# 1.   Introducción.

# 2.   Objetivos.

# 3.   Desarrollo.

# 3.1.          Comisión Nacional de Seguros y Fianzas.

La Comisión Nacional de Seguros y Fianzas (CNSF) es un órgano de la Secretaria de Hacienda y Crédito Público encargado de regular y supervisar las operaciones del sector asegurador y afianzador de México. Su objetivo es preservar la solvencia y estabilidad financiera de las instituciones de Seguros y Fianza mediante el cumplimiento del marco normativo, el cual establece los requisitos y obligaciones que deben cumplir las compañías aseguradoras y afianzadoras, así como los derechos de los contratantes.

Entre sus funciones se encuentran proteger los intereses de los aseguradores, promover la competencia y trasparencia del sector asegurador y afianzador, extender sus servicios a la mayor parte posible de la población, así como realzar inspecciones y auditorias periódicas para verificar el cumplimiento de las normas. En caso de algún incumplimiento o irregularidad la Comisión puede aplicar sanciones y/o medidas correctivas para garantizar la seguridad y confianza del público usuario de estos servicios financieros.

## 3.2.          Reportes Regulatorios.

## 3.3.          Reporte regulatorio 8.

## 3.3.1.      Validaciones.

## 3.3.2.      Límites de tiempo.

## 3.4.          Manejo de Información.

## 3.4.1.      Privacidad y seguridad de la información.

## 3.4.2.      Volumen de información.

## 3.5.          R lenguaje de programación.

## 3.5.1.      Introducción a R.

Ahora que ya vimos un poco de la historia de R veamos como programar en este lenguaje.

R Studio que es la plataforma que nos permite interactuar con R esta divida en 4 secciones: La consola, donde podrás escribir código y ver el resultado de forma inmediata, una fuente de código, donde puedes escribir código sin que se ejecute, una sección de ayuda, donde podrás ver los archivos que tienes disponibles, los paquetes que tienes instalados, buscar funciones y visualizar gráficas, y por último, el ambiente en el que trabajas, en esta sección se mostraran todos los objetos que hayas almacenado y las conexiones que puedas tener con otras aplicaciones, como git, SQL, etc.

El código se escribe en los scripts o markdowns:

Si tu trabajo es puramente código es más recomendable trabajar en scripts para posiblemente después pasar a un paquete.

Si tienes código, pero quieres tener la documentación de ese código, o crear reportes entonces un markdown es la mejor opción. En este tipo de documentos puedes tener texto, código y en el mismo documento el resultado del código.

R también tiene diferentes outputs, como libros, presentaciones, páginas web y más.

## 3.5.2.      Funciones.

Una función nos permite automatizar procesos para evitar la repetición de código, crear funciones nos da como beneficio:

* Evitar repeticiones, en este projecto estamos trabajando con tres tipos de riesgo, las funciones nos permiter correr el mismo código cambiando los valores del tipo de riesgo.
* Evitar errores, ya que al tener un solo código no corres el riesgo de dejar una sección sin actualizar,
* Puede ser utilizada para otros projectos u objetivos.

Las funciones se componen de tres partes: El nombre, tiene que ser un nombre que aporte información del objetivo de tu función. Los argumentos, que son variables dentro del código, y por ultimo, el cuerpo, que es donde se escribe todo el proceso.

Los argumentos nos permiten generalizar el código y obtener distintos resultados dependiendo de su valor. Cuando una función es creada como una mejor practica es recomendable poner los argumentos en el siguiente orden:

* Datos, ya sea una base de datos o un vector.
* Variables pricipales dentro de tu código.
* Variables que tienen un valor predefino pero que pueden ser modificadas.

Para crear las validaciónes del rr8 estas son las funciones necesarias para el entendimiento del código.

### Importar datos

La herramienta más común para manejar bases de datos es excel y en este caso no fue la excepción. Para importar valores dentro de una hoja de datos se utiliza la función read\_excel correspondiente al paquete readxl el cual contiene funciones para leer hojas completas, pero también permite importar rangos dentro de los archivos.

{r}

# Ejemplo de como importar un archivo dentro del folder "data"

read\_excel("data/dataset.xlsx", range = cell\_cols("C:E"))

Tambien es muy útil la función excel\_sheets que nos enseña una lista de los nombres de las pestañas dentro de un archivo.

{r} excel\_sheets("data/dataset.xlsx") # return the sheet names

Y para finalizar, podemos exportar datos con el paquete writexl, con la función write\_xlsx podemos guardar las tablas creadas en R en un archivo de excel.

{r} write\_xlsx(df\_percent, "data/dataset.xlsx") # export to excel

### Manipular Datos

Dentro del paquete dplyr tenemos una serie de funciones que nos permiten manipular los datos y que fueron clave para las validaciones.

### filter

Para crear las validaciones una de las principales funciones es el filter, esta función nos ayuda a crear un subconjunto de los renglones que cumplen con cierta condición.

{r} # Con esto corremos la validacion que todas los montos de suma asegurada sean # mayor a 0 incendio\_df %>% filter(sum\_ase >= 0)

### select

Esta función nos permite extraer un subconjunto de las columnas dentro de la tabla, también nos permite asignar un orden distinto a las columnas.

{r} incendio\_df %>% select(num\_pol, fec\_ini\_vig, fec\_fin\_vig, prim\_dev)

### mutate

Para crear y manipular columnas podemos usar la función mutate, esta utiliza columnas existentes en la tabla para crear nuevos valores.

