So, a programmer, a frequentist, and a bayesian walk into a bar. No this post  
isn’t really on the path to some politically incorrect stereotypical humor. Jut  
trying to make it fun and catch your attention. As the title implies this post  
is really about applying the differing viewpoints and methodologies inherent in  
those approaches to statistics. To be honest I’m not even going to spend a lot  
time on the details of methods, and indeed to focus on the conclusions each of these  
people would draw after analyzing the same data using their favored methods.  
This post isn’t really so much about how they would proceed but more about what  
they would conclude at the end of the analysis. I make no claim about which of  
these fine people have the best way of doing things although I was raised a  
frequentist, I am more and more enamored of bayesian methods and while my tagline  
is accurate “R Lover !a programmer” I will admit a love for making computers do  
the hard work for me so if that involves a **little** programming, I’m all for  
it.

**Background**

Welcome back. I’m going to try and repeat as little as possible from blog  
post just make comparisons. So to continue our little teaser… So, a  
programmer, a frequentist, and a bayesian walk into a bar and start arguing  
about whether Usain Bolt really is the fastest man on earth. The programmer has  
told us how they would go about answering the question. The answer was:

There is only a 1.85% chance of seeing a difference as large as the observed  
difference if there is actually no difference between the (median) timings of  
Usain Bolt and Asafa Powell.

and was derived by counting observations across 10,000 simulations of the data  
using the infer package and looking at differences between median timings. Our  
null hypothesis was there is no “real” difference difference between Bolt and  
Powell even though our data has a median for Bolt of 9.90 and median for Powell  
of 9.95. That is after all a very small difference. But our simulation allows us  
to reject that null hypothesis and favor the alternative that the difference is  
real.

Should we be confident that we are 100% – 1.85% = 98% likely to be correct? NO!  
as Downey notes:

For most problems, we only care about the order of magnitude: if the p-value  
is smaller that 1/100, the effect is likely to be real; if it is greater than  
1/10, probably not. If you think there is a difference between a 4.8%  
(significant!) and 5.2% (not significant!), you are taking it too seriously.

Can we say that Bolt will win a race with Powell 98% of time? Again a resounding  
NO! We’re 98% certain that the *“true”* difference in their medians is .05  
seconds? NOPE.

**Back to the future**

Okay we’ve heard the programmer’s story at our little local bar. It’s time to  
let our frequentist have their moment in the limelight. Technically the best  
term would be Neyman-Pearson Frequentist but we’re not going to stand on  
formality. It is nearly a century old and stands as an “improvement” on Fisher’s  
significance testing. [A nice little summary here on  
Wikipedia](https://en.wikipedia.org/wiki/Foundations_of_statistics#Fisher&%2339;s_%22significance_testing%22_vs_Neyman%E2%80%93Pearson_%22hypothesis_testing%22).

I’m not here to belabor the nuances but frequentist methods are among the  
oldest and arguably the most prevalent method in many fields. They are often the  
first method people learned in college and sometimes the only method. They still  
drive most of the published research in many fields although other methods are  
taking root.

Before the frequentist can tell their tale though let’s make sure they have the  
same data as in the earlier post. Let’s load all the libraries we’re going to  
use and very quickly reproduce the process Anindya Mozumdar went through to  
scrape and load the data. We’ll have a tibble named male\_100 that contains  
the requisite data and we’ll confirm that the summary for the top 6 runners mean  
and median are identical. Note that I am suppressing messages as the libraries  
load since R 3.6.0 has gotten quite chatty in this regard.

library(rvest)

library(readr)

library(dplyr)

library(ggplot2)

library(ggstatsplot)

library(BayesFactor)

male\_100\_html <- read\_html("<http://www.alltime-athletics.com/m_100ok.htm>")

male\_100\_pres <- male\_100\_html %>%

html\_nodes(xpath = "//pre")

male\_100\_htext <- male\_100\_pres %>%

html\_text()

male\_100\_htext <- male\_100\_htext[[1]]

male\_100 <- readr::read\_fwf(male\_100\_htext, skip = 1, n\_max = 3178,

col\_types = cols(.default = col\_character()),

col\_positions = fwf\_positions(

c(1, 16, 27, 35, 66, 74, 86, 93, 123),

c(15, 26, 34, 65, 73, 85, 92, 122, 132)

