

# Computing in the Statistics Curriculum: Lessons Learned from the Educational Sciences

Amelia McNamara, Andrew Zieffler, Matthew Beckman, Chelsey  
Legacy, Elle Butler Basner, Robert delMas, & V.N. Vimal Rao



UNIVERSITY OF  
**St. Thomas**



UNIVERSITY OF MINNESOTA  
**Driven to Discover®**



**PennState**

Computing is fundamental to contemporary statistical practice and scientific inquiry and should be explicitly taught

(ASA, 2017; Horton, 2015; NASEM, 2018; Nolan & Temple Lang, 2010)

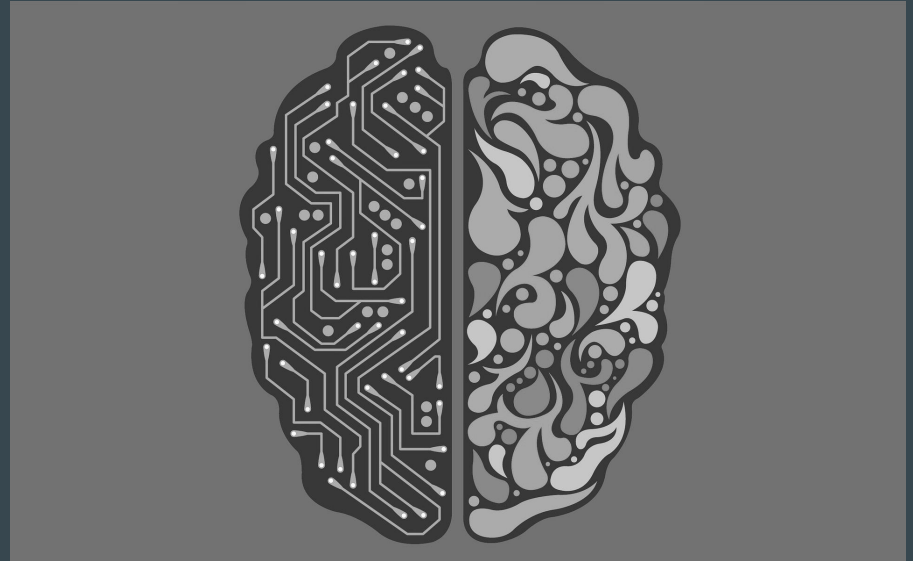


# Cognitive Load: Impediment to Learning

There is a finite amount of information that can be processed or stored in working memory at a time (cognitive load).

Different types of cognitive load (Hermans, 2021):

- *Intrinsic*: characteristics of the information being learned.
- *Extraneous*: the way information is presented.



# Statistical Computing and Cognitive Load

- Statistical computing adds cognitive load to the learning process in statistics (e.g., Woodard & Lee, 2021)
  - Computational considerations
    - Syntax
    - Syntactical structure
    - Debugging
    - Computational thinking
  - Coding seems to be difficult for many students
- We can work to manage and lessen cognitive load by thoughtfully considering the specific coding content we teach and how we teach it.

# Cognitive *Un*-loading: Make Purposeful Pedagogical Decisions

Pedagogical decisions need to be made about coding content.

- What logistical considerations do you need to account for?
- What will be taught (scope)?
- How will it be sequenced?



# Example Pedagogical Decisions

- How will students compute in the course?
  - Desktop / Cloud / Both
- Where and when do students need practice with code?
  - In-class / out-of-class
  - Individual / group
  - Templates / Blank documents (RMD, R script)
- How will coding be introduced in class?
  - Live coding/ Worked examples / Group activities
- What code content do you start the course with?
  - Data structures (e.g., vectors, data frames) / “Cake” (data visualization, EDA)

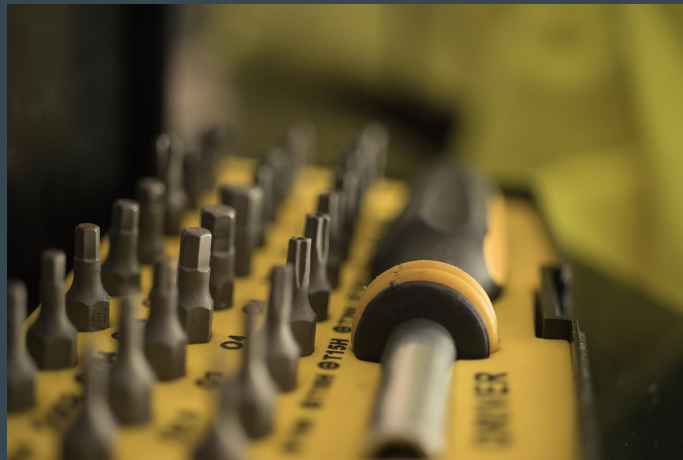
# Cognitive *Un*-loading: Use Consistent Syntactic Structure

