UWEO StatR201 Lecture 3

Maximum Likelihood Estimation - and its Use in Regression

Assaf Oron, January 2013

Another Busy Evening…

Catch-Up from Lecture 2:

- Outliers in X
- Is Non-Normality a Big Deal?
- Regression Simulation, looking at Non-Normality and Multiple Testing

Then:

- Estimation and the Intuitive ("Moments") Estimator
- Likelihood and Maximum Likelihood
- Non-Regression Examples
- MLEs and Regression
- Quantile and Robust Regression

Our First Regression Simulation

In Lecture 2, the briefly mentioned subject of **Multiple Testing** generated a lively discussion. We saw a **Manhattan plot** used in genetics, as a visual aid to account for multiple testing. First, here's a more organized explanation of the problem:

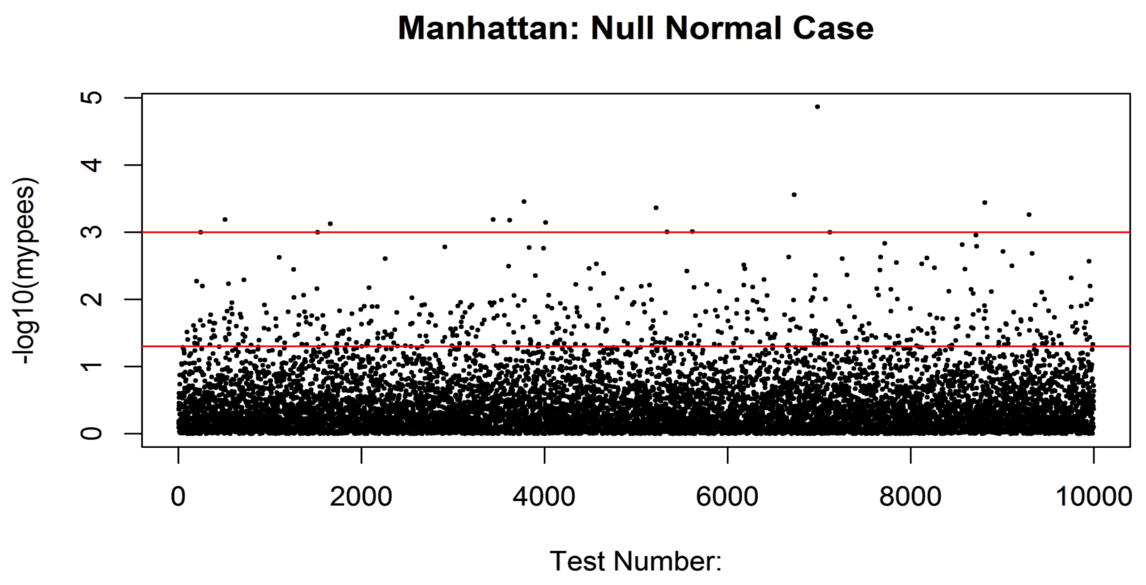
P-values (or some variation thereof) are almost universally used to help detect significant signals. But one must keep in mind:

1. P-values are calculated **under the Null**, i.e., assuming that the Null is true.
2. Under the very same Null, **p-values are uniform random variables in [0,1]**.
3. Inevitably, even if the Null *is* true for all your p-values(=tests) - **the more p-values you look at, the more likely you are to see values closer and closer to zero(=*"significant"*).**

- The math is ridiculously simple. It's the concept – and its discouraging practical implications – that evades many a good scientist.
- Ok, let's generate our very own Manhattan plot!

Regression Simulation: Starting with a Null Case

```
x = matrix(rnorm(1e+05), nrow = 10)  # x, y independent of each other
y = matrix(rnorm(1e+05), nrow = 10)  # Now, a first example using 'mapply'; can be done with loop too
mypees = mapply(function(z, w) summary(lm(w ~ z))$coef[2, 4], split(x, col(x)),
    split(y, col(y)))
plot(-log10(mypees), pch = 19, cex = 0.3, main = "Manhattan: Null Normal Case",
    xlab = "Test Number:")
abline(h = c(-log10(0.05), 3), col = 2)
```



So? Do you see anything **interesting??** Let’s compare to some of these online figures.

