Denver  User Group

Creating an Autocorrelation Plot in ggplot2

Peter DeWitt
dewittpe@gmail.com

18 January 2011

- Show the thought process and coding to make a custom `ggplot2` style ACF plot.
  - Show some of the errors and solutions to them.
  - Give an overview of the construction of the plots using layers.
  - Show how to use the base R functions to generate the data needed for the `ggplot2` graphics.
- Provide an example of creating a LaTeX, in this case a Beamer, document using `Sweave`.

Need an Autocorrelation Plot

- While working on a homework assignment for an introductory Bayesian course I needed an autocorrelation plot to investigate the convergence of a MCMC algorithm.
- I had used `ggplot2` for creating all the plots in the report, but when looking for the autocorrelation in the simulated parameter values I found that there was no option in the `ggplot2` library for ACF Plots.
- Not wanting to use the base  `acf()` plot I though I should make my own using `ggplot2`.
- The function in this talk shows the one I used for the assignment and then a more generalized function I started to write for future use.

The Data Set and some Syntax

The goal of the assignment was to find the posterior distribution for two parameters θ and τ. The simulated values from the MCMC simulations where stored in a data frame called `params`.

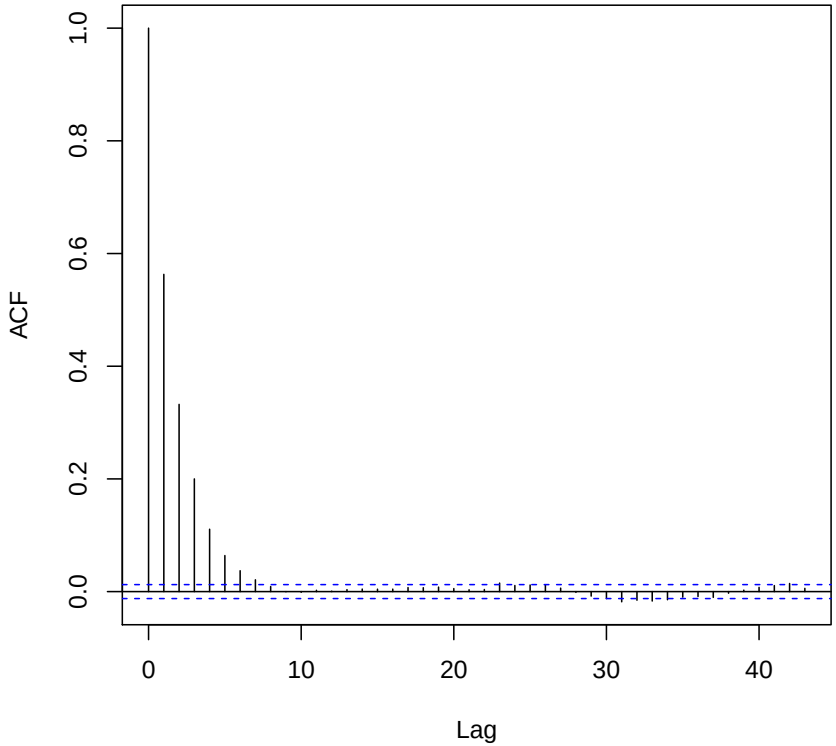
```
> dim(params)
```

```
[1] 25001     2
```

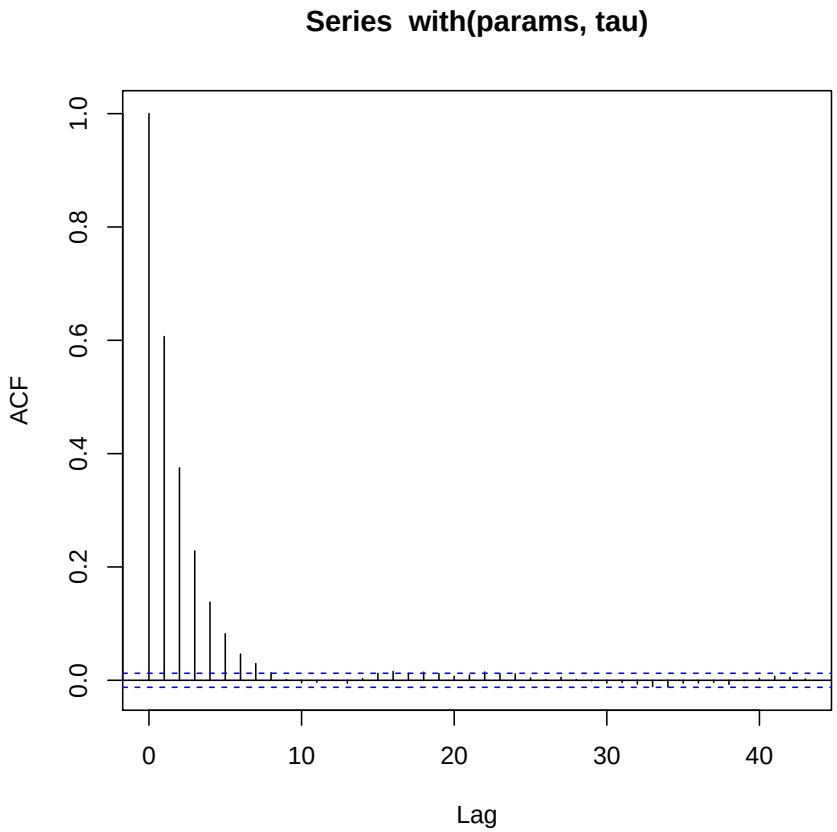
```
> head(params)
```

```
      theta       tau
1 0.01000000 1.0000000
2 0.01238948 0.7005268
3 0.01180436 0.6339757
4 0.01200773 0.7051600
5 0.01648895 0.5985906
6 0.01013832 0.6795824
```

```
> acf(with(params, theta))
```




```
> acf(with(params, tau))
```



Small note: the use of the `with` command is used here because of color coding issues in the text editor I use and the LaTeX command $.

Where to start?

- The ACF Plots in the base  package could be improved on.
- What do I need to create one?
  - Calculate the correlation for each lag between 1 and, uh. . . , a big number
  - Store the different correlations in a data frame and then. . .
- WAIT! Before wasting time to build a whole new function see what is already generated from the `acf()` function.

