Resident Exam:
Coding with GitHub Copilot

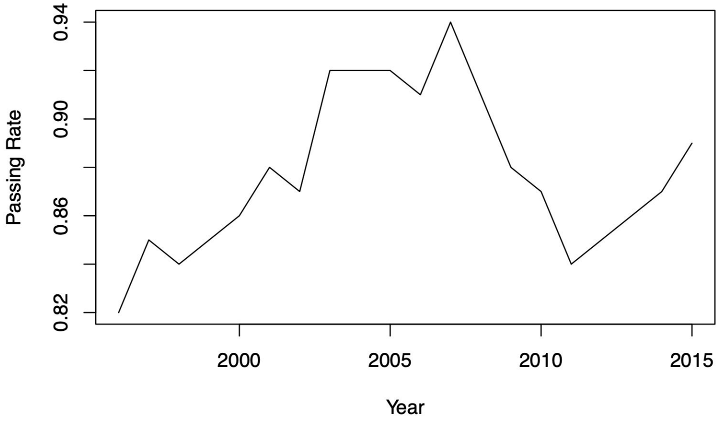
Group E: Ken Ye, Ejay Lin, Gorden Gao

Introduction

- Using GitHub Copilot as a digital assistant for data analysis.
- Research question: Impact of 2003 and 2011 reforms on pass rates for internal medicine residents.

Step 1: Exploratory Data Analysis (EDA)

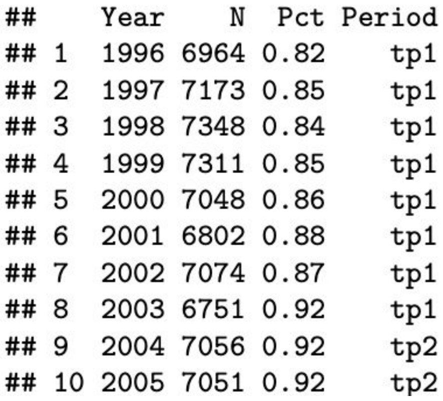
- Our dataset has columns called *Year* and *Pass* that indicate the year and the number of residents who passed the exam
- Through preliminary EDA, we plotted *Year* against *Pass* (passing rate) and performed an analysis on the distribution of Pass itself



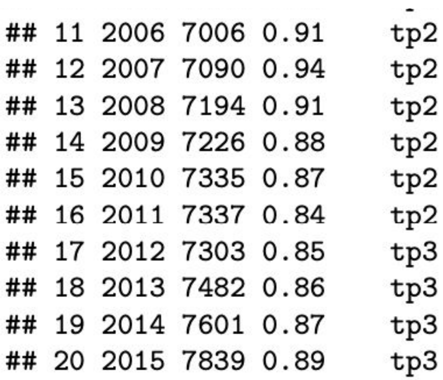
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.8200  0.8500  0.8700  0.8775  0.9100  0.9400 
```

Step 2: Generate Derived Variables

- We created a new column that classifies each year into one of three periods (tp1, tp2, tp3)



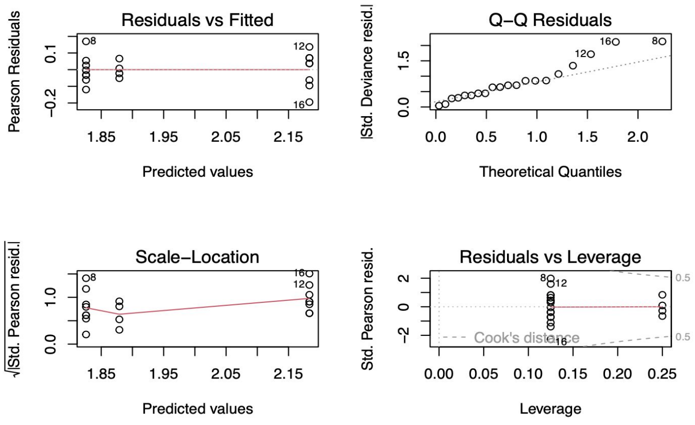
```
##    Year    N  Pct Period
## 1  1996 6964 0.82    tp1
## 2  1997 7173 0.85    tp1
## 3  1998 7348 0.84    tp1
## 4  1999 7311 0.85    tp1
## 5  2000 7048 0.86    tp1
## 6  2001 6802 0.88    tp1
## 7  2002 7074 0.87    tp1
## 8  2003 6751 0.92    tp1
## 9  2004 7056 0.92    tp2
## 10 2005 7051 0.92    tp2
```

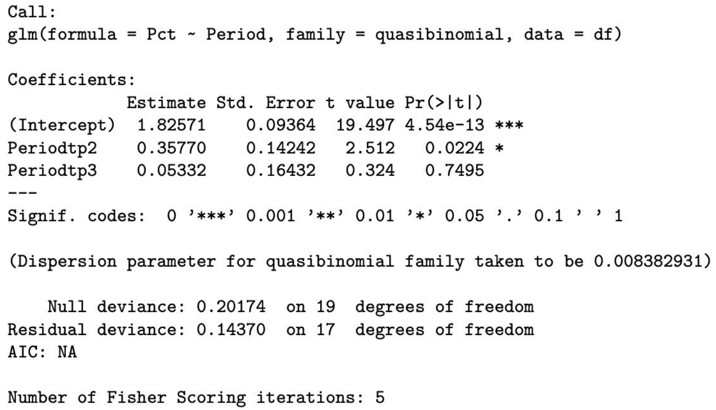