Using code with the same syntactic structure (common grammar) lessens cognitive load

- Can focus on learning new functions (verbs) and their purpose

Some syntactic structures can emphasize the relationship between syntax and concepts

- E.g., Roles of variables



# Example Pedagogical Decision

- What syntax or packages will be used? In R:
  - base
  - formula (mosaic, ggformula, lattice)
  - tidyverse (ggplot2, dplyr)

## R Syntax Comparison : : CHEAT SHEET

### Dollar sign syntax

```
goal(data$x, data$y)
```

SUMMARY STATISTICS:

one continuous variable:

```
mean(mtcars$mpg)
```

one categorical variable:

```
table(mtcars$cyl)
```

two categorical variables:

```
table(mtcars$cyl, mtcars$am)
```

one continuous, one categorical:

```
mean(mtcars$mpg[mtcars$cyl==4])
```

```
mean(mtcars$mpg[mtcars$cyl==6])
```

```
mean(mtcars$mpg[mtcars$cyl==8])
```

PLOTTING:

one continuous variable:

```
hist(mtcars$displ)
```

one categorical variable:

```
barplot(mtcars$cyl)
```

two continuous variables:

```
plot(mtcars$displ, mtcars$mpg)
```

two categorical variables:

```
mosaicplot(table(mtcars$am, mtcars$cyl))
```

one continuous, one categorical:

```
histogram(mtcars$displ[mtcars$cyl==4])
```

```
histogram(mtcars$displ[mtcars$cyl==6])
```

```
histogram(mtcars$displ[mtcars$cyl==8])
```

```
boxplot(mtcars$displ[mtcars$cyl==4])
```

```
boxplot(mtcars$displ[mtcars$cyl==6])
```

```
boxplot(mtcars$displ[mtcars$cyl==8])
```

WRANGLING:

subsetting:

```
mtcars[mtcars$mpg>30, ]
```

making a new variable:

```
mtcars$efficient[mtcars$mpg>30] <- TRUE
```

```
mtcars$efficient[mtcars$mpg<=30] <- FALSE
```

SMITH COLLEGE

### Formula syntax

```
goal(y~x|z, data=data, group=~w)
```

SUMMARY STATISTICS:

one continuous variable:

```
mosaic: mean(~mpg, data=mtcars)
```

one categorical variable:

```
tally: tally(~cyl, data=mtcars)
```

two categorical variables:

```
mosaic: tally(cyl~am, data=mtcars)
```

one continuous, one categorical:

```
mosaic: mean(mpg~cyl, data=mtcars)
```

PLOTTING:

one continuous variable:

```
lattice: histogram(~displ, data=mtcars)
```

```
lattice: bplot(~displ, data=mtcars)
```

one categorical variable:

```
mosaic: bargraph(~cyl, data=mtcars)
```

two continuous variables:

```
lattice: xyplot(mpg~displ, data=mtcars)
```

two categorical variables:

```
mosaic: bargraph(~am, data=mtcars, group=cyl)
```

one continuous, one categorical:

```
lattice: histogram(~displ[cyl, data=mtcars]
```

```
lattice: bplot(cyl~displ, data=mtcars)
```

SMITH COLLEGE

### Tidyverse syntax

```
data %>% goal(x)
```

SUMMARY STATISTICS:

one continuous variable:

```
mtcars %>% dplyr::summarize(mean(mpg))
```

one categorical variable:

```
mtcars %>% dplyr::group_by(cyl) %>%
```

```
dplyr::summarize(n())
```

two categorical variables:

```
mtcars %>% dplyr::group_by(cyl, am) %>%
```

```
dplyr::summarize(n())
```

one continuous, one categorical:

```
mtcars %>% dplyr::group_by(cyl) %>%
```

```
dplyr::summarize(mean(mpg))
```