What is in the acf() function?

```
> p <- acf(with(params, theta), plot = FALSE)
> summary(p)
```

```
         Length Class  Mode
acf      44     -none- numeric
type      1     -none- character
n.used    1     -none- numeric
lag      44     -none- numeric
series    1     -none- character
snames    0     -none- NULL
```

```
> p
```

```
Autocorrelations of series âÃŸwith(params, theta)âÃŹ, by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.563  0.332  0.200  0.111  0.064  0.037  0.021  0.009  0.000 -0.001
    11     12     13     14     15     16     17     18     19     20     21
 0.002  0.001  0.003  0.004  0.004  0.004  0.007  0.007  0.008  0.005  0.003
    22     23     24     25     26     27     28     29     30     31     32
 0.004  0.015  0.011  0.012  0.012  0.006 -0.001 -0.008 -0.012 -0.018 -0.015
    33     34     35     36     37     38     39     40     41     42     43
-0.017 -0.014 -0.010 -0.009 -0.010 -0.003  0.003  0.008  0.011  0.014  0.006
```

Start of the acf plot

Data Manipulation and first steps with a bar plot

```
> baseACF <- with(p, data.frame(lag, acf))
> head(baseACF, 3)
```

```
  lag       acf
1   0 1.0000000
2   1 0.5630029
3   2 0.3322593
```

```
> qplot(lag, acf, data = baseACF, geom = "bar")
stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
Error in pmin(y, 0) : object 'y' not found
```

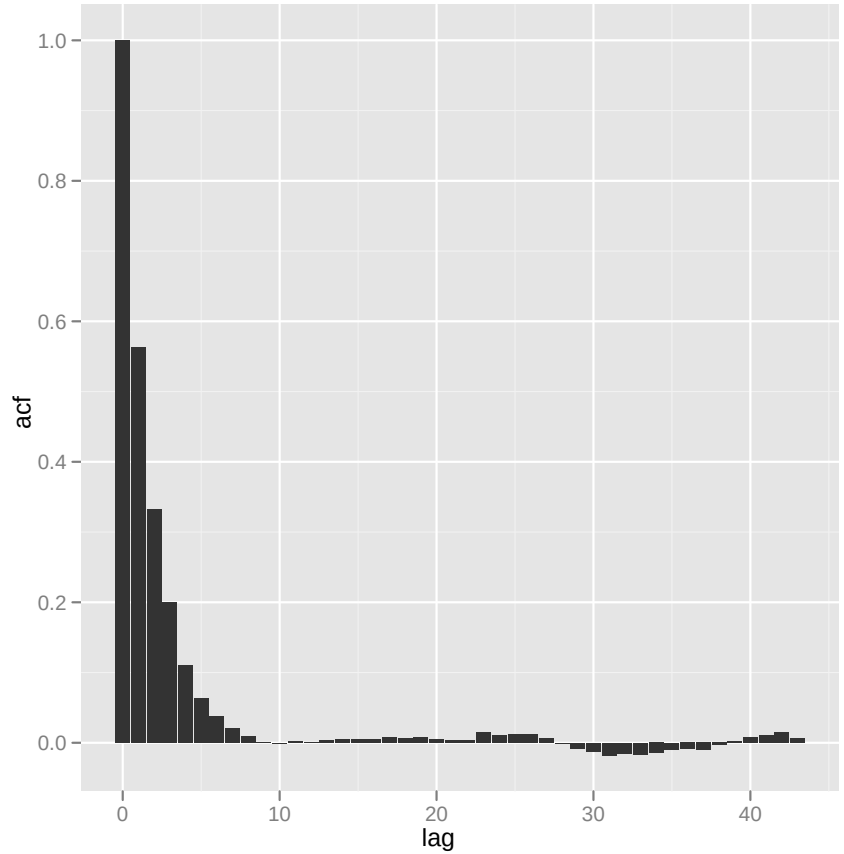
Fix this error by using the argument: `stat_identity`. This will keep the lags as is and will not bin them together.

```
> qplot(lag, acf, data = baseACF, geom = "bar", stat = "identity")
Warning message:
Stacking not well defined when ymin != 0
```

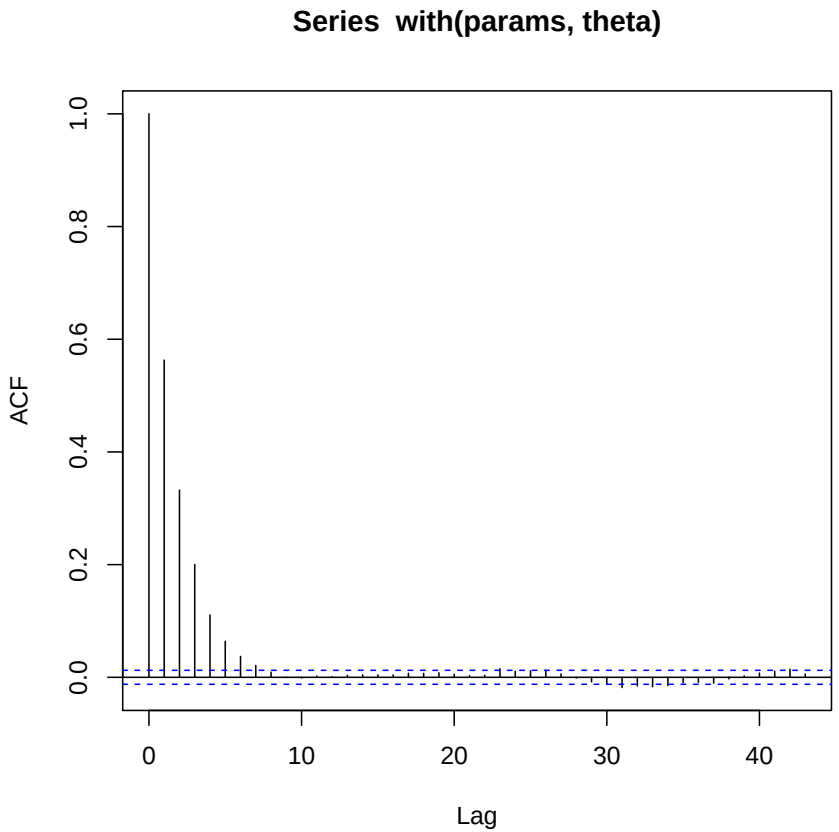
Fix this error with the argument: `position_identity`.

First Iteration of the ggplot2 style ACF Plot

```
> q <- qplot(x = lag, y = acf, data = baseACF, geom = "bar", stat = "identity",
+     position = "identity")
> print(q)
```