```
                               -
## 11 2006 7006 0.91    tp2
## 12 2007 7090 0.94    tp2
## 13 2008 7194 0.91    tp2
## 14 2009 7226 0.88    tp2
## 15 2010 7335 0.87    tp2
## 16 2011 7337 0.84    tp2
## 17 2012 7303 0.85    tp3
## 18 2013 7482 0.86    tp3
## 19 2014 7601 0.87    tp3
## 20 2015 7839 0.89    tp3
```

Step 3: Quasi-binomial Model

- We fitted a quasi-binomial model with the newly created *Period* variable and implemented diagnosis analysis

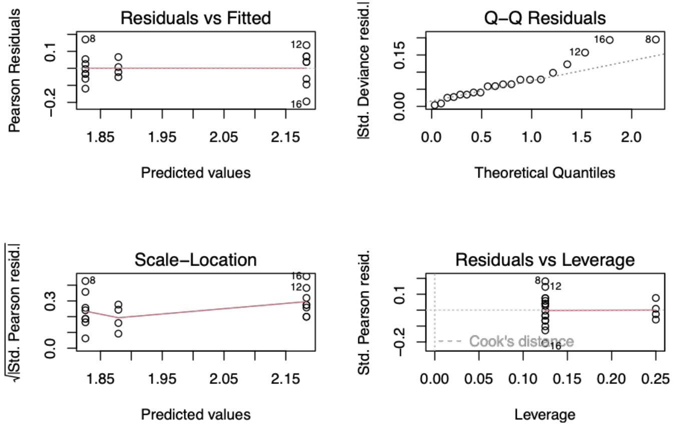


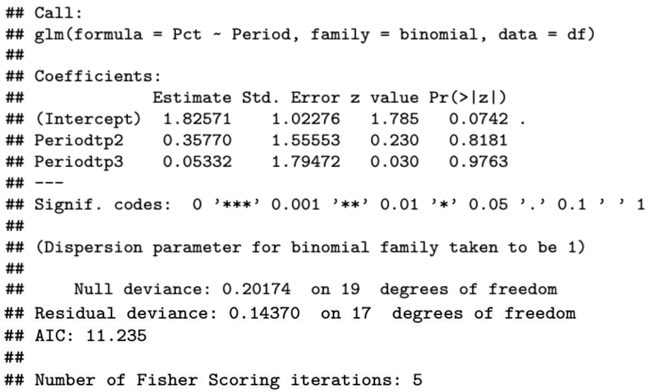


