



Cursos Integrados em Vigilância em Saúde

Curso —

**Construção de painéis (dashboards) para
monitoramento de indicadores de saúde**

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Reitor Irineu Manoel de Souza

Vice-Reitora Joana Célia dos Passos

Pró-Reitora de Pós-graduação Werner Kraus

Pró-Reitor de Pesquisa e Inovação Jacques Mick

Pró-Reitor de Extensão Olga Regina Zigelli Garcia

CENTRO DE CIÊNCIAS DA SAÚDE

Diretor Fabrício de Souza Neves

Vice-Diretora Ricardo de Souza Magini

DEPARTAMENTO DE SAÚDE PÚBLICA

Chefe do Departamento Rodrigo Otávio Moretti Pires

Subchefe do Departamento Sheila Rúbia Lindner

Coordenadora do Curso Alexandra Crispim Boing

INSTITUTO TODOS PELA SAÚDE (ITPS)

Diretor presidente Jorge Kalil (Professor titular da Faculdade de Medicina da Universidade de São Paulo; Diretor do Laboratório de Imunologia do Incor)

ASSOCIAÇÃO BRASILEIRA DE SAÚDE COLETIVA (ABRASCO)

Presidente Rosana Teresa Onocko Campos

EQUIPE DE PRODUÇÃO

Denis de Oliveira Rodrigues

Kamila de Oliveira Belo

Marcelo Eduardo Borges

Oswaldo Gonçalves Cruz

Alexandra Crispim Boing

Antonio Fernando Boing



Curso

Construção de painéis (dashboards) para monitoramento de indicadores de saúde



Dados Internacionais de Catalogação-na-Publicação (CIP)

C758 Construção de painéis (dashboards) para monitoramento de indicadores de saúde/ Marcelo Eduardo Borges, Kamila de Oliveira Belo, Denis de Oliveira Rodrigues e Oswaldo Gonçalves Cruz . Santa Catarina ; São Paulo ; Rio de Janeiro : UFSC ; ITPS ; Abrasco; 2022. 76p. (Cursos Integrados em Vigilância em Saúde).

Publicação Online

[10.52582/curso-analise-dados-vigilancia-modulo11](https://repositorio.ufsc.br/handle/10.52582/curso-analise-dados-vigilancia-modulo11)

1. Vigilância em saúde 2. Análise de dados I. Título

Sumário

Painéis de dados	06
1. O que são painéis de dados?	08
2. Criando o dashboard no R	09
3. Preparando o dashboard	11
3.1 O cabeçalho.....	16
3.2 Espaço para a escrita.....	17
3.3 Espaço para os códigos	18
4. Padronizando o layout.....	23
4.1 Layout orientado por colunas	24
4.2 Layout orientado por linhas	29
5. Definindo o conteúdo para o dashboard	32
5.1 Conhecendo os objetivos do painel.....	33
5.2 Organizando os dados necessários	34
5.3 Inserindo caixas de valores.....	38
5.4 Inserindo gráficos estáticos e interativos.....	46
5.4.1 Gráficos estáticos	46
5.4.2 Gráficos interativos	51
5.5 Inserindo tabelas estáticas e interativas	57
5.5.1 Tabelas estáticas.....	58
5.5.2 Tabelas interativas.....	63
6. Publicando o dashboard	68
6.1 Tema	69
6.2 Logomarcas	72
7. Outros componentes	75

Painéis de dados

Você já imaginou ter parte dos seus dados de vigilância em saúde visualizados de forma interativa em um painel *online*? Agora imagine acessá-lo remotamente em uma reunião com a atenção primária da sua região e fazer análises comparativas de forma interativa. Ou ainda apresentar estas análises sem se preocupar em salvar um **Power Point** ou imprimir o relatório para leitura conjunta.

Seria perfeito, não é mesmo?! Sim, a área da saúde está se transformando digitalmente! Neste curso, você aprenderá a organizar suas análises e transformar seus relatórios personalizados em painéis interativos que podem ser utilizados entre a equipe da vigilância ou para produção de uma comunicação com a sociedade.

Com o apoio da linguagem de programação R você poderá vivenciar todas as situações acima. Neste curso iremos colocar em prática um dos pilares mais importantes para a Vigilância em Saúde: dar transparência à avaliação da situação de saúde. Com o uso do R o profissional de vigilância em saúde poderá fornecer acesso às informações oportunas no momento, de forma intuitiva, com análises confiáveis e metodologicamente bem descritas de forma a ampliar o acesso à informação à toda população.

Além disso, você desenvolverá os *dashboards* que podem transformar os informes epidemiológicos em painéis que apoiarão a tomada de decisão baseada em dados que serão atualizados no tempo que você programar ou em tempo real.

Caso sua rotina de trabalho na vigilância seja intensa e a tarefa de gerenciar vários bancos de dados em saúde se torna complexo e desafiador, então este curso foi estruturado para você!

Ao final deste curso, você será capaz de:

1. conceituar o que são *dashboards*;
2. utilizar o pacote `flexdashboard` no R;
3. gerar gráficos e tabelas interativos;
4. construir um painel com indicadores de saúde.

Atenção



Para seguir com este curso, você deve conhecer as ferramentas básicas para uso do R e do RStudio, além de possuir conhecimentos básicos de rotinas de análises e visualização de dados utilizando a linguagem de programação R. É recomendável conhecer metodologias básicas para a construção de relatórios automatizados utilizando o RMarkdown disponível no curso “**Produção automatizada de relatórios na vigilância em saúde**”. Você poderá também acessar a qualquer momento o curso “**Análise de dados para a vigilância em saúde - curso básico**” e obter os códigos desejados para a confecção do seu painel. Caso não tenha feito o curso, sugerimos fortemente que se inscreva nele.

1. O que são painéis de dados?

Os painéis de dados, também chamados de *dashboards*, são painéis visuais que permitem comunicar informações, métricas, mapas e indicadores de saúde e que apoiam análises referentes a um determinado assunto na vigilância em saúde. Por exemplo, é possível incluir nos painéis dados de séries temporais, número de casos de determinado agravo, pirâmides etárias e gráficos comparativos que variam no tempo e no espaço. Além disso, os painéis também permitem o monitoramento e a exploração de dados em tempo real ou com atualização constante.

O painel de indicadores tem se mostrado uma ferramenta poderosa por permitir que seja realizado o acompanhamento da evolução de quaisquer doenças, agravos ou eventos em saúde de forma dinâmica. Neste ponto, os *dashboards* são amplamente utilizados, pois diferentemente dos arquivos estáticos - como relatórios técnicos, textos impressos ou arquivos digitais em PDF's - os painéis permitem interação com a tela.



Destacamos que os dados apresentados em um painel precisam ser claros, coesos e devem ter sido tratados com métodos bem descritos. É preciso que seja possível uma compreensão rápida e fácil da informação, de modo a facilitar o trabalho de gestores e profissionais da área da saúde na comunicação de saúde a toda população.

2. Criando o dashboard no R

O R é uma linguagem de programação poderosa. Com ele é possível automatizar suas análises transformando-as em relatórios. Utilizando o pacote `flexdashboard` seremos capazes de construir painéis *online*s que tornem os relatórios que você já sabe fazer acessíveis a qualquer pessoa que obtiver o *link* fornecido.

O pacote `flexdashboard` é parte da linguagem `markdown`. Ele é responsável pela criação de painéis dinâmicos para visualizações de dados de forma simples e flexível. Assim, com o pacote `flexdashboard` poderemos:

- Utilizar a estrutura de um documento `RMarkdown` para a publicação de painéis dinâmicos, os chamados **dashboards**.
- Incorporar uma série de componentes, como gráficos, dados tabulares, mapas, caixas para apresentação de índices e valores, caixas de seleção e filtragem e anotações de texto.
- Especificar *layouts* para a disposição dos componentes ou, ainda, redimensioná-los de forma inteligente para serem exibidos em navegadores de internet ou em dispositivos móveis.
- Integrar o R com ferramentas de *dashboards* mais avançadas, como o pacote `Shiny`.



A linguagem *markdown* é uma “linguagem de marcação” utilizada para informar ao computador como ele deve interpretar e estruturar seus arquivos ou documentos.

Uma página de *internet*, por exemplo, utiliza uma linguagem de marcação que chamamos de **HTML**. Esta linguagem informa ao navegador de *internet* quais os elementos presentes em um site, qual a localização de cada um e a sua formatação.

O **R***markdown* permite que possamos unir a linguagem *markdown* a pedaços de códigos que utilizam a linguagem R!

Para saber mais, acesse o curso **“Produção automatizada de relatórios na vigilância em saúde”** e obtenha os códigos desejados.

Resumidamente, os passos para a criação de um painel de dados no R envolvem:

1. a instalação de pacotes específicos para a criação de um arquivo do tipo **.Rmd** (*dashboard*),
2. a escrita de um *script* definindo as bases de dados,
3. os objetos e componentes que serão utilizados no painel,
4. a estruturação do conteúdo a ser apresentado,
5. a padronização de um layout e
6. a transformação do *script* em uma página no formato HTML (renderização).

Mas vamos com calma. Você verá este processo todo exemplificado neste curso! Com o R, os conhecimentos desse curso e seu estudo você poderá construir painéis como estes: - [Dashboard de arboviroses do município do Rio de Janeiro](#), - [Dashboard de covid-19 no mundo](#), - [Dashboard de covid-19 em Portugal](#), - [Dashboard do ministério da saúde do Malawi](#), - [Dashboard de covid-19 no Brasil](#), - [Dashboard de covid-19 da Fiocruz](#), e - [Dashboard de análise de dados do Sistema de Informação sobre Mortalidade \(SIM\) - MorbIS](#).

Vamos lá?!

3. Preparando o dashboard

Para a construção de um painel, temos alguns passos a serem seguidos. Antes de mais nada, precisamos carregar e instalar todos os pacotes específicos que utilizaremos para a produção do nosso painel de indicadores de saúde.

Observe o *script* abaixo e replique-o em seu RStudio:

```
# Instalando e carregando os pacotes necessários
if(!require(flexdashboard)) install.packages("flexdashboard")
if(!require(foreign)) install.packages("foreign")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(lubridate)) install.packages("lubridate")
if(!require(plotly)) install.packages("plotly")
if(!require(knitr)) install.packages("knitr")
if(!require(DT)) install.packages("DT")
```

Agora, com os pacotes devidamente instalados e carregados, iremos criar um documento que irá receber as informações do nosso *dashboard*. Este passo é semelhante à maneira como geramos um relatório RMarkdown.

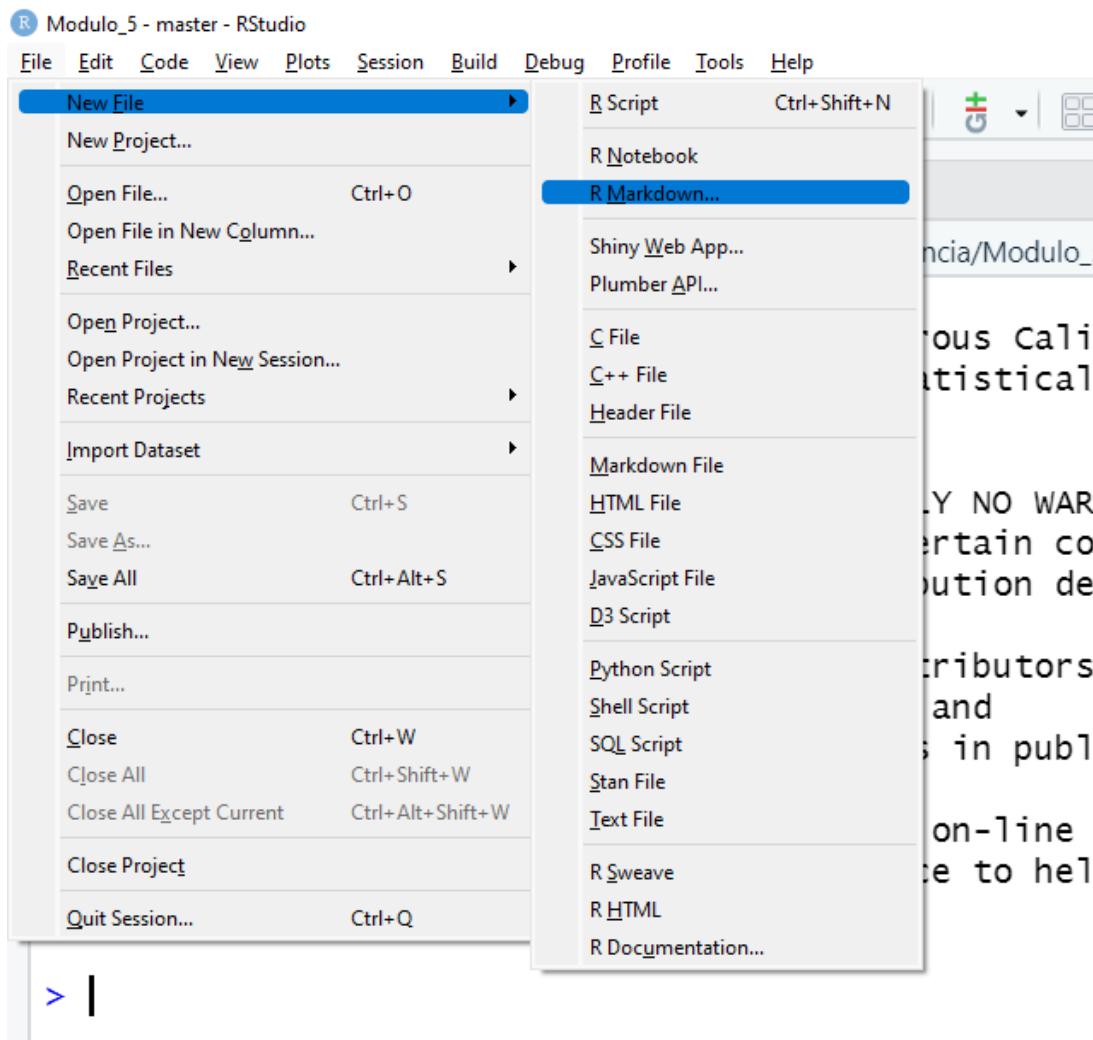
Após a instalação do pacote `flexdashboard` necessitamos reiniciar o RStudio: feche o seu RStudio e reabra-o para que a opção de criar um novo arquivo com o modelo do tipo *Flexdashboard* esteja disponível no menu do programa. A partir disso, todas as vezes em que escolher iniciar um projeto novo esta opção estará presente, mesmo que o pacote ainda não tenha sido carregado.

Vamos lá criar nosso *dashboard*. Siga os passos abaixo no seu RStudio conforme a Figura 1:

1. vá até o menu do RStudio e clique em File (Arquivo),
2. em seguida em New File (Novo Arquivo), e
4. depois em R Markdown...

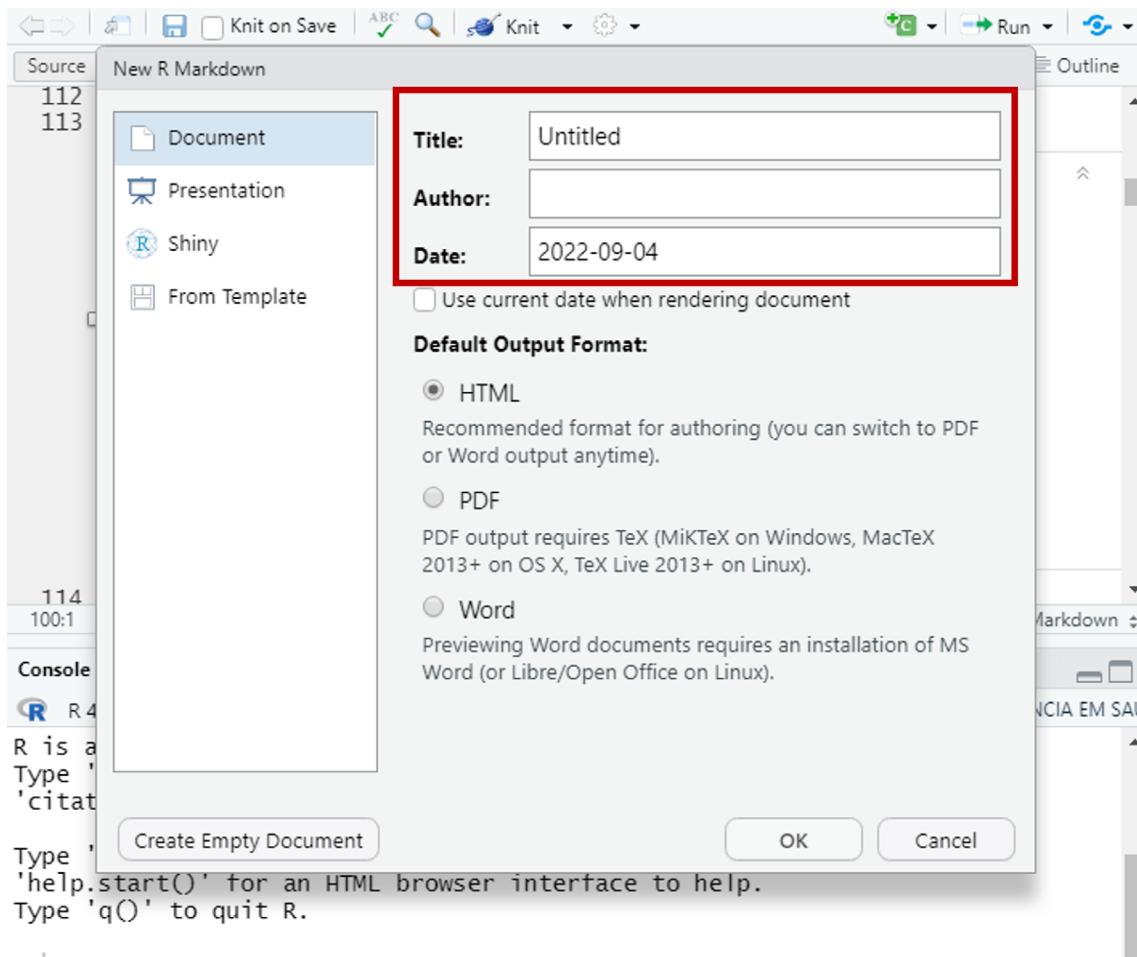
Se o seu RStudio estiver em português utilize o mesmo passo a passo, porém verá o caminho da seguinte forma: Arquivo > Novo Arquivo > R Markdown....

