R is great for report generation. Shiny allows us to easily create web apps that generate a variety of reports with R.

This post details a demo Shiny app that generates an Excel report, a PowerPoint report, and a PDF report:

The full Shiny app source code is listed below. Also, we included a more basic Shiny app that generates an Excel report at the end of this post. Follow up posts will include similar simple Shiny apps generating PowerPoint and PDF reports.

Source Code

Global.R

|  |
| --- |
| library(shiny) |
|  | library(shinydashboard) |
|  | library(tibble) |
|  | library(dplyr) |
|  | library(highcharter) |
|  | library(DT) |
|  | library(lubridate) |
|  | library(tidyr) |
|  | library(shinyWidgets) |
|  | library(tychobratools) |
|  | library(openxlsx) |
|  | library(officer) |
|  | library(flextable) |
|  |  |
|  | trans <- readRDS("./data/trans.RDS") |
|  |  |
|  | excel\_helpers <- source("helpers/format\_excel.R", local = TRUE)$value |
|  |  |
|  | state\_choices <- unique(trans$state) |
|  | ay\_choices <- trans %>% |
|  | mutate(year = year(accident\_date)) %>% |
|  | pull("year") %>% |
|  | unique() %>% |
|  | sort() |
|  |  |
|  | my\_colors <- c("#434348", "#7cb5ec") |
|  |  |
|  | hcoptslang <- getOption("highcharter.lang") |
|  | hcoptslang$thousandsSep <- "," |
|  | options(highcharter.lang = hcoptslang) |
|  |  |
|  | valueBox2 <- function (value, subtitle, icon = NULL, backgroundColor = "#7cb5ec", textColor = "#FFF", width = 4, href = NULL) |
|  | { |
|  |  |
|  | boxContent <- div( |
|  | class = paste0("small-box"), |
|  | style = paste0("background-color: ", backgroundColor, "; color: ", textColor, ";"), |
|  | div( |
|  | class = "inner", |
|  | h3(value), |
|  | p(subtitle) |
|  | ), |
|  | if (!is.null(icon)) { |
|  | div(class = "icon-large", icon) |
|  | } |
|  | ) |
|  | if (!is.null(href)) { |
|  | boxContent <- a(href = href, boxContent) |
|  | } |
|  | div( |
|  | class = if (!is.null(width)) paste0("col-sm-", width), |
|  | boxContent |
|  | ) |
|  | } |
|  |  |
|  | downloadButton2 <- function (outputId, label = "Download", class = NULL, icon = icon("download"), ...) { |
|  | tags$a( |
|  | id = outputId, |
|  | class = paste("btn btn-default shiny-download-link", class), |
|  | href = "", |
|  | target = "\_blank", |
|  | download = NA, |
|  | icon, |
|  | label, |
|  | ... |
|  | ) |
|  | } |
|  |  |
|  | display\_names <- tribble( |
|  | ~data\_name, ~display\_name, |
|  | "claim\_num", "Claim Number", |
|  | "accident\_date", "Accident Date", |
|  | "state", "State", |
|  | "claimant", "Claimant Name", |
|  | "report\_date", "Report Date", |
|  | "status", "Status", |
|  | "payment", "Payment", |
|  | "case", "Case Reserve", |
|  | "transaction\_date", "Transaction Date", |
|  | "trans\_num", "Transcation Number", |
|  | "paid", "Paid Loss", |
|  | "reported", "Reported Loss" |
|  | ) |
|  |  |
|  | #' show\_names |
|  | #' |
|  | #' @param nms character vector of names from the data |
|  | #' |
|  | #' @examples |
|  | #' show\_names(names(trans)) |
|  | #' |
|  | show\_names <- function(nms) { |
|  | nms\_tbl <- data\_frame(data\_name = nms) |
|  |  |
|  | nms\_tbl <- left\_join(nms\_tbl, display\_names, by = "data\_name") %>% |
|  | mutate(display\_name = ifelse(is.na(display\_name), data\_name, display\_name)) |
|  | nms\_tbl$display\_name |
|  | } |
|  |  |
|  | #' loss\_run |
|  | #' |
|  | #' view losses as of a specific date |
|  | #' |
|  | #' @param val\_date date the valuation date of the loss run. Claim values from `trans` |
|  | #' will be values as of the `val\_date` |
|  | #' @param trans data frame of claims transactions |
|  | #' |
|  | #' @importFrom dplyr `%>%` filter group\_by top\_n ungroup mutate arrange |
|  | #' |
|  | #' @return data frame of claims (1 claim per row) valued as of the `val\_date` |
|  | #' |
|  | loss\_run <- function(val\_date, trans\_ = trans) { |
|  | trans\_ %>% |
|  | filter(transaction\_date <= val\_date) %>% |
|  | group\_by(claim\_num) %>% |
|  | top\_n(1, wt = trans\_num) %>% |
|  | ungroup() %>% |
|  | mutate(reported = paid + case) %>% |
|  | arrange(desc(transaction\_date)) |
|  | } |