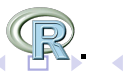

```
> acf(with(params, theta))
```



Adding Confidence Intervals

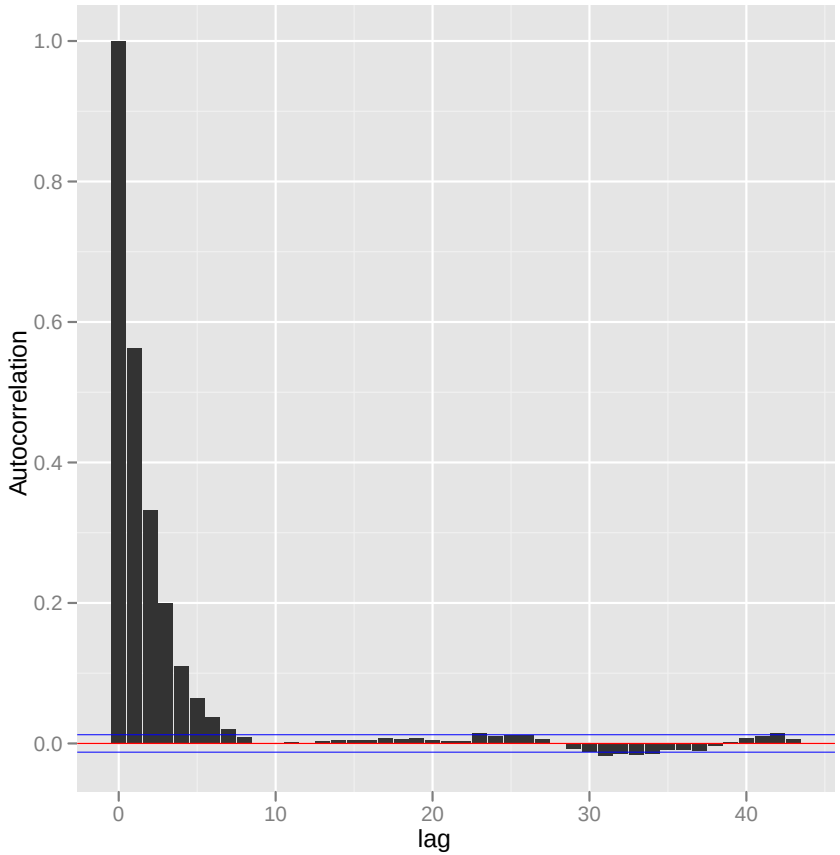
It will be beneficial to start writing a function.

```
> qacf <- function(x, conf.level = 0.95) {
+     ciline <- qnorm((1 - conf.level)/2)/sqrt(length(x))
+     bacf <- acf(x, plot = FALSE)
+     bacfdf <- with(bacf, data.frame(lag, acf))
+     q <- qplot(lag, acf, data = bacfdf, geom = "bar", stat = "identity",
+         position = "identity", ylab = "Autocorrelation")
+     q <- q + geom_hline(yintercept = -ciline, color = "blue",
+         size = 0.2)
+     q <- q + geom_hline(yintercept = ciline, color = "blue",
+         size = 0.2)
+     q <- q + geom_hline(yintercept = 0, color = "red", size = 0.3)
+     return(q)
+ }
```

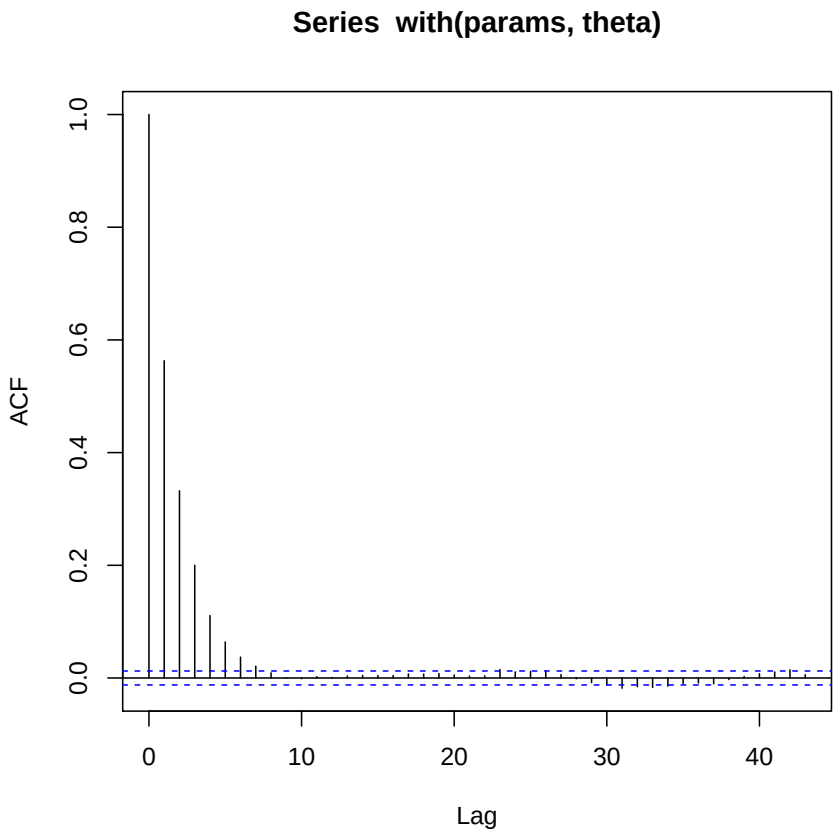
The syntax has each layer added to the plot one at a time. `Sweave` does not preserve linebreaks in a single line of code. This is because of the use of the `parse()` and `deparse()` functions calls in  .

What does it look like now?

```
> p <- qacf(with(params, theta))
> print(p)
```




```
> acf(with(params, theta))
```



How many lags to display?

How many lags should be displayed? There is a default setting in the `acf()` function.
A few lines of code from the base R acf function:

```
> acf
function (x, lag.max = NULL, type = c("correlation", "covariance",
    "partial"), plot = TRUE, na.action = na.fail, demean = TRUE,
    ...)
{

    sampleT <- nrow(x)
    nser <- ncol(x)
    if (is.null(lag.max))
        lag.max <- floor(10 * (log10(sampleT) - log10(nser)))
    lag.max <- min(lag.max, sampleT - 1)
    if (lag.max < 0)
        stop("'lag.max' must be at least 0")
```

What about the lags?

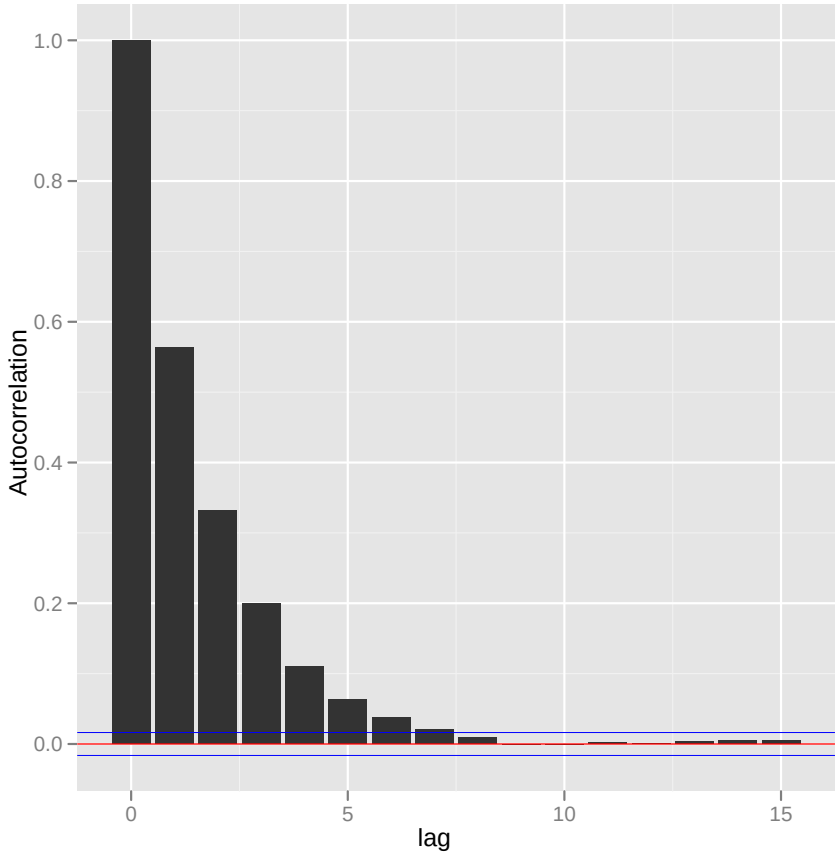
```
> qacf <- function(x, conf.level = 0.95, max.lag = NULL, min.lag = 0) {
+     ciline <- qnorm((1 - conf.level)/2)/sqrt(length(x))
+     bacf <- acf(x, plot = FALSE, lag.max = max.lag)
+     bacfdf <- with(bacf, data.frame(lag, acf))
+     if (min.lag > 0) {
+         bacfdf <- bacfdf[-seq(1, min.lag), ]
+     }
+     q <- qplot(lag, acf, data = bacfdf, geom = "bar", stat = "identity",
+         position = "identity", ylab = "Autocorrelation")
+     q <- q + geom_hline(yintercept = -ciline, color = "blue",
+         size = 0.2)
+     q <- q + geom_hline(yintercept = ciline, color = "blue",
+         size = 0.2)
+     q <- q + geom_hline(yintercept = 0, color = "red", size = 0.3)
+     return(q)
+ }
```