PLOTTING:

one continuous variable:

```
ggplot2: qplot(~mpg, data=mtcars, geom="histogram")
```

```
ggplot2: qplot(x=displ, data=mtcars, geom="boxplot")
```

one categorical variable:

```
ggplot2: qplot(x=cyl, data=mtcars, geom="bar")
```

two continuous variables:

```
ggplot2: qplot(mpg~displ, ympg, data=mtcars, geom="point")
```

two categorical variables:

```
ggplot2: qplot(x=am, data=mtcars, geom="bar") +
```

```
facet_grid(~cyl)
```

one continuous, one categorical:

```
ggplot2: qplot(mpg~displ, data=mtcars, geom="histogram") +
```

```
facet_grid(~cyl)
```

```
ggplot2: qplot(mpg~displ, x=factor(cyl), data=mtcars, geom="boxplot")
```

WRANGLING:

subsetting:

```
mtcars %>% dplyr::filter(mpg>30)
```

making a new variable:

```
mtcars <- mtcars %>%
```

```
dplyr::mutate(efficient = if_else(mpg>30, TRUE, FALSE))
```

The variety of R syntaxes give you many ways to "say" the same thing

read across the cheat sheet to see how different syntaxes approach the same problem

Make a histogram of the `bill_length_mm` variable from the `penguins` dataset:

```
ggplot(penguins) + geom_histogram(aes(x = bill_length_mm))
```

```
histogram(~ bill_length_mm, data = penguins)
```

```
gf_histogram(~ bill_length_mm, data = penguins)
```

```
hist(penguins$bill_length_mm)
```



```
library(psych)
library(BHH2)
nhanes2017= read.csv("nhanes2017.csv", as.is = F)
table(nhanes2017$exerciseGT60)
par(mfrow = c(1, 2))
hist(nhanes2017$pulse[nhanes2017$exerciseGT60 == "YES"], xlim = c(min(nhanes2017$pulse), max(nhanes2017$pulse)))
hist(nhanes2017$pulse[nhanes2017$exerciseGT60 == "NO"], xlim = c(min(nhanes2017$pulse), max(nhanes2017$pulse)))
par(mfrow = c(1, 1))
tapply(nhanes2017$pulse, nhanes2017$exerciseGT60, summary)
tapply(nhanes2017$pulse, nhanes2017$exerciseGT60, describe)
t.test(pulse ~ exerciseGT60, data = nhanes2017)
```

**Authentic example:** Analyze the difference in **pulse** by **exerciseGT60** from **NHANES**

```
# Load libraries
library(psych)


# Import data
nhanes2017= read.csv("nhanes2017.csv")

# Get levels and sample sizes
table(nhanes2017$exerciseGT60)


# Plot histograms
par(mfrow = c(1, 2))
hist(nhanes2017$pulse[nhanes2017$exerciseGT60 == "YES"], xlim = c(min(nhanes2017$pulse), max(nhanes2017$pulse)))
hist(nhanes2017$pulse[nhanes2017$exerciseGT60 == "NO"], xlim = c(min(nhanes2017$pulse), max(nhanes2017$pulse)))
par(mfrow = c(1, 1))

# Compute summary statistics
tapply(nhanes2017$pulse, nhanes2017$exerciseGT60, FUN = describe)

# Carry out two-sample t-test
t.test(pulse ~ exerciseGT60, data = nhanes2017)
```



Use comments  
in your syntax



In RStudio turn on rainbow  
parentheses  
Options > Code > Display

```
# Load libraries
library(mosaic)

# Import data
nhanes2017 = read_csv("nhanes2017.csv")

# Get levels and sample sizes
tally(~ exerciseGT60, data = nhanes2017)

# Plot histograms
gf_histogram(~ pulse | exerciseGT60, data = nhanes2017)

# Compute summary statistics
favstats(~ pulse | exerciseGT60, data = nhanes2017)

# Carry out two-sample t-test
t_test(~ pulse | exerciseGT60, data = nhanes2017)
```



## The Most Important Template

The following template is important because we can do so much with it.

`[ ] ( [ ] ~ [ ] , data = [ ] )`

It is useful to name the components of the template:

`goal ( y ~ x , data = mydata )`

We're hiding a bit of complexity in the template, and there will be times that we will want to gussy things up a bit. We'll indicate that by adding `...` to the end of the template. Just don't let `...` become a distractor early on.