Regression Simulation: Your Turn

First, replicate the Normal case on your machine. Then, start mixing it up:

- Symmetric variables, but heavier tails (e.g., logistic or double-exponential)
- Skewed variables - mild (gamma with large shape parameter) or extreme (exponential)
- Compare the amount of significant p-values to the Normal case. Does it matter whether we make both x and y “not nice”, or just one of them?
- Then, make it **not Null** by defining y as a linear function of x, **plus** its own random noise.
- First plot the p-values as before. How many go “undetected”? (i.e., below the lower $p = 0.05$ red line). The probability of detecting significance when the alternative is really true, is known as **the Power of the test**.
- Then, return the $\hat{\beta}$ values instead of (or together with) the p-values, and do a $\texttt{qqnorm}$ of them. For this one, repeat while increasing n to 100 (you can reduce M the ensemble size from 10000 to 1000 in this part).

Then we go on break. What we don’t get to here, will be finished at home.

Questions? (online/in-class)

And now, to the "Real" Lecture 3. What's an Estimate?

This is a classic philosophical quandary of trying to explain the obvious - or rather, to explain the thought patterns we all routinely and subconsciously practice.

All of us estimate, and generate **predictions** from our estimates, all the time, without batting an eyelash *(how else would we ever catch a ball thrown towards us? Or correct our first, opposing-coach-donated kick attempt to drill the second one perfectly?)*

As Eli nicely described last quarter, **statistical estimation** follows, roughly speaking, along these lines:

- Describe reality via **a probability model**
- Observe the data
- Use the data to calculate estimates for the model's **parameters.**

The concept is simple; the application can also be simple - or as complicated as they come…

Method-of-Moments, a.k.a. "Intuitive" Estimators

The birth of modern statistics is marked, among other things, by the establishment of universal principles for the estimation of *any* model, and the theoretical proof of estimator properties. **Maximum Likelihood Estimators (MLEs)** have been at the heart of this revolution.

Unfortunately, MLEs are *not* always the most immediately intuitive choice. That award goes to moments estimators, or MMEs.

In moment-based estimation, you observe the role each parameter plays in your model - and replace it by the analogous summary statistic from the data. For example:

- With a Normal model, you use the mean of the data $\bar{X}$ to estimate the location parameter μ.
- With a Bernoulli/Binomial model for n coin tosses, you match the observed proportion of "heads", say, h/n, to the parameter p describing the probability of tossing "heads".
- And so forth. This type of estimate is attributed to Carl Pearson, back in the 19th Century. What else would anyone need? Why replace a simple common-sense solution?

Why MLEs? And what's a Likelihood?

Well, it turns out the MLEs (introduced by the ubiquitous R.A. Fisher) have better asymptotic properties.

As $n \to \infty$, their **bias** goes to zero (this is called *consistency*), and their **variance** is the smallest among all estimators sharing the first property (this is called *efficiency*). We will learn more about bias and variance during this course. Actually, maybe today.

Of course, all these nice properties hold only if our model for the data is correct.

Another point to consider: many problems are too complicated to find an "intuitive" (Moments) estimator. On the other hand, the MLE can (almost) *always* be found.

But what are MLEs? Even before that, what's a Likelihood?

The likelihood is simply the probability model for the data. It is the **joint** probability for observing all the data we have, assuming that our model is exactly correct.

…so why call it a Likelihood, if it’s just a Probability?

Aha. Yes, it is a semantic trick - but also a paradigm shift. See, the probability for the data is a function of the data **viewed as random and yet-to-be-observed**, and assuming the model is fixed and its parameters constant (although usually not directly observable). Now flip things around and call it a Likelihood:

$$f(y \mid \beta) = L(\beta; y)$$

where f is the density or probability function, and y is our vector of random observations.

So while the probability density is a function of the observations, the Likelihood is a function of the parameters.

Note the slight notation difference: while y is *conditional* on β, when we flip y around we don’t get the vice versa. This is because, even as we treat y as *fait accomplix*, it doesn’t mean that β becomes random. **Heavens no. That would turn Fisher into a Bayesian :)**

So what kind of beast *is* the Likelihood? It is an **algebraic function of β, which is an algebraic variable** – with y playing the role of known constants.