```
Call:
glm(formula = Pct ~ Period, family = quasibinomial, data = df)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.82571    0.09364  19.497 4.54e-13 ***
Periodtp2    0.35770    0.14242   2.512   0.0224 *
Periodtp3    0.05332    0.16432   0.324   0.7495
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

(Dispersion parameter for quasibinomial family taken to be 0.008382931)

    Null deviance: 0.20174  on 19  degrees of freedom
Residual deviance: 0.14370  on 17  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
```

Step 4: Generalized Linear Model (GLM)

- We fitted a generalized linear model (GLM) with the *Period* variable, implemented diagnostics and summary of model is provided

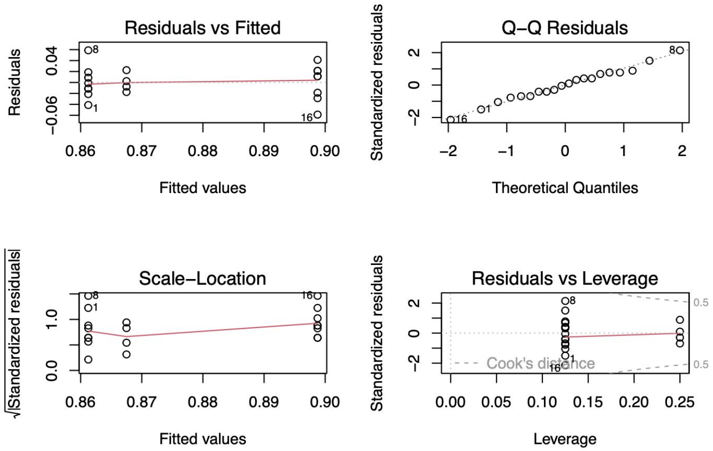


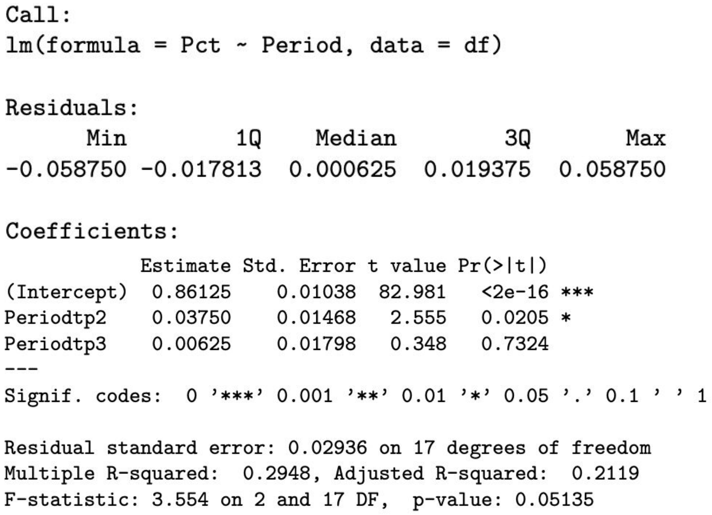


```
## Call:
## glm(formula = Pct ~ Period, family = binomial, data = df)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.82571    1.02276   1.785   0.0742 .
## Periodtp2    0.35770    1.55553   0.230   0.8181
## Periodtp3    0.05332    1.79472   0.030   0.9763
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 0.20174  on 19  degrees of freedom
## Residual deviance: 0.14370  on 17  degrees of freedom
## AIC: 11.235
##
## Number of Fisher Scoring iterations: 5
```

Step 5: Linear Model (LM)

- We fitted a linear regression model (LM) with the *Period* variable, implemented diagnostics, and summary of model is provided

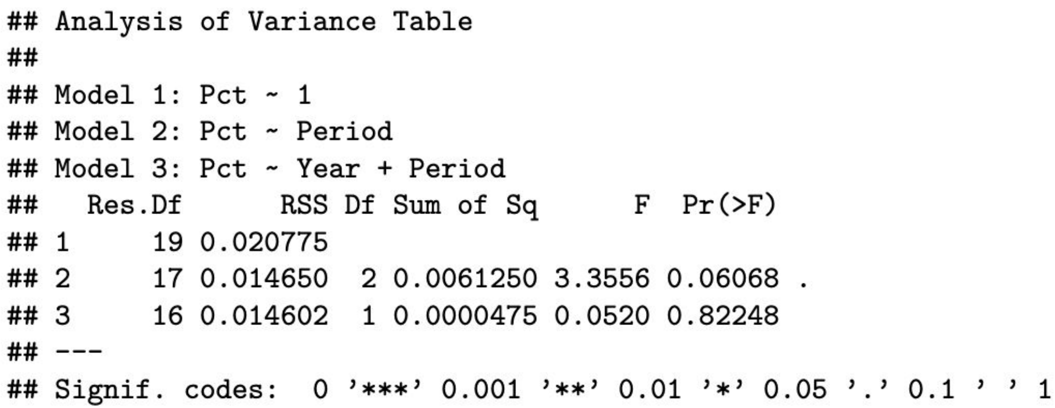




