |
|  | #' @keywords internal |
|  | #' @export |
|  | #' @importFrom shiny NS tagList |
|  | mod\_create\_intervals\_ui <- function(id){ |
|  | ns <- NS(id) |
|  | sidebarLayout( |
|  | sidebarPanel(width = 3, |
|  | fluidRow( |
|  | selectInput(ns("functional\_relationship"), |
|  | label = "Choose type of relationship", |
|  | choices = c("Linear: y = a + bx + eps" = "linear", |
|  | "Log-linear: log(y) = a + bx + eps" = "log\_linear", |
|  | "Sine: y = a + b\*sin(x) + eps" = "sine"))), |
|  | fluidRow( |
|  | numericInput(ns("select\_a"), |
|  | "Value of a:", |
|  | 1), |
|  | numericInput(ns("select\_b"), |
|  | "Value of b:", |
|  | 0.5) |
|  | ), |
|  | fluidRow( |
|  | sliderInput(ns("x\_range"), |
|  | label = "Range of x", |
|  | min = 0, max = 10, value = c(0, 1), step = 0.1, dragRange = T) |
|  | ), |
|  | fluidRow( |
|  | sliderInput(ns("sample\_size"), |
|  | label = "Select Sample Size", |
|  | min = 10, max = 1000, value = 250, step = 10) |
|  | ), |
|  | fluidRow( |
|  | sliderInput(ns("epsilon\_error"), |
|  | label = "Define error sigma", |
|  | min = 0, max = 10, value = 0.3, step = 0.1) |
|  | ), |
|  | fluidRow( |
|  | h3("Include on chart:"), |
|  | checkboxInput(ns("include\_real\_function"), |
|  | label = "Original Function", value = T), |
|  | checkboxInput(ns("include\_geom\_point"), |
|  | label = "Points", value = F), |
|  | checkboxInput(ns("include\_ci"), |
|  | label = "Confidence interval", value = T), |
|  | checkboxInput(ns("include\_pi"), |
|  | label = "Prediction Interval", value = T), |
|  | checkboxInput(ns("include\_fit"), |
|  | label = "Model's fit", value = T) |
|  | ) |
|  | ), |
|  | mainPanel( |
|  | fluidRow(h3("Linear regression - model fit and intervals"), |
|  | p("This app was generated by Adi Sarid, as a tool to demonstrate the differences between prediction and confidence intervals. |
|  | Change the controls on the right to see how the predictions change."), |
|  | p("The source code for this app is available in the a github repository:", |
|  | a("https://github.com/adisarid/prediction\_confidence\_intervals\_demo", |
|  | href = "https://github.com/adisarid/prediction\_confidence\_intervals\_demo", |
|  | target = "\_blank")), |
|  | p("The use of this tool is permitted via the cc-by-sa, with attribution to ", |
|  | a("Adi Sarid", href = "https://adisarid.github.io", |
|  | target = "\_blank"))), |
|  | DT::DTOutput(ns("debug\_text")), |
|  | plotOutput(ns("result\_plot"), height = "500px"), |
|  | shinydashboard::box(tagList(p("Solid black = real functional relationship"), |
|  | p("Dashed blue = confidence intervals"), |
|  | p("Dashed red = prediction interval"), |
|  | p("Solid brown = model fit"), |
|  | p("Points = randomized sample")), |
|  | title = "Chart legend") |
|  | ) |
|  | ) |
|  | } |
|  |  |
|  | # Module Server |
|  |  |
|  | #' @rdname mod\_create\_intervals |
|  | #' @export |
|  | #' @keywords internal |
|  |  |
|  | mod\_create\_intervals\_server <- function(input, output, session){ |
|  | ns <- session$ns |
|  |  |
|  | # randomize noise ---- |
|  | eps\_noise <- reactive({ |
|  | rnorm(n = input$sample\_size, sd = input$epsilon\_error) |
|  | }) |
|  |  |
|  |  |
|  | # generate real relationship ---- |
|  | real\_functional\_relationship <- reactive({ |
|  | x\_tib <- tibble::tibble(x = seq(from = input$x\_range[1], |
|  | to = input$x\_range[2], |
|  | by = 0.05)) |
|  | if (input$functional\_relationship == "linear"){ |
|  | x\_tib %>% |
|  | mutate(y\_values\_actual = x\*input$select\_b + input$select\_a) |
|  | } else if (input$functional\_relationship == "log\_linear"){ |
|  | x\_tib %>% |
|  | mutate(y\_values\_actual = exp(x\*input$select\_b + input$select\_a)) |
|  | } else { |
|  | x\_tib %>% |
|  | mutate(y\_values\_actual = sin(x)\*input$select\_b + input$select\_a) |
|  | } |
|  | }) |
|  |  |
|  | # generate sample ---- |
|  | sample\_tibble <- reactive({ |
|  | x\_tib <- tibble(x = runif(n = input$sample\_size, |
|  | min = input$x\_range[1], |
|  | max = input$x\_range[2] |
|  | )) |
|  | if (input$functional\_relationship == "linear"){ |
|  | x\_tib %>% |
|  | mutate(y\_randomized = x\*input$select\_b + input$select\_a + eps\_noise()) |
|  | } else if (input$functional\_relationship == "log\_linear"){ |
|  | x\_tib %>% |
|  | mutate(y\_randomized = exp(x\*input$select\_b + input$select\_a + eps\_noise())) |
|  | } else { |
|  | x\_tib %>% |
|  | mutate(y\_randomized = sin(x)\*input$select\_b + input$select\_a + eps\_noise()) |
|  | } |
|  | }) |
|  |  |
|  | linear\_model\_predictions <- reactive({ |
|  | if (input$functional\_relationship == "linear"){ |
|  | linear\_model <- lm(y\_randomized ~ x, data = sample\_tibble()) |
|  | } else if (input$functional\_relationship == "log\_linear"){ |
|  | linear\_model <- lm(log(y\_randomized) ~ x, data = sample\_tibble()) |
|  | } else { |
|  | linear\_model <- lm(y\_randomized ~ sin(x), data = sample\_tibble()) |
|  | } |
|  | confidence\_interval <- predict(linear\_model, interval = "confidence", |
|  | newdata = real\_functional\_relationship()) %>% |
|  | tibble::as\_tibble() %>% |
|  | select(fit, lwr, upr) %>% |
|  | rename\_at(vars(2:3), ~paste0("ci\_",.)) |
|  |  |
|  | prediction\_interval <- predict(linear\_model, interval = "prediction", |
|  | newdata = real\_functional\_relationship()) %>% |
|  | tibble::as\_tibble() %>% |
|  | select(lwr, upr) %>% |
|  | rename\_at(vars(1:2), ~paste0("pi\_",.)) |
|  |  |
|  | prepped\_tibble <- real\_functional\_relationship() %>% |
|  | bind\_cols(confidence\_interval, |
|  | prediction\_interval) |
|  |  |
|  | if (input$functional\_relationship == "log\_linear"){ |
|  | prepped\_tibble <- prepped\_tibble %>% |
|  | mutate\_at(vars(ci\_lwr, ci\_upr, pi\_lwr, pi\_upr, fit), exp) |
|  | } |
|  |  |
|  | prepped\_tibble |
|  |  |
|  | }) |
|  |  |
|  | # output$debug\_text <- DT::renderDT({ |
|  | # DT::datatable(linear\_model\_predictions()) |
|  | # }) |
|  |  |
|  | output$result\_plot <- renderPlot({ |
|  |  |
|  | final\_plot <- ggplot() |
|  |  |
|  | if (input$include\_real\_function){ |
|  | final\_plot <- final\_plot + |
|  | geom\_line(data = linear\_model\_predictions(), |
|  | aes(x = x, y = y\_values\_actual), color = "black", inherit.aes = F, size = 1.5) |
|  | } |
|  |  |
|  | if (input$include\_geom\_point){ |
|  | final\_plot <- final\_plot + |
|  | geom\_point(data = sample\_tibble(), |
|  | aes(x = x, y = y\_randomized), alpha = 0.5, inherit.aes = F) |
|  | } |
|  |  |
|  | if (input$include\_ci){ |
|  | final\_plot <- final\_plot + |
|  | geom\_line(data = linear\_model\_predictions(), |
|  | aes(x = x, y = ci\_lwr), inherit.aes = F, color = "blue", linetype = 2, size = 1) + |
|  | geom\_line(data = linear\_model\_predictions(), |
|  | aes(x = x, y = ci\_upr), inherit.aes = F, color = "blue", linetype = 2, size = 1) |
|  | } |
|  |  |
|  | if (input$include\_pi){ |
|  | final\_plot <- final\_plot + |
|  | geom\_line(data = linear\_model\_predictions(), |
|  | aes(x = x, y = pi\_lwr), inherit.aes = F, color = "red", linetype = 2, size = 1) + |
|  | geom\_line(data = linear\_model\_predictions(), |
|  | aes(x = x, y = pi\_upr), inherit.aes = F, color = "red", linetype = 2, size = 1) |
|  | } |
|  |  |
|  | if (input$include\_fit){ |
|  | final\_plot <- final\_plot + |
|  | geom\_line(data = linear\_model\_predictions(), |
|  | aes(x = x, y = fit), inherit.aes = F, color = "brown", size = 1) |
|  | } |
|  |  |
|  | final\_plot + |
|  | theme\_bw() + |
|  | xlab("x") + |
|  | ylab("y") |
|  |  |
|  | }) |
|  |  |
|  | } |