Figura 1: Tela para abrir um documento RMarkdown.



Após criar um documento RMarkdown, abrirá em seu RStudio uma nova janela com opções de criação de **novos projetos**. Nesta janela são inseridos os dados básicos do documento conforme a Figura 2. Em seu computador insira o título, o nome do autor e a data do seu painel, com isso configuraremos o nosso documento.

Figura 2: Tela de criação de um documento RMarkdown.

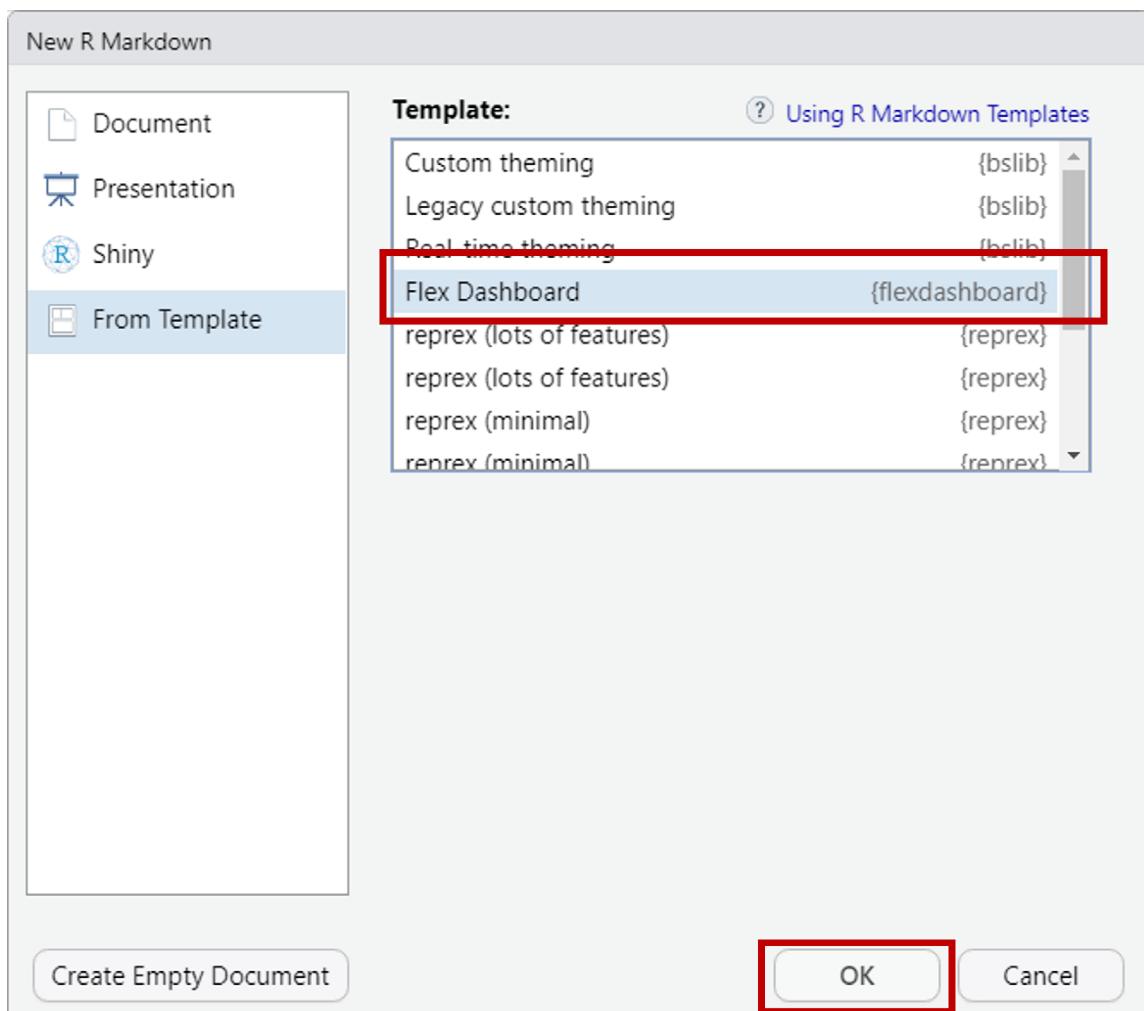


Agora precisamos criar o arquivo de abertura do **Flex Dashboard**. Observe que nesta tela de configuração o seu painel é à esquerda.

Nele você visualizará algumas opções, selecione a opção **From Template**. Agora, na caixa que se abriu à direita escolha como **Template** a opção **Flex Dashboard** conforme a Figura 3. Em seguida, clique em **OK** para criar um arquivo um arquivo do tipo **.Rmd (dashboard)**.



Figura 3: Tela de seleção para abertura do Flex Dashboard.



Perceba que após clicar em **OK**, o RStudio abrirá um arquivo no formato **.Rmd**, com uma estrutura padrão de um *script* para criação de painel. Veja na Figura 4 como você deverá visualizar os elementos presentes no arquivo que criou em seu RStudio.

Figura 4: Tela com *script* gerado automaticamente após selecionar o Flex Dashboard.

```
1 ---  
2 title: "Untitled"  
3 output:  
4   flexdashboard::flex_dashboard:  
5     orientation: columns  
6     vertical_layout: fill  
7 ---  
8  
9 {r setup, include=FALSE}  
10 library(flexdashboard)  
11  
12 Column {data-width=650}  
13  
14 1  
15  
16 2  
17 3  
18 {r}  
19 ...  
20  
21  
22 Column {data-width=350}  
23  
24  
25 4  
26 5  
27 {r}  
28 ...  
29  
30  
31 6  
32 7  
33 {r}  
34 ...  
35  
36  
37
```

Observe que a estrutura do *script* é semelhante a de um arquivo de relatório **.Rmd**. Perceba que neste arquivo da Figura 4 há três partes principais:

1. Cabeçalho em **YAML** (sigla para *Yet Another Markup Language*).
 2. Blocos para textos e elementos da linguagem *markdown* (com fundo branco).
 3. Blocos de códigos (*chunks*) em **R** (fundo acinzentado, delimitado por três aspas).

Agora vamos detalhar a diferença entre elas e como podemos personalizar estas partes.

3.1 O cabeçalho

No cabeçalho, conforme a Figura 5, você verá os detalhes sobre o documento escritos no formato YAML (sigla para *Yet Another Markup Language*). Observe que esta parte do arquivo é delimitada por três hifens (---). Além disso ela contém informações essenciais sobre o seu arquivo, como o título (*title*) e o formato de saída (*output*) do arquivo que precisarão ser configuradas de acordo com a sua necessidade.

Observe no seu RStudio se o seu cabeçalho está parecido com o da Figura 5. Lembre-se de que a Figura 5 ainda não está personalizada.

Figura 5: Visualização do cabeçalho em YAML no arquivo RMarkdown.

```
1 ---  
2 title: "Untitled"  
3 output:  
4   flexdashboard::flex_dashboard:  
5     orientation: columns  
6     vertical_layout: fill  
7 ---
```



Se ainda não sabe o que é o RMarkdown, ou tem dúvidas sobre como construir um relatório no R, revise e aprofunde este conteúdo acessando o curso “Produção automatizada de relatórios na vigilância em saúde”.

Aumente seus conhecimentos!

3.2 Espaço para a escrita

Agora observe no arquivo que foi criado uma parte de escrita com fundo branco, conforme a Figura 6. Nela você deve utilizar a linguagem *markdown* para destacar elementos do texto e formatar componentes para construção de seu painel. Visualize o seu arquivo e perceba se a configuração é semelhante à mostrada na Figura 6.

Figura 6: Visualização da inserção da linguagem markdown.

```
11 ~~~  
12  
13 Column {data-width=650}  
14 ~~~  
15  
16 ~~~ ### Chart A  
17  
18 ~~~ {r}  
19
```

Perceba que a escrita neste bloco pode ser feita da mesma maneira como a que escreve seus relatórios, por exemplo, no Microsoft Word. Neste espaço você deve incluir tudo que achar relevante para seu painel. Lembre-se de observar as correções ortográficas e que, quando fazemos um painel, o conteúdo explicativo deve articular informações verbais com os visuais, ou seja, você deve publicar seus dados e análises de forma simples garantindo que o usuário do painel compreenderá seu conteúdo quando contenham análises complexas ou que transmitam a gravidade de uma doença ou evento.



Inspire-se nos **Infográficos** que costumam conter textos, ilustrações, gráficos, sons, ícones e outros tipos de mídia. Valorizam-se, assim, as informações relevantes trazendo leveza ao leitor.

Veja alguns exemplos:

- [Materiais de comunicação da Organização Pan-Americana de Saúde \(OPAS\)](#)
- [Materiais de comunicação do Instituto Nacional de Câncer \(INCA\)](#)

Calma que as coisas vão seclareando aos poucos. Sigamos em frente.

3.3 Espaço para os códigos

Mas e onde iremos inserir os códigos para analisar dados no R?

Observe a Figura 7 e perceba o espaço que delimita onde poderemos incluir os blocos de códigos, também chamados de *chunks*, no seu RStudio. Este espaço permitirá que você escreva ali os nossos comandos, contendo tabelas, gráficos ou qualquer outro elemento que componha a sua análise de dados.

Estes *chunks* são delimitados entre os símbolos ` ` ` {r} e ` ` `. Dentro destes blocos de código é possível adicionar códigos da linguagem R que serão interpretados para a produção do dashboard, semelhante à forma como escrevemos os relatórios no RMarkdown. Observe a Figura 7 e localize esta imagem em seu RStudio.

Figura 7: Visualização dos blocos de códigos (*chunks*) no arquivo RMarkdown.

```
13 column {data-width=650}
14 -
15
16 - ### Chart A
17
18 - ``{r}
19
20 -
21
22 column {data-width=350}
23 -
24
25 - ### Chart B
26
27 - ``{r}
28
29 -
30
31 - ### Chart C
```

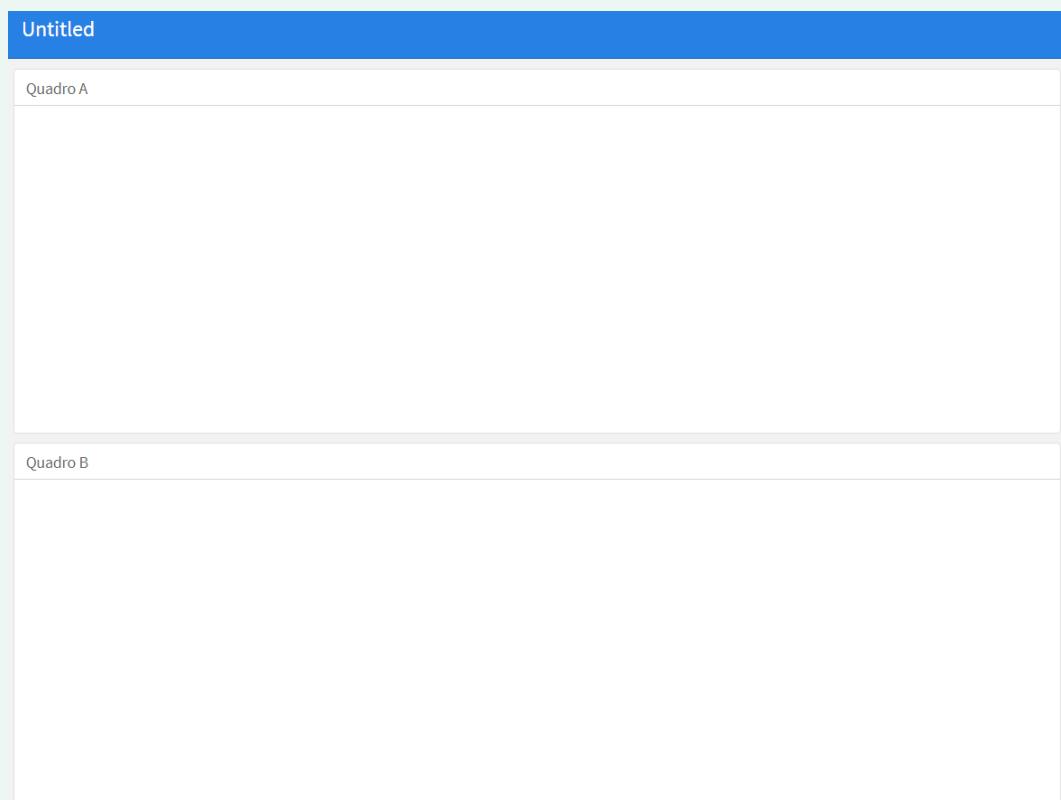


É possível especificar o tamanho do painel, principalmente quando necessitamos que ele seja exibido em telas menores como celular. Para isso, utilizamos o argumento `vertical_layout`, que fica no cabeçalho `YAML` para torná-lo visualizável em qualquer tela.

Escolhendo o argumento `vertical_layout igual(:) a fill` (`vertical_layout: fill`) os quadros irão preencher toda a tela do seu navegador. Assim seu *dashboard* ocupará 100% do espaço vertical disponível. **Esta é a opção que estamos utilizando neste curso!**

O argumento `vertical_layout: fill` é ideal quando você optou por um *dashboard* apenas um ou dois quadros empilhados verticalmente conforme a Figura 8 abaixo.

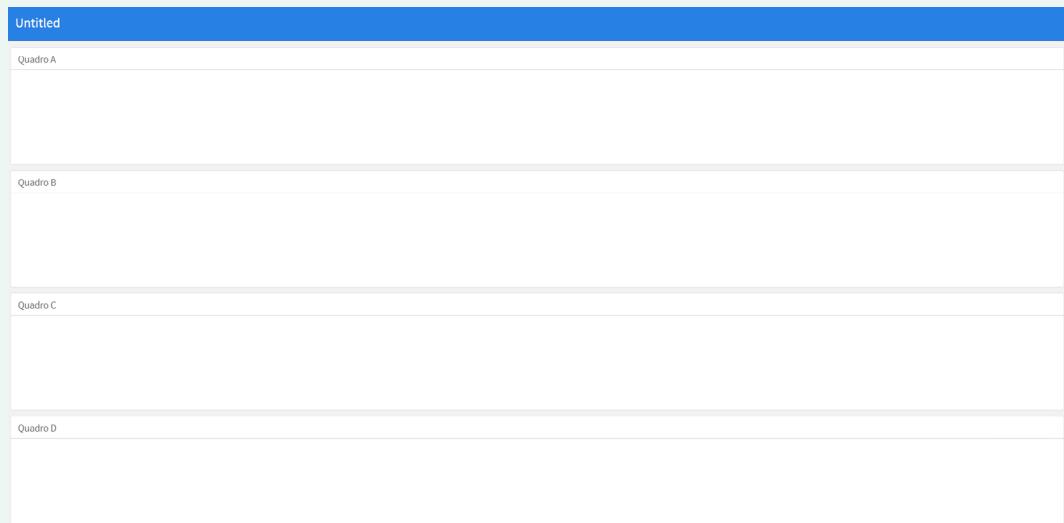
Figura 8: Visualização de um *dashboard* configurado com o argumento `vertical_layout: fill`.





