



Язык R для анализа данных

Утилита формирования отчётов Quarto

• REC

Проверить, идет ли запись

Меня хорошо видно
& слышно?



Ставим “+”, если все хорошо “-”, если есть
проблемы

Тема вебинара

Утилита формирования отчётов Quarto



Андрей Павлюченко

Старший медиааналитик

Об опыте:

Мониторинг и исследования медиа (Brand Analytics, Медиалогия, R, Python)

Люблю делать: Text Mining, графовый анализ, интерактивные дашборды

a.pawluczenko@gmail.com / @a.pawluczenko



Правила вебинара



Активно участвуем



Офф-топик обсуждаем в Telegram



Вопросы пишем в чат или
поднимаем руку и говорим голосом



Включенная камера – желательно, но
не обязательно



Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара

Знакомство

Что такое Quarto и что будет с Rmarkdown?

Возможности Quarto

Обзор процесса создания отчетов

Контент, оформление и настройки

Создание сайтов в Quarto

Рефлексия

Цели вебинара

К концу вебинара вы сможете:

1. Разбираться в возможностях, форматах и опциях Quarto.
2. Создавать базовые документы в Quarto в разных форматах.
3. Наполнять их различным статическим и динамическим контентом.
4. Использовать возможности модульного и параметрического дизайна Quarto.
5. Использовать свои знания и документацию Quarto для создания продвинутых документов.

Смысл

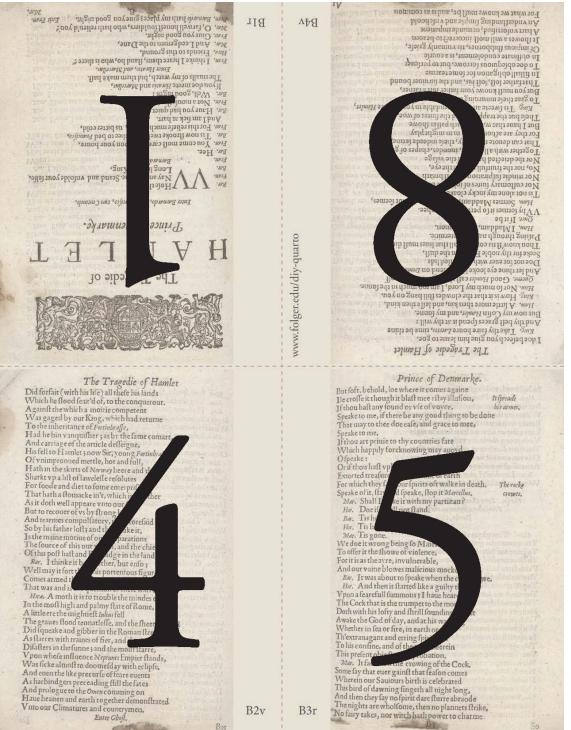
Зачем это нужно? Для того, чтобы:

1. Готовить качественные отчеты с настраиваемым дизайном, чтобы эффективно представлять результаты анализа.
2. Создавать интерактивные документы, чтобы пользователи могли исследовать данные и принимать решения на основе интерактивных элементов.
3. Адаптировать документы под различные цели и требования, обеспечивая гибкость и персонализацию.



Знакомство с Quarto

Что за Quarto?



4
5

Quarto (отсылка к формату *in quarto*) – система публикации документов от Posit (бывшая RStudio, создатели *Tidyverse*).



Источник: Tom Mock на [GitHub](#)



Что за Quarto?

Quarto – не библиотека R, а автономное приложение с интерфейсом командной строки. Например, даже без установленных R или Python вы можете:

1. Создать документ `document.md` в своем любимом редакторе.
2. Добавить YAML-метаданные (как Rmarkdown, но см. ниже).
3. Сверстать документ в нужном вам формате.

```
Terminal
1 quarto render document.md
```



Что за Quarto?

RStudio IDE новых версий включает Quarto, а также предоставляет для него редактор кода с автозаполнением и удобный визуальный редактор.

Quarto имеет более широкие по сравнению с Rmarkdown возможности. Можно воспринимать его как:

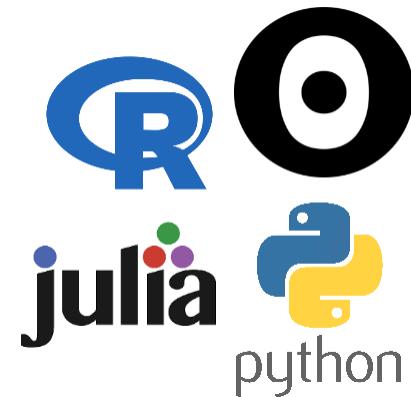
- Rmarkdown++;
- MS Word с гибридным интерфейсом;
- Dreamweaver для аналитиков и ученых.



Что нового: не только R

Quarto умеет обрабатывать код на следующих языках:

- R;
- Python;
- Julia;
- JavaScript (Observable).



Многоформатность

Quarto поддерживает множество форматов вывода, включая HTML, PDF, EPUB и другие. В Rmarkdown для большинства этих форматов необходимы дополнительные библиотеки:

Возможности Quarto	Реализация в Rmarkdown
HTML	✓
PDF	✓
DOCX/ODT	✓
PowerPoint	✓
сайт/блог	distill , blogdown
HTML-презентация	revealjs
книга	bookdown
интерактивная презентация	flexdashboard

Модульный дизайн

Quarto позволяет разбивать анализ на маленькие, повторно используемые компоненты, которые можно собирать в сложные документы. Это делает повторное использование и комбинирование частей вашего анализа легким.

Сложный проект на R может состоять из нескольких скриптов, содержащих загрузку и предобработку данных, специальные функции, моделирование и визуализацию.

```
1 source('data-wrangle.R', encoding = 'UTF-8')
```

Модульный дизайн

Как «настоящие» языки программирования, Quarto позволяет точно так же использовать сторонние файлы `.qmd`: собирать их в большие проекты, использовать вывод одной ячейки кода много раз.

Пример: включить содержимое из файла `_data.qmd`

```
1 {{< include _data.qmd >}}
```

Модульный дизайн

Quarto позволяет использовать параметры, с которыми можно готовить разные отчеты из одного и того же ноутбука, например:

- отчеты для разных регионов;
- отчеты для разных периодов;
- моделирование с разными допущениями или алгоритмами.

```
1 ---  
2 title : "Анализ рынка труда"  
3 ...  
4 params:  
5   job : "Бухгалтер"  
6   area: "Москва"  
7 ---
```

Модульный дизайн

Значения параметров используются в качестве шаблона в коде ноутбука, например:

```
1 df <- open_dataset('data/db/meta', partitioning = 'job_id') |>
2   filter(
3     job == params$job,
4     area == params$area
5   )
```

Затем документ можно сверстать с разными параметрами из R:

```
1 quarto::quarto_render(
2   input = "doc.qmd",
3   output_format = "pdf",
4   execute_params = list(job = "Бухгалтер", area = "Москва")
5 )
```

Или из командной строки:

Terminal

```
1 quarto render doc.qmd -P job:"Бухгалтер" -P area:"Москва" --output doc.pdf
```

Модульный дизайн

Некоторые преимущества модульных ноутбуков включают:

1. Повторное использование: можно повторно использовать те же ячейки в нескольких ноутбуках.
2. Гибкость: можно повторно комбинировать ячейки разными способами для разных выводов или целей.
3. Абстракция: скрывать детали реализации от читателей документа.
4. Удобство поддержки: нужно вносить изменения только в одном месте.



