Prof. Wolff's Style Guide and Review of Common Mistakes

An Unofficial Document

Daniel Hintz

03 January, 2024

Contents

Style Guide

This Style Guide is not comprehensive and are not the explicit views of Professor Wolff. The intent of this document is to clarify the expectations of students in addition to the hope that it can be extended with time.

Tables

- Print named vectors where possible to converse space on paper

  i.e., like the following

```
mtcars[1,]
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1    4    4
```

Or

```
mtcars[1,]
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4

Table 1

But not

```
t(mtcars[1,])
```

```
##      Mazda RX4
## mpg      21.00
## cyl       6.00
## disp    160.00
## hp      110.00
## drat      3.90
## wt        2.62
## qsec     16.46
## vs        0.00
## am        1.00
## gear      4.00
## carb      4.00
```

Or

```
t(mtcars[1,])
```

	Mazda RX4
mpg	21.00
cyl	6.00
disp	160.00
hp	110.00
drat	3.90
wt	2.62
qsec	16.46
vs	0.00
am	1.00
gear	4.00
carb	4.00

Table 2

- If multiple row vectors can be combined to a table, so long as the column headers pertain to all rows and all rows are bear the same context, do so (it conserves space).

  For example, instead of, ...

```
(qc1 <- quantile(mtcars[,1]))
```

0%	25%	50%	75%	100%
10.4	15.425	19.2	22.8	33.9

Table 3

```
(qc2 <- quantile(mtcars[,2]))
```

0%	25%	50%	75%	100%
4	4	6	8	8

Table 4

```
(qc3 <- quantile(mtcars[,3]))
```

0%	25%	50%	75%	100%
71.1	120.825	196.3	326	472

Table 5

Instead combine them to a table to conserve space (it also makes comparing results easier)

```
apply(mtcars,2, quantile)[,1:3]
```

	mpg	cyl	disp
0%	10.400	4	71.100
25%	15.425	4	120.825
50%	19.200	6	196.300
75%	22.800	8	326.000
100%	33.900	8	472.000

Table 6