Caso você tenha um número grande de quadros que excedam o limite vertical do *dashboard* é aconselhado utilizar o argumento `vertical_layout: scroll`, para que o painel se organize sequencialmente ao longo de uma página e consigamos visualizá-lo pela barra de rolagem do seu navegador de internet como na Figura 9 logo abaixo.

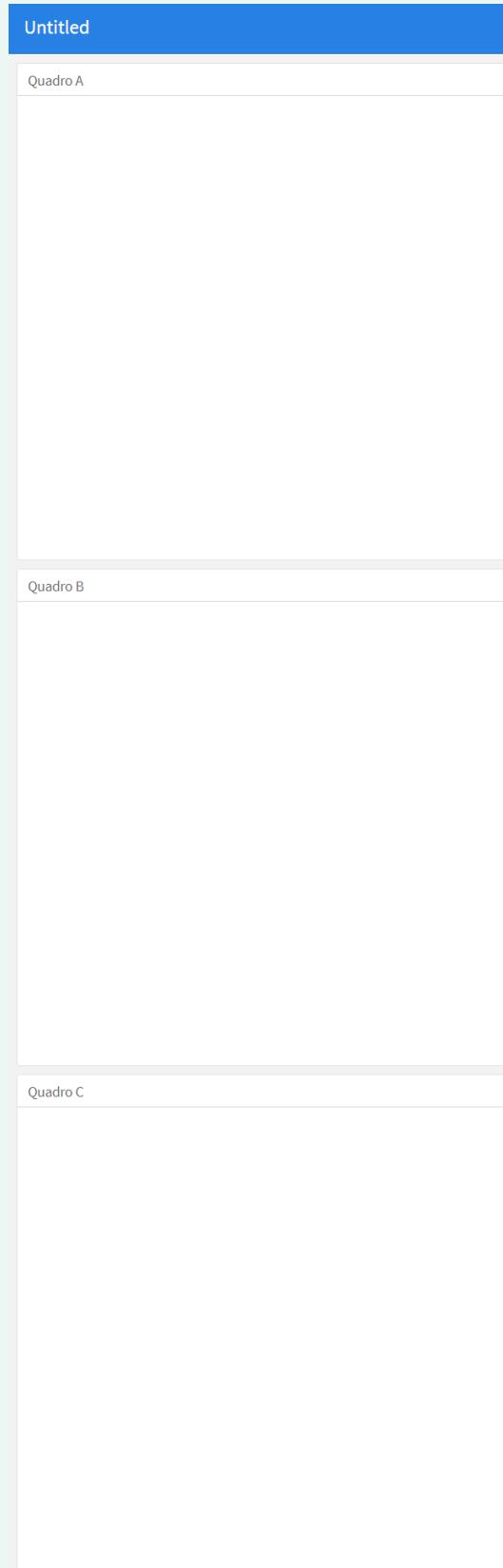
Figura 9: Visualização de um *dashboard* configurado com o argumento `vertical_layout: scroll`.



Observe agora a visualização do *dashboard* com layout “scroll”, na Figura 10, onde a janela de visualização está ocupando um espaço menor da tela. Neste caso, os quadros são automaticamente empilhados um sobre o outro e é possível visualizar os diferentes quadros utilizando a barra de rolagem do navegador.



Figura 10: Visualização de um *dashboard* configurado
com o argumento `vertical_layout: scroll` com a
janela maximizada, incluindo a barra de rolagem.



4 Padronizando o layout

Agora que já criou seu arquivo que receberá os dados iremos aprender a configurar a orientação e criar um *layout* para o seu *dashboard*. Vamos lá!

Na construção do painel de dados é importante que organizemos os elementos visuais em uma página. A esta organização damos o nome de *layout*. Nele dispomos os elementos visuais que avaliamos necessários a partir de princípios organizacionais para que esta composição atinja os objetivos específicos de comunicação. Aliás você já sabe qual o objetivo do painel que estamos construindo?

Responder a esta pergunta será importante para tomar decisões sobre o seu formato e possibilidades de apresentação da informação.

Nesta seção do curso você aprenderá duas possibilidades para a montagem de seu *dashboard* e poderá optar por uma delas quando o objetivo de seu painel estiver delimitado. As duas possibilidades para construção de painéis no R são:

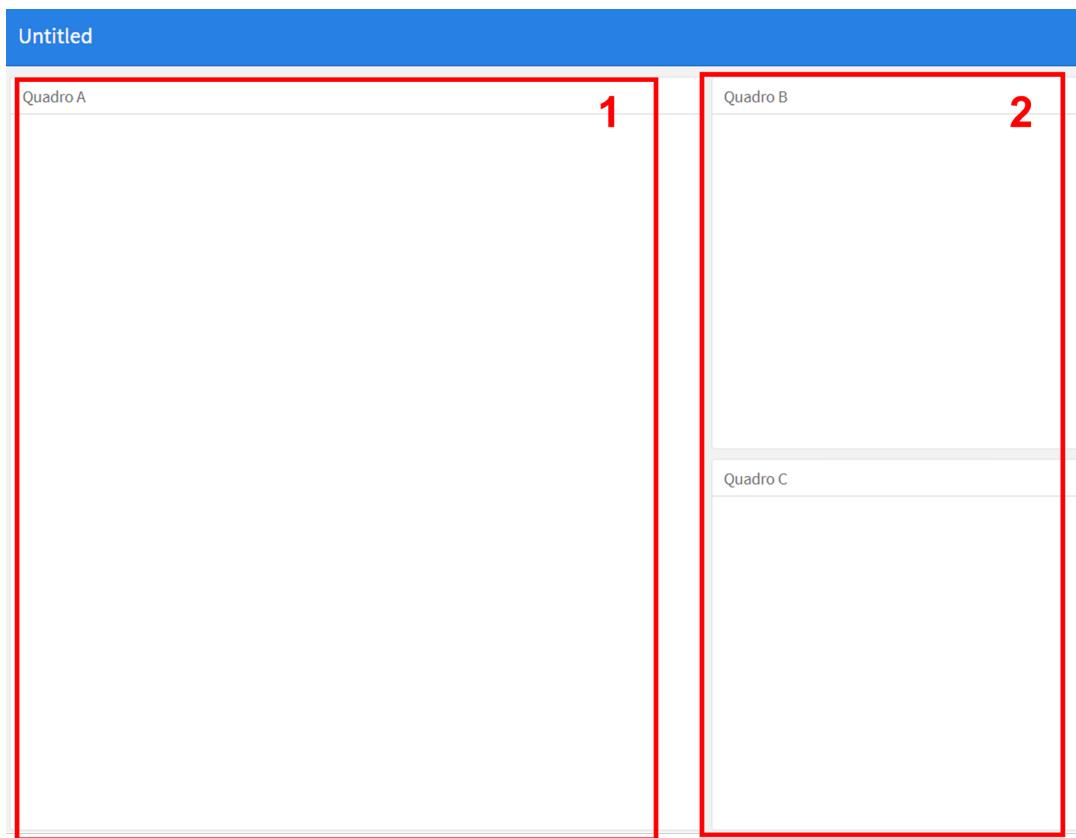
- O *layout* orientado por **colunas** ou
- O *layout* orientado por **linhas**.

Acompanhe as subseções abaixo para construir seu painel da forma que se adeque melhor às suas necessidade de comunicação.

4.1 Layout orientado por colunas

Construir um *dashboard* orientado por colunas significa ordenar o conteúdo do seu painel dentro de quadros orientados por blocos. Nesta etapa, quando fazemos referência às colunas nos referimos a estruturas de organização feito blocos, conforme se pode visualizar na Figura 11.

Figura 11: Painel com orientação em colunas.





Perceba que nesta organização o *layout* é dividido em duas colunas:

1. na primeira tem-se o “Quadro A”, mais largo e mais alto, e
2. na segunda têm-se dois quadros (Quadros B e C), que subdividem a coluna e são mais estreitos.

Para obter um *dashboard* com esta apresentação você precisará inserir o código `orientation: columns` no cabeçalho YAML, localizado no início do seu documento e conforme destacado em vermelho na Figura 12. Observe também na Figura 12, em marrom, que no *script* que estamos montando no *dashboard* cada bloco de texto é definido pela palavra `Column` (coluna em inglês), seguida por - - - hifens na linha seguinte.

Assim, seu *script* deve ser escrito de forma semelhante à Figura 12. Vamos lá e inclua estas etapas no seu RStudio.

Figura 12: Script para criação de um painel orientado por colunas.

```
1 ---  
2 title: "Novo Dashboard"  
3 output:  
4   flexdashboard::flex_dashboard:  
5     orientation: columns  
6     vertical_layout: fill  
7 ---  
8  
9 `r setup, include=FALSE}  
10 library(flexdashboard)  
11  
12  
13 Column {data-width=350}  
14 ---  
15  
16 ### Quadro A  
17  
18 `r}  
19  
20 ---  
21  
22 Column {data-width=350}  
23 ---  
24  
25 ### Quadro B  
26  
27 `r}  
28  
29 ---  
30  
31 ### Quadro C  
32  
33 `r}  
34  
35 ---  
36  
37 |
```

Você deve ter percebido que no exemplo de um modelo de *dashboard* da Figura 9, o *script* apresenta as referências: `### Quadro A`, `### Quadro B` e `### Quadro C`. Essas referências existem para indicar uma divisão dos blocos de textos em quadros. Para esta ação de subdivisão utiliza-se a marcação `###` seguida de um nome para este novo quadro.

Mas e se você necessitar também ajustar a largura de cada coluna? Este tipo de configuração é feita incluindo um novo comando: o `Column {data-width= "350"}.` Nele solicitamos ao R que formate a largura de nosso bloco em um tamanho igual a 350 pixels (350px). Dessa forma, para ajustar a largura da coluna, bastará sempre substituir o valor numérico por um valor desejado.

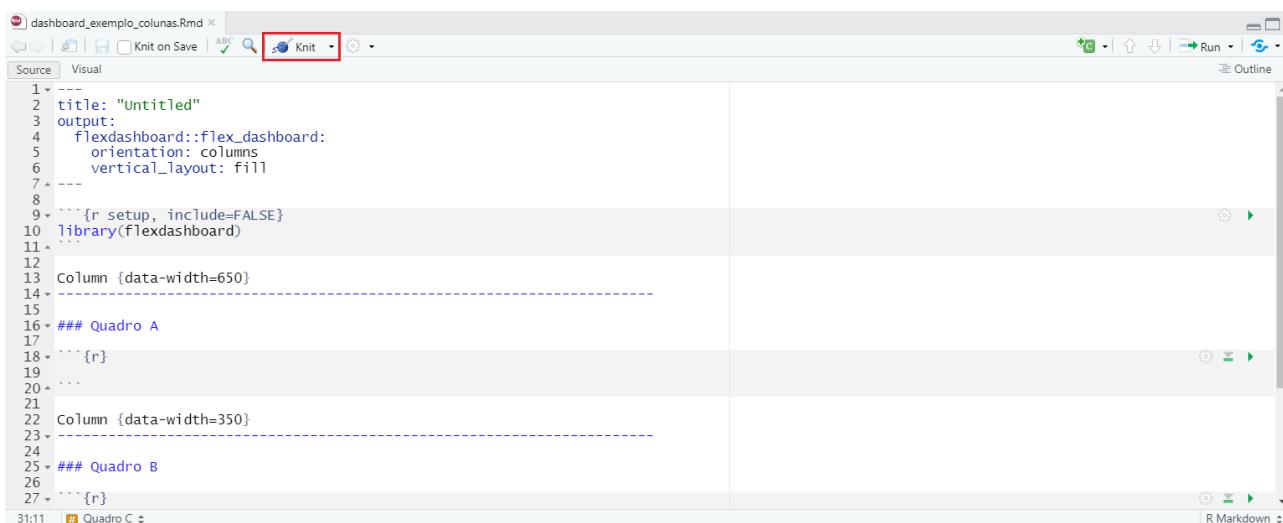


Ao renderizar o seu *script* e transformando-o em um arquivo do tipo `.html` você poderá visualizar e avaliar se o tamanho escolhido para cada bloco é adequado. Você deverá ajustá-lo à sua necessidade! É normal ir testando e verificando o quanto de espaços os blocos estão ocupando na sua tela, afinal o tamanho correto dependerá da quantidade de informação que deseja inserir em cada quadro ou bloco.

Assim, os ajustes irão de acordo com o tamanho da tela que você estiver utilizando. Teste, experimente os diversos tamanhos e escolha o mais adequado para você. Lembre-se que a qualquer momento você poderá retornar ao código e alterar o seu valor em pixel (px).

Pronto. Já temos a nossa base estrutural para o seu primeiro *dashboard*. Agora precisaremos transformá-lo em um painel. Para isso, a partir deste documento atual, geraremos um novo produto que deverá ser um arquivo do tipo `.html`. Ou seja, um arquivo que abrimos em nosso navegador de internet. Para isto, basta selecionar o botão *Knit* na parte superior da aba do *script*. Observe a Figura 13 e localize o botão *Knit* em seu *RStudio*.

Figura 13: Localização do botão *knit* no RStudio.

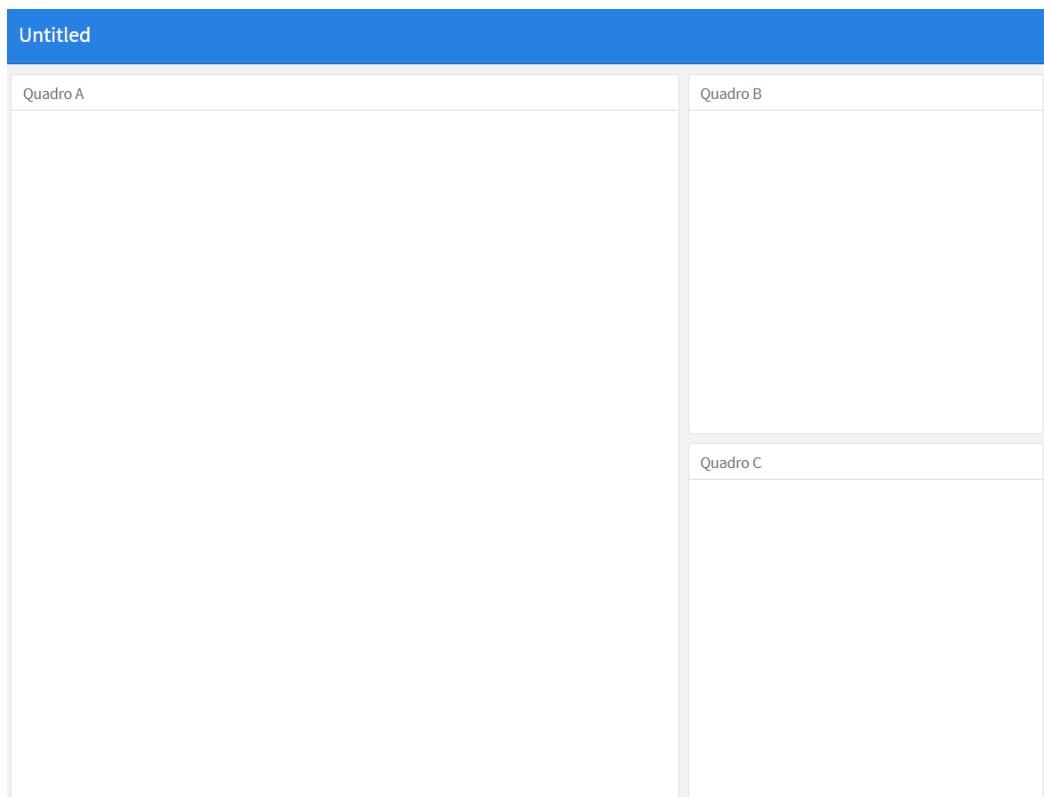


Ao clicar no botão *knit*, você estará fazendo a conversão do arquivo estrutural do *dashboard* que é do tipo `.Rmd` para o tipo `.html`. Esta ação é chamada na linguagem de programação R de *renderização*. Renderizar é uma etapa que torna permanente o trabalho de processamento digital, ou seja, depois que você criou seu *script*, organizou os conteúdos e finalizou todas as etapas de edição, é necessário que eles sejam convertidos em um arquivo final para publicação digital que poderá resultar em um arquivo de imagem, áudio, um documento de texto (`.word`, `.pdf`) ou numa página da internet (`.html`).