Расширенное форматирование

Quarto изначально включает все необходимое для научных публикаций и воспроизводимого анализа:

- обширную поддержку перекрестных ссылок и цитат;
- расширенное аннотирование рисунков, таблиц и кода;
- нативную поддержку графов и блок-схем ([mermaid](#) и [Graphviz](#));
- расширенные возможности написания формул *LATEX*.

Пример: Запись матрицы
(пакет [amsmath](#))

```
\begin{equation}
A = \begin{pmatrix}
4 & -3 & -3 \\
1 & 2 & 1 \\
1 & 1 & 2
\end{pmatrix}
\end{equation}
```

$$A = \begin{pmatrix} 4 & -3 & -3 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$



Что со всем этим делать?

- Quarto «из коробки» позволяет создавать более 40 форматов документов (`format: ваш-формат`).
- Для каждого формата доступны по несколько десятков опций.
- Знание CSS/SASS, *LATEX* или Pandoc дает почти безграничные возможности настройки внешнего вида отчетов.
- Quarto динамично развивается, появляются новые возможности.



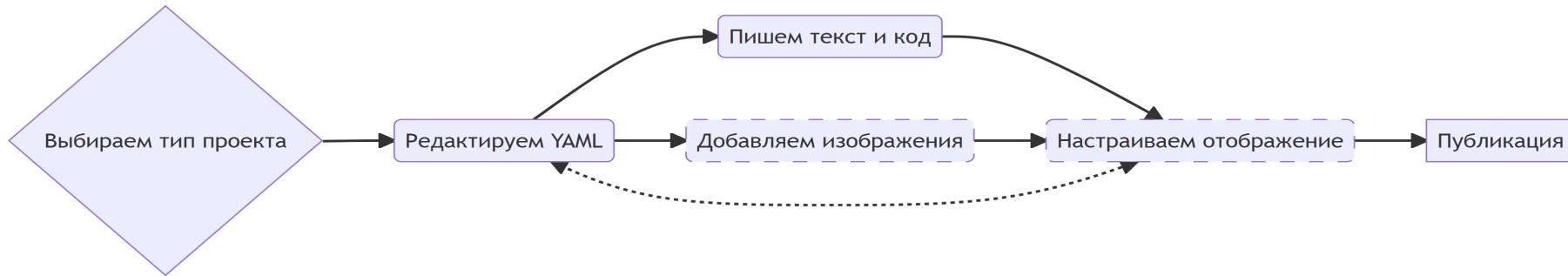
technocrat Regular

2022-09-07

Yes. There are almost *too many* ways to customize Quarto. The [gallery](#) has some examples. [YAML settings are abundant](#) and it's possible to mix in all sorts of CSS and JavaScript.

Алгоритм работы

Общий алгоритм проектов Quarto



Общий алгоритм проектов Quarto

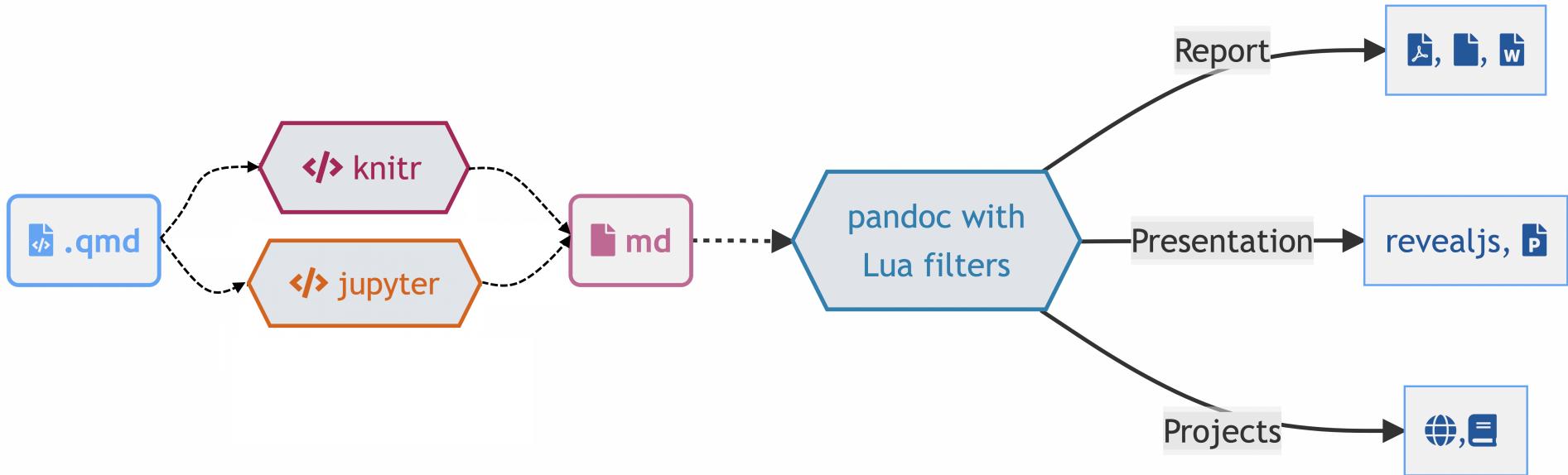
Кстати, код диаграммы:

```
1 ````{mermaid}
2 %%| eval: false
3 flowchart LR
4 A{Выбираем тип проекта} --> B(Редактируем YAML)
5 B --> C(Пишем текст и код)
6 B --> D(Добавляем изображения)
7 C --> E(Настраиваем отображение)
8 D --> E
9 E <--> B
10 E --> F[Публикация]
11
12 classDef optional stroke-dasharray:10;
13 class D optional
14 class E optional
15 ````
```



Как работает Quarto

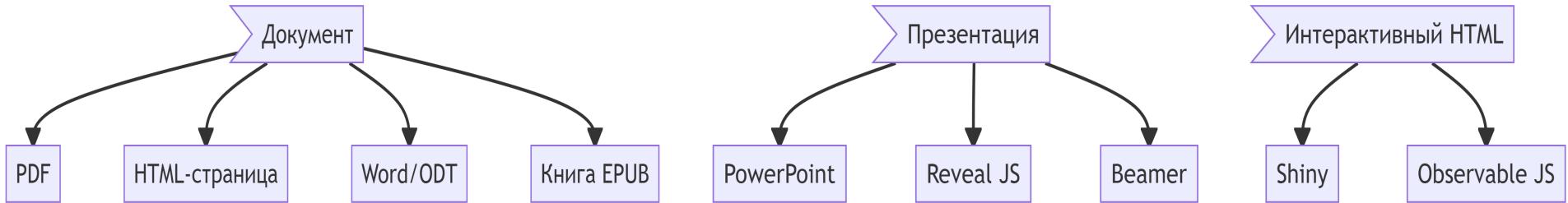
При работе с R система Quarto сначала создает markdown-документ, который затем конвертируется в нужный формат программой [pandoc](#).



Источник: Том Мок на [GitHub](#)



Типы проекта



Проекты других доступных типов можно создать вручную, прописав **format**: [наш-формат]

Также можно создавать комплексные публикации (проекты), состоящие из многих (обычно HTML) документов.

