



Quarto Starter Kit for IR Professionals

Colorado State University Office of IRPE

2025-05-19

Table of contents

1	Introduction to Quarto	3
1.1	What is Quarto?	3
2	Getting Started with Quarto	3
2.1	Installation Instructions	3
2.2	Essential R Packages	3
3	Basic Document Features	4
3.1	Text Formatting with Markdown	4
3.2	Code Chunks and Output	4
3.3	Basic Charts	5
3.4	Tables	6
4	Creating Parameterized Reports	7
4.1	Setting Up Parameters	7
4.2	Using Parameters in Your Document	8
4.3	Rendering with Parameters	8
5	Example: Enrollment Report	9
5.1	Enrollment Trends Visualization	10
5.2	Enrollment Summary Table with Calculations	11
6	Automation Techniques	12
6.1	File Organization	12
6.2	Automation Script	13
6.3	Scheduling (Windows)	14
7	Troubleshooting Tips	14
7.1	Common Issues and Solutions	14
7.1.1	PDF Output Problems	14
7.1.2	Table Formatting in Word	14
7.1.3	Data Loading Errors	15
8	Resources for Learning More	15
8.1	Beginner-Friendly Resources	15
8.2	Video Tutorials	15
8.3	Getting Help	15
9	Thank You!	15

1 Introduction to Quarto

This document serves as both a demonstration of Quarto’s capabilities and a reference guide for Institutional Research professionals. You can use the rendered document as a guide, and also examine the source code to see how each feature is implemented.

1.1 What is Quarto?

Quarto is an open-source scientific and technical publishing system built on Pandoc. It allows you to create dynamic documents that can include:

- Formatted text with Markdown
- Executable code (R, Python, Julia)
- Interactive visualizations
- Mathematical equations
- Citations and references

Best of all, from a single source file (like this one), you can produce documents in multiple formats: HTML, PDF, Word, PowerPoint, and more.

2 Getting Started with Quarto

2.1 Installation Instructions

To use Quarto, you’ll need to install:

1. R from [CRAN](#)
2. RStudio from [Posit](#)
3. Quarto from [Quarto.org](#)

Newer versions of RStudio (2022.07 or later) include Quarto by default.

2.2 Essential R Packages

These packages are commonly used in IR reporting workflows:

```
# Basic packages for data handling and visualization
install.packages(c(
  "tidyverse",    # Data manipulation and visualization
  "knitr",        # Dynamic report generation
  "rmarkdown",    # Markdown processing
  "flextable"     # Word-friendly tables
))

# You only need to install packages once on your computer
```

Remember: You only need to install packages once, but you need to load them with `library()` in each document.

```
library(tidyverse)
library(knitr)
library(rmarkdown)
library(flextable)
```

3 Basic Document Features

3.1 Text Formatting with Markdown

Quarto uses Markdown for text formatting:

- **Bold text** with `**bold**`
- *Italic text* with `*italic*`
- Lists with `-` or `1.`
- [Links](#) with `[text](url)`
- Headers with `#`, `##`, etc.

3.2 Code Chunks and Output

This is a code chunk that runs R code and displays its output:

```
# Calculating summary statistics for a sample dataset
numbers <- c(15, 23, 42, 56, 89, 12)

# Summary statistics
mean_value <- mean(numbers)
```

```

median_value <- median(numbers)
std_dev <- sd(numbers)

# Create a table of results
data.frame(
  Statistic = c("Mean", "Median", "Standard Deviation"),
  Value = c(mean_value, median_value, std_dev)
)

```

	Statistic	Value
1	Mean	39.50000
2	Median	32.50000
3	Standard Deviation	29.52118

3.3 Basic Charts

Creating visualizations is straightforward. Here's a simple bar chart:

```

# Create sample data
enrollment <- data.frame(
  College = c("Arts & Sciences", "Business", "Engineering", "Education", "Health Sciences"),
  Students = c(5234, 3211, 2543, 1432, 2211)
)

# Create a bar chart
ggplot(enrollment, aes(x = reorder(College, -Students), y = Students)) +
  geom_col(fill = "steelblue") +
  theme_minimal() +
  labs(
    title = "Enrollment by College",
    x = "College",
    y = "Number of Students"
  ) +
  # Add data labels on top of bars
  geom_text(aes(label = Students), vjust = -0.5) +
  # Rotate x-axis labels for better readability
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