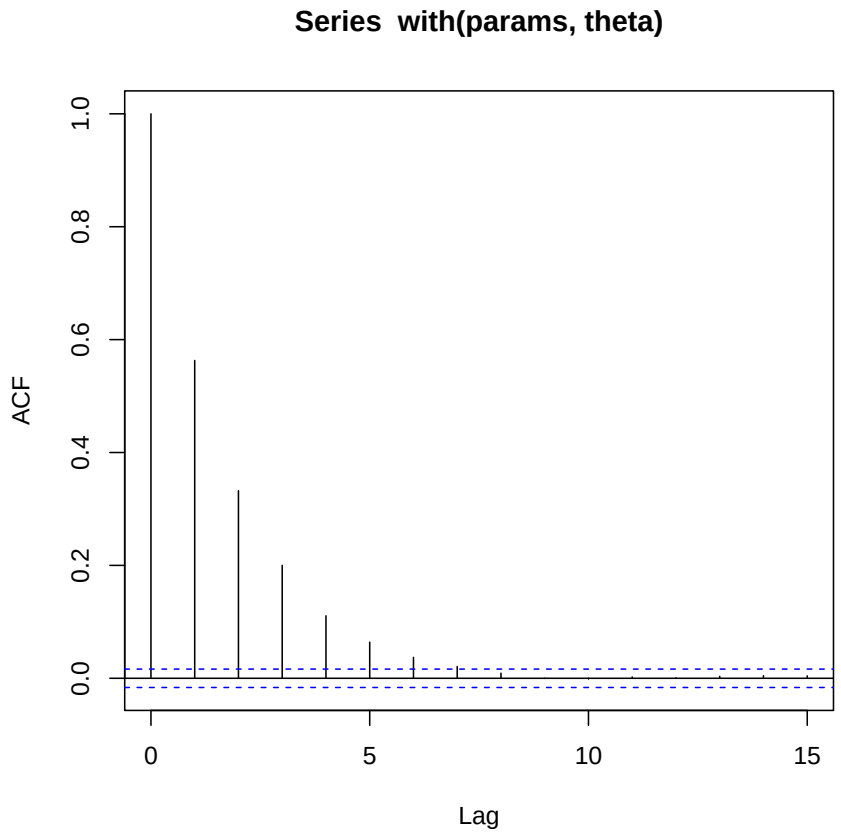
A Couple Quick Examples

Example 1

```
> p <- qacf(with(params, theta), conf.level = 0.99, max.lag = 15)
> print(p)
```

```
> acf(with(params, theta), ci = 0.99, lag
```