Типы проекта

Из одного ноутбука можно сверстать несколько документов в разных форматах. При этом отображение можно настроить отдельно для каждого формата.

```
1 format:  
2   html:  
3     theme: pulse  
4   pdf:  
5     fig-format: png
```



Обратите внимание

Только формат HTML позволяет создавать интерактивные документы.

При этом компоненты Shiny требуют хостинга на удаленном сервере. Компоненты, основанные на JavaScript, будут интерактивными у читателей без дополнительных сложностей.

Типы проекта



Обратите внимание

Для верстки PDF нужно установить *T_EX*-дистрибутив. Лучше всего [tinytex](#):

Terminal

```
1 quarto install tinytex --update-path
```

```
1 format:
2   html:
3     theme: united
4   pdf:
5     papersize: A5
6     fig-format: png
7     include-in-header:
8       - file: packages.tex
```

Элементы Quarto: контент и оформление

Структура ноутбука

YAML-заголовок

Блок кода

markdown-текст

pandoc-форматирование

Встроенные формулы

Опции кода
начинаются с #|

Нативный LaTeX

```
1 ▾ ...
2   title: "Gaussian Processes"
3   subtitle: "Nonlinear, nonparametric regression and optimization of expensive black-box functions"
4   author: "John Wacak"
5   date: today
6   format:
7     revealjs:
8       margin: 0.05
9   execute:
10    echo: true
11 ▾ ...
12
13 ▾ **{Julia}
14  # echo: false
15  # output: false
16  using Pkg
17  Pkg.activate(".")
18
19 ▾ ## Overview
20
21 MLjGaussianProcesses is a package for using Gaussian Processes in MLJ! The plan is to build on top of code from the [JuliaGP's group](https://github.com/juliahpc/gaussianprocesses.github.io).
22
23 :::: {.columns}
24  ::: {.column width="40%"}
25
26
27 1. Linear Regression
28 2. Making it Bayesian
29
30 :::
31  ::: {.column width="40%"}
32
33 6. The Function-space View
34 7. Gaussian Processes
35
36 :::
37 :::
38
39 # Linear Regression
40
41 ***Problem**: Given a dataset  $\mathbf{x} \in \mathbb{R}^n$ ,  $y \in \mathbb{R}$ , how can we determine the line (hyperplane) of best fit?
42
43 :::: {.columns}
44  ::: {.column width="40%"}
45
46 -  $\mathbf{x}_i$  is the input (feature) vector
47 -  $y_i$  is the target
48
49 :::
50  ::: {.column width="50%"}
51
52 ▾ **{Julia}
53 # echo: false
54 # fig-cap: "A scatterplot"
55 X = rand(200)
56 Y = 3 .+ X .+ 0.25 * (rand(200) .- 0.5)
57 X = X'
58
59 p = plot(X, y,
60   seriestype: scatter,
61   color:red,
62   xlabel:"x",
63   ylabel:"y",
64   size:(500,400),
65   ...)
66 ▾ ...
67
68 :::
69 :::
70
71 ***Answer**: Linear regression is best understood in terms of "Linear Algebra". We collect our dataset into a  $10 \times n$  dimensional [Design Matrix](https://en.wikipedia.org/wiki/Design_matrix) so that
72
73 \begin{equation}
74 \mathbf{X} \in \mathbb{R}^{10 \times n}
75 \mathbf{y} \in \mathbb{R}^{10 \times 1}
76 \mathbf{X} = \mathbf{x}_1 \ & \ \mathbf{x}_2 \ & \ \dots \ & \ \mathbf{x}_n \\
77 \end{equation}
```



YAML-заголовки

YAML («**Y**AML **A**in’t **M**arkup **L**

Формат записи – **опция: значение**.

Код конфигурации YAML отделяется тремя дефисами сверху и снизу:

```
1  ---
2  title : "Hello, Quarto"
3  author: "otus.ru"
4  format:
5    pptx:
6      reference-doc: ../templates/template.pptx
7      mermaid-format: png
8  ---
```

YAML-заголовки

В качестве значения может быть указан список:

```
1  опция:
2    - элемент1
3      опция-элемент1: значение
4    - элемент2
```

Многострочный текст печатается с новой строки после знака |:

источник: Mickaël Canouil на [GitHub](#)

```
1  ---
2  format:
3    html:
4      html-math-method: mathjax
5      include-in-header:
6        - text: |
7          <script>
8            window.MathJax = {
9              loader: {
10                load: ['[tex]/upgreek', '[tex]/boldsymbol', '[tex]/physics']
11              },
12              tex: {
13                packages: {
14                  '+': ['upgreek', 'boldsymbol', 'physics']
15                }
16              }
17            };
18          </script>
19  ---
```



YAML-заголовки

К метаданным можно обращаться через синтаксис Quarto `{{<...>}}` с ключевым словом `meta`:

```
1  [{{< meta author >}}]{style="font-family:Courier;font-size:3em;float:right;"}
2
3 ! [Otus: R для анализа данных]({{< meta format.revealjs.logo >}}){#fig-otus-r-owl}
```



otus.ru

Рис. 1: Otus: R для анализа данных

YAML-заголовки

Можно создать в папке с ноутбуком файл `_variables.yml` и наполнить его данными в YAML-формате:

```
_variables.yml
1 lecturer:
2   name: Андрей Павлюченко
3   role: Старший медиааналитик
```

Тогда к значениям из этого файла можно будет обращаться с **ключевым словом** `var`:

```
1 [Вебинар ведет {{< var lecturer.name >}}]{style="font-size:0.75em;"}
```

Вебинар ведет Андрей Павлюченко

Пример настройки: HTML

Опции HTML позволяют настраивать макет страницы, базовые элементы стиля и многое другое.

```
1  ---
2  format:
3      html:
4          theme:
5              light: pulse
6              dark: cyborg
7          css: styles.css
8          mainfont: corbel
9          fontsize: 1.1rem
10         fontcolor: "#090103"
11         linkcolor: "#E81444"
12         monofont: consolas
13         title-block-style: plain
14         title-block-banner: "#DDBEED"
15         title-block-banner-color: "#AA26EE"
16         margin-top: 40px
17     ---
```

Готовые темы Quarto представлены на сайте [Bootswatch](#).



Пример настройки: PDF

В PDF тоже можно настраивать макет (и размер) страницы, базовые элементы стиля, а также добавлять файлы *T_EX* для более тонкой настройки.

```
1  ---
2  format:
3      pdf:
4          mainfont: Georgia
5          monofont: Consolas
6          linkcolor: Bittersweet
7          citecolor: Mahogany
8          urlcolor: MediumOrchid
9          papersize: A5
10         fig-format: png
11         include-in-header:
12             - text: |
13                 \usepackage[english,russian]{babel}
14                 \usepackage{fontspec}
15                 \setsansfont{Courier New}
16                 \usepackage{amsmath,amsfonts,amssymb,amsthm,mathtools}
17                 \usepackage{icomma}
18     ---
```

Интересно ваше мнение:



Какой формат вам кажется самым интересным для работы?



Какой формат, по вашему мнению, вам больше всего пригодится в работе?

Фрагменты кода

Как и для Rmarkdown, возможность интерпретировать код и встраивать результаты его исполнения в документ является основной функциональной особенностью Quarto.

