

Using (not abusing) ChatGPT

Alison Ketz

Motivation: Why learn to code?

What's the point of learning to code when we now have AI to do it for us?

1. Why Code?
2. Uses of programming
3. Context

Benefits:

Quality
Reliability
Efficiency



ChatGPT is a B- (formerly C) student

"How do I implement Bayesian inference for estimating the mean of a normally distributed dataset in R using JAGS?"

```
# Load necessary libraries
library(rjags)
library(coda)

# Generate synthetic data
set.seed(42)
data <- rnorm(100, mean = 10, sd = 2)

# JAGS model
model_string <- "
model {
  for (i in 1:N) {
    y[i] ~ dnorm(mu, tau)
  }
  mu ~ dnorm(0, 0.01)
  tau <- pow(sigma, -2)
  sigma ~ dunif(0, 1)
}
```

```
# Prepare data for JAGS
data_list <- list(y = data, N = length(data))

# Initialize JAGS model
model <- jags.model(textConnection(model_string), data = data_list, n.chains = 3)

# Burn-in
update(model, 1000)

# Sample from the posterior
samples <- coda.samples(model, variable.names = c("mu", "sigma"), n.iter = 1000)

# Summarize results
summary(samples)
```

What is wrong with this?

ChatGPT is a B- (formerly C) student

"How do I implement Bayesian inference for estimating the mean of a normally distributed dataset in R using JAGS?"

```
# Load necessary libraries
library(rjags)
library(coda)

# Generate synthetic data
set.seed(42)
data <- rnorm(100, mean = 10, sd = 2)

# JAGS model
model_string <- "
model {
  for (i in 1:N) {
    y[i] ~ dnorm(mu, tau)
  }
  mu ~ dnorm(0, 0.01)
  tau <- pow(sigma, -2)
  sigma ~ dunif(0, 1)
}
```

```
# Prepare data for JAGS
data_list <- list(y = data, N = length(data))

# Initialize JAGS model
model <- jags.model(textConnection(model_string), data = data_list, n.chains = 3)

# Burn-in
update(model, 1000)

# Sample from the posterior
samples <- coda.samples(model, variable.names = c("mu", "sigma"), n.iter = 1000)

# Summarize results
summary(samples)
```



This prior is too restrictive

ChatGPT is a B- (formerly C) student

"How do I implement Bayesian inference for estimating the mean of a normally distributed dataset in R using JAGS?"

```
# Load necessary libraries
library(rjags)
library(coda)

# Generate synthetic data
set.seed(42)
data <- rnorm(100, mean = 10, sd = 2)

# JAGS model
model_string <- "
model {
  for (i in 1:N) {
    y[i] ~ dnorm(mu, tau)
  }
  mu ~ dnorm(0, 0.01)
  tau <- pow(sigma, -2)
  sigma ~ dunif(0, 1)
}
```

```
# Prepare data for JAGS
data_list <- list(y = data, N = length(data))

# Initialize JAGS model
model <- jags.model(textConnection(model_string), data = data_list, n.chains = 3)

# Burn-in
update(model, 1000)

# Sample from the posterior
samples <- coda.samples(model, variable.names = c("mu", "sigma"), n.iter = 1000)

# Summarize results
summary(samples)
```

Why is this fantastic?

What are some best practices?

- Ask **clear** and **precise** questions, **SPECIFICITY**
Restrict questions to small **limited** tasks
- Understand basic subject matter first
- Iterative querying - Step-by-step refinement
- Verification and validation -
Thoroughly test any code ChatGPT provides
- Documentation
- Privacy and security

What are some best practices?

- Ask **clear** and **precise** questions, **SPECIFICITY**
Restrict questions to small **limited** tasks
- Understand basic subject matter first
- Iterative querying - Step-by-step refinement
- Verification and validation -
Thoroughly test any code ChatGPT provides
- Documentation
- Privacy and security

Further Considerations

- Ethical Considerations
- Sensitive Data - Anything you upload or feed to ChatGPT, it will , in turn “own” and become part of it’s generating model
Trick: load a small snippet aka head(data-frame)
- Student - Will using it help you to learn?
- Laziness - Loops of wrong answers

How do I use it?

Small code chunks

Generalizing functions/analysis

Cleaning data

Formatting data

Translating between programming languages

Generating analysis ideas

Debugging esp. erroneous error messages

Step-by-step instructions for technical debt

Avoiding stack exchange

Finding podcasts for my kids

Is it worth it? Absolutely!

how do i make a for loop in R vs python?

```
# R example: Printing numbers from 1 to 5
for (i in 1:5) {
  print(i)
}

# R example: Iterating over a vector
fruits <- c("apple", "banana", "cherry")
for (fruit in fruits) {
  print(fruit)
}
```

```
# Python example: Printing numbers from 1 to 5
for i in range(1, 6):
  print(i)

# Python example: Iterating over a list
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
  print(fruit)
```

Key Differences:

- **Syntax:** Python uses colons (:) and indentation to define the blocks of code, whereas R uses curly braces ({ }) to enclose the code block.
- **Range Function:** Python's `range()` function is commonly used to generate sequences of numbers, whereas R uses the `:` operator or functions like `seq()`.
- **Printing:** In Python, `print()` is a function, while in R, `print()` is also a function but often used in combination with `paste()` for string concatenation.

Both languages offer flexible looping constructs, but the choice between them often depends on the specific requirements of the task and the preferred programming environment.

It's a model