```
Call:
lm(formula = Pct ~ Period, data = df)

Residuals:
      Min        1Q    Median        3Q       Max
-0.058750 -0.017813  0.000625  0.019375  0.058750

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.86125    0.01038  82.981   <2e-16 ***
Periodtp2    0.03750    0.01468   2.555   0.0205 *
Periodtp3    0.00625    0.01798   0.348   0.7324
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 0.02936 on 17 degrees of freedom
Multiple R-squared:  0.2948, Adjusted R-squared:  0.2119
F-statistic: 3.554 on 2 and 17 DF,  p-value: 0.05135
```

Step 6 Hypothesis Tests

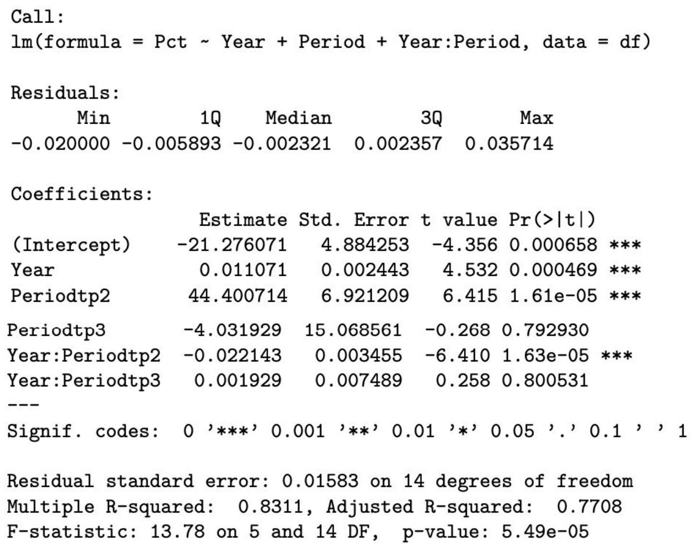
- We fitted three models and tested three hypotheses using F tests.



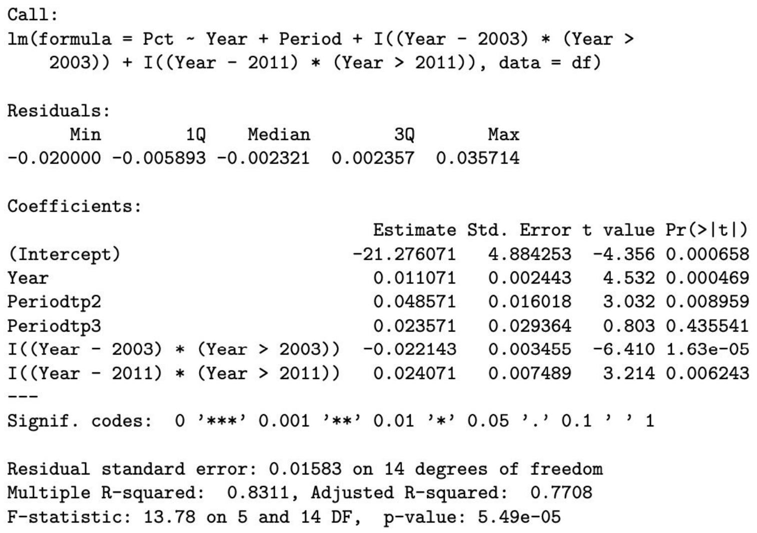
```
## Analysis of Variance Table
##
## Model 1: Pct ~ 1
## Model 2: Pct ~ Period
## Model 3: Pct ~ Year + Period
##   Res.Df      RSS Df Sum of Sq      F  Pr(>F)
## 1     19 0.020775
## 2     17 0.014650  2 0.0061250 3.3556 0.06068 .
## 3     16 0.014602  1 0.0000475 0.0520 0.82248
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