```
# or
#tb.1 = t(rbind(qc1,qc2,qc3)); colnames(tb.1) = colnames(mtcars)[1:3];tb.1
```

Plotting

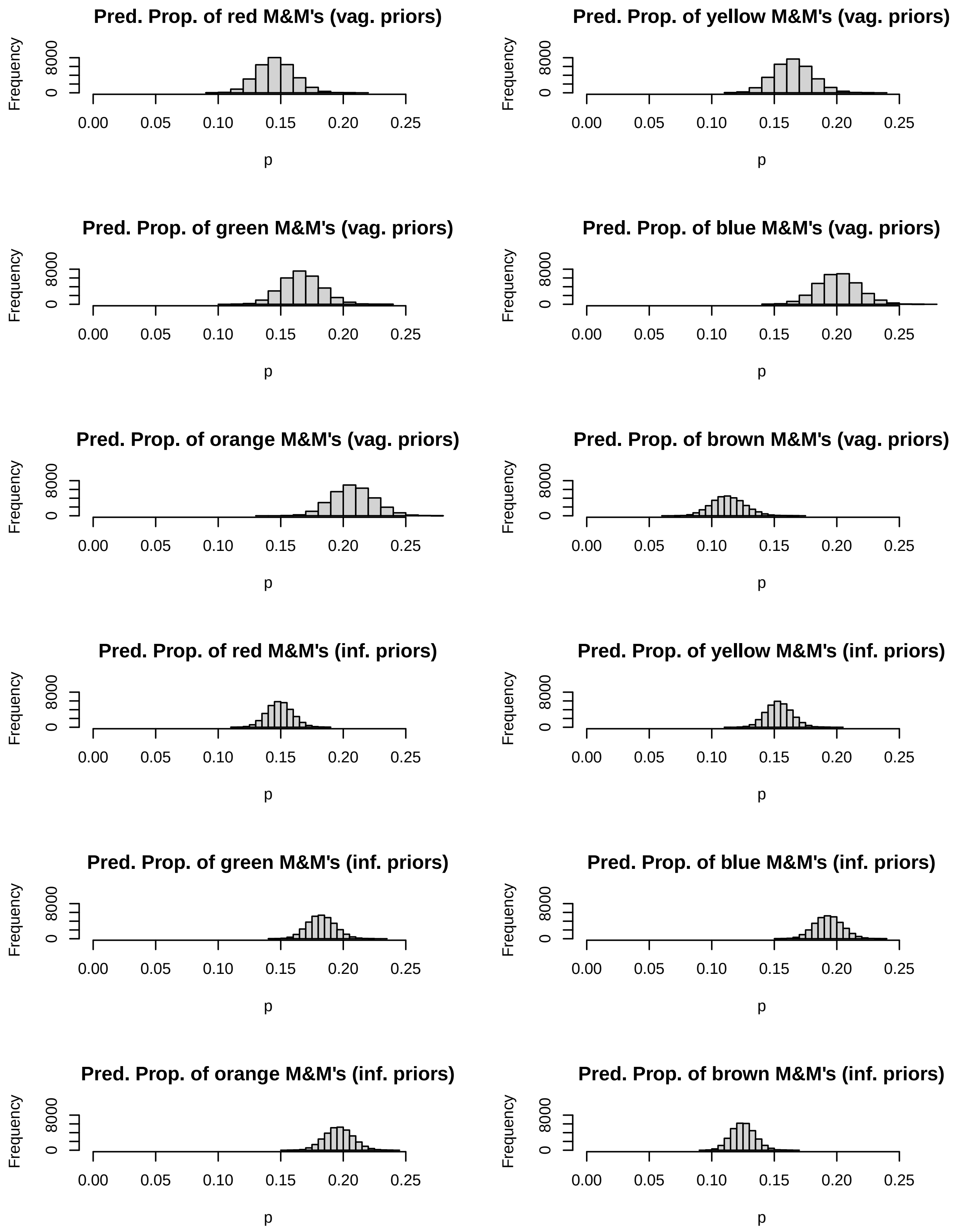
- If multiple plots can be overlaid to one figure do so, it conserves space and makes comparing results easier.

Below is a example of Multinomial regression for the counts of the different colours of M&M's in predicting the proportions for each color with and without counts from the previous year acting as the prior. There are six colors red, yellow, green, blue, orange, and brown, all M&M's can be presumed to have be collected on Halloween 2023.

For the sake of brevity, assume we have run the Bayesian multinomial regression and we have a list of the posterior distributions for p, the predicted proportion of e M&M's for each respective color, saved in objects `p.l.vague`, for the predictions with vague priors and `p.l.inf` for predictions with informative priors.

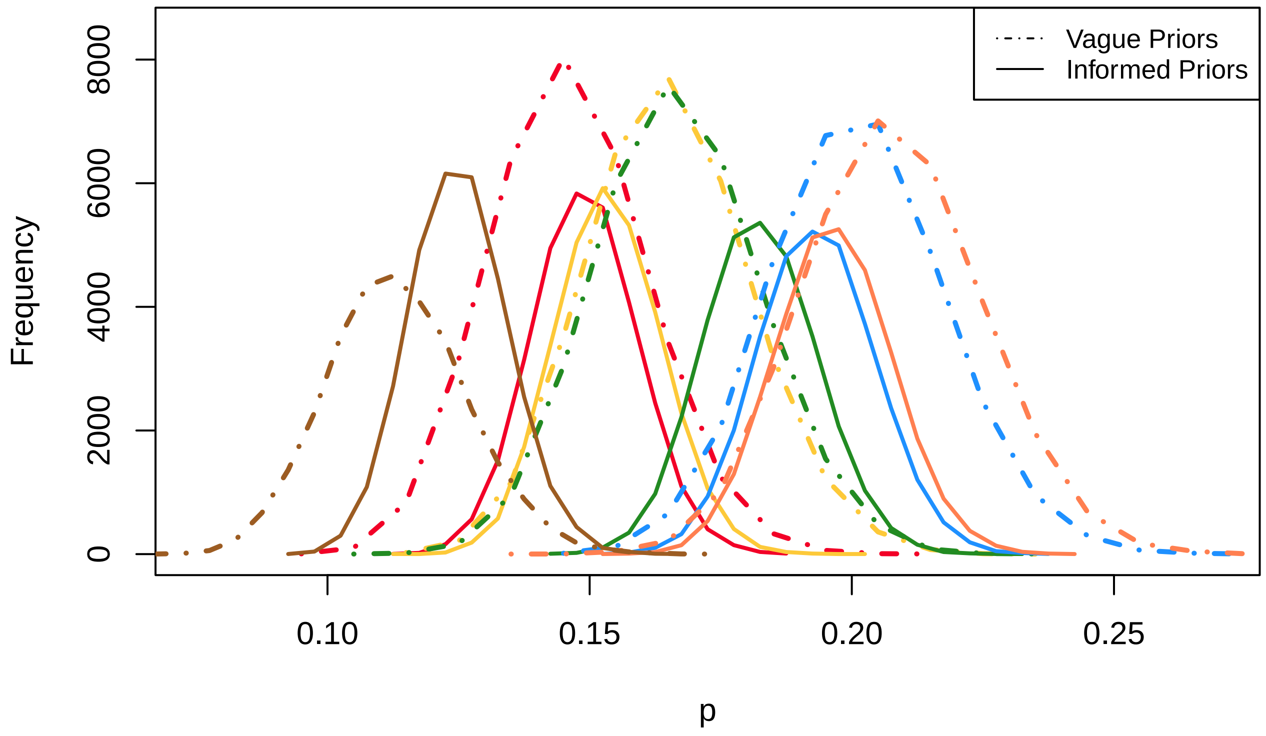
Given there is 6 different colors and 2 models, this makes for 12 histograms! See figure 1 below:

Figure 1: Proportion of M&M's, Informed vs Vague Proirs (Sep. Histograms)



The better way to illustrate this information is in one plot, see Figure 2 below:

Figure 2: Proportion of M&M's, Informed vs Vague Proirs (Single Plot)



From Figure 2, we can easily compare the different predicted proportions of different M&M colors, as well as comparing the results of predictions with vague priors versus informed priors.