Now, **The MLE is the parameter value that maximizes the likelihood.** Perhaps not as intuitive as moments-based estimations, but still simple. We all know how to maximize algebraic functions, right?

MLE Example 1: Binomial Coin Tosses

Say we have a coin-tossing machine and we want to evaluates its fairness, by having it toss n pennies, and h of them come out heads. We use the Binomial model (i.i.d. outcomes, each with p probability of tossing heads). What is the MLE of p?

To save you time, the likelihood is

$$L(p; n, h) = \binom{n}{h} p^h (1-p)^{n-h}$$

Find the MLE $\hat{p}$.

Try to hack at it… *(hint: take the log of the likelihood first)*

Questions? (online/in-class)

MLE Example 2: The Unknown-Interval Uniform

So for the Binomial, the MLE and intuitive estimates are the same. This happens for the Normal, too, and in many other cases. But not always. Consider this sad story.

It is a snow day, and bus schedules become irrelevant. You know that Metro is trying to keep a regular frequency for your bus line. So the bus comes at fixed intervals, but no one knows what the frequency is.

You arrive at the bus stop, and decide to determine the frequency by asking all present how long they've been waiting. How will you use these data to estimate the time interval between buses? We assume people arrive at the stop at a constant rate, independently of each other.

The wait-time in this story can be modeled as a uniform random variable, in the interval $(0, \beta)$ minutes, with β constant but unknown.

First: what is the Moment/intuitive estimator? *(find one! Hint: what's the first moment?)*

MLE Example 2: The Unknown-Interval Uniform

Now to the likelihood. The form of the density for a single observation is simply $1/\beta$, so for n independent observations it is $1/\beta^n$.

Whoa! This is a montone-decreasing function that goes to infinity as $n \to 0$, and to zero as $n \to \infty$. No maximum, right?

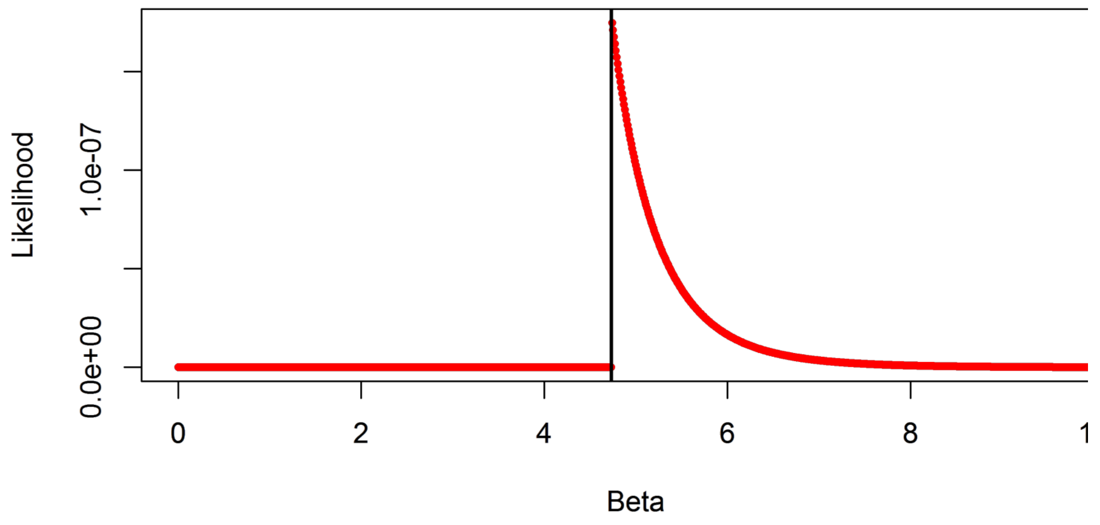
Almost. Our estimate of β **cannot be smaller than any of the observations** (why?) So in practice, there *is* a maximum. It is the smallest allowed value, or in other words $\max(y)$. Formally, this can be solved by explicitly noting that the density has a **restricted range:**

$$f(y) = 1/\beta \;\; I(y \in [0, \beta]),$$

where the algebraic **indicator function** is one if the logical condition holds, and zero otherwise.

```
beta = 5
y = runif(8, 0, beta)

x = (0:1000)/100
plot(x, ifelse(x < max(y), 0, x^(-10)), pch = 19, col = 2, cex = 0.5, xlab
    ylab = "Likelihood")
abline(v = max(y), lwd = 2)
```



So Which is Better Here: the “Intuitive” or the MLE?