В Quarto опции для отдельного фрагмента кода, если они нужны, записываются внутри блока в формате YAML и начинаются с `#|`:

```
1 #| label: load-packages
2 #| include: false
3
4 library(tidymodels)
```

Можно спрятать код под спойлер:

▼ Показать код

```
1 ````{r}
2 #| label: ex-fold
3 #| code-fold: show
4 #| code-summary: 'Показать код'
5 x <- rchisq(1e3, 13)
6 ````
```

Фрагменты кода

В Quarto очень легко подсвечивать строки кода:

```
1  ````{r}
2  #| code-line-numbers: "2|3-4|6|7"
3  #| label: ex-highlight
4  #| eval: false
5
6  hist(x)
7  ggplot2::qplot(x, geom = 'histogram')
8  ````
```

Также можно выбрать или настроить оформление кода

```
1  ---
2  format:
3    html:
4      code-block-bg:          "#fef"
5      code-block-border-left: "#faf"
6      highlight-style:       breeze
7  execute:
8    echo: fenced
9  ---
```

```
```{r}
#| label: ols-model

ols <- lm(mpg ~ wt, data = mtcars)
````
```

Контейнеры pandoc

При работе в Quarto рекомендуется использовать универсальный формат контейнеров `pandoc`

1. Блочный контейнер – аналог `<div>` ограничивается двоеточиями (не менее 3) с опциональными атрибутами в фигурных скобках:

```
1 :::: {style="border: 1px solid;font-size:0.2em;"}
2 Этот маленький текст будет окружен прямоугольной рамкой
3 ::::
4
5 :::: {.callout-note title="Обратите внимание"}
6 А это будет примечание
7 ::::
```

Этот маленький текст будет окружен прямоугольной рамкой



Обратите внимание

А это будет примечание



Контейнеры pandoc

При работе в Quarto рекомендуется использовать универсальный формат контейнеров `pandoc`

2. Строчный контейнер – аналог `` записывается в квадартных скобках, за которыми следуют атрибуты в фигурных скобках:

```
1 [Сначала id, потом класс, потом все остальное] {#id .class key1="val1" key2="val2"}  
2  
3 [Красный текст]{style="color:#d00;"}
```

Красный текст

Форматирование изображений

Синтаксис изображений похож на гиперссылки, но с восклицательным знаком в начале:

```
1 ! [Собачка] (doge.png "Изображение собаки") {height=2cm}
```



Собачка

Надпись в квадратных скобках считается подписью к изображению. Ее можно оставить пустой.

Текст в кавычках после url изображения – его всплывающее название
(``)



Форматирование изображений

Можно добавлять атрибуты, например, выравнивание:

```
1 ! [Собачка] (doge.png "Изображение собаки побольше") {fig-align="left" fig-alt="Изображение собаки"}
```

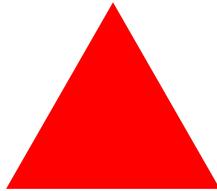


Собачка

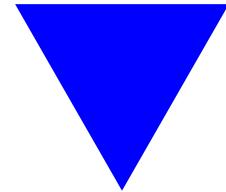
Форматирование изображений

Можно сгруппировать несколько изображений и расположить их определенным образом:

```
1 :::: {#fig-two-triangles layout-ncol=2}
2
3 ! [Первая картинка] (tred.png) {#fig-triangle-red width=20}
4
5 ! [Вторая картинка] (tblue.png) {#fig-triangle-blue width=20}
6
7 Две картинки
8 ::::
```



(1) Первая картинка



(2) Вторая картинка

Рис. 2: Две картинки

Форматирование изображений

В качестве макета можно указать массив, каждый элемент которого – ряд в сетке:

```
1 :::: {#fig-images layout=[[20,20], [40]]" style="font-size:0.5em;"}
2 ! [Первая картинка] (quarto.png) {#fig-01 width=40}
3
4 ! [Вторая картинка] (quarto.png) {#fig-02 width=40}
5
6 ! [Картина во втором ряду] (quarto.png) {#fig-long width=80}
7
8 Три картинки
9 ::::
```

quarto

quarto

(1) Первая картинка

(2) Вторая картинка

quarto

(3) Картина во втором ряду

Рис. 3: Три картинки



Форматирование изображений

Если назвать контейнер с изображениями `#fig-...`, Quarto представит его как одно изображение из нескольких частей (как на предыдущем слайде) и пронумерует соответственно.

На изображения можно ссылаться: [Рис. 2](#)

Кстати, по умолчанию названия сопровождаются префиксом «Figure». Это можно изменить в настройках проекта:

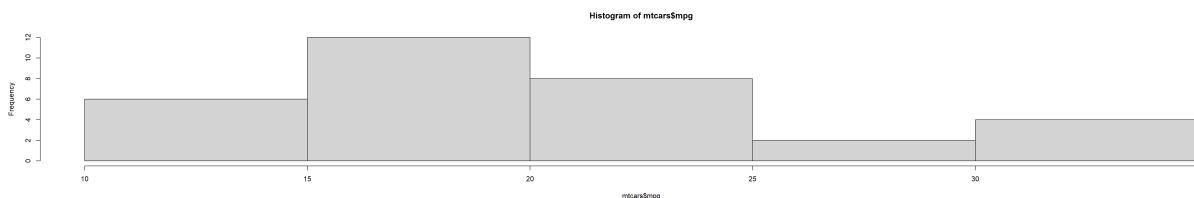
```
1 crossref:  
2   fig-title: Рис.          # Подпись к изображению (по умолчанию "Figure")  
3   tbl-title: Таблица      # Подпись к таблице (по умолчанию "Table")  
4   subref-labels: arabic    # (по умолчанию "a, b, ..., z")  
5   fig-prefix: Рис.         # Префикс ссылки на изображение  
6   tbl-prefix: Таблице     # Префикс ссылки на таблицу
```



Форматирование изображений

Для изображений, генерируемых кодом, параметры задают внутри блока:

```
1  ````{r}
2  #| fig-cap: "Две диаграммы"
3  #| fig-subcap:
4  #|   - "Первая диаграмма"
5  #|   - "Вторая диаграмма"
6  #| layout: "[20, 10]"
7  #| fig-width: 30
8
9  hist(mtcars$mpg)
10 hist(mtcars$disp)
11 ````
```



Первая диаграмма

Две диаграммы



Вторая диаграмма



Форматирование изображений

Блок кода, чье название начинается с `fig-`, генерирует пронумерованные изображения, на которые можно ссылаться:

```
1  ````{r}
2  #| label: fig-two-plots
3  #| fig-cap: "Две диаграммы"
4  #| fig-subcap:
5  #|   - "Первая диаграмма"
6  #|   - "Вторая диаграмма"
7  #| fig-width: 50
8  #| layout: "[10, 10]"
9  hist(mtcars$mpg)
10 hist(mtcars$disp)
11 ````
```