A renderização é importante para a transformação do seu *script* em painel. Após a renderização do painel construído até aqui você obterá um resultado como o que visualizamos na Figura 14 abaixo. Veja e observe se o seu *script* também se transformou em um `.html`. Demais! Você já tem uma estrutura de *dashboard* pronta. Parabéns!



Figura 14: Painel renderizado com seu *layout* orientado por colunas.



Deixamos no menu lateral “Arquivos”, do Ambiente Virtual do curso um *script* chamado `dashboard_exemplo_colunas.Rmd` pronto para você editá-lo. Você deve fazer o *download* do material do curso diretamente da plataforma *moodle*. Abra este *script* e percorra o passo a passo desenvolvido nesta seção do curso. Aproveite!

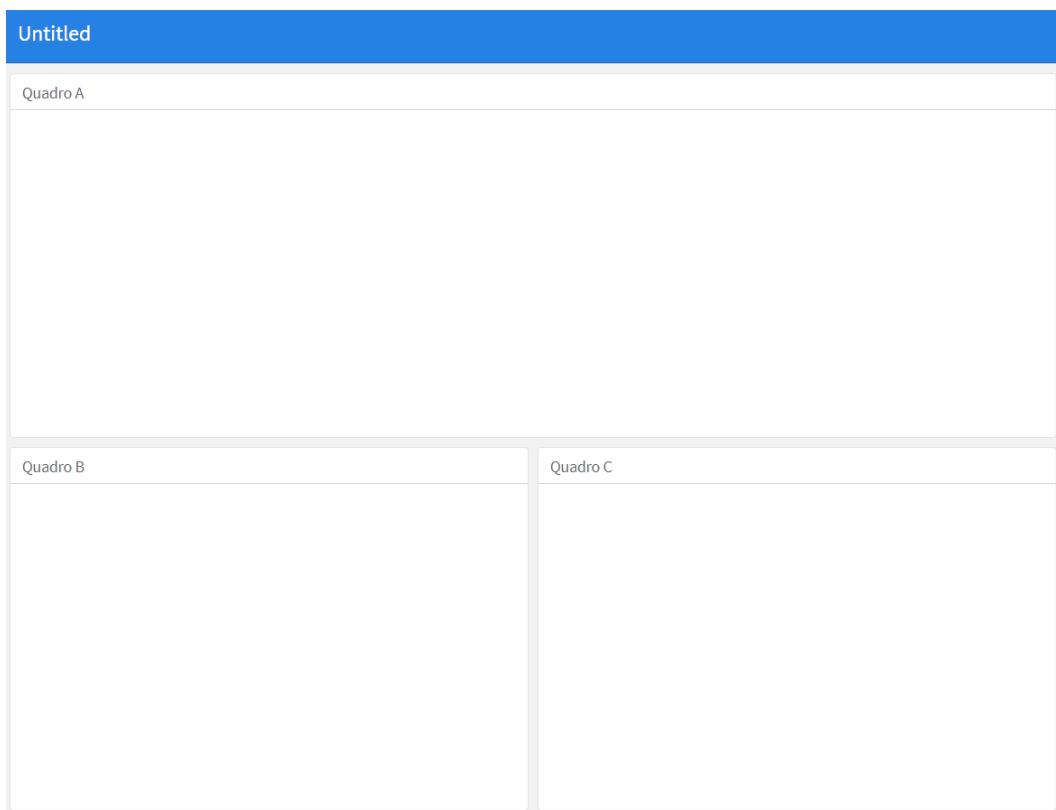


4.2 Layout orientado por linhas

Outra possibilidade de orientação para o seu *dashboard* é em linhas. Com esta organização, ao invés de colunas não utilizaremos a organização de nossos blocos de códigos em linhas verticalmente, como você pode visualizar na Figura 12. Percebeu a diferença?

Aqui na Figura 15 o *layout* possui os mesmos três quadros que já estávamos trabalhando, porém o “Quadro A” ocupa toda a extensão da tela e, logo abaixo dele, os “Quadro B” e “Quadro C” dividem o espaço. Observe a Figura 15:

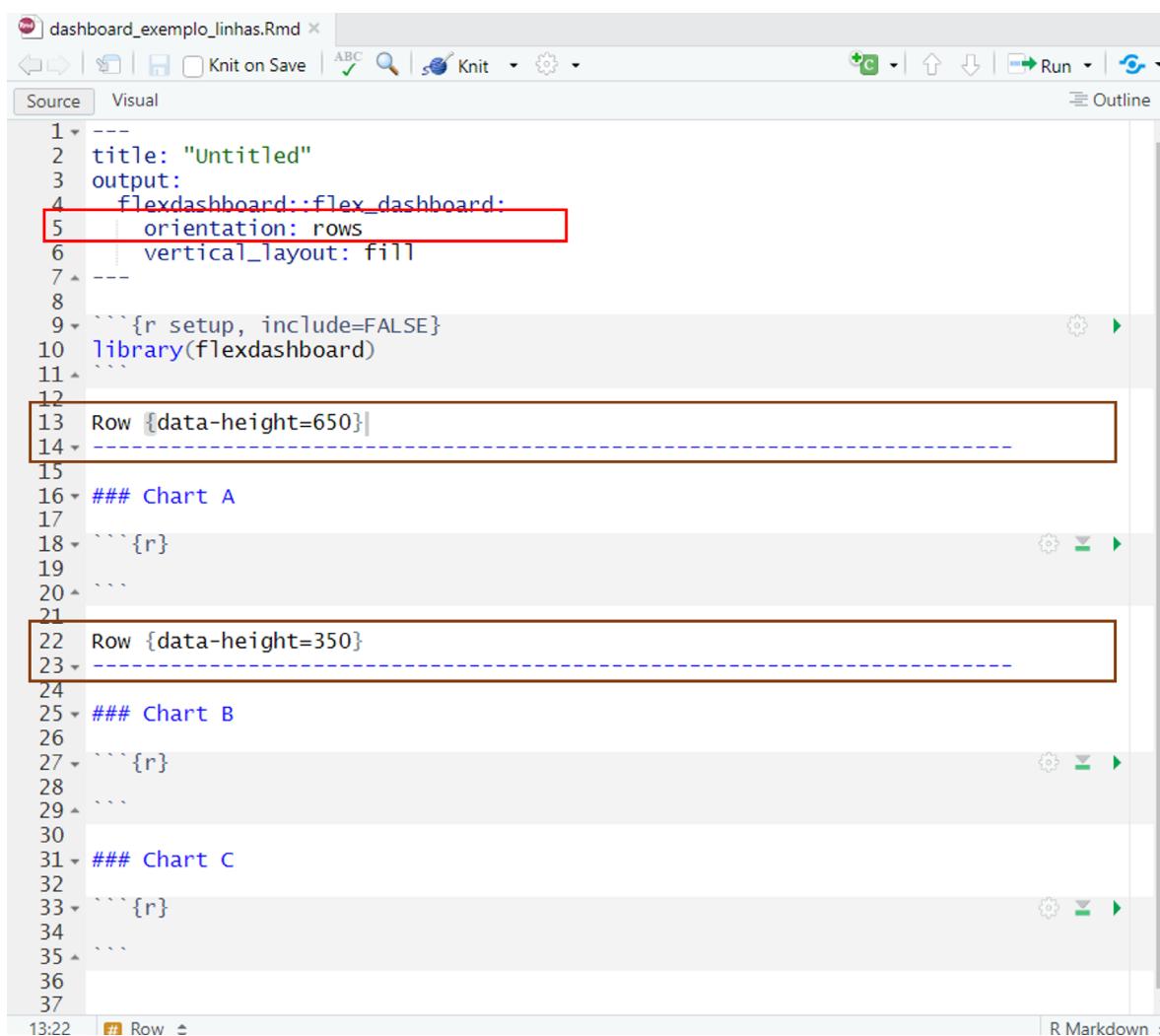
Figura 15: Painel com layout orientado por linhas.



Agora veja o exemplo apresentado na Figura 13. Observe a estrutura de um *script* construído com as substituições que realizamos, em um arquivo de *dashboard* que agora passará a ser orientado por linhas.

Essa configuração pode ser definida inserindo o código `orientation: rows` no cabeçalho YAML, localizado no início do seu documento conforme destacado em vermelho na Figura 13. Perceba que a operação de edição do *layout* é bem simples, e para isso bastará substituir o nome `Column` (coluna) por `Row` (linhas) nos trechos de códigos do seu arquivo, observados nos destaque em marrom (Figura 16). Lembre-se que cada bloco de texto está sendo definido pela palavra `Row` seguida por `- - -` hifens na linha seguinte.

Figura 16: Tela de um *script* para painel orientado por linhas.



```

1 ---  

2 title: "Untitled"  

3 output:  

4   flexdashboard::flex_dashboard:  

5     orientation: rows  

6     vertical_layout: fill  

7 ---  

8  

9 ```{r setup, include=FALSE}  

10 library(flexdashboard)  

11 ````  

12  

13 Row [data-height=650]  

14 - - -  

15  

16 ### Chart A  

17  

18 ```{r}  

19  

20 ````  

21  

22 Row [data-height=350]  

23 - - -  

24  

25 ### Chart B  

26  

27 ```{r}  

28  

29 ````  

30  

31 ### Chart C  

32  

33 ```{r}  

34  

35 ````  

36  

37

```

Neste *Layout* orientado por linhas a principal configuração agora será o ajuste de altura das linhas. Este ajuste poderá ser feito indicando os parâmetros necessários para o código: `Row {data-height=350}`. Aqui você deverá substituir o valor numérico “350” por qualquer valor desejado em formato pixel.

Mas e se você desejar criar novos quadros ou subdividir o quadro de uma linha? Para isso é necessário utilizar a marcação `### (hashtags)` acompanhada do nome deste novo quadro que deseja criar, indicando também o seu tamanho.

Vamos lá. Para praticar em seu arquivo *dashboard* execute os passos que aprendemos nesta subseção.



O RStudio nos permite substituir palavras de forma mais rápida e fácil. Para isso, ele nos oferece um atalho: basta digitar em seu teclado, ao mesmo tempo, as teclas: **CTRL + F**, para que se abra um pequeno painel, conforme na Figura 17, para localização e substituição de termos. É uma espécie de “busca” (*find*, em inglês).

Figura 17: Painel de busca no RStudio.



Agora vamos fazer a substituição da palavra `Column` para `Row`:

- na primeira caixa (item **1** em vermelho, onde está a palavra *Find*), digite o termo “`Column`”,
- na segunda caixa (item **2** em vermelho, onde está a palavra *Replace*), digite o termo “`Row`”, e
- em seguida, selecione o botão “All” (item **3** em vermelho), que irá substituir todas as ocorrências em seu documento para estes termos.

Foi fácil, não é mesmo?! Os termos já foram substituídos.

Pronto. Agora selecione o botão *Knit* na parte superior da aba do *script* e renderize o seu código para transformá-lo em *dashboard* gerando um arquivo do tipo `.html`.

5. Definindo o conteúdo para o dashboard

Agora que já construímos a estrutura básica do nosso *dashboard* e aprendemos a renderizá-lo, podemos incluir o seu conteúdo com as informações que queremos publicizar. Nesta seção do curso você aprenderá a incorporar todos os elementos que permitem comunicar seus dados criando um painel com indicadores de saúde: textos, gráficos, imagens, tabelas, mapas, botões com índices, entre outros.

Esta é uma etapa fundamental, pois os dados apresentados devem estar alinhados aos objetivos da comunicação, de forma a determinar quais dados são de importância para a vigilância em saúde e para quem estamos publicando essas informações que serão organizadas no painel de indicadores.

Para construirmos o conteúdo do nosso painel, iremos seguir esta etapa em cinco partes:

- Parte 1: conhecer os objetivos do painel;
- Parte 2: organizar os dados necessários;
- Parte 3: inserir caixa de valores;
- Parte 4: inserir gráficos estáticos e interativos;
- Parte 5: inserir tabelas estáticas e interativas.

Os códigos utilizados nessas etapas resultarão em um arquivo completo para você construir um *dashboard* utilizando qualquer tema.



Lembre-se que nesta etapa a sua tarefa principal é organizar as informações relevantes de forma clara, simples e resumida.

5.1 Conhecendo os objetivos do painel

Vimos até aqui que os painéis de indicadores são utilizados para facilitar o monitoramento e o acompanhamento rotineiro de informações de saúde que são relevantes. Com a implementação do *dashboard* poderemos verificar a existência de mudanças de cenários epidemiológicos, podendo avaliá-las no tempo e espaço (em regiões específicas do seu município, estado ou qualquer outra).

Assim, os painéis são resultado da incorporação de processos sistemáticos e contínuos de acompanhamento de indicadores na vigilância em saúde, podendo apoiar a execução de políticas e ações dos serviços de saúde. Com eles podemos fornecer informações em tempo oportuno para subsidiar tomadas de decisão, encontrar soluções e reduzir de problemas.

Como exercício prático neste curso você construirá **um painel para avaliação de dengue no Estado de Rosas** (fictício). Para isso você precisará organizar informações que possam descrever a situação das notificações do estado entre os anos 1993 e 2012 para a sociedade civil de todo o estado. Vamos lá?!

5.2 Organizando os dados necessários

Antes de inserir os elementos de visualização, é necessário definir algumas configurações. Isto inclui instalar e carregar os pacotes que serão utilizados, importar os dados necessários e executar as funções específicas para o seu *dashboard*. Faremos um painel com *layout* orientado por linhas (*rows*).

Como iremos construir um painel de avaliação da situação da dengue no Estado de Rosas (fictício), algumas análises precisam ser produzidas a partir do banco de dados. Assim, durante toda a nossa confecção do painel, utilizaremos os dados exportados do Sinan Net do Estado de Rosas: é o arquivo de nome `{NINDNET.dbf}` que está disponível no menu lateral “Arquivos” do Ambiente Virtual do curso. CLIQUE e faça *download* para seguirmos com as análises.

Vamos lá. Primeiramente, você deverá checar se o cabeçalho `YAML` do seu arquivo estrutural de *dashboard* contém todos os elementos apresentados na Figura 16 deste curso. Caso não esteja, adeque o seu documento!

Com o arquivo semelhante ao visualizado na Figura 16, você já poderá escrever os comandos no seu `RStudio` para a produção do *dashboard*. Insira todo o conteúdo dos *scripts* que utilizaremos aqui em seu `Rstudio`, somente a partir da **linha 10** do arquivo, conforme a Figura 18.

Figura 18: Painel de busca no RStudio.



```

Source Visual
1 --- 
2 title: "Untitled"
3 output:
4   flexdashboard::flex_dashboard:
5     orientation: rows
6     vertical_layout: fill
7 ---
8
9 ````{r setup, include=FALSE}
10 library(flexdashboard) ← Red arrow here
11
12
13 Row {}
14
15

```

Lembre-se que os *scripts* devem ser escritos dentro dos blocos de códigos ou *chunks* no R (área de fundo acinzentado, delimitado por três aspas) no arquivo do painel.

Principais funções do pacote `lubridate` que utilizaremos aqui:

- `ymd`: converte um valor em uma variável do tipo `Date` quando os valores estão dispostos na ordem ANO-MÊS-DIA (AAAA-MM-DD),
- `epiweek()`: converte uma data no valor correspondente da semana epidemiológica,
- `epiyear()`: converte uma data no valor correspondente do ano epidemiológico,
- `month()`: extrai apenas o valor numérico do mês de uma determinada data,
- `as.numeric()`: converte valores em variáveis do tipo numérico (`numeric`).

Vamos lá. Para incorporar os dados de dengue ao nosso painel realizaremos os seguintes passos:

1. importaremos o banco de dados `{NINDNET.dbf}` utilizando a função `read.dbf()` do pacote `foreign`,
2. selecionaremos as notificações de casos referentes ao código CID10 = A90 (Dengue) utilizando a função `filter()` do pacote `dplyr`, e
3. transformaremos as variáveis que possuem datas (dias, semanas e ano) epidemiológicas para o formato necessário aninhando funções do pacote `lubridate`, dentro da função `mutate()`, também pertencente ao pacote `dplyr`.

Conforme a Figura 15, escreva todos os códigos do *script* abaixo no *chunk* do seu RStudio:

```
# Carregando os pacotes necessários
require(flexdashboard)
require(foreign)
require(tidyverse)
require(lubridate)

# Importando o banco de dados {NINDNET.dbf} com a função `read.dbf()` do pacote `foreign`
# armazenando os dados de origem em um objeto {nindi} do tipo "data.frame"
nindi <- read.dbf(file = 'Dados/NINDINET.dbf')

# Armazenando apenas dados filtrados com o agravo A90 em novo objeto {dengue}
dengue <- nindi |>
  filter(ID_AGRAVO == 'A90') |>
  mutate(
    DT_SIN_PRI = ymd(DT_SIN_PRI),
    sem_epi = epiweek(DT_SIN_PRI),
    ano_epi = epiyear(DT_SIN_PRI),
    mes = month(DT_SIN_PRI),
    NU_ANO = as.numeric(NU_ANO)
  )
```

Após inserir os comandos em seu *chunk*, você deverá obter um arquivo em seu RStudio semelhante ao que visualizamos na Figura 19 abaixo.