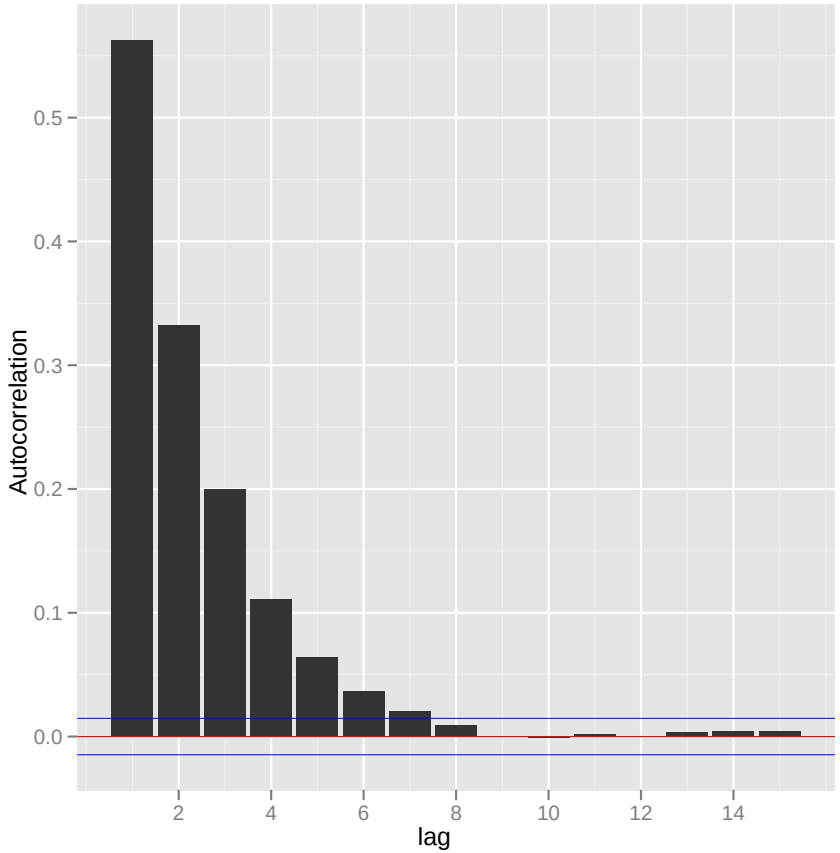




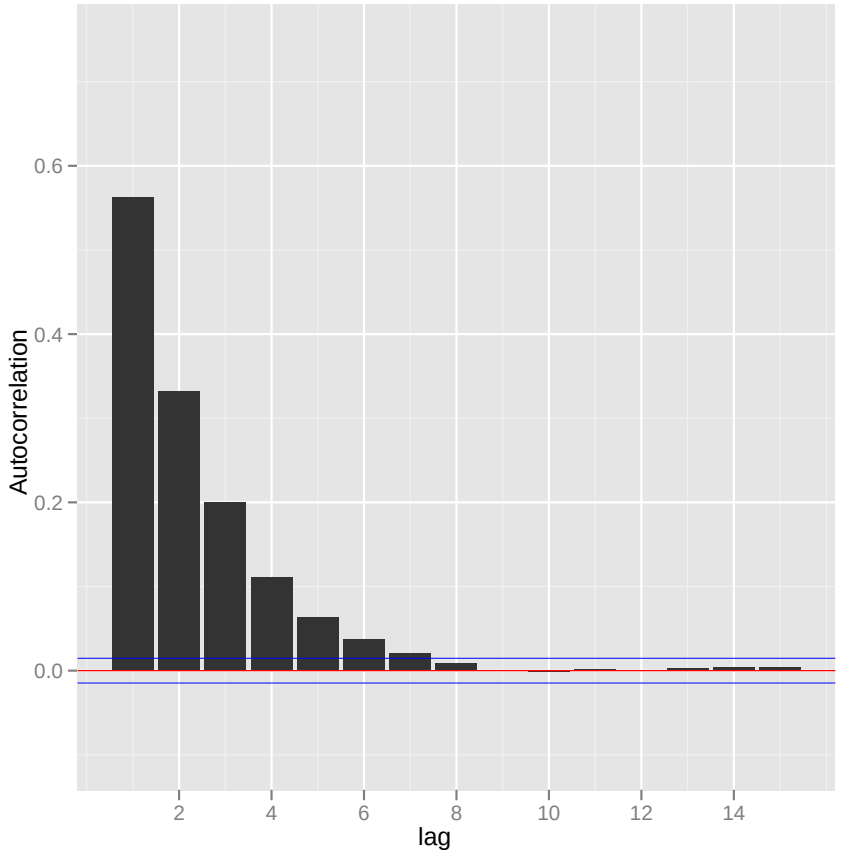
A Couple Quick Examples

Example 2

```
> p <- qacf(with(params, theta), conf.level = 0.98, min.lag = 1,
+     max.lag = 15)
> print(p)
```




```
> p <- p + ylim(-0.1, 0.75)
> print(p)
```



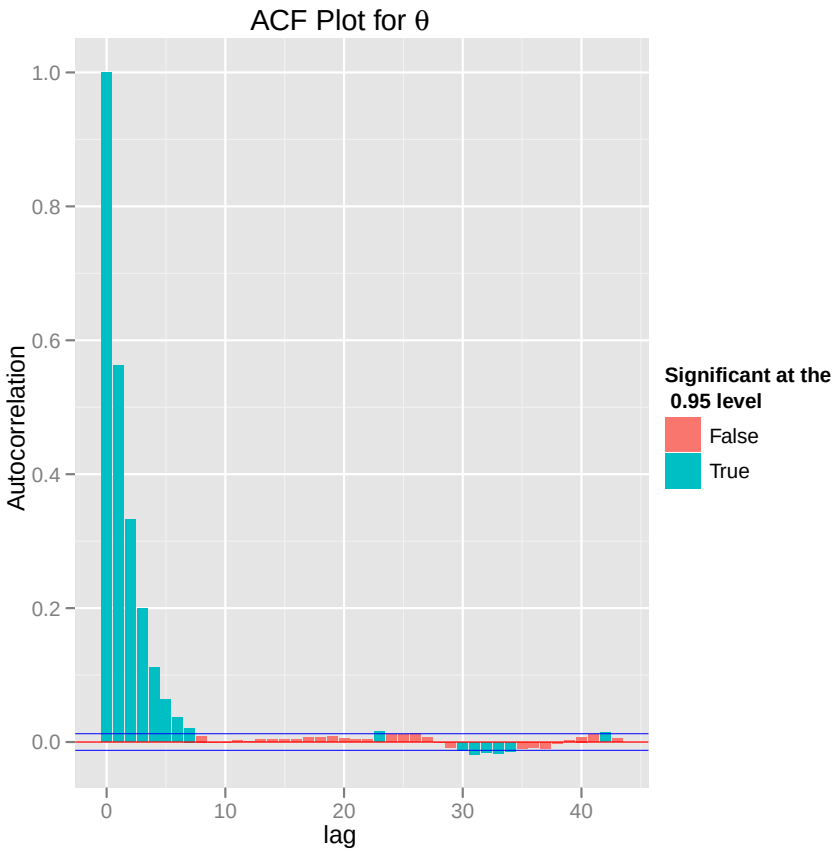
Which Lags are Significance?

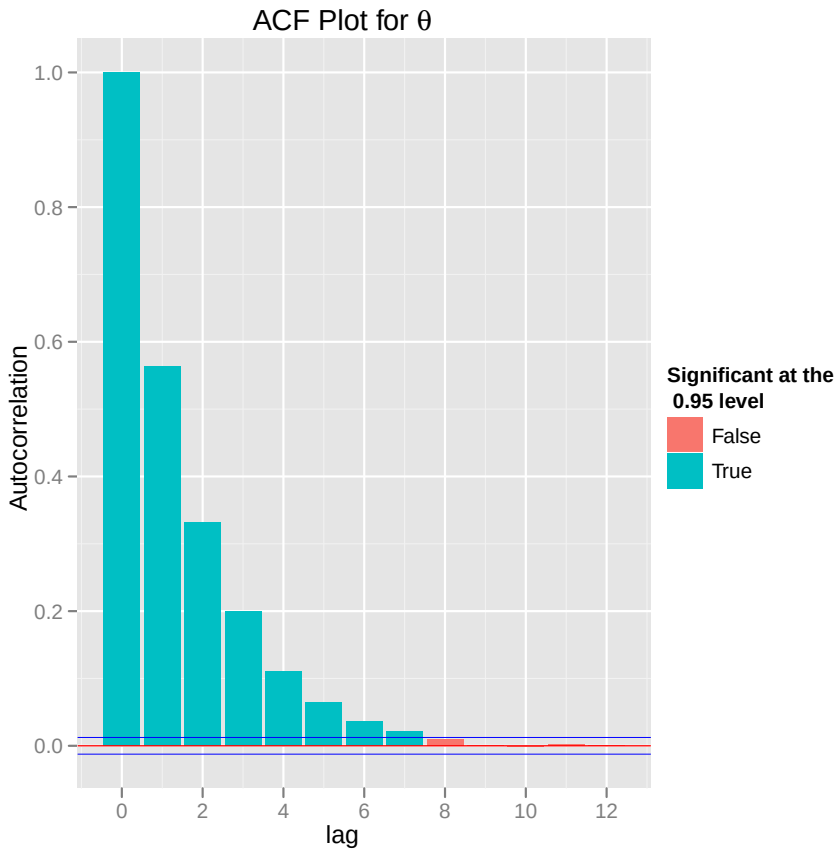
Mark the lags which are significantly different from 0. The ability to added a title to the plot has also been added.

```
> qacf <- function(x, conf.level = 0.95, max.lag = NULL, min.lag = 0,
+     title = "") {
+     ciline <- qnorm((1 - conf.level)/2)/sqrt(length(x))
+     bacf <- acf(x, plot = FALSE, lag.max = max.lag)
+     bacfdf <- with(bacf, data.frame(lag, acf))
+     if (min.lag > 0) {
+         bacfdf <- bacfdf[-seq(1, min.lag), ]
+     }
+     significant <- (abs(bacfdf[, 2]) > abs(ciline))^2
+     bacfdf <- cbind(bacfdf, significant)
+     q <- qplot(lag, acf, data = bacfdf, geom = "bar", stat = "identity",
+         position = "identity", ylab = "Autocorrelation", main = title,
+         fill = factor(significant))
+     q <- q + geom_hline(yintercept = -ciline, color = "blue",
+         size = 0.2)
+     q <- q + geom_hline(yintercept = ciline, color = "blue",
+         size = 0.2)
+     q <- q + geom_hline(yintercept = 0, color = "red", size = 0.3)
+     q <- q + scale_fill_hue(name = paste("Significant at the\n",
+         conf.level, "level"), breaks = 0:1, labels = c("False",
+         "True"))
+     return(q)
+ }
```