Since our model is true, we know that as $n \to \infty$ the MLE will win out. But for small samples, frankly, it is ridiculous (think about it when $n = 1$).

It is also *guaranteed* to be biased, which is what makes it less intuitive (how do I know? And to what direction? Hey, are we agreed of what “bias” means?).

But more fundamentally, how do we gauge estimator performance? The standard metric is the **Mean Squared Error (MSE)**. As its name implies, it is simply the average of the square errors. Just like in regression, this square-based metric decomposes very nicely:

$$MSE\left(\hat{\beta}, \beta\right) = \overline{\left(\hat{\beta} - \beta\right)^2} = (Bias)^2 + Variance$$

But sometimes, other variations on this metric are more useful/responsible. Can you think of some?

One of our homework questions will be to simulate this bus-stop scenario and evaluate estimators.

Questions? (online/in-class)

MLEs and Regression

For regression, the MLE $\hat{\beta}$ is our familiar least-squares estimator – if we assume Normal errors.

As seen earlier, non-Normality doesn't hurt $\hat{\beta}$ too much. There is also **the Gauss-Markov Theorem**, which states that $\hat{\beta}$ is the best linear unbiased estimator (BLUE!) under conditions fairly similar to those of the CLT (so no Normality necessary).

However, **that doesn't mean $\hat{\beta}$ is the actual MLE for all those cases in which it is the BLUE. This is an excellent example for the distinction between assumptions, properties and behavior.**

For example: with **logistically-distributed noise**, which is symmetric, smooth and algebraically more tractable than Normal - but with thicker tails - there is no closed-form MLE. It can be found only numerically.

```
# logistic (-log(likelihood)) function with a covariate vector 'cv'
logislike = function(bet, dat, cv) {
    length(dat) * log(bet[3]) + 2 * sum(log(cosh((dat - bet[1] - bet[2] * cv)/(2 *
        bet[3]))))
}
z = 2 * rnorm(20)
y = z + rlogis(20)
summary(lm(y ~ z))$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1127     0.5036  0.2238   0.8255
## z             0.6713     0.2491  2.6954   0.0148
```

```
mle = optim(c(0, 0, 1), fn = logislike, dat = y, cv = z)
mle$par  # Intercept, beta, and scale ('sd') parameters
```

```
## [1] 0.03813 0.64320 1.18603
```

When would one Need a More Robust Solution?

“Classical” ordinary regression provides a least-squares solution. As we know, these solutions are sensitive to outliers. In the extreme case (when outliers are so prevalent that the mean and/or variance are undefined), the regression - just like the mean of an ordinary sample – **breaks down.** *(see the HW regression-simulation problem)*

Even without gross outliers, if the underlying noise distribution is skewed, the regression estimate – while unbiased about the behavior of the mean, **does not necessarily reflect the behavior of the center of the population** – which might be better approximated by the median.

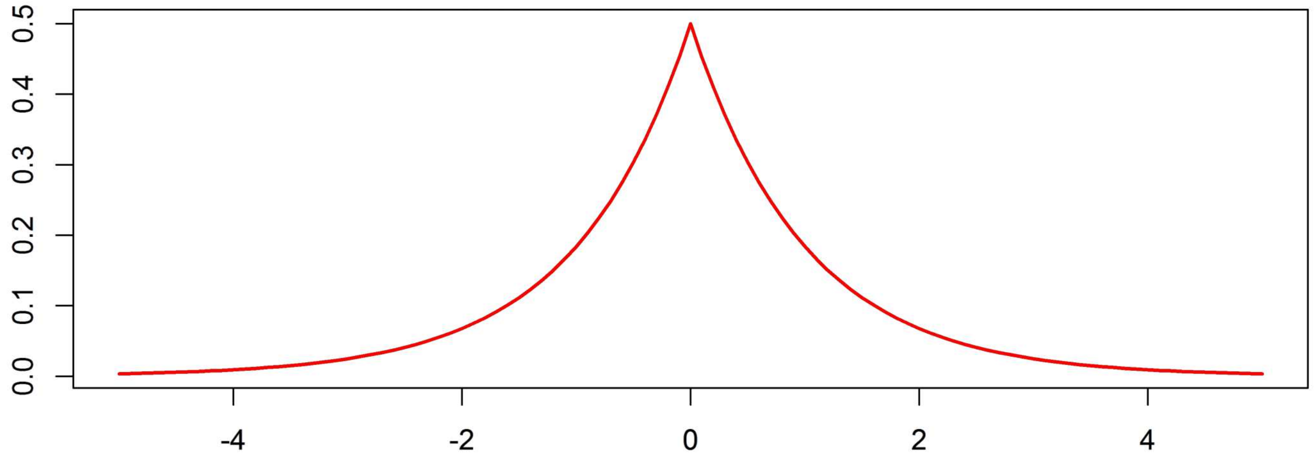
In short, is there a more robust regression, that plays the same role vs. least-squares regression as the median vs. the mean?