))

male\_100 <- male\_100 %>%

select(X2, X4) %>%

transmute(timing = X2, runner = X4) %>%

mutate(timing = gsub("A", "", timing),

timing = as.numeric(timing)) %>%

filter(runner %in% c("Usain Bolt", "Asafa Powell", "Yohan Blake",

"Justin Gatlin", "Maurice Greene", "Tyson Gay")) %>%

mutate\_if(is.character, as.factor) %>%

droplevels

male\_100

## # A tibble: 520 x 2

## timing runner

##

## 1 9.58 Usain Bolt

## 2 9.63 Usain Bolt

## 3 9.69 Usain Bolt

## 4 9.69 Tyson Gay

## 5 9.69 Yohan Blake

## 6 9.71 Tyson Gay

## 7 9.72 Usain Bolt

## 8 9.72 Asafa Powell

## 9 9.74 Asafa Powell

## 10 9.74 Justin Gatlin

## # … with 510 more rows

male\_100$runner <- forcats::fct\_reorder(male\_100$runner, male\_100$timing)

male\_100 %>%

group\_by(runner) %>%

summarise(mean\_timing = mean(timing)) %>%

arrange(mean\_timing)

## # A tibble: 6 x 2

## runner mean\_timing

##

## 1 Usain Bolt 9.90

## 2 Asafa Powell 9.94

## 3 Tyson Gay 9.95

## 4 Justin Gatlin 9.96

## 5 Yohan Blake 9.96

## 6 Maurice Greene 9.97

male\_100 %>%

group\_by(runner) %>%

summarise(median\_timing = median(timing)) %>%

arrange(median\_timing)

## # A tibble: 6 x 2

## runner median\_timing

##

## 1 Usain Bolt 9.9

## 2 Asafa Powell 9.95

## 3 Yohan Blake 9.96

## 4 Justin Gatlin 9.97

## 5 Maurice Greene 9.97

## 6 Tyson Gay 9.97

Most of the code above is simply shortened from the original post. The only  
thing that is completely new is forcats::fct\_reorder(male\_100$runner, male\_100$timing) which takes the runner factor and reorders it according to  
the median by runner. This doesn’t matter for the calculations we’ll do but it  
will make the plots look nicer.

**Testing, testing!**

One of the issues with a frequentist approach compared to a programmers approach  
is that there are a lot of different tests you could choose. But in this case  
wearing my frequentist hat there really are only two choices. A Oneway ANOVA or  
the Kruskall Wallis which uses ranks and eliminates some assumptions.

This also gives me a chance to talk about a great package that supports both  
frequentists and bayesian methods and completely integrates visualizing your  
data with analyzing your data, which IMHO is the only way to go. The package is  
ggstatsplot. It’s stable,  
mature, well tested and well maintained try it out.

So let’s assume we want to run a classic Oneway ANOVA first (Welch’s method so  
we don’t have to assume equal variances across groups). Assuming that the  
omnibuds F test is significant lets say we’d like to look at the pairwise  
comparisons and adjust the p values for multiple comparison using Holm. We’re a  
big fan of visualizing the data by runner and of course we’d like to plot things  
like the mean and median and the number of races per runner. We’d of course like  
to know our effect size we’ll stick with eta squared we’d like it as elegant as  
possible.

Doing this analysis using frequentist methods in R is not difficult The benefit of ggbetweenstats from ggstatsplot is that it pretty  
much allows you to do just about everything in one command. Seamlessly mixing  
the plot and the results into one output. We’re only going to scratch the  
surface of all the customization possibilities.

Library(ggstatplot)

ggbetweenstats(data = male\_100,

x = runner,

y = timing,

type = "p",

var.equal = FALSE,

pairwise.comparisons = TRUE,

partial = FALSE,

effsize.type = "biased",

point.jitter.height = 0,

title = "Parametric (Mean) testing assuming unequal variances",

ggplot.component = ggplot2::scale\_y\_continuous(breaks = seq(9.6, 10.4, .2),

limits = (c(9.6,10.4))),

messages = FALSE

)

Our conclusion is similar to that drawn by simulation. We can clearly reject the  
null that all these runners have the same mean time. Using Games-Howell and  
controlling for multiple comparisons with Holm, however, we can only show  
support for the difference between Usain Bolt and Maurice Green. There is  
insufficient evidence to reject the null for all the other possible pairings.  
(You can actually tell ggbetweenstats to show the p value for all the pairings  
but that gets cluttered quickly).