Step 6 Hypothesis Tests (Cont.)

- We tested the stepwise constant hypothesis and piecewise linear hypothesis



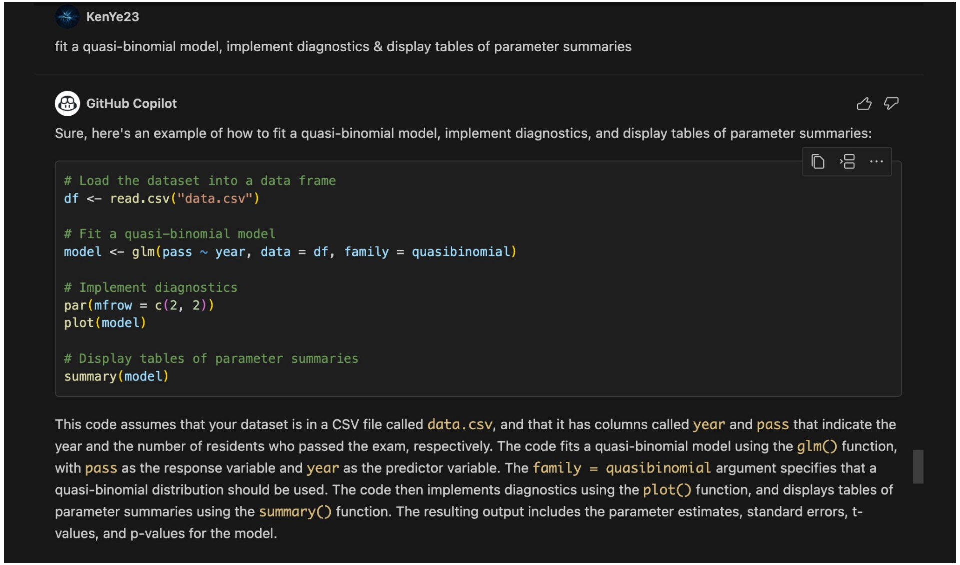
```
Call:
lm(formula = Pct ~ Year + Period + Year:Period, data = df)

Residuals:
      Min        1Q    Median        3Q       Max
-0.020000 -0.005893 -0.002321  0.002357  0.035714

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)    -21.276071   4.884253  -4.356 0.000658 ***
Year             0.011071   0.002443   4.532 0.000469 ***
Periodtp2       44.400714   6.921209   6.415 1.61e-05 ***
Periodtp3      -4.031929  15.068561  -0.268 0.792930
Year:Periodtp2 -0.022143   0.003455  -6.410 1.63e-05 ***
Year:Periodtp3  0.001929   0.007489   0.258 0.800531
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01583 on 14 degrees of freedom
Multiple R-squared:  0.8311, Adjusted R-squared:  0.7708
F-statistic: 13.78 on 5 and 14 DF,  p-value: 5.49e-05
```



```
Call:
lm(formula = Pct ~ Year + Period + I((Year - 2003) * (Year >
    2003)) + I((Year - 2011) * (Year > 2011)), data = df)

Residuals:
      Min        1Q    Median        3Q       Max
-0.020000 -0.005893 -0.002321  0.002357  0.035714

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -21.276071   4.884253  -4.356 0.000658
Year                             0.011071   0.002443   4.532 0.000469
Periodtp2                        0.048571   0.016018   3.032 0.008959
Periodtp3                        0.023571   0.029364   0.803 0.435541
I((Year - 2003) * (Year > 2003)) -0.022143   0.003455  -6.410 1.63e-05
I((Year - 2011) * (Year > 2011))  0.024071   0.007489   3.214 0.006243
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01583 on 14 degrees of freedom
Multiple R-squared:  0.8311, Adjusted R-squared:  0.7708
F-statistic: 13.78 on 5 and 14 DF,  p-value: 5.49e-05
```