Yes.

Quantile Regression

The basic quantile regression happens to be the MLE under **double-exponential, or Laplace-distributed** noise. This is not a very *realistic* distribution:

```
library(VGAM)
par(mar = c(3, 3, 1, 1))
curve(dlaplace, -5, 5, col = 2, lwd = 2)
```



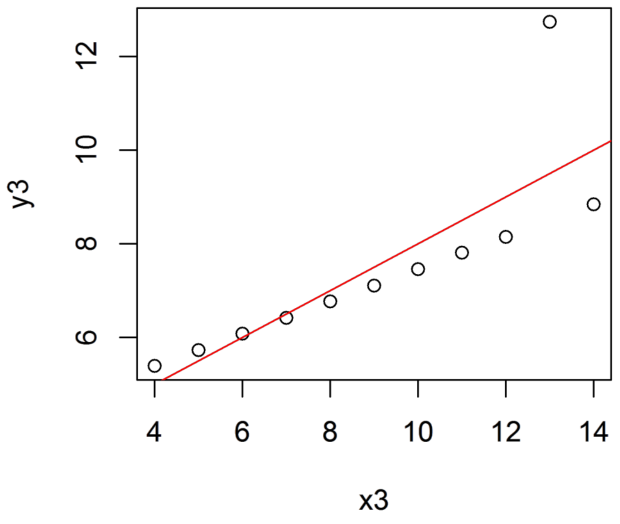
However, again this MLE is chosen not for the assumption but for its behavior: it tracks the expected median rather than the expected mean. **Algebraically, it minimizes the sum of absolute errors, rather than square errors.**

Quantile Regression

```
# The Anscombe example w/the outlier
summary(lm(y3 ~ x3, data = anscombe))$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0025     1.1245   2.670 0.025619
## x3            0.4997     0.1179   4.239 0.002176
```

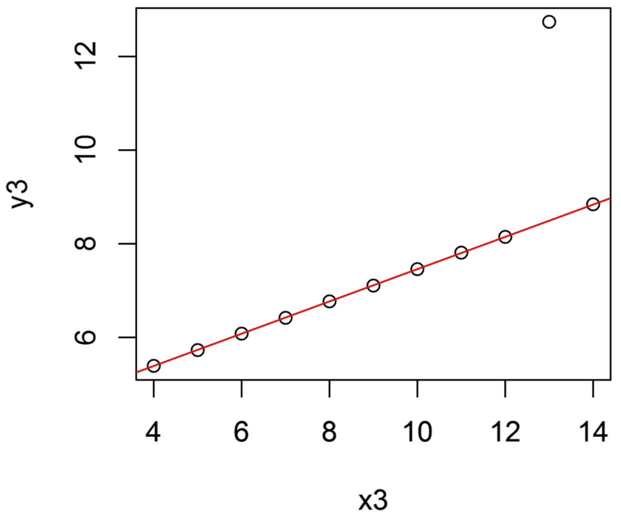
```
plot(y3 ~ x3, data = anscombe)
abline(lm(y3 ~ x3, data = anscombe), col = 2)
```



```
library(quantreg)
summary(rq(y3 ~ x3, data = anscombe))$coef
```

```
##             coefficients lower bd upper bd
## (Intercept)        4.010    3.990   4.0100
## x3                 0.345    0.345   0.7204
```

```
plot(y3 ~ x3, data = anscombe)
abline(rq(y3 ~ x3, data = anscombe), col = 2)
```



Quantile Regression - and Robust Regression “Proper”

The quantile regression function `rq` has the parameter `tau` which defaults to 0.5 (signifying the median). Changing it allows the user to estimate other quantiles of the noise distribution. Algebraically, a non-median quantile regression minimizes an asymmetrically-weighted sum of absolute errors (different weights for positive and negative errors).