Massaging data for various measures that they can presented in one plot is not always possible, though when it is (while sometimes a tedious process) it most often proves to be far more effective at communicating insights.

Fonts and Precision

- Set seeds before every stochastic process both in and outside of loops and not just once at the start of the document.
- Check if a seed was specified and use that seed throughout.
- Round to at least 6 figures.
- Use Mathematical notation with "math font", i.e., a "math environment" (`$$`) for Rmarkdown or LaTeX, or with an equation block when using Microsoft word, or with a code specific font such as Courier New for other editors.

The results should look like $\mathrm{E}[X]$, not E[X]. Moreover, be consistent wherever possible, i.e, operators should be denoted separate from variables. For example the expectation is an operator so it should be $\mathrm{E}[X]$ not $E[X]$. Also, you can use square or round brackets with the operators such as $\mathrm{E}[X], \mathrm{E}(X)$ or $\mathrm{Var}[X], \mathrm{Var}(X)$, just be consistent.

Coding practices

- Show all of your code!
- D.R.Y: Don't repeat yourself

  If a function or an object assignment is already identically typed above, don't redefine it below.
- Be consistent with naming conventions (ie, snake_case, kebab-case, camelCase, PascalCase, leopard.case, SCREAMING_SNAKE_CASE, flatcase, Pascal_Snake_Case, camel_Snake_Case, TRIAN-CASE, UPPERCASE, Http-Header-Case). Note, that snake_case, camelCase, PascalCase, and leopard.case are most popular for R programming.
- Shaun prefers the following option settings:

```
options(show.signif.stars = FALSE, digits = 6)
```

Also, be careful to not cut off code output with `options("max.print")`.

Think carefully about whether you want to pass variables to a function via the function signature or instead have them being called from the global environment.

For example, instead of

```
y = 1; z = 2; i = 3; j = 4; k = 5

# Method 1: Passing Variables as Arguments
bah <- function(x,y,z,i,j,k) x + y + z + i + j + k
bah(0.5,1,2,3,4,5)
```

```
## [1] 15.5
```

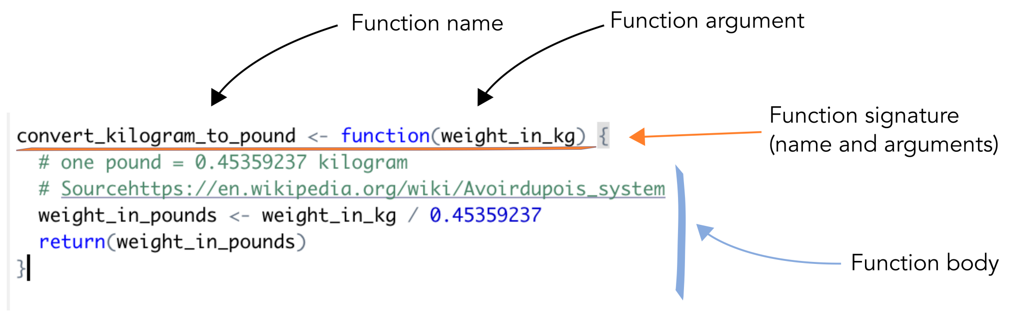
If you expect the variables `y, z, i, j, k` not change throughout the course of a project, or if a "switch" methodology is preferred, given all of these variables are already defined in the global environment above the function call, `method 2` may be preferred (see below).

```
# Method 2: Using Globally Defined Variables
bah <- function(x) x + y + z + i + j + k
bah(0.5)
```

```
## [1] 15.5
```

Notice `y, z, i, j, k` were not defined as arguments in the function signature (see photo below).

Figure 3: Function Structure & Terminology



```r
convert_kilogram_to_pound <- function(weight_in_kg) {
  # one pound = 0.45359237 kilogram
  # Sourcehttps://en.wikipedia.org/wiki/Avoirdupois_system
  weight_in_pounds <- weight_in_kg / 0.45359237
  return(weight_in_pounds)
}
```

When choosing between passing variables as arguments to a function or using globally defined variables within a function in R, there are several pros and cons to consider for each method:

Method 1: Passing Variables as Arguments

Pros:

1. **Clarity and Readability:** The function's dependencies are explicitly stated in its signature, making it clear what data it operates on.
2. **Flexibility:** The function can be used in different contexts with different sets of values, enhancing reusability.
3. **Reduced Side Effects:** Since the function does not rely on external variables, there's less risk of unintended interactions with other parts of the code.
4. **Easier Debugging and Testing:** It's easier to test the function in isolation since all inputs are clearly defined and controlled.

Cons:

1. **Verbosity:** If a function requires many arguments, the signature can become unwieldy.
2. **Repetitive Code:** If the same sets of variables are frequently passed to different functions, it can lead to redundant code.
3. **Maintenance:** Changing the function's requirements means updating every call to the function, which can be time-consuming in large projects.

Method 2: Using Globally Defined Variables

Pros:

1. **Simplicity:** The function call is simpler with fewer arguments, which can be more convenient if the function is called frequently.
2. **Less Redundant Code:** If the same variables are consistently used across multiple functions, this method can reduce repetition.

Cons:

1. **Hidden Dependencies:** The function's dependency on global variables is not obvious from its signature, which can lead to confusion and errors.

2. **Reduced Flexibility:** The function is less adaptable to different contexts or datasets since it's tied to specific global variables.
3. **Risk of Side Effects:** Changes to global variables can inadvertently affect the function's behavior, leading to bugs that are hard to trace.
4. **Difficulties in Testing:** Testing the function in isolation is harder since it depends on the state of the global environment.
5. **Poor Scalability:** As the codebase grows, managing global state becomes increasingly complex and error-prone.

Summary

Method 1 is generally preferred in software development for its clarity, flexibility, and ease of testing and maintenance. It adheres to good practices like explicitness and function independence.

Method 2 might be convenient for quick, small-scale projects or scripts where variables are unlikely to change and simplicity is key (i.e., for Homework Assignments). However, it carries higher risk related to maintainability, scalability, and the potential for bugs as a project scales.

- Remove unnecessary code comments before submission (keep the code lean).
- If a function or piece of code can be written in 1 line (use of `;` is encouraged), do so as long as the code is still neat and doesn't go off the page (Shaun code style guide is different from the tidyverse style, guide see https://design.tidyverse.org, that means Rmarkdown users don't use the code linter `knitr::opts_chunk$set(tidy="styler")`).

  For example,

```
# use
rmse.f <- function(y, yh) sqrt(mean((y - yh)^2))
# as opposed to
rmse.f <- function(y, yh) {
  sqrt(mean((y - yh)^2))
}