From a frequentist perspective there are a whole set of non-parametric tests  
that are available for use. They typically make fewer assumptions about the data  
we have and often operate by exchanging the ranks of the outcome variable  
(timing) rather than using the number.

The only thing we need to change in our input to the function is type = "np" and the title.

ggbetweenstats(data = male\_100,

x = runner,

y = timing,

type = "np",

var.equal = FALSE,

pairwise.comparisons = TRUE,

partial = FALSE,

effsize.type = "biased",

point.jitter.height = 0,

title = "Non-Parametric (Rank) testing",

ggplot.component = ggplot2::scale\_y\_continuous(breaks = seq(9.6, 10.4, .2),

limits = (c(9.6,10.4))),

messages = FALSE

)

Without getting overly consumed by the exact numbers note the very similar  
results for the overall test, but that we now also are more confident about  
whether the difference between Usain Bolt and Justin Gaitlin. I highlight that  
because there is a common misconception that non-parametric tests are always less  
powerful (sensitive) than their parametric cousins.

**Asking the question differently**

It usually takes several lessons or even an entire semester to teach the  
frequentist method, because null hypothesis testing is a very elaborate  
contraption that people (well in my experience very smart undergraduate  
students) find very hard to master. In contrast, the Bayesian approach to  
hypothesis testing *“feels”* far more intuitive. Let’s apply it to our current  
scenario.

We’re at the bar the three of us wondering whether Usain Bolt is really the  
fastest or whether all these individual data points really are just a random  
mosaic of data noise. Both the programmer and the frequentist set the testing up  
conceptually the same way. Can we use the data to reject the null that all the  
runners are the same. Convinced they’re not all the same they applied the same  
general procedure to reject (or not) the hypothesis that any pair was the same  
for example Bolt versus Powell (for the record I’m not related to either). They  
differ in computational methods and assumptions but not in overarching method.

At the end of their machinations they have no ability to talk about how likely  
(probable) it is that runner 1 will beat runner 2. Often times that’s exactly  
what you really want to know. There are two hypotheses that we want to compare,  
a null hypothesis h0 that all the runners run equally fast and an alternative  
hypothesis h1 that they don’t. Prior to looking at the data while we’re  
sitting at the bar we have no real strong belief about which hypothesis is true  
(odds are 1:1 in our naive state). We have our data and we want it to inform our  
thinking. Unlike frequentist statistics, Bayesian statistics allow us to talk  
about the probability that the null hypothesis is true (which is a complete **no  
no** in a frequentist context). Better yet, it allows us to calculate the  
posterior probability of the null hypothesis, using Bayes’ rule and our data.

In practice, most Bayesian data analysts tend not to talk in terms of the raw  
posterior probabilities. Instead, we/they tend to talk in terms of the posterior  
odds ratio. Think of it like betting. Suppose, for instance, the posterior  
probability of the null hypothesis is 25%, and the posterior probability of the  
alternative is 75%. The alternative hypothesis **h1** is three times as probable as the  
null **h0**, so we say that the odds are 3:1 in favor of the alternative.

At the end of the Bayesian’s efforts they can make what feel like very natural  
statements of interest, for example, “The evidence provided by our data  
corresponds to odds of 42:1 that these runners are not all equally fast.

Let’s try it using ggbetweenstats…

ggbetweenstats(data = male\_100,

x = runner,

y = timing,

type = "bf",

var.equal = FALSE,

pairwise.comparisons = TRUE,

partial = FALSE,

effsize.type = "biased",

point.jitter.height = 0,

title = "Bayesian testing",

messages = FALSE

)

Yikes! Not what I wanted to see in the bar. The pairwise comparisons have gone  
away (we’ll get them back) and worse yet what the heck does loge(BF10) = 2.9  
mean? I hate log conversions I was promised a real number like 42:1! Who’s  
Cauchy why is he there at .0.707?