Example

```
> p <- qacf(with(params, theta), title = expression(paste("ACF Plot for ",
+     theta)))
> print(p)
```

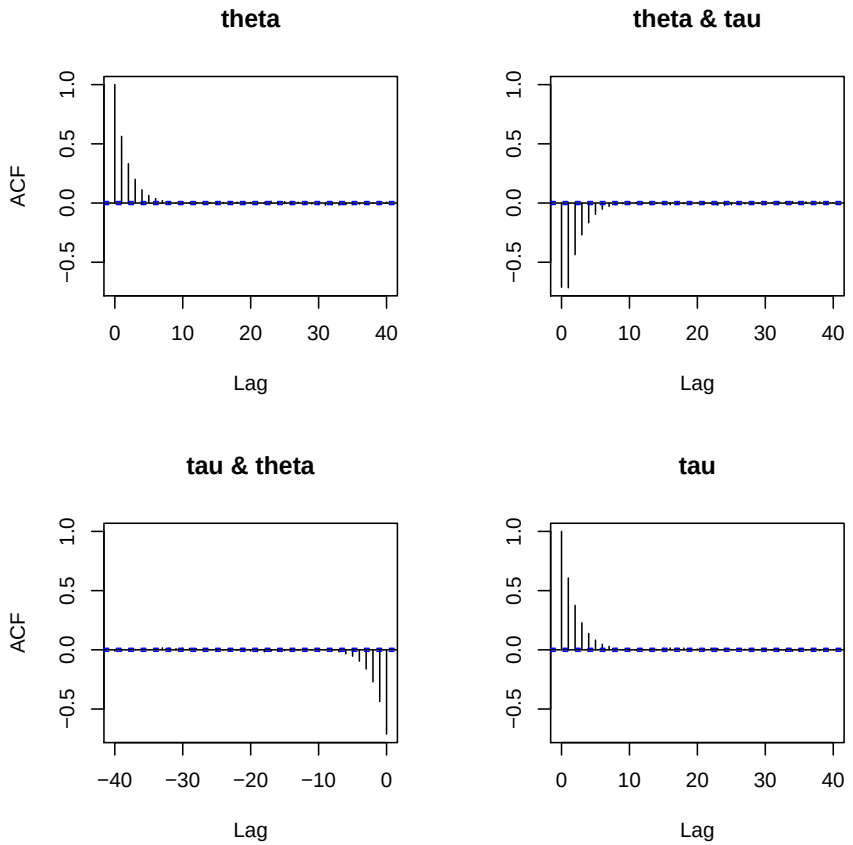




ACF plots for more than one variable

- The `qacf()` function so far will only work for a single vector.
- What if we are interested in a data frame with more than one series in it? The plot shown on the right is the base `acf()` plot for two series.
- How can we create a similar plot in `ggplot2`?
  - Use some data manipulation and a few tweaks to the `qacf()` function.

```
> acf(params)
```



What do we get from the base acf() function?

```
> summary(acf(params, plot = FALSE))
       Length Class  Mode
acf    164    -none- numeric
type     1    -none- character
n.used   1    -none- numeric
lag    164    -none- numeric
series   1    -none- character
snames   2    -none- character
> with(acf(params, plot = FALSE), acf)
, , 1

              [,1]          [,2]
 [1,] 1.0000000000 -0.7111475314
 [2,] 0.5630029497 -0.4355914620
 [3,] 0.3322593196 -0.2702013561
 [4,] 0.2000358269 -0.1619737468
 [5,] 0.1106916075 -0.0954319788
 [6,] 0.0639472781 -0.0532004509
 [7,] 0.0370311938 -0.0336147073
 [8,] 0.0209297569 -0.0174571412
 [9,] 0.0090523897 -0.0063593840
```

What do we get from the base `acf()` function?

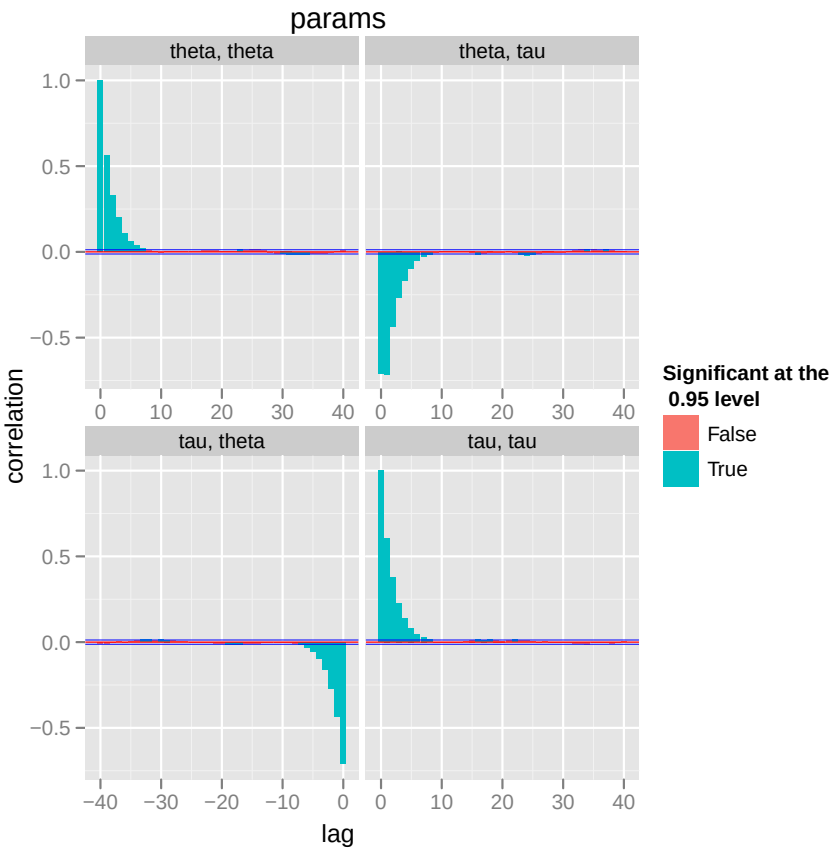
- First bit of good news, the names for the object returned by the `acf()` function are the same as when only one series was passed into the function.
  - `acf`: a 3D array with the numeric values for the `type` of plot. The [,,1] is the first column of plots in the produced graphic.
  - `type`: is the plot for correlation? covariance? or a partial autocorrelation plot?
  - `n.used`: length of the series
  - `lag`: a vector of the lags to be plotted with the correct value stored in `acf`.
  - `series`: name of the object passed into the `acf()` function.
  - `snames`: names of the columns in the data frame passed into the `acf()` function.
- By manipulating the data in `acf` and `lag` into a new data frame we should be able to generate a similar graphic using `ggplo2`.