Figura 19: Visualização da tela de script de construção do dashboard.

The screenshot shows the RStudio interface with the 'Source' tab selected. The code in the editor is as follows:

```
1 ---  
2 title: "Painel de Dengue"  
3 output:  
4   flexdashboard::flex_dashboard:  
5     orientation: rows  
6     vertical_layout: fill  
7 ---  
8 ---  
9 ```{r setup, include=FALSE}  
10 # Carregando os pacotes necessários  
11 require(flexdashboard)  
12 require(foreign)  
13 require(tidyverse)  
14 require(lubridate)  
15  
16 # Importando o banco de dados com a função `read.dbf()` do pacote `foreign`  
17 nindi <- read.dbf(file = '../Dados/NINDINET.dbf')  
18  
19 # Armazenando apenas dados para dengue  
20 dengue <- nindi |>  
21   filter(ID_AGRAVO == 'A90') |>  
22   mutate(  
23     DT_SIN_PRI = ymd(DT_SIN_PRI),  
24     sem_epi = epiweek(DT_SIN_PRI),  
25     ano_epi = epiyear(DT_SIN_PRI),  
26     mes = month(DT_SIN_PRI),  
27     NU_ANO = as.numeric(NU_ANO)  
28   )  
29  
30 ---  
31 ---  
32 |
```

Se você conseguiu obter um arquivo como o apresentado na Figura 19, rode o *script*. Você perceberá que foi criado um *data.frame* {dengue} com 12.781 casos e 65 variáveis. Com esta ação já possuímos os dados necessários, agora vamos utilizá-los para gerar os elementos que irão compor o nosso *dashboard*.

Atenção



Caso tenha encontrado dificuldade de chegar a um arquivo com os *scripts* que utilizamos, não se preocupe e continue no curso!

Deixamos pronto para você um arquivo de estudo com todos os elementos que aplicamos nesta subseção: o `dashboard_parte1_configurando_o_ambiente.Rmd`. Você poderá encontrá-lo acessando o menu lateral “Arquivos”, do Ambiente Virtual do curso e fazer o *download*.

5.3 Inserindo caixas de valores

Para tornar a visualização de dados intuitiva para a população civil de Rosas que acessará o painel que estamos produzindo é preciso torná-la atrativa, destacando as principais informações que serão fornecidas.

Um dos recursos mais utilizados em *dashboards* para isso são as caixas de valores, ou botões, ou *valuebox* (em inglês) como as apresentadas na Figura 20. Elas possibilitam a divulgação de um ou mais valores simples, como o número absoluto ou relativo de casos ou óbitos. Dessa forma, essas informações são destacadas.

Observe na Figura 20 como ficam visualizadas as caixas de valores ou *valuebox* em diferentes cores para serem utilizadas em seu *dashboard*.

Figura 20: Exemplos de cores para as caixas de valor (*valuebox*).



Para incluir uma caixa como estas da Figura 16, no R podemos utilizar a função: `valueBox()`. Ela nos apoiará a exibir valores simples, incluindo seu título e também um ícone ou imagem associado, se assim desejar.

Para a personalização do conteúdo das valuebox precisaremos parametrizar os seus seguintes argumentos da função `valueBox()`:

- `value`: o valor a ser exibido no formato numérico,
- `caption`: o título a ser exibido no formato texto (`character`), e
- `color`: cor de fundo da caixa.

Já para personalizar a cor de fundo das *valuebox*, precisamos escolher entre cinco cores pré-determinadas utilizando os argumentos da função `valueBox()`:

- `primary`: para escolher o azul,
- `info`: para escolher o roxo,
- `success`: para escolher o verde,
- `warning`: para escolher o laranja, e
- `danger`: para escolher a cor vermelha.

Você também pode inserir imagens ou ícones (*icon*, em inglês) para tornar o seu *dashboard* mais intuitivo. Veja alguns exemplos na Figura 21 que podemos utilizar. Estes são ícones relacionados à área da saúde utilizando a partir da plataforma [font awesome](#):

Figura 21: Exemplos de ícones que podem ser adicionados ao *dashboard*.

Medical Icons

 ambulance	 h-square	 heart	 heart-o
 heartbeat	 hospital-o	 medkit	 plus-square
 stethoscope	 user-md	 wheelchair	 wheelchair-alt

Agora vamos inserir três caixas de valores em nosso *dashboard* com dados que contenham o total de casos notificados e o total de curas e óbitos de dengue, todos fazendo referência ao ano de 2012 no Estado de Rosas. Siga o passo a passo a seguir:

1. Primeiro, observe os *scripts* abaixo em que realizamos o cálculo do “total de casos notificados de dengue” e replique-os no *chunk* do seu arquivo `.Rmd`:

```
# Criando objeto numérico com o total de casos de dengue
total_casos <- dengue |>

# Filtrando pelo ano epidemiológico de 2012 com a função filter()
filter(ano_epi == 2012) |>

# Contando o número de linhas (registros) com a função nrow()
nrow()
```

2. Segundo, precisamos incluir o valor calculado no passo 1 em uma caixa de valores de cor azul. Para isso inserimos o código abaixo. Observe que para a função `valueBox()`, escolhemos:

o objeto que será utilizado para demonstrar o valor (`total_casos`),

- a legenda (`caption = "Casos notificados"`),
- a cor (`color = "primary"`), e
- o ícone (`icon = "fa-exclamation-circle"`).

Replique o código abaixo no *chunk* do seu arquivo `.Rmd`:

```
# Inserindo a caixa de valores
valueBox(total_casos,
          caption = "Casos notificados",
          color = "primary",
          icon = "fa-exclamation-circle")
```

3. Agora observe que uniremos os passos 1 e 2 em um único *chunk* para reduzir o tamanho do nosso código. Nomearemos este bloco de **BLOCO 1**.

Acompanhe o *script* abaixo e replique o código no *chunk* do seu arquivo `.Rmd`:

```
# Criando objeto numérico com o total de casos de dengue
total_casos <- dengue |>

# Filtrando pelo ano epidemiológico de 2012 com a função filter()
filter(ano_epi == 2012) |>

# Contando o número de linhas (registros) com a função nrow()
nrow()

# Inserindo a caixa de valores
valueBox(total_casos,
          caption = "Casos notificados",
          color = "primary",
          icon = "fa-exclamation-circle")
```

- 4.** Agora, de forma semelhante iremos criar uma caixa de valor ou botão que irá armazenar o nosso cálculo do “total de casos de dengue com cura em 2012” nomeando-o como **BLOCO 2**.

Acompanhe o *script* abaixo e replique o código no *chunk* do seu arquivo **.Rmd**:

```
# Criando objeto numérico com o total de casos de dengue com classificação final de "cura"
total_cura <- dengue |>

# Filtrando pelo ano epidemiológico de 2012 com a função filter()
filter(ano_epи == 2012) |>

# Filtrando pela classificação final pelo código de cura (1) com a função filter()
filter(CLASSI_FIN == 1) |>

# Contando o número de linhas (registros) com a função nrow()
nrow()

# Inserindo a caixa de valores
valueBox(total_cura ,
          caption = "Cura",
          color = "success",
          icon = "fa-plus-square")
```

5. No quinto passo vamos inserir mais uma caixa de valor ou botão com os cálculos do “total de óbitos em 2012” e a nomearemos de **BLOCO 3**.

Acompanhe o *script* abaixo e replique o código no chunk do seu arquivo **.Rmd**:

```
# Criando objeto numérico com o total de casos de dengue com classificação final de "óbito"
total_obitos <- dengue |>

# Filtrando pelo ano epidemiológico de 2012 com a função filter()
filter(ano_epи == 2012) |>

# Filtrando pela classificação final pelo código de óbito (2) com a função filter()
filter(CLASSI_FIN == 2) |>

# Contando o número de Linhas (registros) com a função nrow()
nrow()

# Inserindo a caixa de valores
valueBox(total_obitos,
          caption = "Óbitos",
          color = "warning",
          icon = "fa-heartbeat")
```



6. Pronto, já possuímos todos os botões necessários para o nosso painel. Agora é preciso que você inspecione seu *script* e observe se você obteve em seu computador um arquivo *.Rmd* como o apresentado na Figura 22.

Figura 22: Visualização do *script* do dashboard de dengue com edição de ícones.

```
Source | Visual
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 Row|
34 #
35
36 ###
37
38 `'{r}
39 # BLOCO 1
40 # Criando objeto numérico com o total de casos de dengue
41 total_casos <- dengue |>
42
43 # Filtrando pelo ano epidemiológico de 2012 com a função filter()
44 filter(ano_epi == 2012) |>
45
46 # Contando o número de linhas (registros) com a função nrow()
47 nrow()
48
49 # Inserindo a caixa de valores
50 valueBox(total_casos,
51           caption = "Casos notificados",
52           color = "primary",
53           icon = "fa-exclamation-circle")
54 ...
55
56 ###
57
58 `'{r}
59 # BLOCO 2
60 # Criando objeto numérico com o total de casos de dengue com classificação final de "cura"
61 total_cura <- dengue |>
62
63 # Filtrando pelo ano epidemiológico de 2012 com a função filter()
64 filter(ano_epi == 2012) |>
65
66 # Filtrando pela classificação final pelo código de cura (1) com a função filter()
67 filter(CLASSI_FIN == 1) |>
68
69 # Contando o número de linhas (registros) com a função nrow()
70 nrow()
71
72 # Inserindo a caixa de valores
73 valueBox(total_cura ,
74           caption = "Cura",
75           color = "success",
76           icon = "fa-plus-square")
77 ...
78
79 ###
80
81 `'{r}
82 # BLOCO 3
83 # Criando objeto numérico com o total de casos de dengue com classificação final de "óbito"
84 total_obitos <- dengue |>
85
86 # Filtrando pelo ano epidemiológico de 2012 com a função filter()
87 filter(ano_epi == 2012) |>
88
89 # Filtrando pela classificação final pelo código de óbito (2) com a função filter()
90 filter(CLASSI_FIN == 2) |>
91
92 # Contando o número de linhas (registros) com a função nrow()
93 nrow()
94
95 # Inserindo a caixa de valores
96 valueBox(total_obitos,
97           caption = "Óbitos",
98           color = "warning",
99           icon = "fa-heartbeat")
100 ...
101
102
103
104
```

Observe que a Figura 22 possui um cabeçalho em YAML no início do arquivo `.Rmd` que configura o *dashboard*, com os **BLOCO 1**, **BLOCO 2** e **BLOCO 3** que criamos separados pelo indicador `###` para indicar as subdivisões do painel. Não definiremos nenhum detalhe de altura das linhas neste momento.

7. O sétimo passo será renderizar todos os botões que criamos, para que visualizemos como ficaria o nosso painel. Observe na Figura 23 como ficará o resultado quando clicarmos no botão `knit` do RStudio e renderizarmos o script para publicar o *dashboard* no formato `.html`.

Figura 23: Tela visualizada após renderizar o código da Figura 16.



O resultado é muito interessante, não é mesmo?! Agora, veja se conseguiu obter esse resultado. Torna-se crucial que você consiga executar o passo a passo para inserirmos os demais elementos gráficos do nosso painel da dengue do Estado de Rosas.

Atenção



Caso tenha encontrado dificuldade de chegar a um arquivo com os *scripts* que utilizamos, não se preocupe e continue no curso!

Deixamos para você um arquivo com todos os elementos que aplicamos nesta subseção prontos: o `dashboard_parte2_caixa_de_valores.Rmd`. Encontre-o acessando o menu lateral “Arquivos”, do Ambiente Virtual do curso e faça o seu *download*. Compare os arquivos e veja o que pode ter feito errado e corrija. Estamos aqui para aprender, não se preocupe.



Para inserir um ícone dentro de uma *valuebox* você deverá referenciar utilizando um código com a seguinte estrutura: `icon = "prefixo-nomedoicone"`. Ou seja, indique a palavra `"fa"` como **prefixo**, seguida de hífen (-) e então o nome do **ícone**, todos entre aspas. Por exemplo, um ícone de coração deve ser indicado pelo argumento `"icon = fa-heart"`.

Todos esses ícones já vêm instalado junto ao pacote `flexdashboard`. Experimente também a personalização dos ícones; seguem algumas fontes que também poderão ser utilizadas no seu R:

- [Font awesome](#) utilizando o prefixo `"fa-`
- [Ionicons](#) utilizando o prefixo `"ion-`
- [Bootstrap Glyphicons](#) utilizando o prefixo `"glyphicon-`

5.4 Inserindo gráficos estáticos e interativos

Como forma de tornar uma análise acessível, simplificando avaliações complexas, costumamos utilizar visões gráficas. Estas são as visualizações mais utilizadas para produzir informações em um relatório de vigilância em saúde, não é mesmo?! Aqui no nosso *dashboard* não será diferente. Daremos destaque a uma visão gráfica para apresentar análises de tendência dos casos de dengue e permitir a observação do comportamento do agravo ao longo dos anos (1993-2012).

Os gráficos podem ser facilmente inseridos nos blocos de códigos (*chunks*) utilizando diversos pacotes do R para criação de gráficos. Nesta etapa iremos produzir dois tipos de gráficos: os gráficos estáticos e os interativos. Aqui optaremos por utilizar alguns gráficos produzidos no curso de “Visualização de dados de interesse para a vigilância em saúde” utilizando especificamente o pacote `ggplot2`.

5.4.1 Gráficos estáticos

Chamamos de gráficos estáticos aqueles que representam apenas uma imagem estática, que não possuem interação como cliques ou filtros do usuário, ou seja, sem recursos de animação para analisar os dados. Para nosso *dashboard* criaremos apenas um gráfico em que seja possível visualizar o número de casos de dengue para o Estado de Rosas por semana epidemiológica em diferentes anos. Vamos lá!

Para gerar esta visão gráfica você precisará digitar todos os códigos do *script* abaixo no *chunk* do seu arquivo `.Rmd` em seu RStudio:

```
#utilizando o objeto {dengue} que armazena os casos de dengue exportados do {NINDNET.dbf} de Rosas
dengue |>

# Contando número de casos por ano e semana epidemiológica
count(ano_epi, sem_epi) |>

# Plotando visualização de gráfico por ano e semana epidemiológica
ggplot(aes(
  x = sem_epi,
  y = n,
  color = factor(ano_epi)
)) +
  geom_line() +
  geom_point() +
  theme_minimal() +
  xlab("\nSemana epidemiológica") +
  ylab("") +
  scale_color_discrete("Ano") +
  scale_x_continuous(breaks = c(1, seq(5, 50, 5)))
```

Pronto! Criamos o nosso gráfico de casos notificados. Verifique se você inseriu todo o código acima em seu **RStudio** e obteve um *script* como o apresentado na Figura 24. Caso não tenha conseguido, reescreva as linhas de código no *chunk* do seu arquivo.