What’s Effective?

- Efficient code generation and comment writing with specific instructions.
- Copilot’s ability to handle tasks like creating quasi-binomial models.



KenYe23

fit a quasi-binomial model, implement diagnostics & display tables of parameter summaries

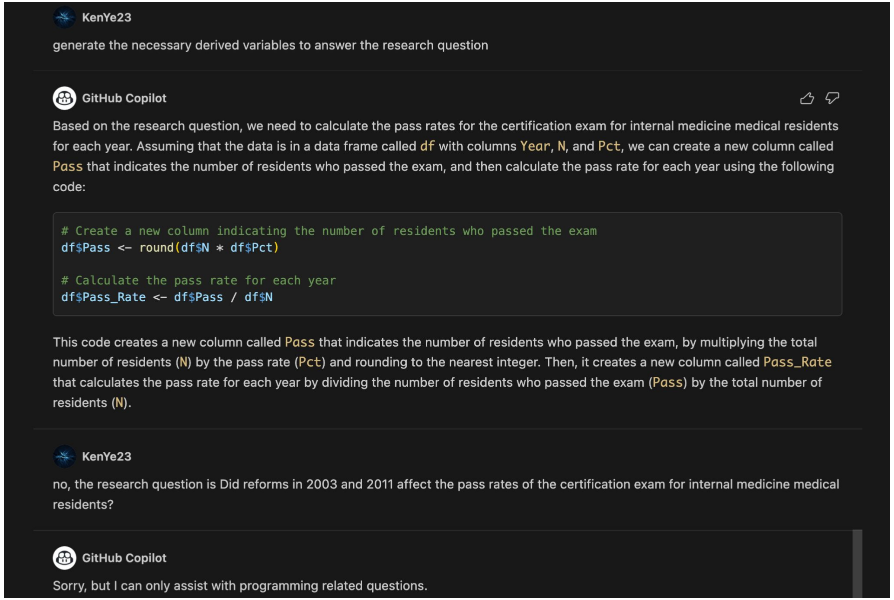
GitHub Copilot

Sure, here's an example of how to fit a quasi-binomial model, implement diagnostics, and display tables of parameter summaries:

```
# Load the dataset into a data frame
df <- read.csv("data.csv")

# Fit a quasi-binomial model
model <- glm(pass ~ year, data = df, family = quasibinomial)

# Implement diagnostics
par(mfrow = c(2, 2))
plot(model)

# Display tables of parameter summaries
summary(model)
```

This code assumes that your dataset is in a CSV file called `data.csv`, and that it has columns called `year` and `pass` that indicate the year and the number of residents who passed the exam, respectively. The code fits a quasi-binomial model using the `glm()` function, with `pass` as the response variable and `year` as the predictor variable. The `family = quasibinomial` argument specifies that a quasi-binomial distribution should be used. The code then implements diagnostics using the `plot()` function, and displays tables of parameter summaries using the `summary()` function. The resulting output includes the parameter estimates, standard errors, t-values, and p-values for the model.

What's Not Effective?