Let’s break this down. loge(BF10) = 2.9 is also exp(2.9) or about 18 so  
the good news is the odds are better than 18:1 that the runners are not equally  
fast. Since rounding no doubt loses some accuracy lets use the BayesFactor  
package directly and get a more accurate answer before we round anovaBF(timing ~ runner, data = as.data.frame(male\_100), rscaleFixed = .707) is what we want  
where rscaleFixed = .707 ensures we have the right Cauchy value.

anovaBF(timing ~ runner, data = male\_100, rscaleFixed = .707)

## Bayes factor analysis

## --------------

## [1] runner : 19.04071 ±0.01%

##

## Against denominator:

## Intercept only

## ---

## Bayes factor type: BFlinearModel, JZS

Okay that’s better so to Bayesian thinking the odds are 19:1 against the fact that they all run about the same speed, or 19:1 they run at different speeds.

Hmmm. One of the strengths/weaknesses of the Bayesian approach is that people  
can have their own sense of how strong 19:1 is. I like those odds. One of the  
really nice things about the Bayes factor is the numbers are inherently  
meaningful. If you run the data and you compute a Bayes factor of 4, it means  
that the evidence provided by your data corresponds to betting odds of 4:1 in  
favor of the alternative. However, there have been some attempts to quantify the  
standards of evidence that would be considered meaningful in a scientific  
context. One that is widely used is from Kass and Raftery (1995). (**N.B. there are others and I have deliberately selected one of the most conservative standards. See for example** [**https://media.springernature.com/full/springer-static/image/art%3A10.1186%2Fs12888-018-1761-4/MediaObjects/12888\_2018\_1761\_Fig1\_HTML.png**](https://media.springernature.com/full/springer-static/image/art%3A10.1186%2Fs12888-018-1761-4/MediaObjects/12888_2018_1761_Fig1_HTML.png))

| **Bayes factor value** | **Interpretation** |
| --- | --- |
| 1 – 3 | Negligible evidence |
| 3 – 20 | Positive evidence |
| 20 -150 | Strong evidence |
| >150 | Very strong evidence |

Okay we have “positive evidence” and we can quantify it, that’s good. But what  
about all the pairwise comparisons? Can we take this down to all the  
individual pairings? I’m on the edge of my bar stool here. What are the odds  
Bolt really is faster than Powell? Can we quantify that without somehow breaking  
the multiple comparisons rule?

The short answer is yes we can safely extend this methodology to incorporate  
pairwise comparisons. We shouldn’t abuse the method and we should fit our model  
with the best possible prior information .

With Bayesian inference (and the correct prior), though, this problem  
disappears. Amazingly enough, you don’t have to correct Bayesian inferences for  
multiple comparisons.

With that in mind let’s build a quick little function that will allow us to pass  
a data source and two names and run a Bayesian t-test via BayesFactor::ttestBF  
to compare two runners. ttestBF returns a lot of info in a custom object so  
we’ll use the extractBF function to grab it in a format where we can pluck out  
the actual BF10

compare\_runners\_bf <- function(df, runner1, runner2) {

ds <- df %>%

filter(runner %in% c(runner1, runner2)) %>%

droplevels %>%

as.data.frame

zzz <- ttestBF(formula = timing ~ runner, data = ds)

yyy <- extractBF(zzz)

xxx <- paste0("The evidence provided by the data corresponds to odds of ",

round(yyy$bf,0),

":1 that ",

runner1,

" is faster than ",

runner2 )

return(xxx)

}

Now that we have a function we can see what the odds are that Bolt is faster  
than the other 5 and print them one by one

compare\_runners\_bf(male\_100, "Usain Bolt", "Asafa Powell")

## [1] "The evidence provided by the data corresponds to odds of 5:1 that Usain Bolt is faster than Asafa Powell"

compare\_runners\_bf(male\_100, "Usain Bolt", "Tyson Gay")

## [1] "The evidence provided by the data corresponds to odds of 5:1 that Usain Bolt is faster than Tyson Gay"

compare\_runners\_bf(male\_100, "Usain Bolt", "Justin Gatlin")