Figura 24: Tela de visualização do script com a criação do gráfico estático.

```

102 Row {data-height=650}
103 ~~~{r, fig.width=7, fig.height=5}
104
105 ### Casos de dengue por semana epidemiológica
106
107 dengue |>
108
109 # Contando número de casos por ano e semana epidemiológica
110 count(ano_epi, sem_epi) |>
111
112 # Plotando visualização de gráfico por ano e semana epidemiológica
113 ggplot(aes(
114   x = sem_epi,
115   y = n,
116   color = factor(ano_epi)
117 )) +
118
119 # Adicionando linhas
120 geom_line() +
121
122 # Adicionando pontos
123 geom_point() +
124
125 # Aplicando novo tema para o gráfico
126 theme_minimal() +
127
128 # Adicionando rótulo para o eixo x
129 xlab("\nsemana epidemiológica") +
130
131 # Adicionando rótulo para o eixo y
132 ylab("") +
133
134 # Definindo o título da legenda
135 scale_color_discrete("Ano") +
136
137 # Definindo o intervalo de valores dos rótulos do eixo x
138 scale_x_continuous(breaks = c(1, seq(5, 50, 5)))
139
140 ~~~
141

```

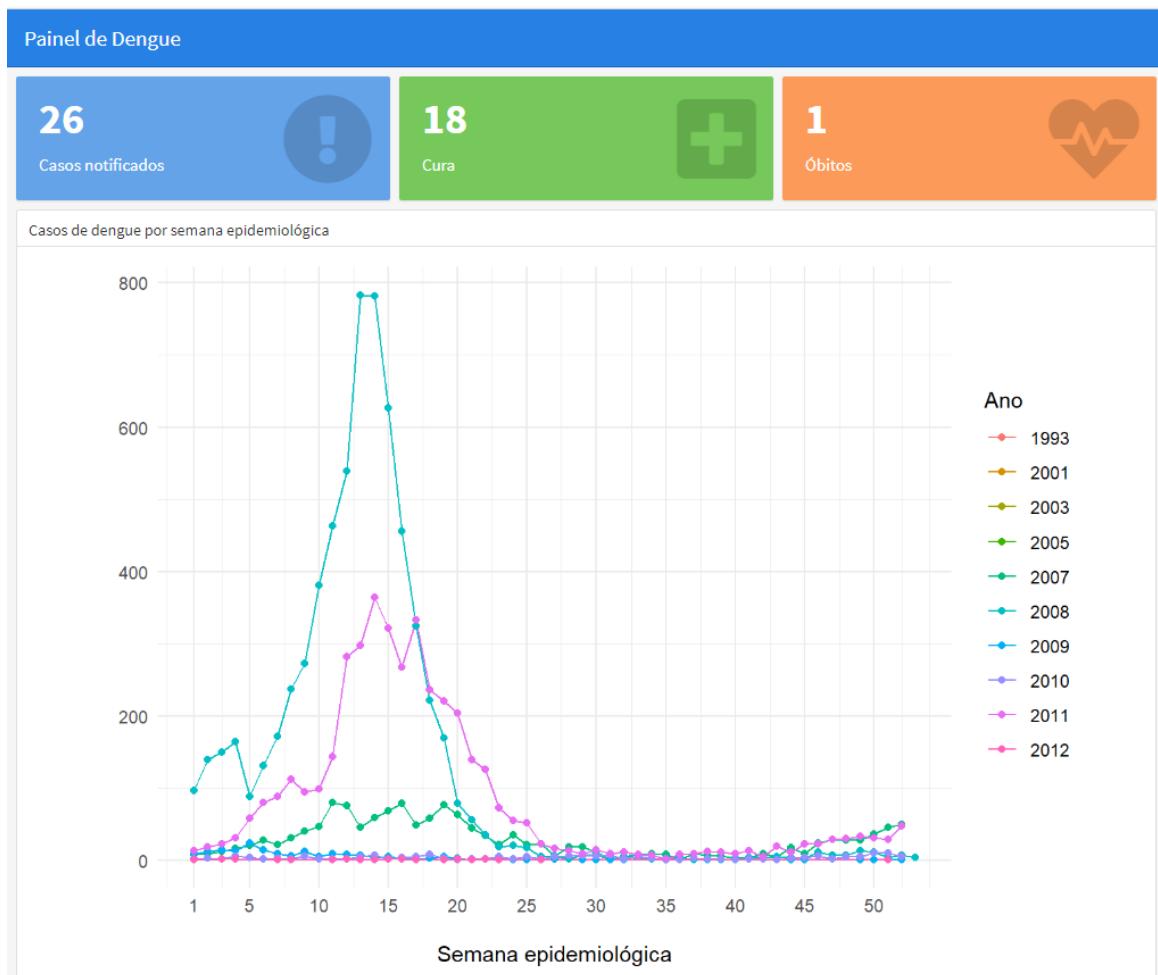
Observe na Figura 24 que, antes do bloco de código em R, as linhas 102-103 do arquivo `.Rmd` especificam a criação de um novo quadro em seu *dashboard*. Desta forma, o gráfico gerado irá aparecer em um quadro próprio, e não no mesmo quadro em que os ícones incluídos anteriormente aparecem.

Outro detalhe importante é que precisamos nomear os quadros (blocos) onde nosso gráfico será posicionado. Para isso, você deve utilizar as `###(hashtags)` e em seguida já incluir o título do quadro. Neste caso, denominamos o nosso quadro de: “Casos de Dengue por Semana Epidemiológica”. Você poderá editar este título da forma que desejar, mas lembre-se que ele deve ser conciso e apresentar de forma clara os dados gráficos que serão visualizados.

Agora vamos transformar nosso trabalho em um *dashboard*. Para isso, renderize o seu arquivo do tipo `.Rmd` que estamos trabalhando, tornando-o um arquivo do tipo `.html`. Clique no botão `knit` do `RStudio` e renderize o *script*! Você deverá obter um arquivo como visualizamos na Figura 25. Veja:



Figura 25: Dashboard de Dengue com a inclusão do gráfico estático com título.



Está ficando bem bonito, não é mesmo? E é você que está fazendo. Vamos em frente tornar o nosso painel de indicadores ainda mais intuitivo para a população de Rosas.



Atenção

Caso tenha encontrado dificuldade de chegar a um arquivo com os *scripts* que utilizamos, não se preocupe e continue no curso!

Deixamos para você um arquivo com todos os elementos que aplicamos nesta subseção prontos: o `dashboard_parte3_grafico_estatico.Rmd`. Encontre-o acessando o menu lateral “Arquivos”, do Ambiente Virtual do curso e faça o seu *download*.



Todos os gráficos que criamos em um arquivo `.Rmd` (que estamos desenvolvendo) são arquivos gráficos no formato de imagem do tipo `.png`. Os *dashboards* por padrão preencherão todo o espaço livre com a imagem gerada mantendo a proporção entre altura e largura adaptando esta visualização para tamanhos de telas em que os *dashboards* são gerados (no nosso caso em uma tela para navegador de internet).

Será importante que você saiba que ao abrir o *dashboard* em uma tela de celular, por exemplo, haverá distorção da imagem, exigindo adequação. Desta forma, existem alguns argumentos que podem ser adicionados ao código do painel que ajudam a fixar o tamanho da imagem:

- o `fig.width`: largura da figura (em polegadas), e
- o `fig.height`: altura da figura (em polegadas).

Estes argumentos devem ser adicionados no início do trecho do código em `R` conforme poderemos visualizar na Figura 26.

**Figura 26: Tela de visualização do script com a criação do gráfico
ajustando com os argumentos `fig.width` e `fig.height`.**

```
32 ## Casos de dengue por semana epidemiológica
33
34 ```{r, fig.width=7, fig.height=5}
35 dengue |>
36 # Contando número de casos por ano e semana epidemiológica
37 count(ano_epi, sem_epi) |>
38 # Plotando visualização de gráfico por ano e semana epidemiológica
39 ggplot(aes(x = sem_epi, y = n, color = factor(ano_epi))) +
40 # Adicionando linhas
41 geom_line() +
42 # Adicionando pontos
43 geom_point() +
44 # Aplicando novo tema para o gráfico
45 theme_minimal() +
46 # Adicionando rótulo para o eixo x
47 xlab("\nSemana epidemiológica") +
48 # Adicionando rótulo para o eixo y
49 ylabel("") +
50 # Definindo o título da legenda
51 scale_color_discrete("Ano") +
52 # Definindo o intervalo de valores dos rótulos do eixo x
53 scale_x_continuous(breaks = c(1,seq(5,50,5)))
54 ```
55
```



Em geral, para gerar em valores de largura e altura ideais para a necessidade do seu *dashboard* fazemos testes e opta-se pelas melhores opções, após muitas tentativas e erros. Teste os diversos formatos de texto, para isso você pode enviar o arquivo do tipo `.html` que contém o seu painel para seu e-mail e experimente em telas diferentes; sugerimos que os testes sejam feitos no navegador do computador. Pratique rescrevendo o *script* que aprendeu a partir da Figura 26 e observe o resultado.

5.4.2 Gráficos interativos

Os gráficos interativos são mais flexíveis aos diversos formatos de tela. Eles são os elementos mais valiosos para a criação de painéis porque permitem a interação do usuário com as informações presentes no gráfico, seja por meio dos filtros, barras de rolagens, cliques ou detalhamento de dados. A partir desta subseção você será capaz de gerar gráficos interativos utilizando o pacote `plotly` que permite criar gráficos por meio da biblioteca de gráficos JavaScript.



A linguagem **JavaScript** é uma das bases do desenvolvimento *web*. Somado à linguagem `HTML` e à `CSS` são responsáveis por boa parte dos sites da internet. Juntas essas linguagens permitem desenvolver páginas completas de internet, com *layouts* robustos e totalmente interativos.

É comum a utilização de *plug-ins*, *frameworks* e bibliotecas da linguagem `JavaScript` para aperfeiçoar ou desenvolver diversos elementos interativos, por exemplo, para páginas da *web*.

Nesta seção iremos transformar o gráfico criado anteriormente (estático) em interativo. Lembre-se que o gráfico anterior foi criado a partir do objeto `{dengue}`. Acompanhe o passo a passo e siga-os em seu RStudio.

1. Primeiro, iremos realizar o carregamento do pacote `plotly` em um bloco de código do seu arquivo `.Rmd` junto ao trecho onde outros pacotes são carregados:

```
require(plotly)
```

2. Segundo, vamos salvar o gráfico estático criado na seção 5.4.1 em um objeto que chamaremos de `{grafico_1}`, e
3. utilizaremos a função `ggplotly()` do pacote `plotly` com o objeto `{grafico_1}` sendo seu argumento.

Atenção



A função `ggplotly()` funciona somente com objetos de gráficos gerados pela função `ggplot()`. Para inserir um argumento não é necessário inserir " " (aspas) no nome do objeto gráfico.

Para gerar o gráfico interativo você precisará digitar todos os códigos do *script* abaixo no *chunk* do seu arquivo `.Rmd` em seu RStudio:

```

# Criando o objeto `grafico_1`
grafico_1 <- dengue |>

# Contando número de casos por ano e semana epidemiológica
count(ano_epsi, sem_epsi) |>

# Plotando visualização de gráfico por ano e semana epidemiológica
ggplot(aes(
  x = sem_epsi,
  y = n,
  color = factor(ano_epsi)
)) +
  geom_line() +
  geom_point()

# Adicionando linhas
theme_minimal() +
  xlab("\nSemana epidemiológica") +
  ylab("") +
  scale_color_discrete("Ano") +
  scale_x_continuous(breaks = c(1, seq(5, 50, 5))) +
  ggplotly(grafico_1)

```

Agora observe se o seu arquivo RStudio se assemelha à Figura 27. O seu *chunk* deve estar assim para produção do gráfico dinâmico:



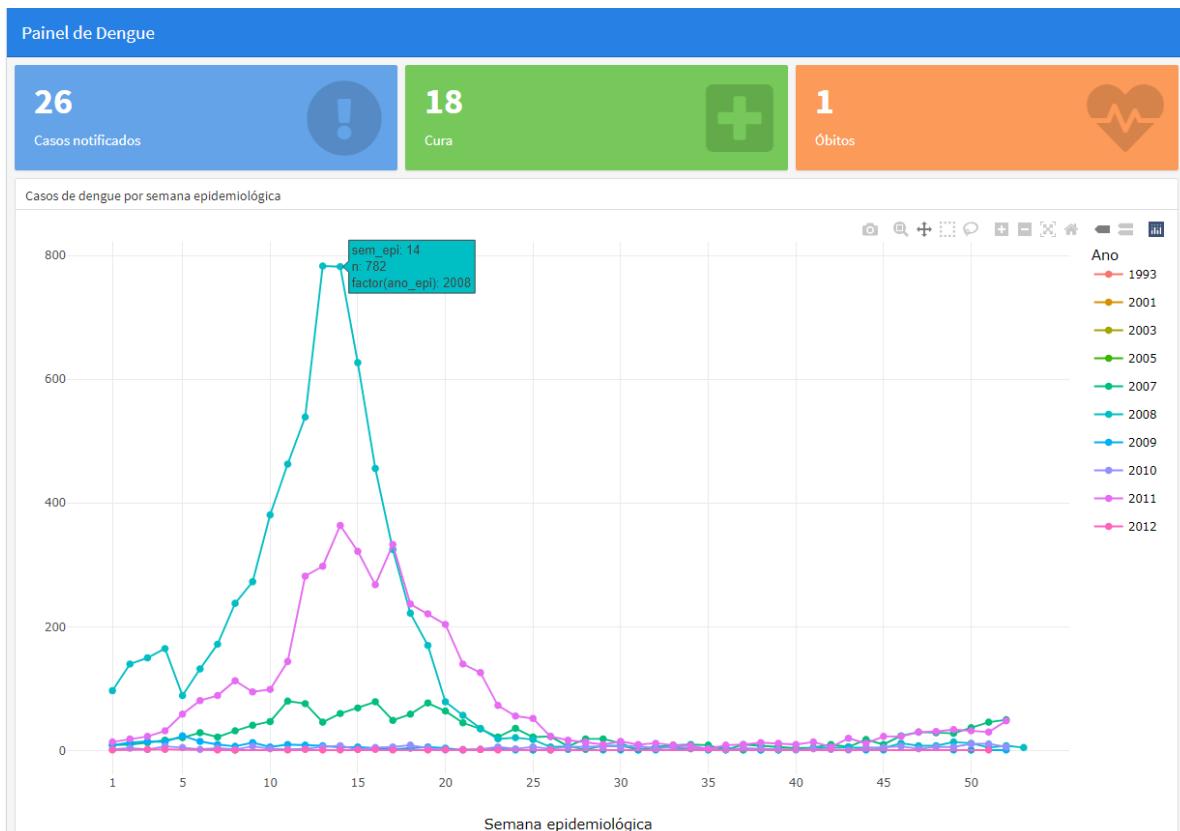
Figura 27: Tela de visualização do *script* com a criação do gráfico dinâmico.

```
-- 102 Row {data-height=650}
103 -
104
105 - ### Casos de dengue por semana epidemiológica
106
107 - ````{r}
108 # Criando o objeto de gráfico
109 grafico_1 <- dengue |>
110
111 # Contando número de casos por ano e semana epidemiológica
112 count(ano_epi, sem_epi) |>
113
114 # Plotando visualização de gráfico por ano e semana epidemiológica
115 ggplot(aes(
116   | x = sem_epi,
117   | y = n,
118   | color = factor(ano_epi)
119 )) +
120
121 # Adicionando linhas
122 geom_line() +
123
124 # Adicionando pontos
125 geom_point() +
126
127 # Aplicando novo tema para o gráfico
128 theme_minimal() +
129
130 # Adicionando rótulo para o eixo x
131 xlab("\nSemana epidemiológica") +
132
133 # Adicionando rótulo para o eixo y
134 ylab("") +
135
136 # Definindo o título da legenda
137 scale_color_discrete("Ano") +
138
139 # Definindo o intervalo de valores dos rótulos do eixo x
140 scale_x_continuous(breaks = c(1, seq(5, 50, 5)))
141
142 # Plotando um gráfico dinâmico
143 ggplotly(grafico_1)
144
145
```

No último passo você deverá renderizar seu arquivo para visualização do gráfico conforme a Figura 28 no seu *dashboard*. Clique no botão `Knit` do RStudio e rendeze o *script*! Veja na Figura 28 como ficará.



**Figura 28: Tela de visualização do dashboard com a criação
do objeto {grafico_1} que contém gráfico dinâmico.**



Observe que agora, no canto superior direito do gráfico existe um menu com opções interativas, como: aumentar o zoom para determinada área, arrastar o gráfico, retornar para escala inicial, entre outras. Além disso, ao passar o mouse pelos pontos do gráfico você visualizará uma pequena caixa com informações específicas sobre aquele ponto (uma espécie de legenda interativa) com informações sobre suas coordenadas no eixo x e y, assim como informações adicionais sobre o grupo ao qual pertence.

Bacana, não é mesmo? A população do Estado de Rosas conseguirá acessar os dados de dengue de forma intuitiva.



Nesta etapa é importante que você explore os recursos do gráfico e pratique modificando os gráficos. O pacote `ggplotly` permite ainda uma série de customizações que estão detalhadas em seu site oficial <https://plotly.com/ggplot2/getting-started/>.

Acesse e escolha outras opções para o *dashboard* que estamos desenvolvendo.

Atenção



Caso tenha encontrado dificuldade de chegar a um arquivo com os *scripts* que utilizamos, não se preocupe e continue no curso!

Deixamos para você um arquivo com todos os elementos que aplicamos nesta subseção prontos: o `dashboard_parte3_grafico_interativo.Rmd`. Encontre-o acessando o menu lateral “Arquivos” do Ambiente Virtual do curso e faça o seu *download*.

5.5 Inserindo tabelas estáticas e interativas

Uma das demandas da sociedade civil do Estados de Rosas é saber o número de casos notificados ao longo dos anos e, se possível, desagregado por município ou bairro de residência. Um recurso visual interessante muito utilizado quando temos um número grande de municípios é o uso de tabelas.