Mod\_My\_First\_Module.R

|  |
| --- |
| # Module UI |
|  |  |
|  | #' @title mod\_my\_first\_module\_ui and mod\_my\_first\_module\_server |
|  | #' @description A shiny Module. |
|  | #' |
|  | #' @param id shiny id |
|  | #' @param input internal |
|  | #' @param output internal |
|  | #' @param session internal |
|  | #' |
|  | #' @rdname mod\_my\_first\_module |
|  | #' |
|  | #' @keywords internal |
|  | #' @export |
|  | #' @importFrom shiny NS tagList |
|  | mod\_my\_first\_module\_ui <- function(id){ |
|  | ns <- NS(id) |
|  | tagList( |
|  |  |
|  | ) |
|  | } |
|  |  |
|  | # Module Server |
|  |  |
|  | #' @rdname mod\_my\_first\_module |
|  | #' @export |
|  | #' @keywords internal |
|  |  |
|  | mod\_my\_first\_module\_server <- function(input, output, session){ |
|  | ns <- session$ns |
|  | } |

Run\_App.R

|  |
| --- |
| #' Run the Shiny Application |
|  | #' |
|  | #' @export |
|  | #' @importFrom shiny shinyApp |
|  | #' @importFrom golem with\_golem\_options |
|  | run\_app <- function(...) { |
|  | with\_golem\_options( |
|  | app = shinyApp(ui = app\_ui, server = app\_server), |
|  | golem\_opts = list(...) |
|  | ) |
|  | } |