# use
lprior <- function(pars) {sigma_sq = max(pars[np],ee); log(1 /(sigma_sq))}
# as opposed to
lprior <- function(pars) {
  sigma_sq = max(pars[np],ee)
  log(1 /(sigma_sq))
}

# use
a = 1; b = 2; c = 3
# as opposed to
```

```
a = 1
b = 2
c = 3

# use
a = b = c = 1
# as opposed to
a = 1
b = 1
c = 1
# or
a = 1; b = 1; c = 1
```

- Don't include error handling in functions (takes up unnecessary space for purpose of grading homework).

  For example,

```
foo <- function(p){
  if(!is.integer(p)){
    stop("only integer values for variable p")
  }
}
```

- Include support of the distribution when writing pdfs/pmfs as functions and avoid the use of for loops.

  For example, when considering the bivariate distribution

$$f_{\mathrm{X,Y}}(x,y) = \frac{1}{\beta^2} e^{-x/\beta} e^{-y/\beta} e^{-e^{-y/\beta}} \quad \text{for } -\infty < y < x < \infty$$

```
fxy <- function(x, y, beta) (1/beta^2)*exp(-x /beta)*exp(-y/beta)*exp(-exp(-y/beta))*(y<x)
```

  Notice the `*(y<x)` avoid the need for an `if()` statement.

Common Mistakes

- Check all pdfs/pmfs have their supports listed.
- When doing a hypothesis test, report the test statistic, the p-value, and the alpha level if using the decision based approach.
- When using the Fisher/evidence-based approach for hypothesis tests and reporting evidence against the null hypothesis, state that the p-value of p **suggests** that ...
  - If there is strong evidence against the null hypothesis:

    ... <describe the implications or interpretation of rejecting H_0 here>.
  - If there is weak evidence against the null hypothesis:

    ... <describe the implications or interpretation of not rejecting H_A here>.

  For example for a cancer treatment measuring the time until a recurrence of the disease:

Given a p-value of 0.66 and a test statistic of 0.44, there is very weak evidence against the null hypothesis that $\tau = 1$. While we cannot accept H_0, our evidence suggests that the hormone treatment does not impact the distribution of the time until a second recurrence. This suggests that the hormone treatment does not impact the distribution of the time until a second recurrence.

- When doing hyperparamter optimization, always report final hyperparameters and use `set.seed` before fitting the final model.
- Don't forget the differentials, ie $dy, dx, ...$ when writing integral (its a common mistake).
- The evidence (`<test_statistic>`, p-value) is **the amount of evidence agaisnt the null hypothesis**
- Say the "the null hypothesis" or H_0, don't just say "the null".
- In statistics, we make inference about population parameters, so report results as such, i.e., "The 95% CI (`<lwr>`,`<upr>`), contains the true population mean, μ for `<subjects of interest>`".
- Always report findings in the context of the given problem.
- If an alpha level is not stated, uses Fisher's evidence based approach instead of a decision based approach for hypothesis testing (Note: if asked for a k% CI, this means $\alpha = 1 - k$).
- Random variables should be upper-case e.g., $Y \sim$, $\bar{X} \sim$, $\Theta \sim$, estimates have hats, e.g., $\hat{\beta}_0, \hat{\tau}$, fixed population parameters are lower-case (usually Greek not Roman), i.e., β, μ.
- Specify what values are being used for parameters in the write up.
- For method of moments, the correct notation is:

$$\widehat{\mathrm{E}[X]} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Not

$$\mathrm{E}[X] = \frac{\sum_{i=1}^{n} x_i}{n}$$

- For a pdf/pmf $f(x_i|\theta)$, $\{f(\text{“Data”} \mid \text{Params})\}$, the LHS is a RV (unknown but not fixed) and the RHS is fixed or known, while for the likelihood, $L(\theta|\underline{x})$, $\{L(\text{Params}|\text{“Data”})\}$, the LHS is unknown and fixed and the RHS is fixed and unknown.

Where $f(x_i|\theta)$ is how probable are different values of "x" given a certain set of parameters "θ" and $L(\theta|\underline{x})$ is how probable are different values of the parameters "θ", given the dataset "x".

Remember, that Likelihood values indicate how well a statistical model fits the data; higher likelihood values indicate a better fit of the model to the data.