Com o R conseguiremos produzir dois tipos de tabelas: as tabelas estáticas e as interativas. As duas poderão ser confeccionadas de maneira bastante rápida e simples.

As tabelas também são recursos que a vigilância em saúde utiliza de forma recorrente e algumas vezes poderiam ser mais bem exploradas. Nesta seção iremos inserir tabelas em nosso *dashboard* permitindo a visualização de diversos tipos de dados de forma resumida e dinâmica. Aqui continuaremos a utilizar os dados exportados do Sinan Net do Estado de Rosas: o {NINDINET}.dbf}.

Vamos lá!

5.5.1 Tabelas estáticas

Tabelas simples ou estáticas são aquelas que não possuem interação. Elas geralmente são inseridas nos dashboards para organizar listas com números de registros de forma detalhada. Para criar tabelas amigáveis e práticas no R podemos utilizar o pacote `knitr`. Este pacote executa uma série de combinações tornando a visualização de dados muito intuitiva e simples.

Para o painel de avaliação da situação da dengue no Estado de Rosas iremos monitorar uma tabela que apresente a classificação final da doença em todos os anos em que houve notificações de casos. Lembre-se de que o banco de dados utilizado é o `{NINDNET.dbf}` que se encontra disponível no menu lateral “Arquivos” do Ambiente Virtual do curso.

Vamos iniciar a criação de nossas tabelas. Para isso precisaremos carregar o pacote `knitr` no início do bloco de código (*chunk*) do seu arquivo `.Rmd`. Replique o comando abaixo no arquivo que está construindo seu *dashboard*:

```
# carregando o pacote `knitr`  
require(knitr)
```

Agora precisaremos criar um novo objeto que chamaremos de `{tabela_dengue}`. Ele armazenará o *data.frame* `{dengue}` que criamos na seção anterior deste curso. Observe o *script* utilizado para gerar a tabela contendo a classificação final de dengue em todos os anos em que houve notificações `{tabela_dengue}`. Replique o comando abaixo no *chunk* do seu arquivo `.Rmd`:

```

# Criando nova tabela com informações sobre dengue
tabela_dengue <- dengue |>

# Criando nova coluna com os nomes da Classificação Final utilizando a função mutate()
mutate(
  Classificacao = case_when(
    CLASSI_FIN == 1 ~ "Cura",
    CLASSI_FIN == 2 ~ "Óbito",
    CLASSI_FIN >= 3 ~ "Outro",
    is.na(CLASSI_FIN) ~ "Ignorado"
  )
) |>

# Contando o número de casos por ano epidemiológico e classificação final com a função count()
count(ano_epi, Classificacao) |>

# Pivoteando a tabela para o formato "Largo"
pivot_wider(names_from = Classificacao,
            values_from = n,
            values_fill = 0) |>

# Selecionando apenas as colunas de interesse
select(ano_epi, Cura, Óbito, Outro, `Ignorado`) |>

# Renomeando a coluna de ano epidemiológico ("ano_epi") para "Ano"
rename(Ano = ano_epi)

```

Pronto. Você conseguiu compreender o que foi feito nesta etapa? Vamos analisar este *script* de forma detalhada:

1. Primeiro, utilizamos a função `mutate()` para criar a coluna chamada **Classificação** e a função `case_when` para renomear (categorizar) os valores de acordo com os códigos utilizados para a coluna com a variável de classificação final, **CLASSI_FIN**.
2. Segundo, executamos a função `count()` para realizar a contagem de cada categoria de classificação criada por ano epidemiológico.
3. Em seguida, transformamos a tabela criada (formato longo) em uma tabela no formato largo utilizando a função `pivot_wider()`.
4. Após, selecionamos apenas as colunas **ano_epí**, **Cura**, **Óbito**, **Outro**, e **Ignorado** com uso da função `select()`.
5. Por último, a coluna de nome **ano_epí** é renomeada para **Ano** utilizando a função `rename()`.

Agora, para que o seu dashboard faça a exibição desta tabela *online* necessitamos salvá-la como um objeto do tipo *tabela*. Para isso utilizaremos a função `kable()` do pacote `kntir`. Como argumento, a função `kable()` deve receber o nome do objeto de tabela `{tabela_dengue}`. Veja como fica o código que faz esta transformação e replique-o no *chunk* do seu arquivo `.Rmd`:

```
# Gerando uma tabela com a função kable()
kable(tabela_dengue)
```

Por fim, como uma boa prática necessitamos unir todos esses códigos em um único *chunk* em nosso arquivo `.Rmd` conforme visualizamos na Figura 29. Ao criarmos um novo bloco de códigos (*chunk*) este sempre deverá ser inserido após a indicação de uma nova linha indicada com o uso do `---`(hifen). E não se esqueça de renomear o nosso novo quadro, indicado por `###`, aqui denominado “Classificação final de casos de dengue por ano epidemiológico”.

Agora é a sua vez! Veja na Figura 29 como deverá ser o seu código com o *script* escrito em um arquivo do tipo `.Rmd` (*RMarkdown*) e replique-o.

**Figura 29: Tela de visualização do *script* com a criação da
tabela simples armazenada no objeto {tabela_dengue}.**

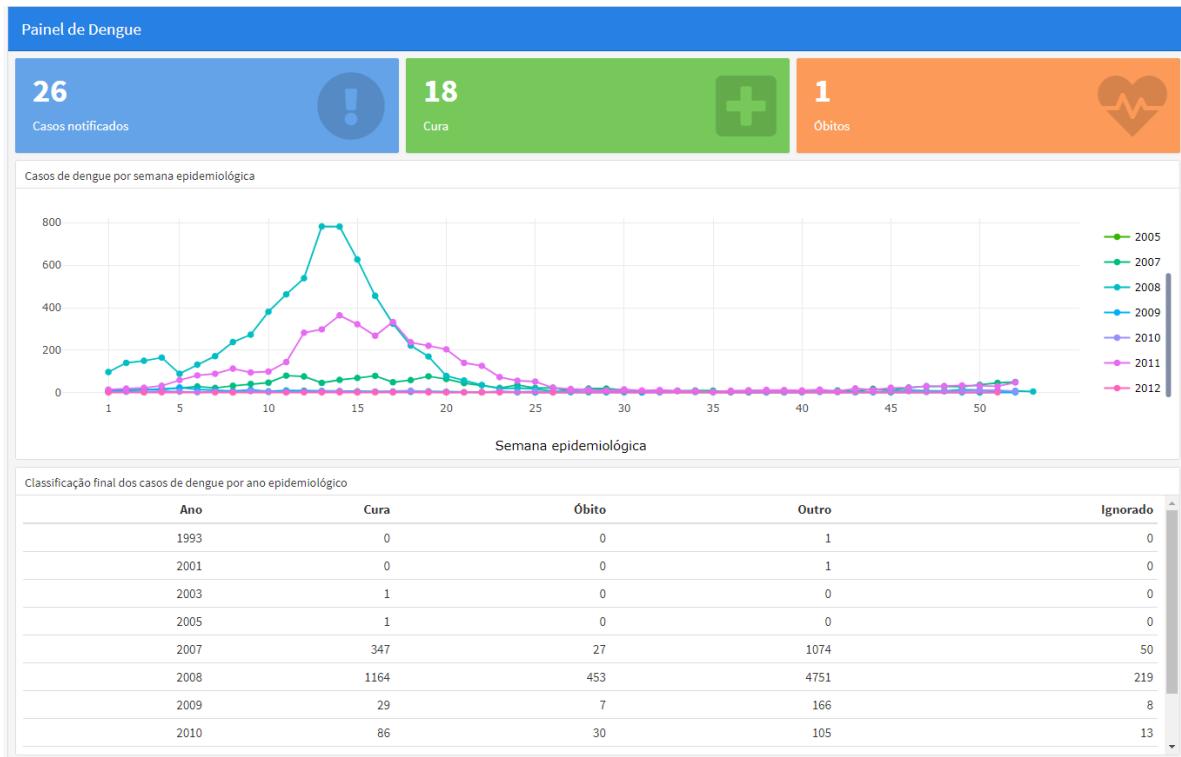
```

147 Row
149
150 ### classificação final dos casos de dengue por ano epidemiológico
151
152 ```{r}
153 # Criando nova tabela com informações sobre dengue
154 tabela_dengue <- dengue |>
155
156 # Criando nova coluna com os nomes da classificação Final utilizando a função mutate()
157 mutate(
158   classificacao = case_when(
159     CLASSI_FIN == 1 ~ "cura",
160     CLASSI_FIN == 2 ~ "Óbito",
161     CLASSI_FIN >= 3 ~ "outro",
162     is.na(CLASSI_FIN) ~ "Ignorado"
163   )
164 ) |>
165
166 # Contando o número de casos por ano epidemiológico e classificação final com a função count()
167 count(ano_epi, classificacao) |>
168
169 # Pivoteando a tabela para o formato "largo"
170 pivot_wider(names_from = classificacao,
171             values_from = n,
172             values_fill = 0) |>
173
174 # Selecionando apenas as colunas de interesse
175 select(ano_epi, Cura, Óbito, outro, `Ignorado`) |>
176
177 # Renomeando a coluna de ano epidemiológico ("ano_epi") para "Ano"
178 rename(Ano = ano_epi)
179
180 # Gerando uma tabela com a função kable()
181 kable(tabela_dengue)
182 ```
183
184

```

Para transformar nosso arquivo em um dashboard para o Estado de Rosas precisaremos renderizá-lo. Clique no botão **knit** do RStudio e renderize o *script*! Veja na Figura 30 como ficará:

Figura 30: Tela de visualização do *dashboard* com a criação da tabela simples armazenada no objeto {tabela_dengue}.



Atenção



Caso tenha encontrado dificuldade de chegar a um arquivo com os *scripts* que utilizamos, não se preocupe e continue no curso!

Deixamos para você um arquivo com todos os elementos que aplicamos nesta subseção prontos: o `dashboard_parte4_tabela_estatistica.Rmd`. Encontre-o acessando o menu lateral “Arquivos” do Ambiente Virtual do curso e faça o seu *download*.

5.5.2 Tabelas interativas

Uma opção também muito valorizada quando construímos um painel de indicadores são as tabelas dinâmicas ou interativas. Elas são ideais quando existem muitas informações a serem exibidas em tabela. Por serem interativas elas permitem que haja a participação do usuário do *dashboard* na escolha pela informação de forma intuitiva.

Para gerar tabelas dinâmicas no R utilizamos o pacote **DT**. Ele é muito interessante porque possui uma interface para a biblioteca JavaScript *DataTables*, que permite exibir tabelas em **HTML** interativas e muito bonitas. Com esse recurso, é possível filtrar informações, configurar paginação e classificá-la, inclusive. Após esta subseção você será capaz de tornar mais atrativa a visualização dos dados de dengue de Rosas.

Vamos praticar!

Primeiro, iremos carregar o pacote **DT** já no início do *chunk* do arquivo **.Rmd**:

```
# carregando o pacote `DT`  
require(DT)
```

Agora, criaremos o objeto `{tabela_bairro}` que receberá os dados do *data.frame* `{dengue}` utilizado para fazer a tabela simples `{tabela_dengue}` que criamos anteriormente. No entanto, agora incluiremos um número maior de informações neste novo objeto. Vamos lá, siga o passo a passo abaixo e replique-o em seu **RStudio**:

1. Excluiremos da tabela todos os valores que, na variável **NOBAIINF** (bairro), sejam iguais a nulo ou branco. Para isso utilizaremos a função `drop_na()` do pacote `tidyverse`.
2. Utilizaremos a função `mutate()` do pacote `dplyr` para transformar a variável **NOBAIINF** em uma variável do tipo fator com a função `factor()`.
3. Contaremos os casos de dengue por município de residência, bairro provável de infecção e ano epidemiológico com a função `count()` do pacote `dplyr`.
4. Após, utilizaremos a função `arrange()` do pacote `dplyr` para ordenar as linhas em ordem crescente de acordo com a coluna **ano_epi** e, em seguida, de acordo com a ordem alfabética para a coluna **NOBAIINF**.
5. E, por último, iremos renomear as colunas selecionadas para **Ano**, **Bairro** e **Casos** utilizando a função `rename()` do pacote `dplyr`.

Observe abaixo como escrevemos o código executando os passos de 1-5 no R. Replique-o em seu arquivo `.Rmd(dashboard)` no RStudio.

```
# Criando uma nova tabela
tabela_bairro <- dengue |>

# Removendo linhas com valores faltantes para bairro
drop_na(NOBAIINF) |>

# Transformando a variável NOBAIINF em fator
mutate(NOBAIINF = factor(NOBAIINF)) |>

# Contando o número de registros por bairro e ano epidemiológico
count(ano_epi, ID_MN_RESI, NOBAIINF, .drop = FALSE) |>

# Reordenando as linhas por ordem crescente de ano epidemiológico
arrange(ano_epi, NOBAIINF) |>

# Renomeando as colunas
rename(Ano = ano_epi,
       Município = ID_MN_RESI,
       Bairro = NOBAIINF,
       Casos = n)
```

Observe que criamos a tabela `{tabela_bairro}` quando rodamos os códigos acima. Agora vamos executar a função `datatable()` do pacote `DT` para criar uma estrutura dinâmica! Esta função tem algumas particularidades quanto à forma em que seus argumentos são definidos. Além de definir a tabela a ser criada como argumento principal, outros adicionais são passados em um objeto do tipo lista (`list`) dentro de um argumento de nome `options`. Parece complexo, não é mesmo? Mas acompanhe os passos a seguir para aplicar estas etapas:

6. Vamos criar uma tabela dinâmica, paginada, com no máximo 10 registros por página utilizando a função `datatable()` e seu argumento `pageLength`. Este argumento deverá estar contido no objeto do tipo lista em `options`. Esta etapa está sendo aplicada para que exibamos em tela apenas uma tabela de 10 em 10 linhas com a inserção de sua paginação no rodapé da tabela mudando com cliques do usuário.

Observe o *script* abaixo executando o passo 6 e replique-o em seu RStudio:

```
# Gerando uma tabela dinâmica com a função datatable()
datatable(tabela_bairro,

          # definindo máximo de 10 registros para serem mostrados
          options = list(pageLength = 10))
```

Como uma boa prática para a produção de um *dashboard* precisamos unir todos os códigos que executamos nos passos de 1 a 6. Observe na Figura 31 como deverá ficar o código para gerar a tabela dinâmica no arquivo `.Rmd`:

**Figura 31: Tela de visualização do *script* com a criação da
tabela dinâmica armazenada no objeto {`tabela_bairro`}.**

```

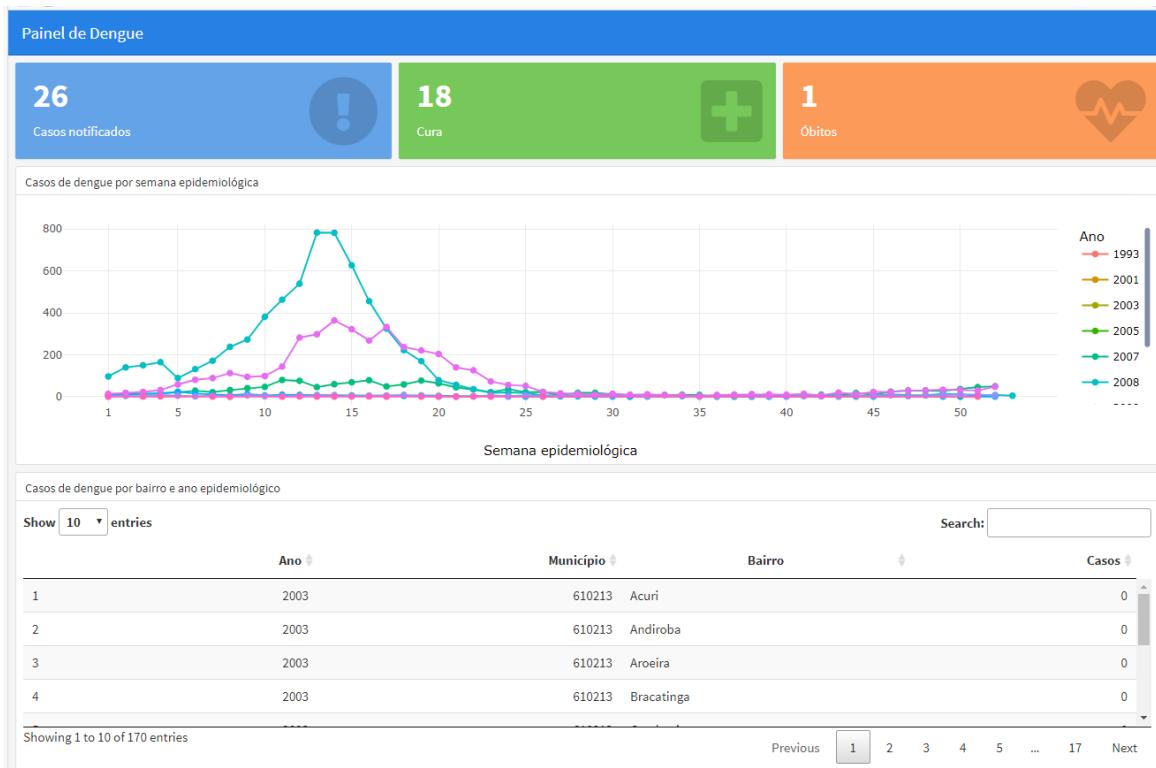
148
149 Row
150 <-->
151
152 <--> ### Casos de dengue por bairro e ano epidemiológico
153
154 <--> ``{r}
155 # Criando uma nova tabela
156 tabela_bairro <- dengue |>
157
158 # Removendo linhas com valores faltantes para bairro
159 drop_na(NOBAIINF) |>
160
161 # Transformando a variável NOBAIINF em fator
162 mutate(NOBAIINF = factor(NOBAIINF)) |>
163
164 # Contando o número de registros por bairro e ano epidemiológico
165 count(ano_epi, ID_MN_RESI, NOBAIINF, .drop = FALSE) |>
166
167 # Reordenando as linhas por ordem crescente de ano epidemiológico
168 arrange(ano_epi, NOBAIINF) |>
169
170 # Renomeando as colunas
171 rename(Ano = ano_epi,
172        ...Município = ID_MN_RESI,
173        ...Bairro = NOBAIINF,
174        ...Casos = n)
175
176 # Gerando uma tabela dinâmica com a função datatable()
177 datatable(tabela_bairro,
178           ...# definindo máximo de 10 registros para serem mostrados
179           ...options = list(pageLength = 10))
180 <-->
181 <-->
182
183

```

Pronto, agora já possuímos a estrutura necessária para produzir a nossa tabela interativa com páginas contendo 10 registros. Aplique a renderização em seu *script* para obter uma visualização como a apresentada na Figura 32. Lembre-se que para renderizar o seu *dashboard* basta clicar no botão `knit` do RStudio!