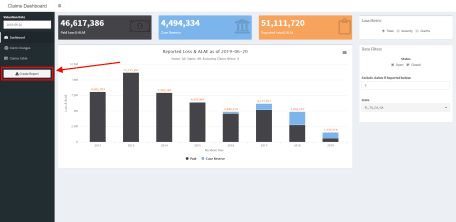
Server.R

|  |
| --- |
| function(input, output, session) { |
|  |  |
|  |  |
|  | val\_tbl <- reactive({ |
|  | req(input$val\_date) |
|  | loss\_run(input$val\_date) |
|  | }) |
|  |  |
|  | source("server/01-dashboard-srv.R", local = TRUE) |
|  | source("server/02-changes-srv.R", local = TRUE) |
|  | source("server/03-claims-srv.R", local = TRUE) |
|  |  |
|  | source("server/04-report-srv/04-report-srv.R", local = TRUE) |
|  | source("server/04-report-srv/04-modal-srv.R", local = TRUE) |
|  | source("server/04-report-srv/04-powerpoint-srv.R", local = TRUE) |
|  | source("server/04-report-srv/04-excel-srv.R", local = TRUE) |
|  |  |
|  | } |

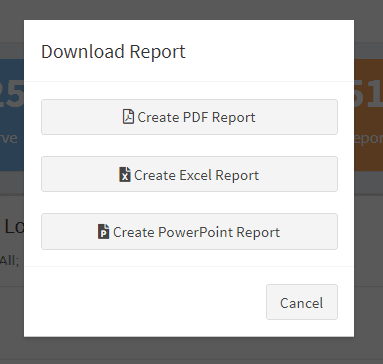
UI.R

|  |
| --- |
| library(shiny) |
|  |  |
|  | header <- dashboardHeader( |
|  | title = "Claims Dashboard" |
|  | ) |
|  |  |
|  | sidebar <- dashboardSidebar( |
|  | dateInput( |
|  | "val\_date", |
|  | "Valuation Date", |
|  | value = Sys.Date(), |
|  | min = min(trans$accident\_date), |
|  | max = Sys.Date(), |
|  | startview = "decade" |
|  | ), |
|  | sidebarMenu( |
|  | menuItem("Dashboard", tabName = "dashboard", icon = icon("dashboard")), |
|  | menuItem("Claim Changes", tabName = "changes", icon = icon("balance-scale")), |
|  | menuItem("Claims Table", tabName = "table", icon = icon("table")), |
|  | br(), |
|  | fluidRow( |
|  | column( |
|  | width = 11, |
|  | actionButton( |
|  | "generate\_report\_modal", |
|  | "Create Report", |
|  | icon = icon("download"), |
|  | #class = "btn-lg", |
|  | style = "width: 95%; color: #111" |
|  | ) |
|  | ) |
|  | ) |
|  | #menuItem("Reports", tabName = "report", icon = icon("file-pdf-o")) |
|  | ) |
|  | ) |
|  |  |
|  | body <- dashboardBody( |
|  | tags$head( |
|  | tags$link( |
|  | rel = "shortcut icon", |
|  | type = "image/png", |
|  | href = "https://res.cloudinary.com/dxqnb8xjb/image/upload/v1510505618/tychobra-logo-blue\_d2k9vt.png" |
|  | ), |
|  | tags$link(rel = "stylesheet", type = "text/css", href = "styles.css"), |
|  | tags$script(src = "custom.js") |
|  | ), |
|  | tabItems( |
|  | source("ui/01-dashboard-ui.R", local = TRUE)$value, |
|  | source("ui/02-changes-ui.R", local = TRUE)$value, |
|  | source("ui/03-claims-ui.R", local = TRUE)$value#, |
|  | #source("ui/04-report-ui.R", local = TRUE)$value |
|  | ) |
|  | ) |
|  |  |
|  | dashboardPage( |
|  | header, |
|  | sidebar, |
|  | body, |
|  | skin = "black" |
|  | ) |

Here are some screenshots of the Shiny app that generates the reports. You click the button in the left sidebar to select the report type (Excel, PowerPoint, or PDF).



And then you click a the report type button to generate the report. Simple as that!