Statistically, it is the MLE of an asymmetric Laplace distribution – which is even less realistic than the “vanilla” Laplace… **However, sometimes this solution’s behavior might be exactly what you need.**

So, if we tweak assumptions to generate behavior (or in other words: **invent practical solutions, then look for models that justify them**) - why not *“cut out the model middle-man”*, and focus on behavior from the start?

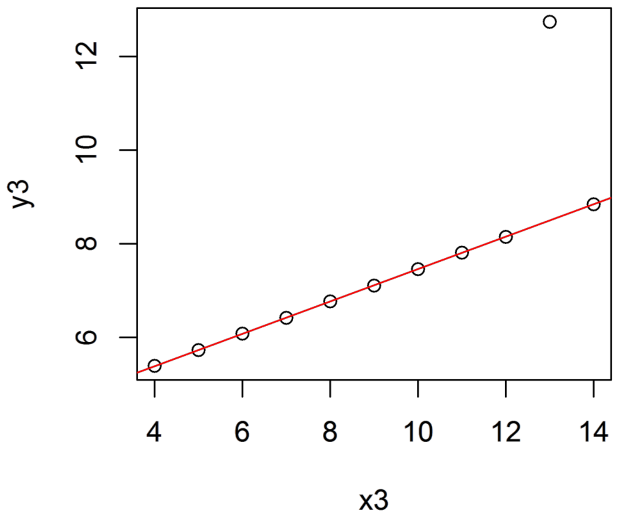
This is the domain of straight-up robust estimation. The leading algorithm approach for regression (and also for location-finding on a single collection of observations) is known generically as **M-estimators** (as you might guess, ordinary least-squares and quantile regression become ‘just’ two more special cases of an M-estimator).

Robust Regression

```r
library(MASS)
summary(rlm(y3 ~ x3, data = anscombe))$coef
```

```
##              Value Std. Error t value
## (Intercept) 4.0035  0.0040425   990.3
## x3          0.3457  0.0004238   815.8
```

```r
plot(y3 ~ x3, data = anscombe)
abline(rlm(y3 ~ x3, data = anscombe), col = 2)
```



Robust regression algorithms down-weight outliers recursively, according to some pre-set formula. There are many different algorithms, with the goal of being as resistant to outliers as (median-)quantile regression – i.e., can handle up to 50% outliers without breaking down – but as efficient(=small-variance) as least-squares regression when the observations are well-behaved.

These estimators are **Not** MLEs for any specific distribution, but their **properties** are proven (asymptotically normal, etc.) - without relying on any probability-model assumption.

Ok, play with these three - `lm, rq, rlm` - for a while, with whatever dataset you want to throw at them…

But first, note the huge difference in confidence in $\hat{\beta}$ between the least-squares, quantile and robust regressions. Can you explain it?

Questions? (online/in-class)

R Annyoance of the Week

I actually changed this one following the office hours.

In HW1 you were asked to look for functions that calculate the VIF online and compare them, and also to make one yourselves. There are two functions called `vif` that are easily found: one in the `car` package (another potentially useful book, btw), and one in the `HH` package.