**Figura 32: Tela de visualização do dashboard com a criação
da tabela dinâmica armazenada no objeto {tabela_bairro}.**



Utilizando a função `datatable()` você poderá configurar a seleção do número de entradas visualizáveis por página da tabela, configurar a buscar por termos, além da possibilidade de navegar entre diferentes páginas de dados. Leia mais sobre o pacote `DT` clicando [aqui](#).



Atenção

Caso tenha encontrado dificuldade de chegar a um arquivo com os *scripts* que utilizamos, não se preocupe e continue no curso!

Deixamos para você um arquivo com todos os elementos que aplicamos nesta subseção prontos: o `dashboard_parte4_tabela_dinâmica.Rmd`. Encontre-o acessando o menu lateral “Arquivos”, do Ambiente Virtual do curso e faça o seu *download*.

6. Publicando o dashboard

Como profissional da vigilância do Estado de Rosas, você conseguiu levantar e preparar todas as análises necessárias para a construção de **um painel para avaliação de dengue no Estado de Rosas** (fictício). Com ele você dará transparência à situação das notificações de dengue entre anos de 1993 e 2012 à sociedade civil do estado.

Até aqui você configurou o *layout* do *dashboard* de Rosas, organizou o conteúdo que foi selecionado, criou gráficos e tabelas para tornar a informação interativa. Agora chegou o momento de escolher qual a maneira mais adequada para publicar este *dashboard* e torná-lo de acesso público.

Vamos lá!

6.1 Tema

É possível personalizar a aparência de um painel com diferentes temas (*templates*, em inglês), ou seja, podemos inserir e alterar suas cores e a fonte dos textos do *dashboard*. Os temas também devem ser definidos já no início do *script*, no cabeçalho `YAML` do seu documento `.Rmd`. Veja aqui uma lista dos diferentes temas que podem ser escolhidos para o seu *dashboard*:

- *cosmo* (padrão)
- *bootstrap*
- *cerulean*
- *journal*
- *flatly*
- *readable*
- *spacelab*
- *united*
- *lumen*
- *paper*
- *sandstone*
- *simplex*
- *yeti*

Por padrão, quando não escolhemos um tema específico para o painel ou configuramos o seu tema = “*default*” (padrão), o `flexdashboard` escolhe de forma autônoma, ou seja, automaticamente seleciona o tema *cosmo*.

Para definir um tema específico para o nosso *dashboard* necessitamos primeiro incluir a linha com o comando para esta ação: `theme: default`. Ela vai no cabeçalho YAML no início do arquivo `.Rmd`, conforme visualizamos na linha de código a seguir:

```
output:  
  flexdashboard::flex_dashboard:  
    theme: default
```

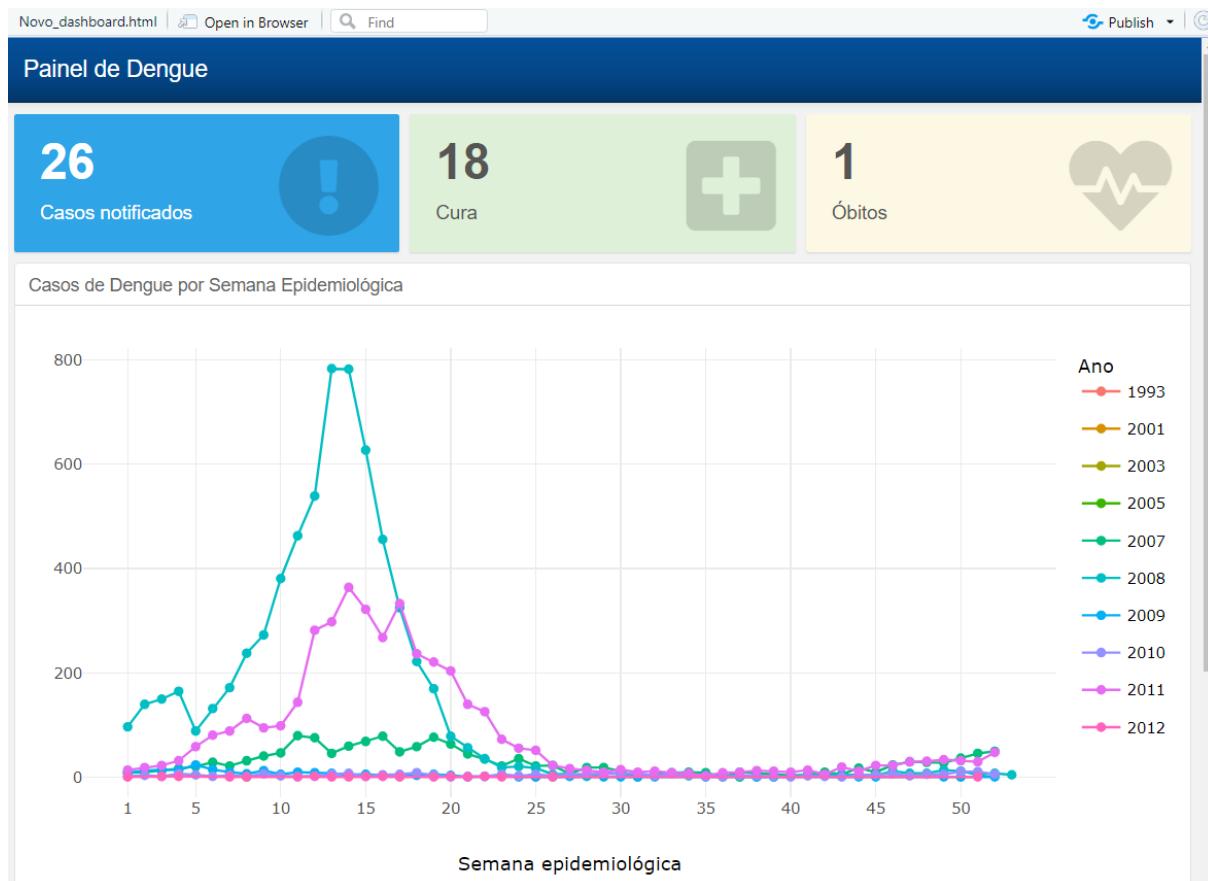
Assim, para a seleção de um tema específico precisaremos substituir o nome `theme: default` pela seleção necessária que queremos aplicar. Para o *dashboard* do Estado de Rosas utilizaremos o tema “*cerulean*”. Para isso basta substituir a palavra `default` por `cerulean` (`theme: cerulean`), conforme código abaixo.

Em seu arquivo `.Rmd` rescreva o cabeçalho YAML seguindo os seguintes comandos abaixo:

```
---  
title: "Painel de Dengue"  
output:  
  flexdashboard::flex_dashboard:  
    theme: cerulean  
    orientation: rows  
    vertical_layout: scroll  
---
```

Agora que já configurou o tema para o painel, renderize o seu *script* clicando no botão **knit** para obter um *dashboard* como o da Figura 33:

Figura 33: Tela de visualização do dashboard com a inclusão do tema cerulean.



Ficou bem mais interessante, não é mesmo? Um *dashboard* simples, com cores suaves e bastante intuitivo torna a informação mais atrativa e acessível à população de Rosas. Experimente os outros temas listados aqui nesta seção e opte pelo que mais lhe agrada. É importante que você sinta harmonia entre as cores ao publicar um *dashboard*. Pratique!

6.2 Logomarcas

Outra possibilidade de personalização do *dashboard* utilizando a linguagem de programação **R** é incrementá-lo inserindo uma logomarca ou *logo*. Geralmente este recurso é utilizado para identificar facilmente a origem ou o conteúdo de um painel. Quase sempre as logomarcas são inseridas ao lado do título do *dashboard* para que todos possam fazer identificação rápido do painel.

Calma. Estamos quase finalizando as edições para publicação final do *dashboard* do Estado de Rosas! Como uma última etapa iremos adicionar um logo salvo em uma imagem que chamamos de `lupa.png`, disponível no menu lateral “Arquivos”, do Ambiente Virtual do curso.

Para isso utilizamos o argumento `Logo` no cabeçalho `YAML` do seu documento `.Rmd` da seguinte forma: `Logo: nome-do-logo.png`. Observe o *script* a seguir em que escrevemos o trecho de código em `YAML` incluindo a logomarca. Replique o código a apresentado em seu arquivo `.Rmd` no seu computador.

```
---
```

```
title: "Painel de Dengue"
```

```
output:
```

```
flexdashboard::flex_dashboard:
```

```
  logo: Imagens/lupa.png
```

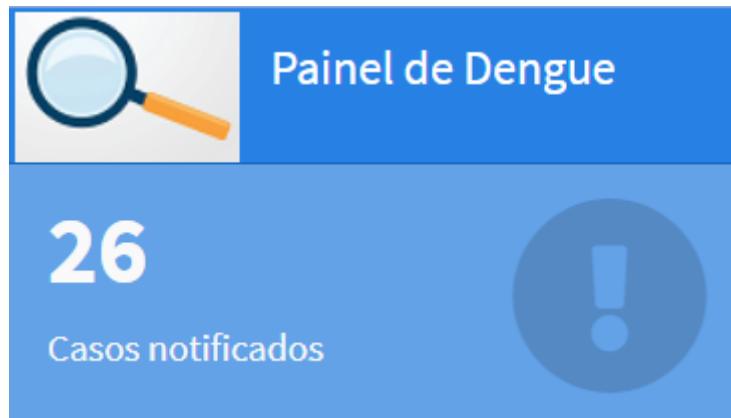
```
  orientation: rows
```

```
  vertical_layout: scroll
```

```
--
```

Para finalizar, precisamos transformar o arquivo `.Rmd` em `.html`. Para isso rendeze o seu *script* para obtermos um *dashboard* com inclusão da logomarca. Você deverá obter um painel como o visualizado aqui na Figura 34 a seguir:

Figura 34: Tela de visualização do *dashboard* com a *Logo*.



Você conseguiu! Parabéns, o Estado de Rosas agora possui um painel interativo para publicizar à sua população a situação da Dengue nos municípios do estado. Treine esta etapa e reproduza o que aprendeu no curso em seu dia a dia de vigilância em saúde, iniciando com pequenas visualizações e vá ampliando aos poucos seus painéis. Certamente você dará destaque ao município ou estado por isso. Este curso revolucionará o seu trabalho!

Atenção



Mais do que se atentar para a disposição e exibição dos elementos em um *dashboard*, você deve analisar se o propósito do painel está sendo cumpridos, tais como: facilitar com que o usuário encontre informações desejadas, que saiba discriminar o que são dados essenciais que ajudem na tomada de decisões ou a compreensão de um cenário epidemiológico. Veja se está oferecendo suporte gerencial e/ou contribuindo com a transparência dos dados à população.

A criação de dashboards vai muito além da criação de gráficos e indicadores em uma página com interação dinâmica!

Busque dispor suas informações de forma clara e intuitiva, facilitando ao máximo com que o usuário consiga encontrar as informações relevantes e necessárias com o mínimo de esforço.

Pronto. Agora selecione o botão *Knit* na parte superior da aba do *script* e renderize o seu dashboard e gere um arquivo `.html` contendo o seu painel.

Uma vez que você escreveu seu código e organizou o seu painel, você poderá disponibilizá-lo *online* para que qualquer pessoa possa encontrá-lo na *web*. Converse com o profissional responsável pela informática em seu município ou estado e solicite a inclusão do *dashboard* aqui produzido em um servidor *web*. Além disso nesta etapa de publicação você poderá solicitar também a atualização automatizada do painel, transformando seu *dashboard* com informações de vigilância em saúde em um *website*.

7. Outros componentes

Para aprimorar o seu *dashboard* e torná-lo ainda mais interativo e intuitivo, com **R** podemos ir muito além do que você aprendeu neste curso. Existem uma gama de ferramentas que podem ser utilizadas para gerar novas funcionalidades. Uma das ferramentas avançadas mais utilizadas é o **Shiny**, um pacote do **R** para o desenvolvimento de aplicações *web*. Listamos abaixo alguns exemplos de painéis em que se utilizou o **Shiny** e para saber mais acesse [aqui](#):

- [Calculadora de pressão assistencial em decorrência do COVID-19](#)
- [Monitoramento e Avaliação Estratégicos - RAG e RDQAS](#)
- [Observatório Escocês de Saúde Pública](#)

Destacaremos também a estrutura *htmlwidgets* que permite conectar o **R** com bibliotecas de visualização de dados em linguagem **JavaScript**. Os gráficos baseados em *htmlwidgets* são ideais para uso no **R** e podem se redimensionar dinamicamente, adaptando-se aos quadros do **flexdashboard**. Neste sentido deixamos como exemplo algumas das bibliotecas mais importantes utilizadas para produção de *dashboards*:

- [Leaflet](#): para criar mapas dinâmicos que suportam panorâmica e zoom, com várias anotações como marcadores, polígonos e *pop-ups*.
- [dygraphs](#): fornece recursos avançados para traçar dados de séries temporais e inclui suporte para muitos recursos interativos, incluindo destaque de série/ponto, zoom e panorâmica.
- [rbokeh](#): uma interface para **Bokeh**, uma estrutura declarativa **Bokeh** para criar gráficos baseados na web.
- [Highcharter](#): uma interface **R** para a popular biblioteca de gráficos **JavaScript Highcharts**.
- [visNetwork](#): uma interface para os recursos de visualização de rede da biblioteca **vis.js**.



Uma forma importante de aprender a desenvolver *dashboards* no R cada vez mais arrojados é sempre analisar os bons exemplos, e tomando como referência alguns painéis. A seguir, listamos uma galeria que pode contribuir como referências importante de modelo:

- [Galeria de *htmlwidgets*](#)

Para aprofundamento do tema, acesse:

- <https://nicar.r-journalism.com/docs/crosstalk-flexdashboard-leaflet-datatable>



Nossos cursos

Pronto, chegamos ao final deste curso! Agora você já conhece as principais ações para construir *dashboards* com o apoio da linguagem de programação R. Quer seguir a diante no aprendizado? Você encontrará outras etapas para aprofundamento das análises de dados em vigilância em saúde nos outros cursos. Aproveite e já faça sua inscrição nos cursos abaixo clicando nos *links*:

- [Análises de dados para Vigilância em Saúde - curso básico.](#)
- [Visualização de dados de interesse para a vigilância em saúde.](#)
- [Produção automatizada de relatórios na vigilância em saúde.](#)
- [Construção de diagramas de controle na vigilância em saúde.](#)
- [Linkage de bases de dados de saúde.](#)
- [Análise espacial de dados para a vigilância em saúde.](#)

Aproveite!