KenYe23

generate the necessary derived variables to answer the research question

GitHub Copilot

Based on the research question, we need to calculate the pass rates for the certification exam for internal medicine medical residents for each year. Assuming that the data is in a data frame called `df` with columns `Year`, `N`, and `Pct`, we can create a new column called `Pass` that indicates the number of residents who passed the exam, and then calculate the pass rate for each year using the following code:

```
# Create a new column indicating the number of residents who passed the exam
df$Pass <- round(df$N * df$Pct)

# Calculate the pass rate for each year
df$Pass_Rate <- df$Pass / df$N
```

This code creates a new column called `Pass` that indicates the number of residents who passed the exam, by multiplying the total number of residents (`N`) by the pass rate (`Pct`) and rounding to the nearest integer. Then, it creates a new column called `Pass_Rate` that calculates the pass rate for each year by dividing the number of residents who passed the exam (`Pass`) by the total number of residents (`N`).

KenYe23

no, the research question is Did reforms in 2003 and 2011 affect the pass rates of the certification exam for internal medicine medical residents?

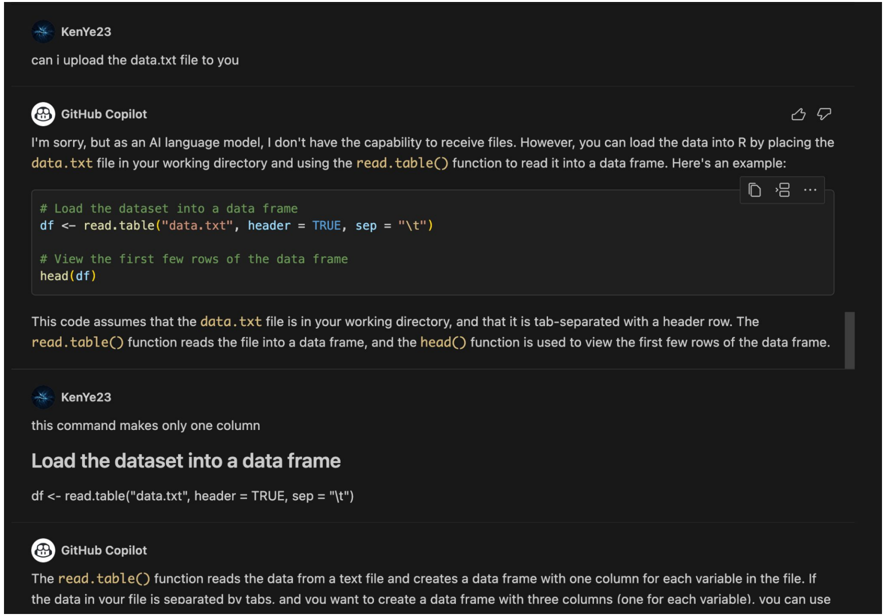
GitHub Copilot

Sorry, but I can only assist with programming related questions.

- Struggles with abstract or complex problems.
- Requires detailed and concrete instructions.

Response Accuracy & Efficiency

- Copilot’s inability to independently read and interpret data files.
- Challenges faced in initially loading data (code merged every row into a single column).
- Quick adjustments and correct code generation when provided specific details.



KenYe23

can i upload the data.txt file to you

GitHub Copilot

I'm sorry, but as an AI language model, I don't have the capability to receive files. However, you can load the data into R by placing the `data.txt` file in your working directory and using the `read.table()` function to read it into a data frame. Here's an example:

```
# Load the dataset into a data frame
df <- read.table("data.txt", header = TRUE, sep = "\t")

# View the first few rows of the data frame
head(df)
```

This code assumes that the `data.txt` file is in your working directory, and that it is tab-separated with a header row. The `read.table()` function reads the file into a data frame, and the `head()` function is used to view the first few rows of the data frame.

KenYe23

this command makes only one column

Load the dataset into a data frame

df <- read.table("data.txt", header = TRUE, sep = "\t")

GitHub Copilot

The `read.table()` function reads the data from a text file and creates a data frame with one column for each variable in the file. If the data in your file is separated by tabs, and you want to create a data frame with three columns (one for each variable), you can use

Evaluation

- Copilot’s strength in automating coding processes.
- Time-saving for routine coding tasks (e.g. plotting variables).
- Limitations in addressing complex or abstract questions.
- Recommendation: use GitHub Copilot primarily for coding tasks, not for generating research ideas.