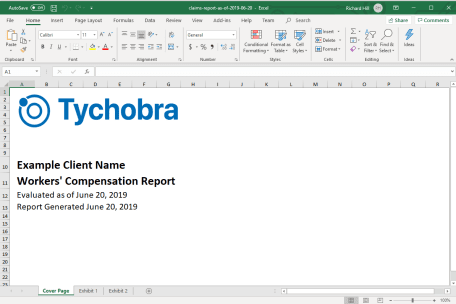


In this app, the generated reports rely on the user selected “Valuation Date” Shiny input (in the left sidebar). Your report generating Shiny app would, of course, include your data and the Shiny inputs necessary for your custom report.

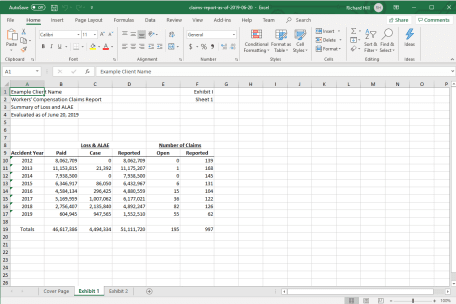
**Excel Report**

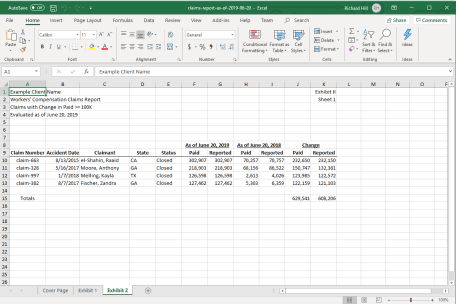
The generated Excel workbook has 3 sheets:

The “Cover Page” sheet includes an image and some custom text and cell styling.



The other 2 pages include tables with multi part headers, totals rows, and some custom styling.





Follow the link below to download a copy of the actual generated Excel file:

<https://github.com/Tychobra/shiny-insurance-examples/blob/master/basic-insurer-dashboard/example%20reports/claims-report-as-of-2019-06-20.xlsx>

The process of creating and customizing the Excel workbook is handled by the openxlsx R package. We like openxlsx because it does not require Java (several of the other R packages for working with Excel depend on Java), and it provides functions to highly customize the Excel workbook.

The following is a simple Shiny app that generates an Excel workbook. You can copy and paste this simple app into your R console to run it. Enjoy!

library(shiny)

library(openxlsx)

# create some example data to download

my\_table <- data.frame(

Name = letters[1:4],

Age = seq(20, 26, 2),

Occupation = LETTERS[15:18],

Income = c(50000, 20000, 30000, 45000)

)

# add a totals row

my\_table <- rbind(

my\_table,

data.frame(

Name = "Total",

Age = NA\_integer\_,

Occupation = "",

Income = sum(my\_table$Income)

)

)

# minimal Shiny UI

ui <- fluidRow(

column(

width = 12,

align = "center",

tableOutput("table\_out"),

br(),

downloadButton(

"download\_excel",

"Download Data to Excel"

)

)

)

# minimal Shiny server

server <- function(input, output) {

output$table\_out <- renderTable({

my\_table

})

output$download\_excel <- downloadHandler(

filename = function() {

"employee\_data.xlsx"

},

content = function(file) {

my\_workbook <- createWorkbook()

addWorksheet(

wb = my\_workbook,

sheetName = "Employee Data"

)

setColWidths(

my\_workbook,

1,

cols = 1:4,

widths = c(6, 6, 10, 10)

)

writeData(

my\_workbook,

sheet = 1,

c(

"Company Name",

"Employee Data"

),

startRow = 1,

startCol = 1

)

addStyle(

my\_workbook,

sheet = 1,

style = createStyle(

fontSize = 24,

textDecoration = "bold"

),

rows = 1:2,

cols = 1

)

writeData(

my\_workbook,

sheet = 1,

my\_table,

startRow = 5,

startCol = 1

)

addStyle(

my\_workbook,

sheet = 1,

style = createStyle(

fgFill = "#1a5bc4",

halign = "center",

fontColour = "#ffffff"

),

rows = 5,

cols = 1:4,

gridExpand = TRUE

)

addStyle(

my\_workbook,

sheet = 1,

style = createStyle(

fgFill = "#7dafff",

numFmt = "comma"

),

rows = 6:10,

cols = 1:4,

gridExpand = TRUE

)

saveWorkbook(my\_workbook, file)

}

)

}

shinyApp(ui, server)

The above app generates this neat little excel workbook:

