Project 2: Retirement Withdrawal Simulator

Stat 133, Fall 2021

About

The purpose of this assignment is to create a **retirement withdrawal simulator using a shiny app.** You will have to create a shiny app, publish it on RStudio `shinyapps.io`, and record a short video—of 3 mins max—showing how your app works and how to make sense of the displayed information.

We are assuming that you have gone through the learning materials covered in weeks 7, 8, 9 and 10.

Please carefully read this document in its entirety before writing any code.

1) The Four Percent Rule

This section aims to provide context about the project and therefore is purely informative; no need to calculate or code anything for this part.

In the *Personal Finance* field, there is a famous rule of thumb known as the **Four Percent Rule**. This is a tool commonly used to help a person avoid running out of money in retirement.

What does the "4% rule" say? Depending on what source you look at, you may find slightly different versions of this rule. However, most of the sources will say something along these lines:

> "You can comfortably withdraw 4% of your savings in your first year of retirement and adjust that amount for inflation for every subsequent year without risking running out of money for at least 30 years."

For this rule to be applicable, we need two major assumptions:

- your investment portfolio (your savings) contains about 60% stocks and 40% bonds (this is commonly referred to as a "60-40 portfolio").
- you'll keep your spending level constant throughout retirement.

2) Motivation

This section provides more contextual information; nothing to calculate here.

Suppose you are helping a financial planner to design a shiny app that she will use when talking about retirement with her clients. The overall goal of this app is to help her clients understand the behavior of their portfolios during this phase of their lives.

Imagine a potential client, which we'll call Paco, who is 60 years old. To make the math easy, let's assume that Paco has saved—throughout his professional career—a retirement portfolio of $1,000,000. And he is now considering the possibility of retiring from his corporate job. If he decides to retire, then he will have no income other than what his portfolio returns.

Let's also assume that Paco's portfolio has an average annual rate of return of 6%. If Paco relies on the 4% rule as a starting point, he will be taking out $40,000 from his portfolio for the first year, and then he will continue to withdraw $40,000 (**adjusted for inflation**) in subsequent years.

If we ignore inflation and portfolio compounding, withdrawing $40,000 each year should allow his $1,000,000 portfolio to last him approximately 25 years. But what if we want to consider the impact of both 1) the rate of return and 2) the rate of inflation?

3) Variable Return and Inflation Rates

This third section describes contextual information that will guide you in coding your shiny app.

We are going to assume that both 1) the **rate of return** and 2) the **rate of inflation** are not constant, rather they will vary every year.

3.1) Variable Rate of Return

We are going to assume that annual rates of return, denoted r_t, follow a **normal distribution** with:

- **average annual return rate**: $\mu_r = 9\%$
- **average annual return volatility**: $\sigma_r = 10\%$

This can be written in compact notation as:

$$\text{annual return at year t} \quad \longrightarrow \quad r_t \sim \mathcal{N}(\mu_r = 0.09,\ \sigma_r = 0.10)$$

BTW: with your shiny app, you will be able to modify these values and get an idea of the expected returns under different return and volatility conditions.

3.2) Variable Rate of Inflation

We are also going to assume a Normal distribution on the annual rates of inflation, denoted as i_t, with:

- **average annual inflation rate**: $\mu_i = 3\%$
- **average annual inflation volatility**: $\sigma_i = 3.5\%$

This can be written in compact notation as:

$$\text{annual inflation at year t} \quad \longrightarrow \quad i_t \sim \mathcal{N}(\mu_i = 0.03,\ \sigma_i = 0.035)$$

BTW: with your shiny app, you will be able to modify these values and get an idea of the expected inflations under different conditions.

4) Simulation with variable rates of return

This fourth section describes examples that will guide you in coding your shiny app.

Let's bring back Paco's withdrawal retirement scenario with an initial amount of $1,000,000. And let's use a withdrawal rate of 4%.

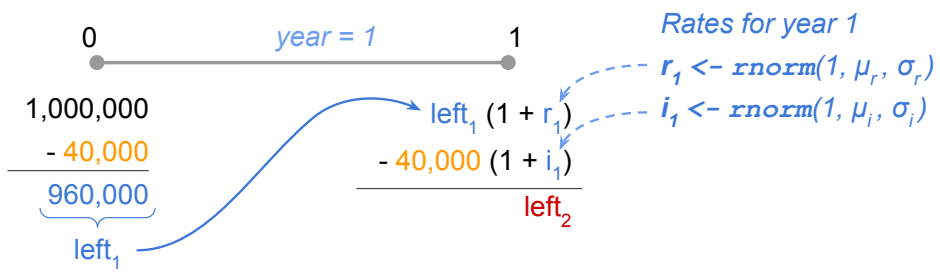
Starting at year $t = 0$

At time $t = 0$ (beginning of Paco's retirement), he withdraws

$$\$1,000,000 \times 0.04 = \$40,000$$

This is the amount of money that he is supposed to spend during his first year of retirement. This means that his portfolio will have a remainder of $\$1,000,000 - \$40,000 = \$960,000$. Let's call this amount left_1 (what is left in the portfolio for year 1). See image below.

$$\text{left}_1 = \$1,000,000 - \$40,000 = \$960,000$$



At the end of the first year $t = 1$

At the end of year $t = 1$, how much money does Paco need to withdraw for his second year of retirement?

On one hand, Paco's portfolio will have a return r_1 during the first year. Thus, at the end of year one, his portfolio will amount to:

$$\text{portfolio's balance at } t = 1 \quad \longrightarrow \quad \text{left}_1(1 + r_1)$$

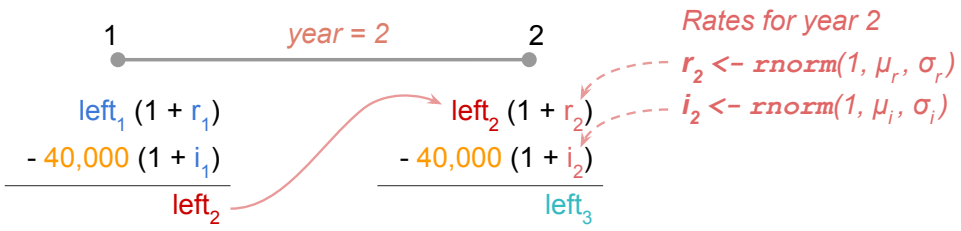
On the other hand, he will take out \$40,000 (adjusting for inflation) to be spent during the second year.

$$\text{amount withdrawn at } t = 1 \quad \longrightarrow \quad 40,000(1 + i_1)$$

Consequently, the remainder of his portfolio, left_2, will be:

$$\text{left}_2 = \text{left}_1(1 + r_1) - 40,000(1 + i_1)$$

This is illustrated in the following diagram:



At the end of the second year $t = 2$

At the end of the second year ($t = 2$), the balance in Paco's portfolio is

$$\text{portfolio's balance at } t = 2 \quad \longrightarrow \quad \text{left}_2(1 + r_2)$$

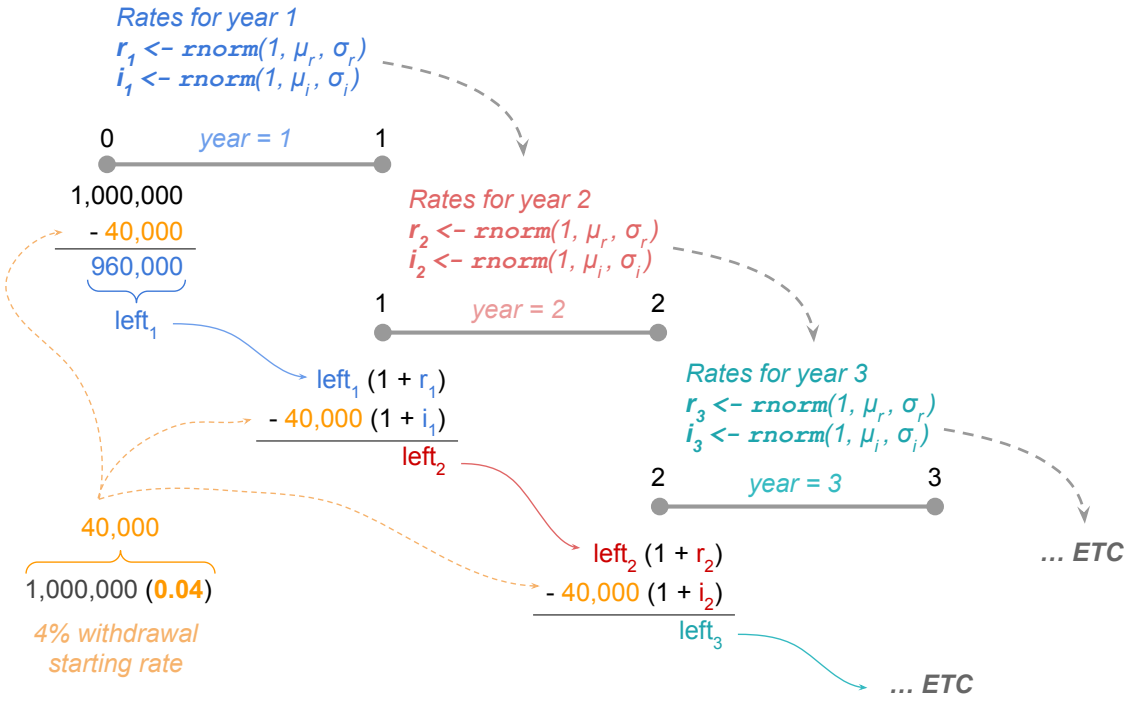
from which he will withdraw \$40,000—adjusted for inflation—to be spent in year three

$$\text{amount withdrawn at } t = 2 \quad \longrightarrow \quad 40,000(1 + i_2)$$

Therefore, the remainder of his portfolio, left_3, will be:

$$\text{left}_3 = \text{left}_2(1 + r_2) - 40,000(1 + i_2)$$

Here's a conceptual diagram illustrating the flow of the retirement portfolio during the first three years:



As you can tell from the figure above, annual rates of return and inflation need to be randomly generated—via `rnorm()`. This is the function in R that allows you to generate random numbers from a Normal distribution.

4.1) Example: Running 3 simulations

One of the goals of this project is to run various simulations that give you a theoretical idea for a certain withdrawal rate.

Say the financial planner assumes a 40 year period (till Paco reaches age 100), with an initial portfolio of \$1,000,000, that returns on average 6% per year, and that the average annual inflation rate is 3%.

$$r_t \sim \mathcal{N}(\mu_r = 6\%, \sigma_r = 10\%) \qquad \text{and} \qquad i_t \sim \mathcal{N}(\mu_i = 3\%, \sigma_i = 3.5\%)$$

Here's a toy example with the annual portfolio balance of 3 simulations (see table below).

The first row (year=0) shows the initial portfolio amount of \$1,000,000. The last row (year=5) shows the balance of the portfolio at the end of year 5. Column 1 corresponds to the 1st simulation, column 2 to the 2nd simulation, etc.

```
      sim1       sim2       sim3  year
1  1000000  1000000.0  1000000.0     0
2  1052806  1061443.5   832570.1     1
3  1111016  1065067.5   826347.0     2
4  1058262  1176813.0   669096.4     3
5  1174549   958862.1   686986.5     4
6  1141468   968977.2   648803.2     5
```

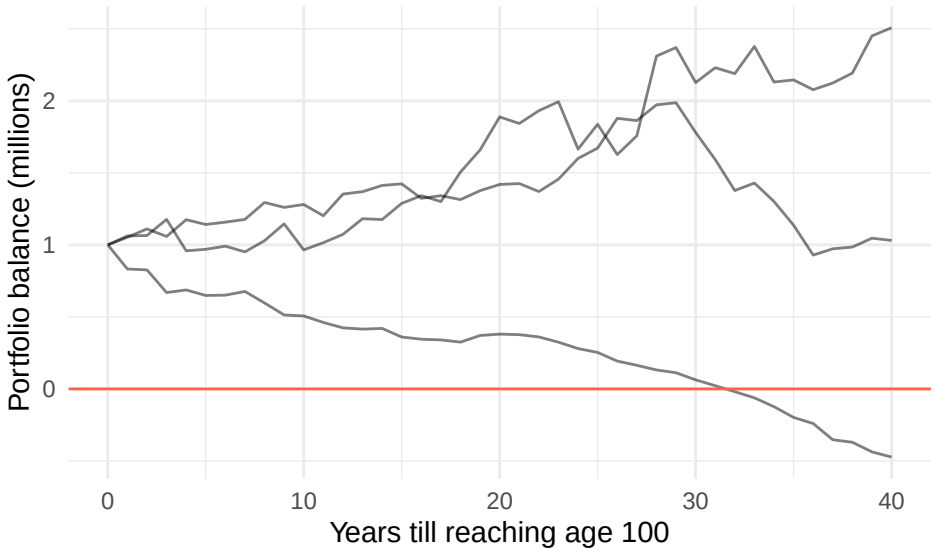
To graph timelines with `"ggplot2"`, it is more convenient to arrange the content in what is called "tall" (or "long") format. One option to do this is with the `pivot_longer()` function from the *tidyverse* package `"tidyr"`:

Assuming that the data displayed above is in an object `portfolios`, we can make it "longer" like so (a few rows shown):

```
sim_dat = pivot_longer(
  portfolios,
  cols = starts_with("sim"),
  names_to = c("simulation"),
  values_to = "amount")
```

```
## # A tibble: 15 x 3
##     year simulation   amount
##    <int> <chr>         <dbl>
##  1     0 sim1       1000000
##  2     0 sim2       1000000
##  3     0 sim3       1000000
##  4     1 sim1       1052806.
##  5     1 sim2       1061443.
##  6     1 sim3        832570.
##  7     2 sim1       1111016.
##  8     2 sim2       1065068.
##  9     2 sim3        826347.
## 10     3 sim1       1058262.
## 11     3 sim2       1176813.
## 12     3 sim3        669096.
## 13     4 sim1       1174549.
## 14     4 sim2        958862.
## 15     4 sim3        686987.
```

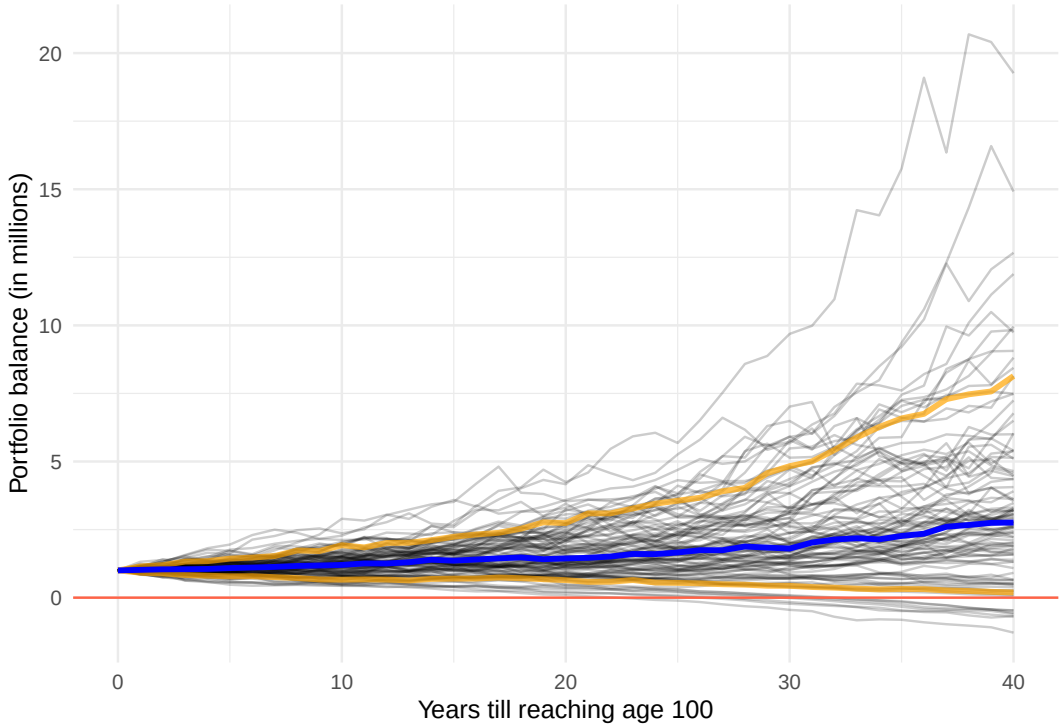
Having the data in this "tall' format, we can easily graph it with `ggplot()` and friends (*btw:* by now you should be abe to figure out the `ggplot()` command that produces the following graph):



As you can tell from the figure above, Paco's retirement portfolio does very well in two of the simulations: he is supposed to have a considerable amount left in his portfolio by the time he is 100 years old. However, in one of the simulations his portfolio "only" lasts 31 years before getting depleted.

4.2) Example: Running 100 simulations

What if we decide to increase the number of simulations to say 100? Here's what could happen.



In an attempt to provide a better way to observe the general behavior of the simulations, we can add reference lines for the 10-th and 90-th percentiles (colored in yellow), as well as the median line (blue line).

What else would you do to improve the visualization of so many lines?

Come up with more ways to improve the visualization and implement them in your shiny app. Also, keep in mind good practices for data visualization (title, subtitle, axis labels, scales, tick marks, colors, etc.

Not shown, but something of interest, could be the computation of numeric summaries that help us make sense of the obtained results from 100 simulations. For instance, we could calculate some quantiles of the portfolio balance along the retirement period. Or we could count in how many simulations Paco ran out of money.

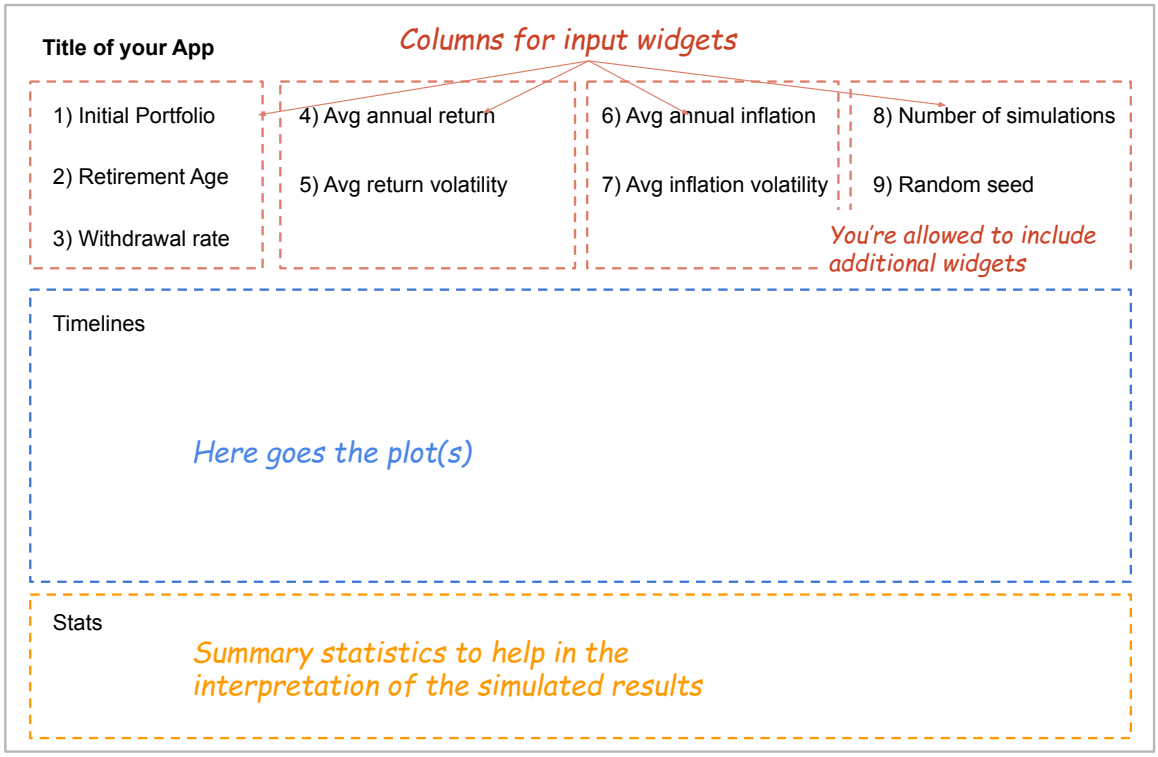
What else could you compute to summarize the "typical" behavior (e.g. measures of center) and the "typical" variability (e.g. measures of spread) of the simulations?

Come up with more numeric summaries and related information to help you understand the simulated results; and implement them in your shiny app.

5) Shiny App

The main goal of this assignment is to create a shiny app to run simulations, and to display the output of what could happen when using a certain withdrawal rate, under a given set of rates of return and inflation.

Your app should have a layout like the following diagram (see specifications below). You can find a template R script file `app.R` in the folder containing this pdf of instructions (bCourses).



As you can tell from the above diagram, the layout of the app involves four distinctive sections—see template file `app.R`:

- **title**: main title for your app (give it a meaningful name)
- **input widgets**: widgets arranged in four columns
- **plot**: an output graph to display the yearly balances
- **stats**: an output area (e.g. for a table, text, etc) to display numeric summaries.

5.1) Input widgets

Your app should include widgets for the inputs listed below. You are given freedom to decide what type of widgets (e.g. `numericInput()`, `sliderInput()`, `radioButtons()`, etc) to include in the shiny app. You are also allowed to include more widgets in addition to the required ones.

1) the **initial portfolio amount**, default value of 1,000,000
2) the **retirement age**, default value of 60 (but you need to be able to specify different age values such as 30, 45, 70, ...). Regardless of the input retirement age, assume that all simulations will be calculated up to reaching 100 years of age. For example, if retirement is 40, then the calculations will be performed for a 100 - 40 = 60 year time period.
3) the **withdrawal rate**, default value of 4%; this rate is used to determine the withdrawn amount every year (which will be adjusted for inflation)

4) the **average annual rate of return**: this is the mean μ_r to be used in `rnorm()` for generating rates in each year; default value of 10%

5) the **average return volatility**: this is the std deviation σ_r to be used in `rnorm()` for generating return rates in each year; default value of 18%.

6) the **average inflation rate**: this is the mean μ_i to be used in `rnorm()` for generating inflation rates in each year; default value of 3%

7) the **average inflation volatility**: this is the std deviation σ_i to be used in `rnorm()` for generating inflation rates in each year; default value of 3.5%.

8) the **number of simulations**, default value of 50.

9) the **value of the random seed**, this is the value to be passed to `set.seed()`. Default value of 12345. In the code of your simulations, you only need to invoke `set.seed()` once. This seed will be used by R to generate new runs of random numbers. Changing this numeric value runs a new set of simulations.

You are allowed to include more widgets that enhance the user experience, or that provide deeper insight into understanding the behavior of a simulated retirement withdrawal scheme.

6) Submission

1) **R file**: You will have to submit the source `app.R` file (do NOT confuse with an `Rmd` file) containing the code of your app.

2) **Link of published app**: You will also have to submit the link of your published app in shinyapps.io (the free version). This publication process is fairly straightforward, and you can watch Garret Grolemund's video on how to do this: https://vimeo.com/rstudioinc/review/131218530/212d8a5a7a/#t=30m35s. Share the link with us in the comments section of the submission in bCourses.

3) **Video**: In addition to the `app.R` file and the link of your published app, you will also have to record a video—maximum length of 3 minutes—in which you show us your published shiny app, how to use it, and a description of its outputs. Upload your video file to your Berkeley Box account, and share the **public link** with us in the comments section of the submission in bCourses.

3a) Ideas for your video. You can tell us:

- How a more conservative portfolio (lower return with less volatility) or a more aggressive one (higher return with more volatility) behave under the 4% rule.
- How changing the withdrawal rate (say to 5% or 6% or higher) affect the longevity of a given portfolio.

- How increasing or decreasing the time horizon affects the simulation? For example, how many years will a certain portfolio last for someone persuing early retirement at age 40?
- What could be some of the (main) limitations when trying to use this simulator in "real" life?

4) **Important:** You do NOT have to submit any Rmd or html files this time. Also, **we will not accept any content sent by email**. We will only grade the `app.R` file submitted to bCourses, the public link of the video in your Berkeley Box account, and the link of your app in `shinyapps.io`.

Resources and Warning

You may want to take a look at the Shiny gallery: https://shiny.rstudio.com/gallery/

BTW: you may find a retirement simulator shiny app in the above gallery. Keep in mind that the approach used behind this app to simulate withdrawing from a retirement portfolio is **different** from the apprach outlined for this Stat 133 project-2. While you can take inspiration from this app, **do not** attempt to replicate the code for your own shiny app.

Shiny widgets gallery:
https://shiny.rstudio.com/gallery/widget-gallery.html

Share you app with `shinyapps.io`:
https://vimeo.com/rstudioinc/review/131218530/212d8a5a7a/#t=30m35s

Of course, you can take a look at other apps displayed in the Shiny gallery to get some inspiration.

Appendix: About the 4% rule

- The Four Percent Rule is a rule of thumb used to determine how much a retiree should withdraw from a retirement account each year.
- The Four Percent Rule was created using historical data on stock and bond returns over the 50-year period from 1926 to 1976.
- The Four Percent Rule seeks to provide a steady income stream to the retiree while also maintaining an account balance that keeps income flowing through retirement.
- The Four Percent Rule helps financial planners and retirees set a portfolio's withdrawal rate.
- Life expectancy plays an important role in determining if this rate will be sustainable, as retirees who live longer need their portfolios to last longer, and medical costs and other expenses can increase as retirees age.