(1) Первая диаграмма



(2) Вторая диаграмма

Рис. 4: Две диаграммы



Диаграммы

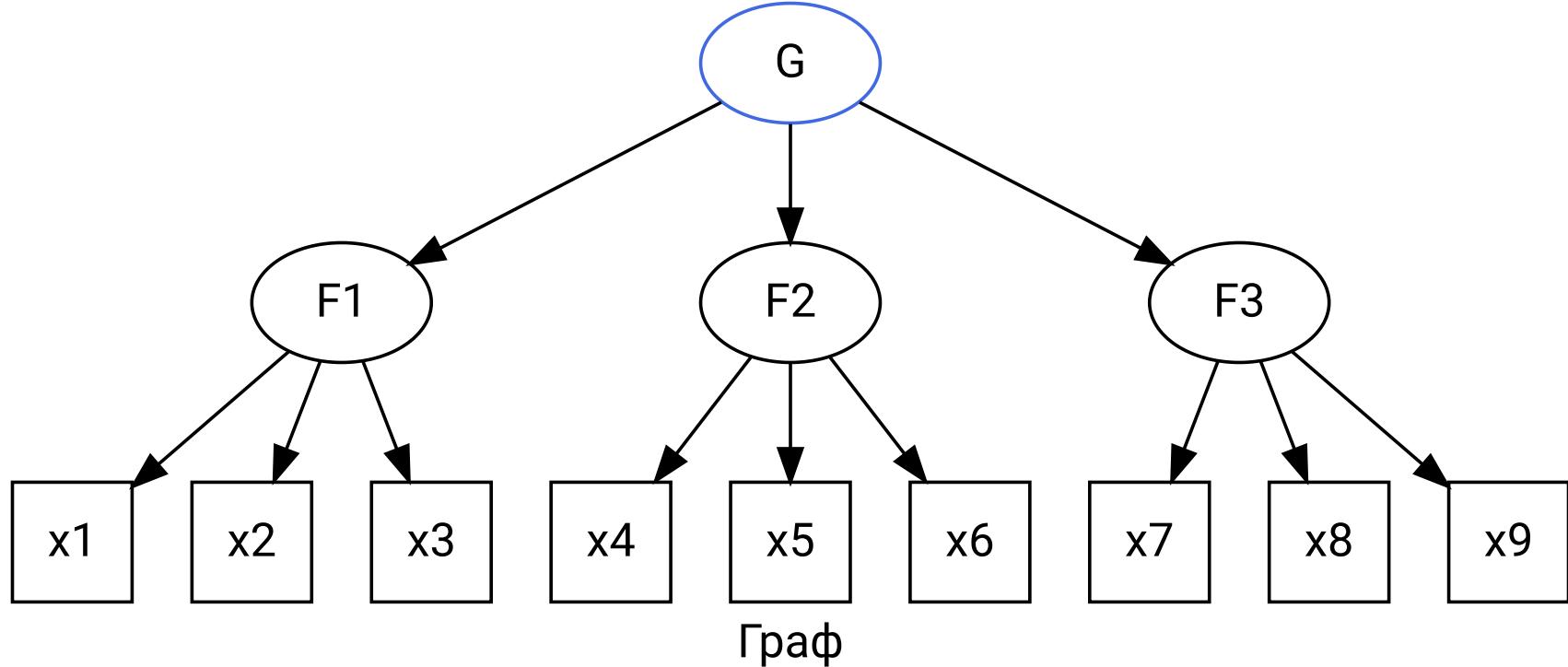
Диаграмма (граф) в Graphviz. Опции начинаются с \\\|:

```
1  ````{dot}
2 //|| label: ex-graphviz-code
3 //|| eval: false
4 // Комментарий
5 digraph G {
6     fontname="Roboto"
7     node [fontname="Roboto"]
8     G [color=royalblue];
9     x1, x2, x3, x4, x5, x6, x7, x8, x9 [shape=square];
10    F1 -> {x1, x2, x3};
11    F2 -> {x4, x5, x6};
12    F3 -> {x7, x8, x9};
13    G -> {F1, F2, F3};
14
15    label = "Граф";
16 }
17 ````
```



Диаграммы

Пример графа в Graphviz:



Интерактивные элементы

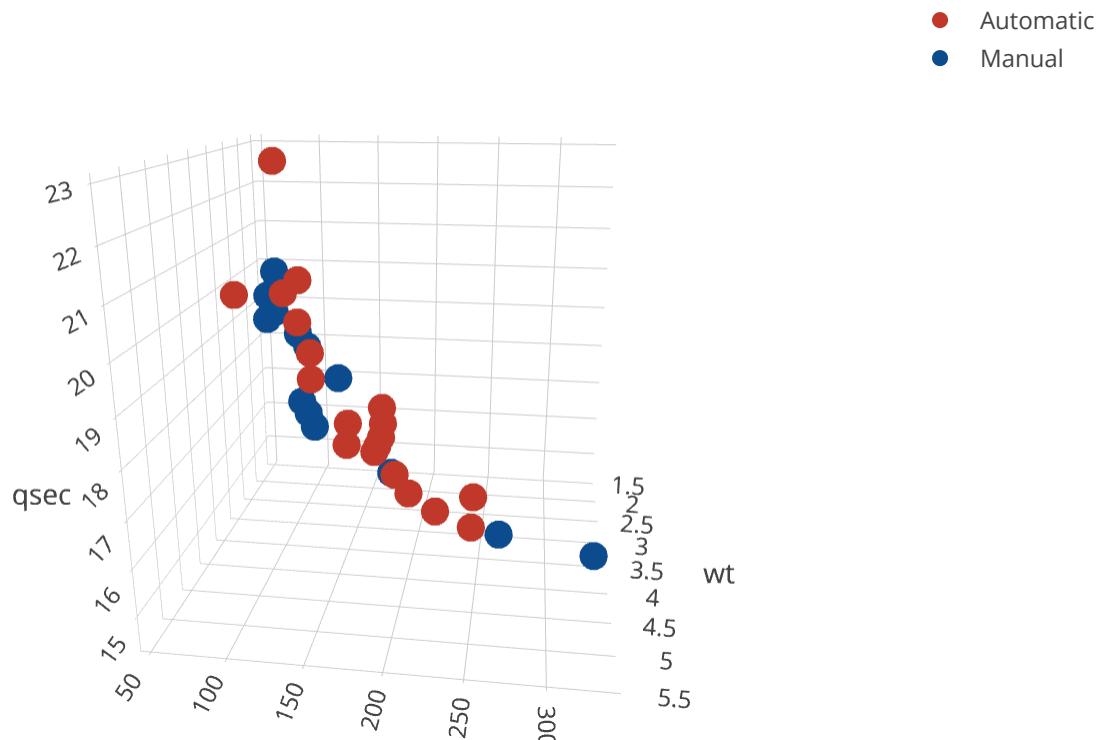
Формат HTML и производные от него (как revealjs) позволяют добавлять в документ интерактивные элементы семейства [htmlwidgets](#)

Интерактивная диаграмма plotly:

```
1 library(plotly)
2 library(dplyr)
3 library(forcats)
4
5 # Небольшая предобработка: сделаем категориальную переменную явно категориальной
6 mtcars <- mtcars |>
7   mutate(am = fct_recode(as.character(am), 'Automatic' = '0', 'Manual' = '1'))
8
9 plot_ly(
10   mtcars,
11   x = ~wt,
12   y = ~hp,
13   z = ~qsec,
14   color = ~am,
15   colors = c('#BF382A', '#0C4B8E'),
16   width = 550, height = 450
17 ) |>
18   add_markers()
```