Now, if you load them both into your session, which one will run when you just type `vif`? If you don't do anything special, it will be the most recent one.

```
library(car)
library(HH)
vif
```

```
## function (xx, ...)
## UseMethod("vif")
## <environment: namespace:HH>
```

You can control that - and in fact, learn how to take packages off the session (no, there is no command called `unlibrary`).

```
search()
```

```
##  [1] ".GlobalEnv"            "package:HH"
##  [3] "package:latticeExtra"  "package:RColorBrewer"
##  [5] "package:leaps"         "package:multcomp"
##  [7] "package:survival"      "package:mvtnorm"
##  [9] "package:grid"          "package:car"
## [11] "package:nnet"          "package:lattice"
## [13] "package:scatterplot3d" "package:VGAM"
## [15] "package:stats4"        "package:splines"
## [17] "package:quantreg"      "package:SparseM"
## [19] "package:MASS"          "package:lmtest"
## [21] "package:zoo"           "package:knitr"
## [23] "package:stats"         "package:graphics"
## [25] "package:grDevices"     "package:utils"
## [27] "package:datasets"      "package:methods"
## [29] "Autoloads"              "package:base"
```

R Annyoance of the Week

`search()` shows you all the packages you have currently loaded or attached. R looks for any function name, in packages according to the order in this list (starting from #2). Every package you load, goes into position 2 and pushes everyone else down.

This is also how you remove a library from the session.

```
detach(2)
vif
```

```
## function (mod, ...)
## {
##     UseMethod("vif")
## }
## <bytecode: 0x000000000795e3b0>
## <environment: namespace:car>
```

You can also do `detach(package:HH)`. Now, you can load `HH` back but **still have R look for `vif` in `car` first** - by loading `HH` in a lower position.

```
library(HH, pos = 20)
vif
```

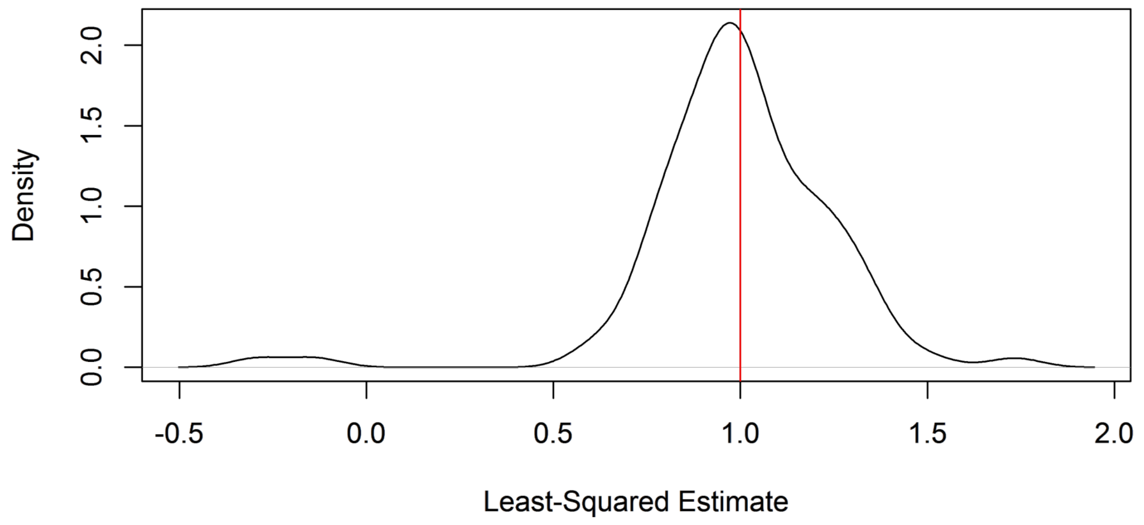
```
## function (mod, ...)
## {
##     UseMethod("vif")
## }
## <bytecode: 0x000000000795e3b0>
## <environment: namespace:car>
```

You can also directly control which `vif` is run, by calling `car::vif` or `HH::vif`.

R Cool Trick of the Week

```
x = matrix(rexp(2000), nrow = 20)
y = matrix(x + rexp(2000), nrow = 20)
z = mapply(function(a, b) lm(b ~ a)$coef[2], split(x, col(x)), split(y, col(y)))
mymain = bquote(paste("Distribution of ", hat(beta), " with Exponential Noise (",
    .(dim(x)[2]), " runs of n=", .(dim(x)[1]), " each)", sep = ""))
plot(density(z), main = mymain, xlab = "Least-Squared Estimate")
abline(v = 1, col = 2)
```

Distribution of $\hat{\beta}$ with Exponential Noise (100 runs of n=20 each)



With `bquote`, you can combine text, math symbols and the actual value of variables - all in the same caption.

To learn more, look up `?plotmath`, `?bquote`.

And thanks to Eli!