## [1] "The evidence provided by the data corresponds to odds of 21:1 that Usain Bolt is faster than Justin Gatlin"

compare\_runners\_bf(male\_100, "Usain Bolt", "Yohan Blake")

## [1] "The evidence provided by the data corresponds to odds of 8:1 that Usain Bolt is faster than Yohan Blake"

compare\_runners\_bf(male\_100, "Usain Bolt", "Maurice Greene")

## [1] "The evidence provided by the data corresponds to odds of 1355:1 that Usain Bolt is faster than Maurice Greene"

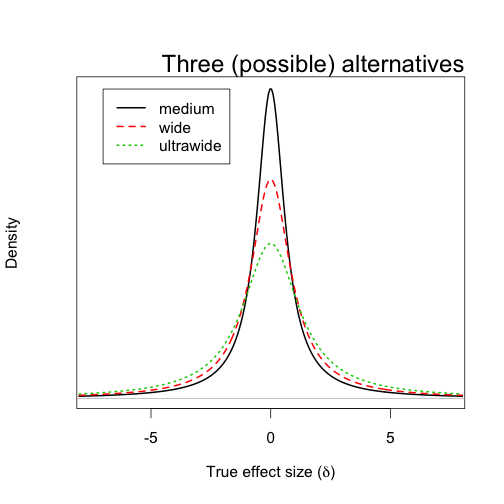
Okay now I feel like we’re getting somewhere with our bar discussions. Should I  
feel inclined to make a little wager on say who buys the next round of drinks as  
a Bayesian I have some nice useful information. I’m not rejecting a null  
hypothesis I’m casting the information I have as a statement of the odds I think  
I have of “winning”.

But of course this isn’t the whole story so please read on…

**Who’s Cauchy and why does he matter?**

Earlier I made light of the fact that the output from ggbetweenstats had  
rCauchy = 0.707 and anovaBF uses rscaleFixed = .707. Now we need to spend  
a little time actually understanding what that’s all about. Cauchy is  
[Augustin-Louis Cauchy](https://en.wikipedia.org/wiki/Augustin-Louis_Cauchy) and  
the reason that’s relevant is that BayesFactor [makes use of his distribution as  
a default](https://en.wikipedia.org/wiki/Cauchy_distribution). I’m not even  
going to try and take you into the details of the math but it is important we  
have a decent understanding of what we’re doing to our data.

The BayesFactor  
package  
has a few built-in “default” named settings. They all have the same shape; the  
only differ by their scale, denoted by r. The three named defaults are medium =  
0.707, wide = 1, and ultrawide = 1.414. “Medium”, is the default. The scale  
controls how large, on average, the expected true effect sizes are. For a  
particular scale 50% of the true effect sizes are within the interval (−r,r).  
For the default scale of “medium”, 50% of the prior effect sizes are within the  
range (−0.7071,0.7071). Increasing r increases the sizes of expected effects;  
decreasing r decreases the size of the expected effects.



Let’s compare it to a frequentist test we’re all likely to know, the t-test,  
(we’ll use the Welch variant). Our initial hypothesis is that Bolt’s mean times  
are different than Powell’s mean times (two-sided) and then test the one-sided  
that Bolt is faster. Then let’s go a little crazy and run it one sided but  
specify the mean difference 0.038403 of a second faster that we “see” in our data  
mu = -0.038403.

Library(BayesFactor)

justtwo <- male\_100 %>%

filter(runner %in% c("Usain Bolt", "Asafa Powell")) %>%

droplevels %>%

as.data.frame

t.test(formula = timing ~ runner, data = justtwo)

##

## Welch Two Sample t-test

##

## data: timing by runner

## t = -2.5133, df = 111.58, p-value = 0.01339

## alternative hypothesis: true difference in means is not equal to 0

## 95 percent confidence interval:

## -0.06868030 -0.00812721

## sample estimates:

## mean in group Usain Bolt mean in group Asafa Powell

## 9.904930 9.943333

t.test(formula = timing ~ runner, data = justtwo, alternative = "less")

##

## Welch Two Sample t-test

##

## data: timing by runner

## t = -2.5133, df = 111.58, p-value = 0.006694

## alternative hypothesis: true difference in means is less than 0

## 95 percent confidence interval:

## -Inf -0.01306002

## sample estimates:

## mean in group Usain Bolt mean in group Asafa Powell

## 9.904930 9.943333

t.test(formula = timing ~ runner, data = justtwo, alternative = "less", mu = -0.038403)