Интерактивные элементы



Интерактивные элементы

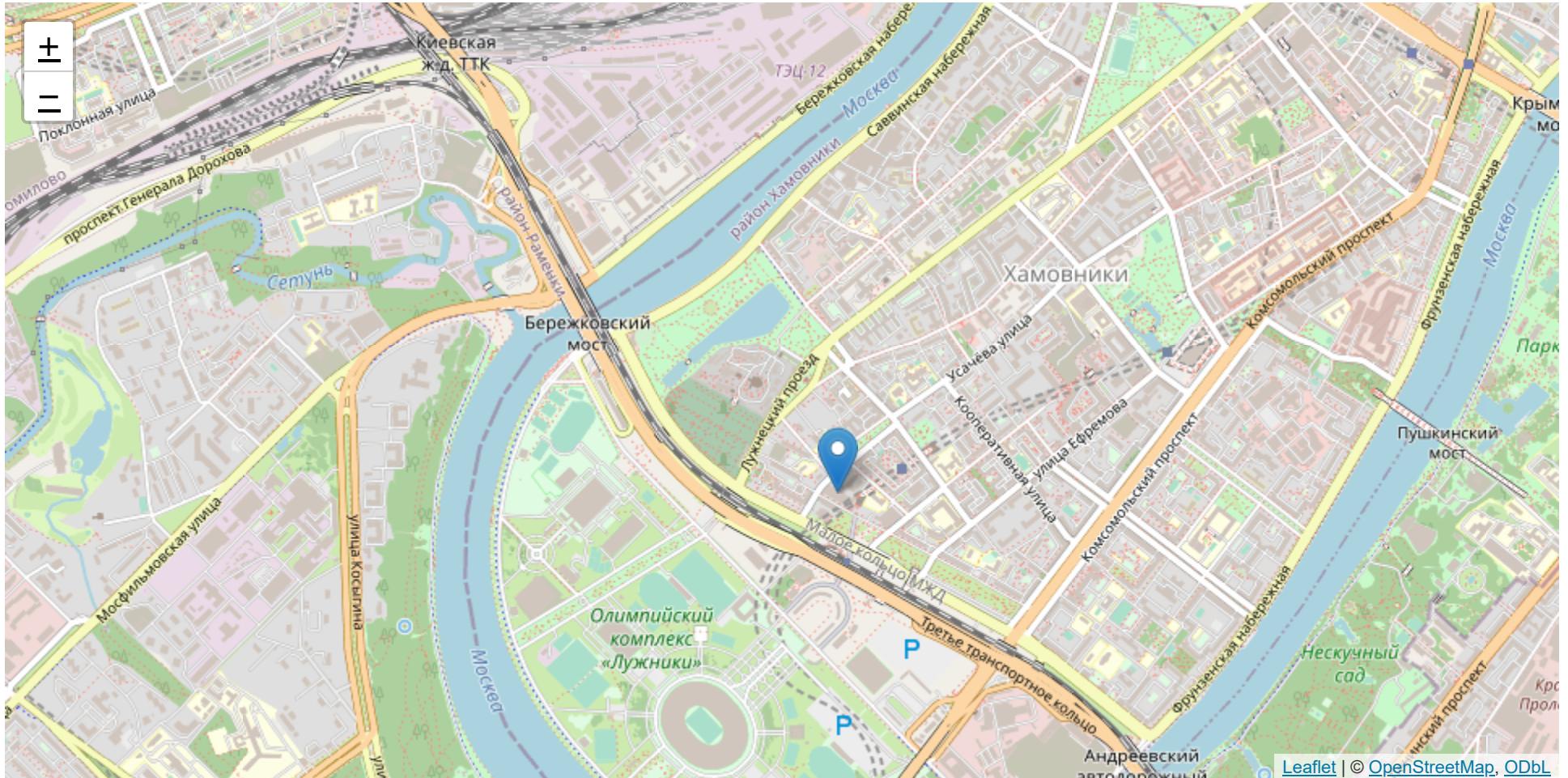
Формат HTML и производные от него (как revealjs) позволяют добавлять в документ интерактивные элементы семейства [htmlwidgets](#)

Интерактивная карта leaflet:

```
1 library(leaflet)
2
3 leaflet() |>
4   addTiles() |> # По умолчанию загружает карту OpenStreetMap, и норм
5   addMarkers(lng=37.560826, lat=55.722685, popup="Я разговариваю с вами отсюда")
```



Интерактивные элементы



Макеты страниц

Макет страницы – это то, как по умолчанию располагаются элементы на странице и друг относительно друга.

По умолчанию Quarto использует [Article layout](#): макет, оптимизированный для записей блога и других веб-публикаций, а также для статей в формате PDF.

```
1 ---  
2 format:  
3   html:  
4     page-layout: article  
5 ---
```

Другие доступные макеты:

- [full](#): как [article](#), но заполняет контентом поля, если они не используются;
- [custom](#): не содержит grid-макета, заполняет всю доступную ширину.



Макеты страниц

Можно делать исключения по местоположению для отдельных блоков, например:

```
1 :::: {.column-screen}
2 ! [Изображение на весь экран] (image.png)
3 :::
```

```
1 #| column: screen-inset
2
3 # Иллюстрация на всю ширину экрана с маленькими полями
4 leaflet() |>
5   addTiles() |>
6   addMarkers(lng=174.768, lat=-36.852, popUp="Родина R")
```

```
1 :::: {.column-margin}
2 ! [Картинка на полях] (image.png)
3 :::
```

```
1 [Заметка на полях] {.aside}
```



Макеты для интерактивного взаимодействия

Если вы используете интерактивные элементы с возможностью управления их отображения пользователем (как в Shiny), можно использовать раскладку, облегчающую интерактивное взаимодействие:

- `.panel-input`: контейнер для элементов управления, располагающийся над контентом;
- `.panel-tabset`: помещает контент в несколько вкладок, между которыми можно переключаться;
- `.panel-sidebar` и `.panel-fill` или `.panel-center`: боковая панель для элементов управления и широкая панель для отображаемого контента.



Макеты для интерактивного взаимодействия

Как и ширину блока контента, панель можно задать:

- в теле ноутбука через контейнер:

```
1 :::: {.panel-sidebar}
2 [элементы]
3 :::
```

- во фрагментах кода, чтобы указать, куда отправятся сгенерированные кодом элементы:

```
1 ---
2 ...
3 format:
4   html:
5     page-layout: custom
6 server: shiny
7 ---
```

```
1 #| panel: sidebar
2 vars <- setdiff(names(iris), "Species")
3 selectInput('xcol', 'X Variable', vars)
4 selectInput('ycol', 'Y Variable', vars, selected = vars[[2]])
5 numericInput('clusters', 'Cluster count', 3, min = 1, max = 9)
```

```
1 #| panel: fill
2 plotOutput('plot1')
```



Условный контент

Иногда вам понадобится настроить видимость того или иного фрагмента документа в зависимости от формата, в котором он будет сверстан. Например, загружать интерактивные элементы для HTML, но прятать их или заменять статичным контентом для PDF.