The finished qacf() function

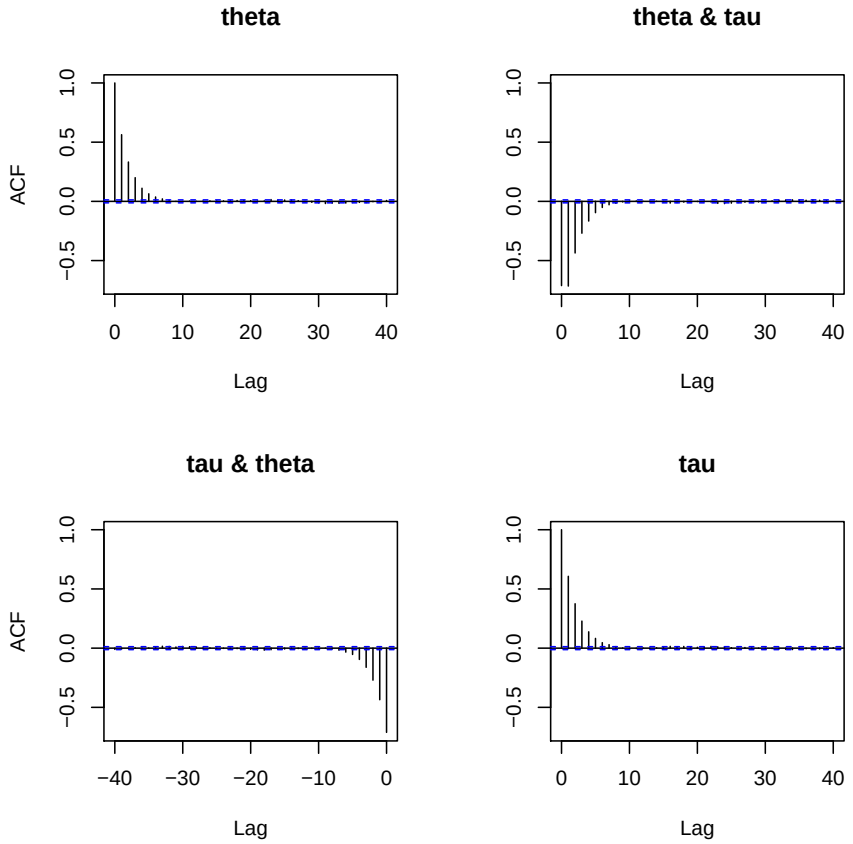
- The code for the qacf() function has change quite a bit to account for the data sets and some plotting options.
- The final code for the acf() function is too long for a Beamer slide. We'll look over the code in the .R file extracted by the Sweave function Stangle().