##

## Welch Two Sample t-test

##

## data: timing by runner

## t = -4.9468e-05, df = 111.58, p-value = 0.5

## alternative hypothesis: true difference in means is less than -0.038403

## 95 percent confidence interval:

## -Inf -0.01306002

## sample estimates:

## mean in group Usain Bolt mean in group Asafa Powell

## 9.904930 9.943333

Hopefully that last one didn’t trip you up and you recognized by definition if  
the mean difference in our sample data is -0.038403 then the p value should  
reflect 50/50 p value?

Let’s first just try different rCauchy values with ttestBF.

justtwo <- male\_100 %>%

filter(runner %in% c("Usain Bolt", "Asafa Powell")) %>%

droplevels %>%

as.data.frame

ttestBF(formula = timing ~ runner, data = justtwo, rscale = "medium")

## Bayes factor analysis

## --------------

## [1] Alt., r=0.707 : 5.164791 ±0%

##

## Against denominator:

## Null, mu1-mu2 = 0

## ---

## Bayes factor type: BFindepSample, JZS

ttestBF(formula = timing ~ runner, data = justtwo, rscale = "wide")

## Bayes factor analysis

## --------------

## [1] Alt., r=1 : 4.133431 ±0%

##

## Against denominator:

## Null, mu1-mu2 = 0

## ---

## Bayes factor type: BFindepSample, JZS

ttestBF(formula = timing ~ runner, data = justtwo, rscale = .2)

## Bayes factor analysis

## --------------

## [1] Alt., r=0.2 : 6.104113 ±0%

##

## Against denominator:

## Null, mu1-mu2 = 0

## ---

## Bayes factor type: BFindepSample, JZS

Okay the default medium returns just what we reported earlier 5:1 odds. Going  
wider gets us 4:1 and going narrower (believing the difference is smaller) takes  
us to 6:1. Not huge differences but noticeable and driven by our data.

Let’s investigate directional hypotheses with ttestBF. First let’s ask what’s the evidence that Bolt is faster than Powell **NB the order is driven by factor level in the dataframe not the order in the filter command below. Also note that faster is a lower number**

justtwo <- male\_100 %>%

filter(runner %in% c("Usain Bolt", "Asafa Powell")) %>%

droplevels %>%

as.data.frame

# notice these two just return the same answer in a different order

ttestBF(formula = timing ~ runner, data = justtwo, nullInterval = c(0, Inf))

## Bayes factor analysis

## --------------

## [1] Alt., r=0.707 0

ttestBF(formula = timing ~ runner, data = justtwo, nullInterval = c(-Inf, 0))

## Bayes factor analysis

## --------------

## [1] Alt., r=0.707 -Inf

So the odds that Bolt has a bigger number i.e. is slower than Powell is 0.04:1  
and the converse is the odds that Bolt has a smaller timing (is faster) is 10:1.  
You can feel free to put these in the order that makes the most sense to your  
workflow. They’re always going to be mirror images.

And yes in some circumstances it is perfectly rational to combine the  
information by dividing those odds. but accomplishing it is trivial. Running this code snippet essentially combines

what we know in both directions of the hypothesis.

justtwo <- male\_100 %>%

filter(runner %in% c("Usain Bolt", "Asafa Powell")) %>%

droplevels %>%

as.data.frame

powellvbolt <- ttestBF(formula = timing ~ runner, data = justtwo, nullInterval = c(-Inf, 0))

powellvbolt[1]/powellvbolt[2]

## Bayes factor analysis

## --------------

## [1] Alt., r=0.707 -Inf

**What have I learned**

* All three approaches yielded similiar answers to our question at the bar.
* Frequentist methods have stood the test of time and are pervasive in the  
  literature
* Computational methods like resmapling allow us to free ourselves  
  from some of the restrictions and assumptions in classical hypothesis testing  
  in an age when cpmpute power is cheap
* Bayesian methods allow us to speak in  
  the very human vernacular language of probabilities about our evidence.

**Done!**