Для этого в Quarto предусмотрено две пары команд (точнее, два класса и два атрибута для контейнеров с контентом), которые можно комбинировать:

1. `.content-visible` или `.content-hidden`: показать или скрыть контент в зависимости от условия в п. 2.
2. `when-format=` или `unless-format=`: показать или скрыть контент (в зависимости от класса в п. 1) при условии, что формат соответствует или не соответствует значению после знака равно.

```
1 :::: {.content-visible when-format="html"}  
2  
3 Содержание этого блока будет видно только в формате HTML  
4  
5 :::
```



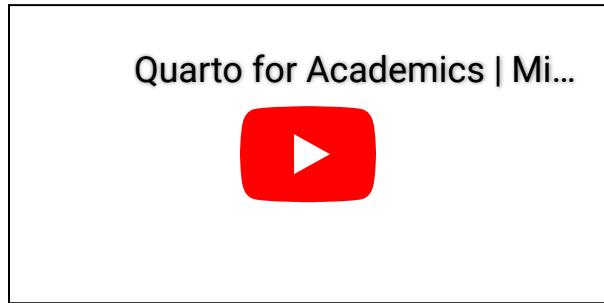
Условный контент

Для удобства несколько распространенных названий форматов считаются псевдонимами, обобщающими ряд других доступных в Quarto форматов. Вот таблица соответствия:

| Псевдоним | Форматы |
|-----------|-----------------------------------|
| latex | latex, pdf |
| pdf | latex, pdf |
| epub | epub |
| html | html, epub, revealjs |
| html:js | html, revealjs |
| markdown | markdown, commonmark, gfm, markua |

Видео

В форматах HTML в документ можно встроить видео из локального файла или по url. В презентации `revealjs` видео можно добавить в качестве фона.



Кроме заголовка фрейма можно настраивать такие параметры, как:

- соотношение сторон (`aspect-ratio="4x3"`);
- ширина и высота (`width="250" height="175"`);
- таймкод (`start="26"`).

Комплексные публикации

Сайты, блоги, книги

Quarto содержит нативную поддержку создания комплексных публикаций:

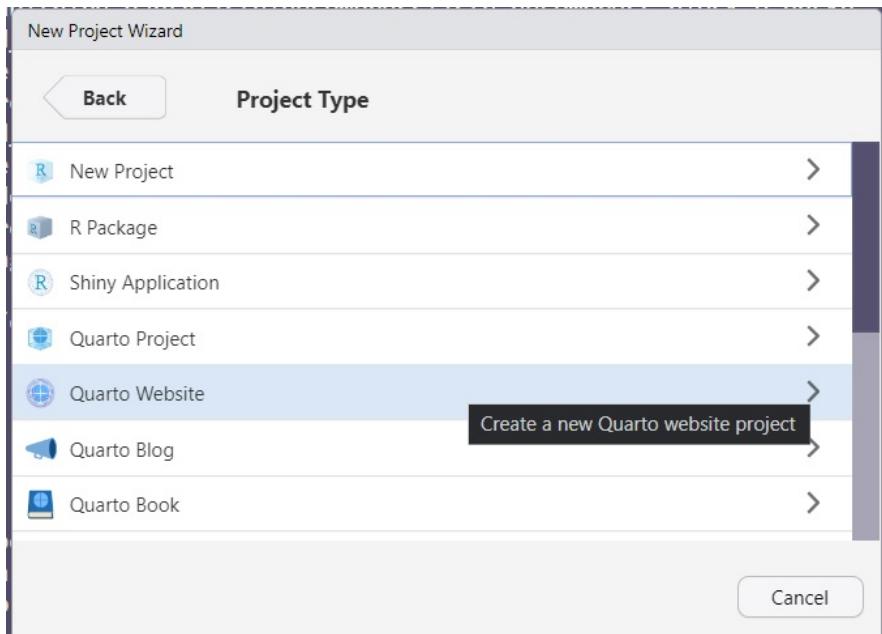
- сайты;
- блоги;
- КНИГИ.

Любой сайт, блог или книга состоит из основных элементов:

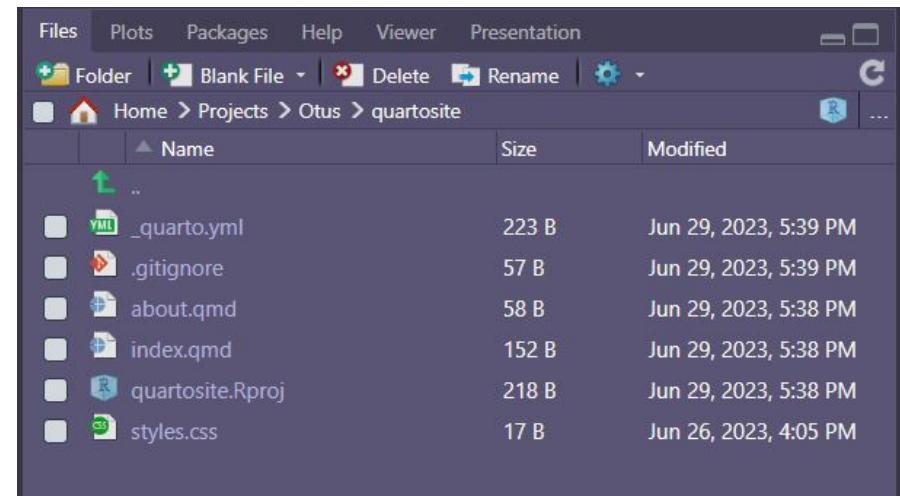
- метаданные проекта;
- контент (файлы `.qmd`);
- элементы навигации (содержание, меню, вкладки, поиск и т. п.);
- ресурсы (изображения, темы, скрипты и т. п.).

Создание сайта в Quarto – Заготовка

Проще всего создать сайт, блог или книгу как проект RStudio:



Выбираем Quarto...



RStudio создает для нас необходимые файлы и пару файлов для примера

Создание сайта в Quarto – Метаданные

Информация о проекте, в том числе ссылки на контент, описание структуры и общий стиль, содержится в файле `_quarto.yml`:

```
1  ---
2 project:
3   type: website
4
5 website:
6   title: "quartosite"
7   navbar:
8     left:
9       - href: index.qmd
10      text: Home
11       - about.qmd
12
13 format:
14   html:
15     theme:
16       light: united
17       dark: vapor
18     css: styles.css
19     toc: true
20 ---
```

Создание сайта в Quarto – Навигация

В Quarto можно легко создать следующие элементы навигации:

- `navbar` (навигационная панель сверху);
- `sidebar` (боковая навигационная панель);
- `page-footer` (подвал сайта);
- `search` (поисковый виджет).

Создание сайта в Quarto – Навигация

Пример задания верхней навигационной панели:

```
1 website:  
2   <...>  
3   # Верхняя панель («шапка»)  
4 navbar:  
5   title: "QuartoWebsite" # Переписывает website: title  
6   logo: www/owls_R.png  
7   logo-alt: "Otus: R for Data Analysis"  
8   # Элементы слева  
9   left:  
10    - href: index.qmd  
11      text: Home  
12    - about.qmd  
13   # Элементы справа  
14   right:  
15    - href: https://otus.ru  
16      icon: mortarboard  
17      target: _blank  
18    - href: https://github.com  
19      icon: github  
20      target: _blank
```

Создание сайта в Quarto – Навигация

Пример задания боковой панели:

```
1 website:
2 <...>
3 # Боковая панель (боковое меню):
4 sidebar:
5   background: dark    # Один из стандартных цветов текущей темы оформления
6   foreground: light   # Еще один цвет темы. Они изменяются, если изменить тему
7   style: docked      # Прикреплена к своему краю
8 contents:
9   - section: "Первый раздел"
10    contents:
11      - entry-01.qmd
12      - entry-02.qmd
13   - section: "Второй раздел"
14    contents:
15      - entry-03.qmd
16      - entry-04.qmd
```



Создание сайта в Quarto – Навигация

Пример задания нижней панели (подвала):

```
1 website:  
2   <...>  
3   # Подвал  
4 page-footer:  
5   background: primary  # «Ведущий» цвет выбранной темы оформления  
6   foreground: "#eee"  
7   # Элементы посередине (также можно указать right и left):  
8   center:  
9     - about.qmd
```