```

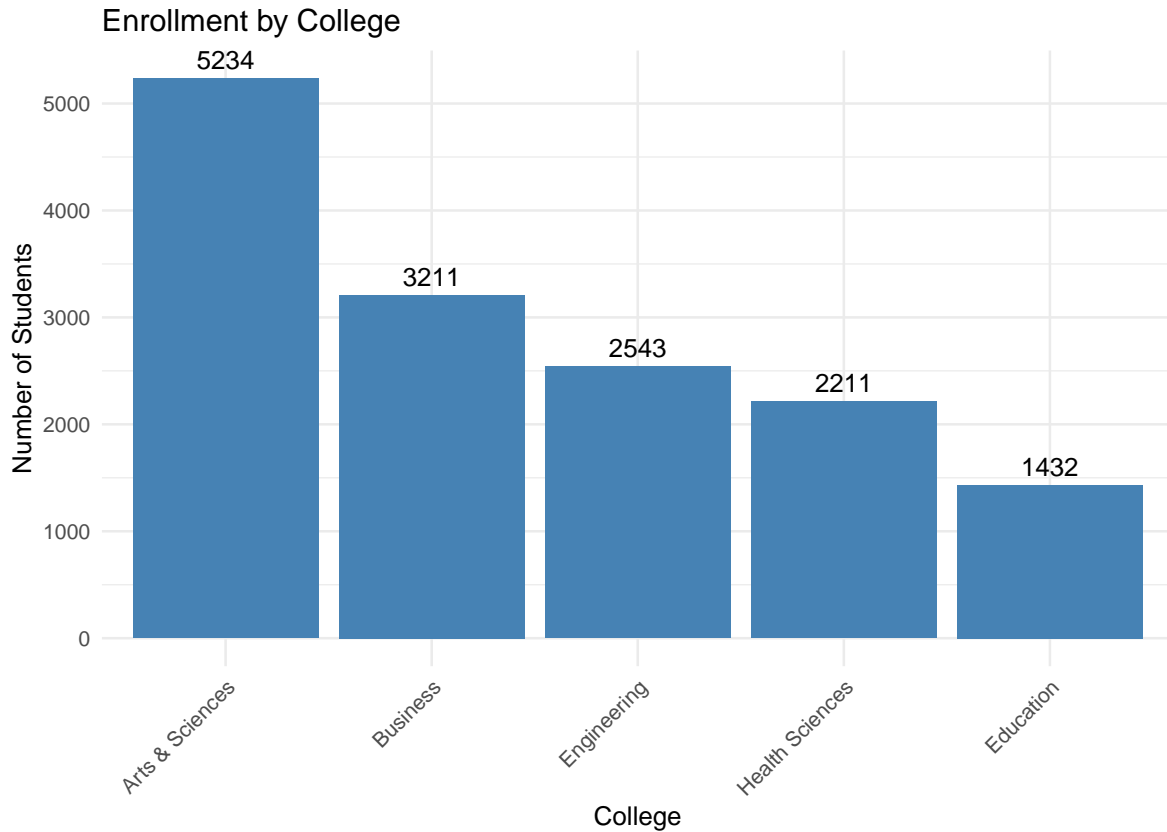


Figure 1: Sample enrollment data by college

3.4 Tables

You can create nicely formatted tables for different output formats:

```
# Create sample retention data
retention <- data.frame(
  College = c("Arts & Sciences", "Business", "Engineering", "Education", "Health Sciences"),
  "First Spring" = c(0.82, 0.85, 0.88, 0.79, 0.84),
  "Second Fall" = c(0.76, 0.78, 0.82, 0.72, 0.77),
  "Third Fall" = c(0.71, 0.74, 0.79, 0.68, 0.72)
)

# For HTML and PDF output, kable works well
kable(retention, digits = 2, caption = "Retention Rates by College")
```

Table 1: Retention Rates by College

College	First.Spring	Second.Fall	Third.Fall
Arts & Sciences	0.82	0.76	0.71
Business	0.85	0.78	0.74
Engineering	0.88	0.82	0.79
Education	0.79	0.72	0.68
Health Sciences	0.84	0.77	0.72

For Word output, the flextable package provides better control:

```
flextable(retention) %>%
  colformat_double(j = 2:4, digits = 2) %>%
  autofit() %>%
  add_footer_lines("Source: Office of Institutional Research") %>%
  theme_vanilla()
```

College	First.Spring	Second.Fall	Third.Fall
Arts & Sciences	0.82	0.76	0.71
Business	0.85	0.78	0.74
Engineering	0.88	0.82	0.79
Education	0.79	0.72	0.68
Health Sciences	0.84	0.77	0.72
Source: Office of Institutional Research			

4 Creating Parameterized Reports

4.1 Setting Up Parameters

Parameters allow you to create one template that generates many different outputs. Parameters are defined in the YAML header:

```
---
title: "College of {$college} Fact Sheet"
params:
  college: "Arts & Sciences"
```

```
academic_year: 2024
---
```

4.2 Using Parameters in Your Document

You can refer to parameters in both text and code:

```
# Define sample parameters (in a real parameterized document, these would come from the YAML)
params <- list(
  college = "Engineering",
  academic_year = 2024
)

# Display parameter value in text using inline R code
cat(paste("This report shows data for the", params$college, "college in the", params$academic_year, "academic year."))
```

This report shows data for the Engineering college in the 2024 academic year.

```
# Use parameters in data filtering
colleges <- data.frame(
  College = c("Arts & Sciences", "Business", "Engineering", "Education", "Health Sciences"),
  Students = c(5234, 3211, 2543, 1432, 2211),
  Year = rep(2024, 5)
)


# Filter data based on parameter
filtered_data <- colleges %>%
  filter(College == params$college, Year == params$academic_year)

# Show the filtered data
filtered_data
```

	College	Students	Year
1	Engineering	2543	2024

4.3 Rendering with Parameters

In RStudio, you can render with parameters by:

1. Clicking the  next to the “Render” button

2. Choosing “Render with Parameters...”
3. Entering your parameter values
4. Clicking “Render”

For batch rendering, you can use an R script:

```
library(quarto)

# List colleges to generate reports for
colleges <- c("Arts & Sciences", "Business", "Engineering")

# Create a report for each college
for(college in colleges) {
  quarto::quarto_render(
    "college_report.qmd", # Your template file
    output_file = paste0(college, "_Report.pdf"),
    execute_params = list(
      college = college,
      academic_year = 2024
    )
  )
}
```

5 Example: Enrollment Report

Let’s create a more complete example combining several features:

```
# Create sample enrollment trend data
set.seed(42)
years <- 2020:2024
colleges <- c("Arts & Sciences", "Business", "Engineering", "Education", "Health Sciences")

# Expand grid to create all combinations
trend_data <- expand.grid(
  Year = years,
  College = colleges
)

# Generate reasonably realistic enrollment numbers with some trends
base_enrollments <- c(6000, 3500, 2800, 1800, 2700)
names(base_enrollments) <- colleges
```

```

# Create enrollment with modest growth for each college
trend_data$Enrollment <- sapply(1:nrow(trend_data), function(i) {
  college <- as.character(trend_data$College[i])
  year_index <- which(years == trend_data$Year[i])

  # Base value plus growth and random noise
  base <- base_enrollments[college]
  growth <- base * 0.02 * (year_index - 1) # 2% growth per year
  noise <- rnorm(1, 0, base * 0.01) # Small random component

  # Special case: make Education decline slightly
  if (college == "Education") {
    growth <- -growth * 0.5
  }

  return(round(base + growth + noise))
})

```

5.1 Enrollment Trends Visualization

```

# Create the line chart
ggplot(trend_data, aes(x = Year, y = Enrollment, color = College, group = College)) +
  geom_line(size = 1.2) +
  geom_point(size = 3) +
  scale_y_continuous(labels = scales::comma) +
  scale_x_continuous(breaks = years) +
  theme_minimal() +
  labs(
    title = "Enrollment Trends by College (2020-2024)",
    y = "Total Enrollment",
    x = "Academic Year"
  ) +
  # Add data labels for 2024
  geom_text(
    data = trend_data %>% filter(Year == 2024),
    aes(label = Enrollment),
    nudge_x = 0.2,
    size = 3.5
  ) +
  theme(

```

```

legend.position = "bottom",
legend.title = element_blank(),
panel.grid.minor = element_blank()
)

```

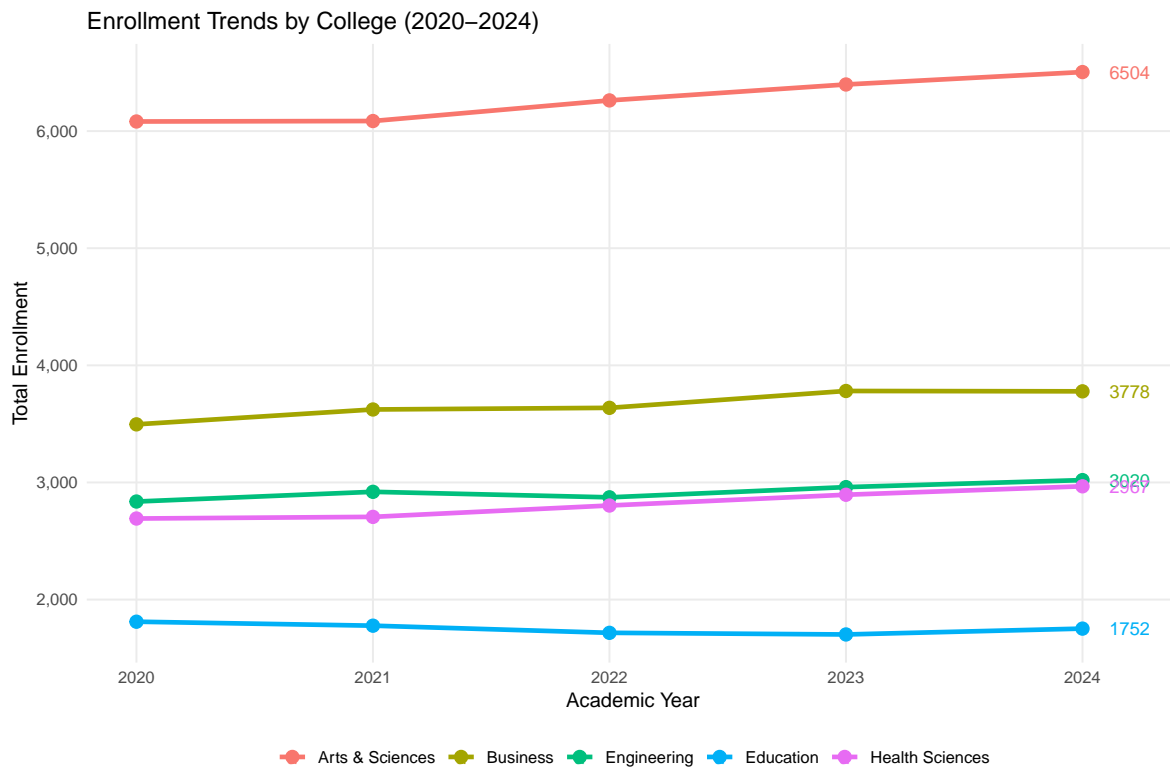


Figure 2: Five-Year Enrollment Trends by College

5.2 Enrollment Summary Table with Calculations

```

# Calculate summary statistics
summary_stats <- trend_data %>%
  group_by(College) %>%
  summarize(
    `2024 Enrollment` = Enrollment[Year == 2024],
    `2020 Enrollment` = Enrollment[Year == 2020],
    `Change` = `2024 Enrollment` - `2020 Enrollment`,
    `Percent Change` = ((`2024 Enrollment` - `2020 Enrollment`) / `2020 Enrollment`) * 100
  )

```

```

) %>%
  arrange(desc(`2024 Enrollment`))

# Create a nicely formatted table
flextable(summary_stats) %>%
  colformat_double(j = 5, digits = 1) %>%
  bg(j = 4, bg = function(x) {
    ifelse(x > 0, "#e6f3e6", ifelse(x < 0, "#f7e6e6", "white"))
  }) %>%
  bg(j = 5, bg = function(x) {
    ifelse(x > 0, "#e6f3e6", ifelse(x < 0, "#f7e6e6", "white"))
  }) %>%
  bold(j = 1) %>%
  add_footer_lines("Source: Office of Institutional Research") %>%
  autofit()

```

College	2024 Enrollment	2020 Enrollment	Change	Percent Change
Arts & Sciences	6,504	6,082	422	6.9
Business	3,778	3,496	282	8.1
Engineering	3,020	2,837	183	6.5
Health Sciences	2,967	2,692	275	10.2
Education	1,752	1,811	-59	-3.3

Source: Office of Institutional Research

6 Automation Techniques

6.1 File Organization

For efficient reporting workflows, organize your files in a clear structure:

```

InstitutionalResearch/
  Templates/
    college_factsheet.qmd
    enrollment_report.qmd
    generate_reports.R

```

```
Reports/  
  2024_Spring/  
    enrollment_summary.pdf  
    college_factsheets/  
  2024_Fall/
```

6.2 Automation Script

Here's a simple script for automatically generating multiple reports:

```
# Simple script to generate multiple reports  
library(quarto)  
  
# List the colleges you want to create reports for  
colleges <- c("Arts & Sciences", "Business", "Engineering")  
academic_year <- 2024  
  
# Create a folder for all the reports  
output_folder <- paste0("Factbook_", academic_year)  
dir.create(output_folder, showWarnings = FALSE)  
  
# First, create the main enrollment report  
print("Creating main enrollment report...")  
quarto::quarto_render(  
  "enrollment_report.qmd",  
  output_file = paste0(output_folder, "/main_enrollment_report.pdf")  
)  
  
# Then create a report for each college  
for (college in colleges) {  
  # Show progress in the console  
  print(paste("Creating fact sheet for", college))  
  
  # Make a simple filename  
  clean_name <- gsub(" |&", "_", tolower(college))  
  filename <- paste0(output_folder, "/", clean_name, "_factsheet.pdf")  
  
  # Create the report with parameters  
  quarto::quarto_render(  
    "college_fact_sheet.qmd",  
    output_file = filename,  
    execute_params = list(  

```

```

        college = college,
        academic_year = academic_year
    )
}

print(" All reports created successfully!")

```

6.3 Scheduling (Windows)

For Windows users, you can schedule your automation script using Task Scheduler:

1. Create a .bat file with the following content:

```
"C:\Program Files\R\R-4.2.1\bin\Rscript.exe" "C:\path\to\your\generate_reports.R"
```

2. Schedule this using Windows Task Scheduler to run monthly or as needed

7 Troubleshooting Tips

7.1 Common Issues and Solutions

Here are solutions to common problems:

7.1.1 PDF Output Problems

If you encounter errors with PDF rendering:

```

# Install TinyTeX (a lightweight LaTeX system)
install.packages("tinytex")
tinytex::install_tinytex()

```

7.1.2 Table Formatting in Word

For better-looking tables in Word documents, use flextable instead of kable:

```

library(flextable)
flextable(your_data) %>% autofit()

```

7.1.3 Data Loading Errors

Always check file paths and preview your data:

```
# Print working directory to check where R is looking for files
print(getwd())

# Preview first few rows of CSV to check structure
head(read.csv("your_file.csv"))
```

8 Resources for Learning More

8.1 Beginner-Friendly Resources

- [Posit’s “Getting Started with Quarto”](#)
- [R for the Rest of Us](#)
- [Introduction to R for Excel Users](#)

8.2 Video Tutorials

- [RStudio YouTube Channel](#)
- [How to Create Your First Quarto Document](#)

8.3 Getting Help

- [Posit Community Forum](#)
- [Stack Overflow with ‘quarto’ tag](#)

9 Thank You!

- Nicole Ross, Associated Director, nmvross@colostate.edu
- Lee Tyson Senior Research Analyst, lkytson@colostate.edu