{r} incendio\_df %>% mutate( check\_date = fec\_fin\_vig > fec\_ini\_vig )

### arrange

Esta función nos permite organizar los datos en orden ascendente o descendente por las columnas especificadas.

{r} incendio\_df %>% arrange( fec\_ini\_vig, desc(fec\_fin\_vig) )

### group\_by + summarise

Con la combinación de estas dos funciones podemos calcular valores por grupo, dependiendo de los valores del group\_by.

{r} incendio\_df %>% group\_by(num\_pol, giro) %>% summarise( tot\_prim\_dev = sum(prim\_dev) )

### rename / clean\_names

Estas son funciones auxiliares para limpiar los nombres de las columnas, por un lado, rename toma el nombre de la columna y le asigna un nuevo valor, por el otro, clean\_names convierte todos los nombres a un estilo más generico (minusculas, sin espacios)

{r} incendio\_df %>% rename( prima\_devengada = prim\_dev )

### Unir tablas

Para generar la validaciones una accion muy común es pegar valores de una tabla externa dependiendo de una llave, con las distintas funciones para unir podemos conseguir resultados diferentes.

### left\_join

Esta funcion nos permite pegar información dentro de una tabla pero conservando los valores unicos de la tabla original.

{r} incendio\_df %>% left\_join( mapeo\_entidad, by = "entidad" )

### full\_join

Usando el full\_join al unir las tablas se conservan todos los valores de ambas tablas.

{r} incendio\_df %>% full\_join( pol\_canceladas, by = num\_pol )

### anti\_join

Esta función es muy útil, esta nos regresa un subconjunto de la tabla original donde los valores de la llave no se encontraron en la tabla auxiliar, esta función sirve para validar que todos los valores dentro de la tabla original también esten en la auxiliar, en este caso todos los catalogos de la comisión.

{r} incendio\_df %>% anti\_join( cat32, by = "entidad" )

### Iterar

La idea principal para construir las validaciones es crearlas para un riesgo, una columna o una tabla y luego iterar sobre los tres riesgos.

### map

Nos permite iterar sobre una variable

### map2

Nos permite iterar sobre dos variables,

## 3.5.3.      Paquetes.

Un paquete es un conjunto de funciones acompañadas de viñetas y tablas en una estructura especifica, los paquetes en R son fáciles de compartir y puedes encontrar dentro de cada uno documentación para su uso.

El uso de paquetes es recomendable para cumplir ….

Un paquete se puede distinguir de un proyecto por los siguientes archivos:

1. DESCRIPTION FILE. Este es un script con todos los metadatos del paquete, el nombre, la versión, los autores, etc.
2. Folder R. Dentro de este folder se van a encontrar todas las funciones, cada función deberá contar con su respectiva documentación.
3. NAMESPACE. Este archivo contiene la lista de elementos que se exportaran de tu paquete, al igual que los paquetes o funciones auxiliares que utilices
4. Rbuildignore. Lista de archivos que son necesarios para construir el paquete, pero no son parte de él.
5. gitignore. Elementos que permiten conectar el paquete a GitHub.

Puedes crear un paquete empezando desde cero con la ayuda de *devtools* y *usethis* [@usethis] ambos contienen funciones que facilitan el trabajo como usethis::create\_package( ) que genera un proyecto en R con la estructura adecuada para hacer un paquete. Cuando usamos esta función se crea un nuevo proyecto como se ve en la imagen.

Como mencionamos un paquete es el conjunto de información y vive dentro de tu librería, para acceder a las funciones usamos la función `library` que nos permite acceder a los paquetes que tienes instalados

|  |  |  |
| --- | --- | --- |
| **Paquete** | **Subpaquete** | **Uso** |
| shinydashboard |  | Facilita la creación de tableros de control (dashboards) |
| shinyWidgets |  | Herramientas para personalizar el ui. |
| shinyFiles |  | Permite navegar por tus archivos dentro de un shiny. |
| tidyverse | readxl | Utilizado para importar hojas de excel. |
| tidyverse | stringr | Permite modificar caracteres |
| tidyverse | purrr | Permite iterar sobre vectores o tablas |
| tidyverse | dplyr | Funciones para manipular datos |
| janitor |  | Funciones para "limpiar" nombres. |
| scales |  | Permite personalizar elementos de las gráficas. |
| DT |  | Tablas con mayor formato e interacción con el usuario. |
| shinyalert |  | Crea notificaciones dependiendo de condiciones especificadas. |
| excelR |  | Tablas interactivas de excel en R |
| lubridate |  | Manejo y manipulación de fechas |
| shinyjs |  | Auxiliar de Java en aplicaciones shiny. |
| reactable |  |  |
| shinyBS |  |  |
| tippy |  |  |

Durante este proyecto

# 4.   Metodología.

## 4.1.          Código para una validación.

## 4.2.          Generalización para todas las validaciones.

## 4.3.          Generalización para los subramos Incendio, Terremoto y Riesgos Hidrometereológicos.

## 4.4.          Creación del paquete.

# 5.   Resultados.

## 5.1.          Documentación de la paquetería.

## 5.2.          Ejemplo de un caso de uso.

# 6.   Conclusiones.

# 7.   Bibliografía.

# Anexos