Создание сайта в Quarto – Навигация

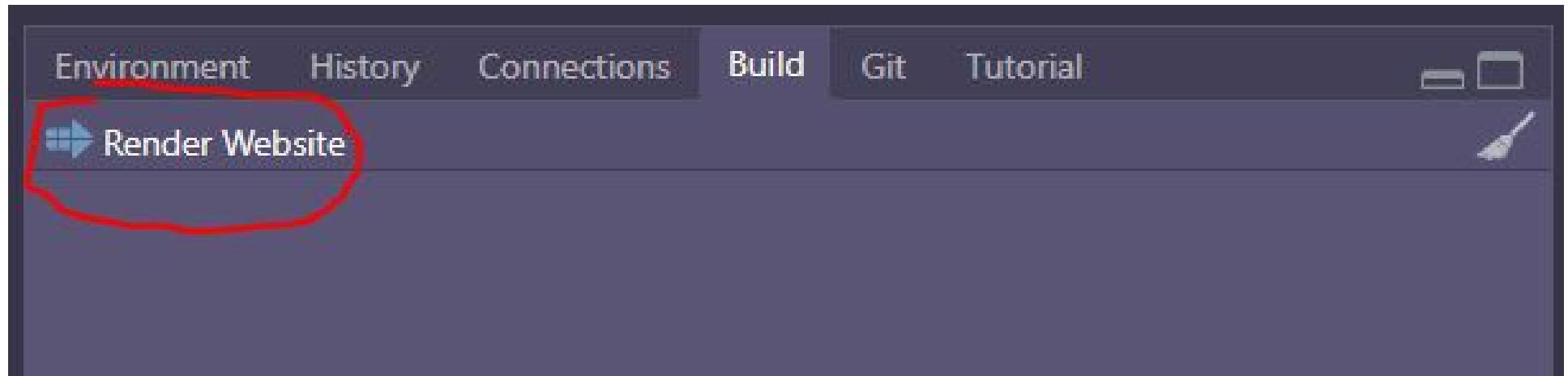
Пример задания поискового виджета:

```
1 website:  
2 <...>  
3 # Поиск  
4 search:  
5   location: navbar # К чему прикрепляется виджет  
6   type: overlay    # Тип: overlay – иконка со всплывающим виджетом, textbox – стандартный text input  
7   limit: 5         # Максимальное количество результатов для показа
```

Создание сайта в Quarto – Рендеринг

Для предпросмотра нашего сайта находим:

1. Меню проекта
2. Вкладку **Build**
3. Кнопку **Render Website**



Создание сайта в Quarto – Публикация

Помимо собственного сервера, опубликовать сайт можно на [Quarto Pub](#), [Posit Connect](#), [GitHub Pages](#), [Netlify](#) и некоторых других платформах.

Например, если мы назовем проект [и репозиторий на GitHub username.github.io](#), где `username` – наше имя на GitHub, мы создадим свой персональный сайт на GitHub Pages.

Если мы назовем проект и репозиторий `name`, где `name` – любое название, мы создадим сайт по адресу `username.github.io/name`.

Создание сайта в Quarto – Публикация

Рассмотрим публикацию на GitHub Pages:

1. Прописываем, что папка с сайтом должна называться “docs” (по умолчанию “_site”)

```
1 project:  
2   type: website  
3   output-dir: docs
```

2. Заходим в свой аккаунт на GitHub и создаем пустой репозиторий (без README)
3. Связываем свой локальный проект с репозиторием:

```
1 git init  
2 git add .  
3 git commit -m "Initial commit"  
4 git branch -M main  
5 git remote add origin https://github.com/username/name.git  
6 git push -u origin main
```



Создание сайта в Quarto – Публикация

4. В настройках репозитория в меню Pages разрешаем развертывание на Pages: выбираем ветку и папку ([Рис. 5](#))

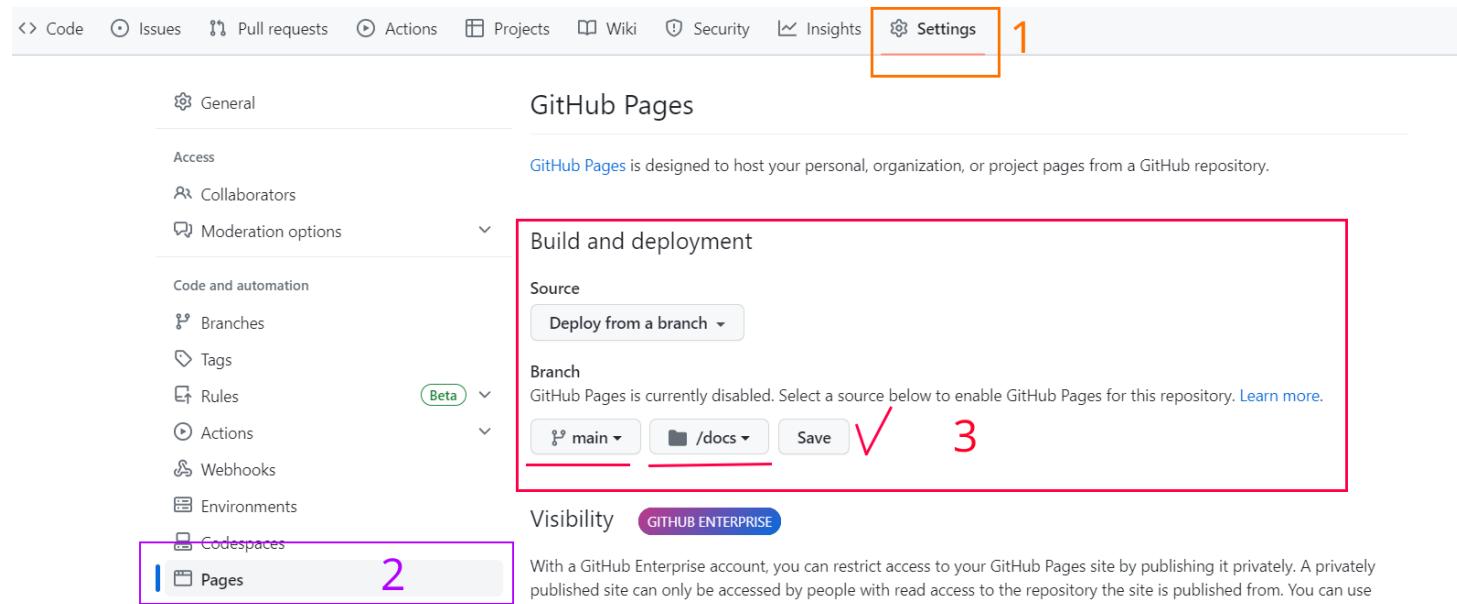


Рис. 5: Настройка отображения на GitHub Pages

Рефлексия

Рефлексия

Спросите себя. Расскажите в чате или голосом:

1. Хотите ли вы применять Quarto в своей работе?
2. Считаете ли вы, что это сложно?
3. О каких еще возможностях системы вы хотите узнать?
4. Какой проект вам хочется оформить в Quarto первым? В каком формате вы бы его сверстали?

A close-up photograph of a snowy owl's head and upper chest. The owl has large, prominent yellow eyes with dark pupils. Its feathers are a mix of white, brown, and black, creating a mottled pattern. The background is a plain, light color.

Спасибо за внимание!

До встречи ;)