`goal ( y ~ x , data = mydata , ... )`

# Cognitive *Un*-loading: An Example with Histograms in R

What you teach:

- `ggplot(penguins) + geom_histogram(aes(x = bill_length_mm))`
- `histogram( ~ bill_length_mm, data = penguins)`
- `gf_histogram( ~ bill_length_mm, data = penguins)`
- `hist(penguins$bill_length_mm)`

The order you teach it in:

- Variables -> **Histograms** -> R intro -> syntax
- R intro -> Variables -> **Histograms** -> syntax

How it's introduced:

- RMarkdown file with a worked example
- Live coding session
- Combination-- RMarkdown with some worked, some blanks

# Cognitive *Un*-loading: Model Computational Thinking Norms

Modeling computational thinking and norms

- Debugging (e.g., [McCauley et al., 2008](#))
- Normalizing emotional reactions to coding (e.g., Not panicking (e.g., [DiNapoli, 2018](#)))



## Questions to help you revise how you teach coding:

- What syntactic structure makes the most sense for my students/course/goals?
- Is the code being presented to students consistent in its structure?
- How does new code connect with previous content?
- Will students see/use this code more than once?
  - **“Stitch in time saves 9”**
- How will students encounter code?
  - Live coding, scaffolded documents, cheatsheet?

# Resources and Places to Start

- [Introductory statistics labs in R](#), Amelia McNamara (formula or tidyverse)
- [Speaking R](#), Amelia McNamara (guidance for live coding and reading code)
- [Statistical Modeling and Computation for Educational Scientists](#), Andrew Zieffler (tidyverse)
- [Simulation Based Inference](#), Randy Pruim (formula)
- [Data Science in a Box](#), Mine Çetinkaya-Rundel (tidyverse)



Chelsey Legacy  
[legac006@umn.edu](mailto:legac006@umn.edu)

Amelia McNamara  
[mcna6887@stthomas.edu](mailto:mcna6887@stthomas.edu)



Elle Butler Basner  
[esb5324@psu.edu](mailto:esb5324@psu.edu)

Thank you!



Andrew Zieffler  
[zief0002@umn.edu](mailto:zief0002@umn.edu)



Robert delMas  
[delma001@umn.edu](mailto:delma001@umn.edu)



V.N. Vimal Rao  
[rao00013@umn.edu](mailto:rao00013@umn.edu)



Matthew Beckman  
[mdb268@psu.edu](mailto:mdb268@psu.edu)



# References

- American Statistical Association. (2017). Curriculum guidelines for undergraduate programs in statistical science. American Statistical Association. <https://www.amstat.org/asa/education/home.aspx/curriculumguidelines.cfm>
- Ayres, P. (2016). Impact of reducing intrinsic cognitive load on learning in a mathematical domain. *Cognitive Psychology*, 20(3), 287-298. <https://onlinelibrary.wiley.com/doi/abs/10.1002/acp.1245>
- Hermans, F. (2021). *The Programmer's Brain*. <https://www.manning.com/books/the-programmers-brain>
- Horton, N. J. (2015). Challenges and opportunities for statistics and statistical education: looking back, looking forward. *The American Statistician*, 69(2), 138-145. <https://doi.org/10.1080/00031305.2015.1032435>
- National Academies of Sciences, Engineering, and Medicine. (2018). *Data science for undergraduates: Opportunities and options*. The National Academies Press. <https://doi.org/10.17226/25104>.
- Nolan, D., & Temple Lang, D. (2010). Computing in the statistics curricula. *The American Statistician*, 64(2), 97-107. <https://doi.org/10.1198/tast.2010.09132>
- Sweller, J. (2011). Cognitive load theory. In J. P. Mestre & B. H. Ross (Eds.), *Psychology of Learning and Motivation*, 55, 37-76. Academic Press. <https://doi.org/10.1016/B978-0-12-387691-1.00002-8>
- Woodard, V., and Lee, H. (2021). How students use statistical computing in problem solving. *Journal of Statistics and Data Science Education*, 29, 1-18. <https://doi.org/10.1080/10691898.2020.1847007>

# Photo Credits

<https://edtechmagazine.com/k12/article/2020/04/how-teachers-can-better-manage-classroom-computing>

<https://pixabay.com/illustrations/artificial-intelligence-ai-robot-2228610/>

<https://pixabay.com/illustrations/start-beginning-arrows-direction-5896294/>

<https://bangaloremirror.indiatimes.com/bangalore/cover-story/starting-from-scratch/articleshow/74016629.cms>

<https://pixabay.com/photos/hardware-screwdriver-kit-4649058/>