qacf() vs. acf()

```
> p <- qacf(params, show.sig = TRUE)
> print(p)
```

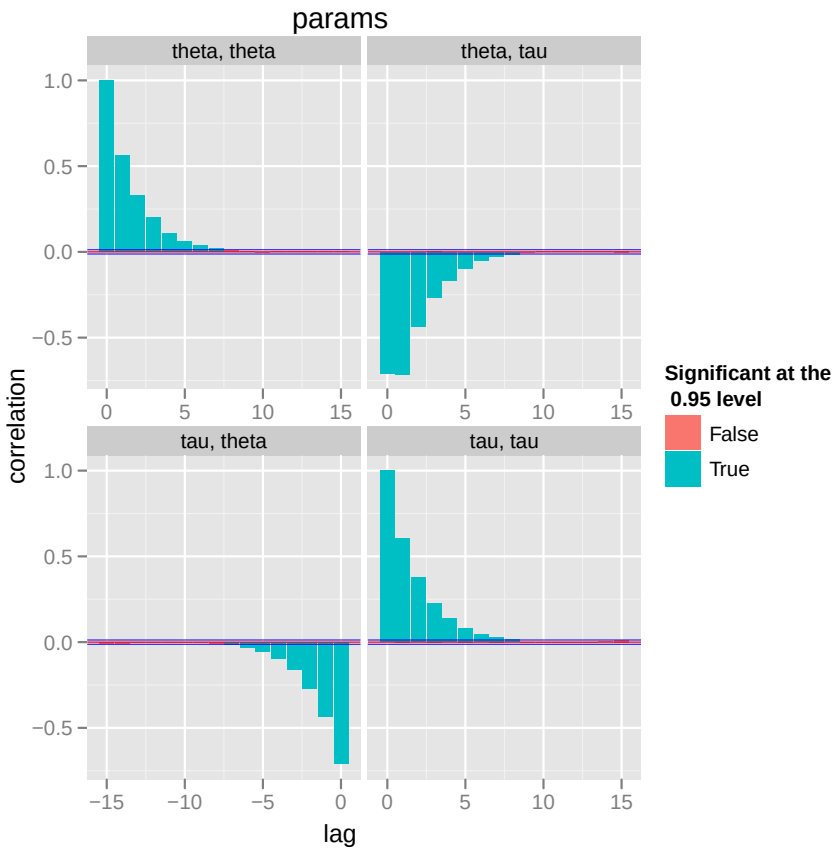



```
> acf(params)
```

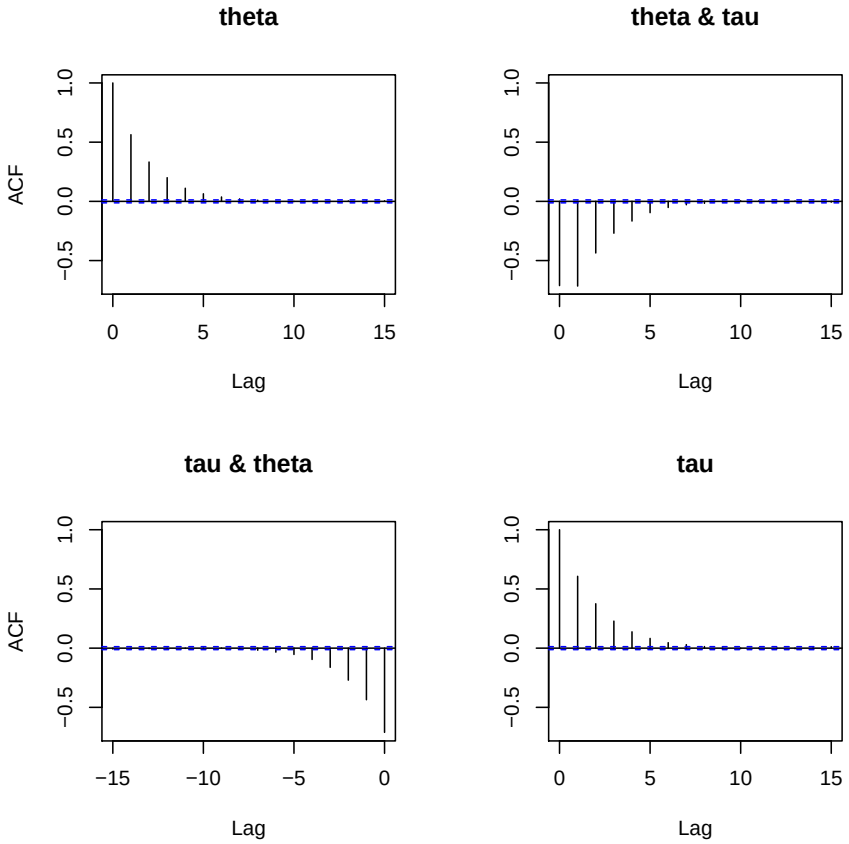


qacf() vs. acf()

```
> p <- qacf(params, lag.max = 15, show.sig = TRUE)
> print(p)
```




```
> acf(params, lag.max = 15)
```

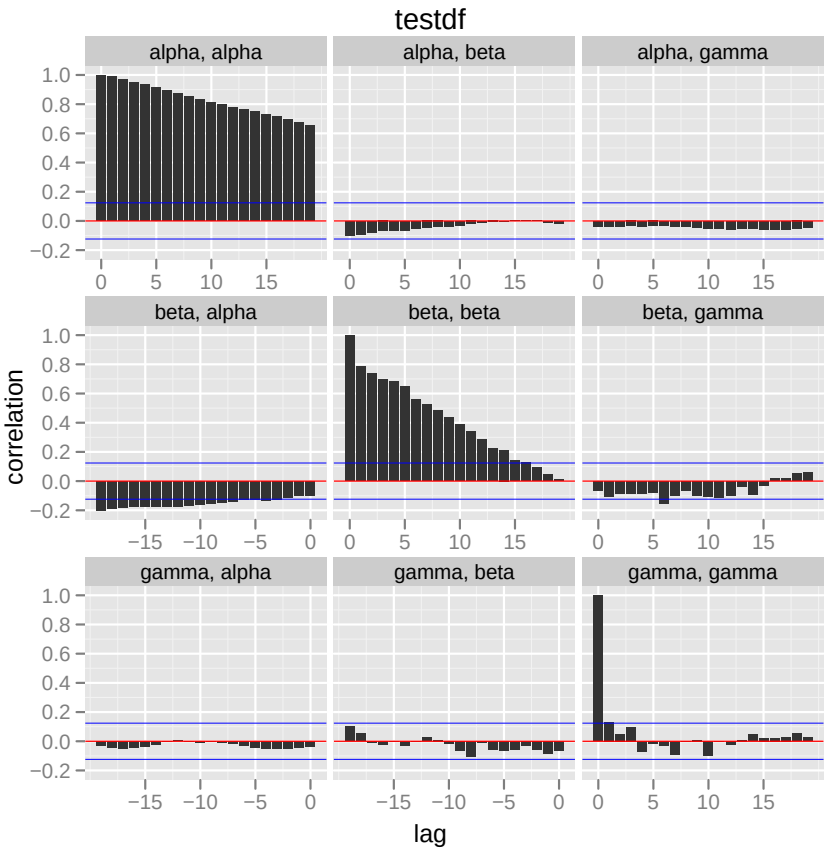


A random set of three series to use:

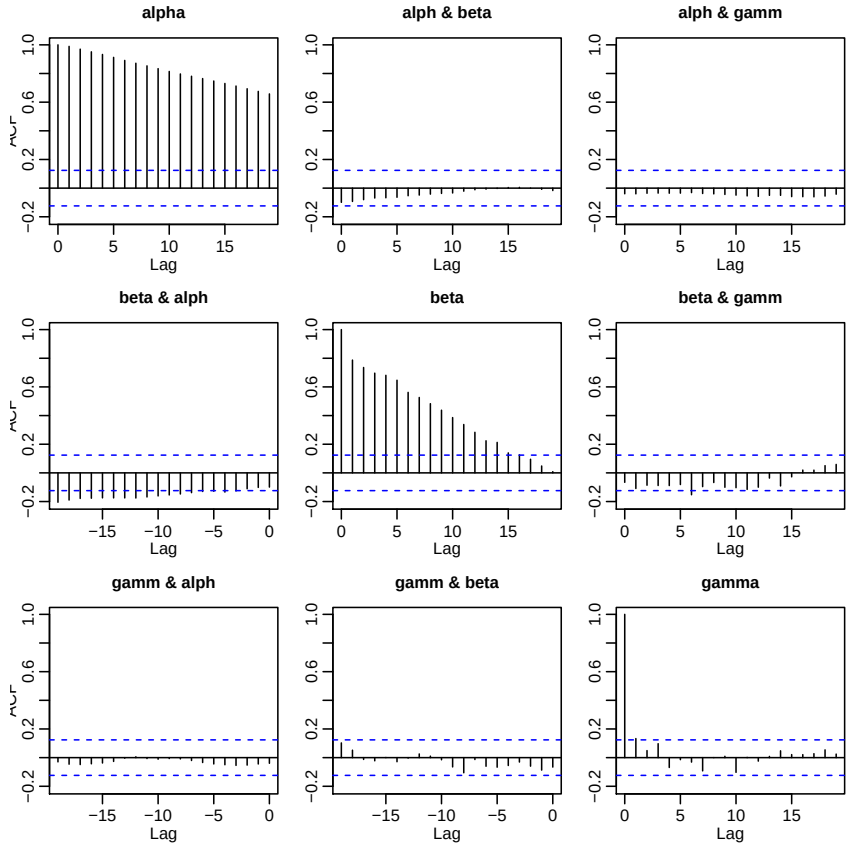
```
> set.seed(42)
> n <- 250
> alpha <- c(5)
> beta <- c(5)
> gamma <- c(5)
> Z.1 <- rnorm(n, 0, 1)
> Z.2 <- rnorm(n, 0, 2)
> Z.3 <- rnorm(n, 0, 5)
> for (i in 2:n) {
+     alpha[i] <- alpha[i - 1] + 2 * Z.1[i] + 3.14 * Z.1[i - 1]
+     beta[i] <- beta[i - 1] - 2 * Z.2[i] + Z.2[i - 1]
+     gamma[i] <- gamma[i - 1] * 0.2 * Z.3[i] + 4.31 * Z.3[i -
+         1]
+ }
> testdf <- data.frame(alpha, beta, gamma)
```

More Complex Examples

```
> p <- qacf(testdf)
> print(p)
```

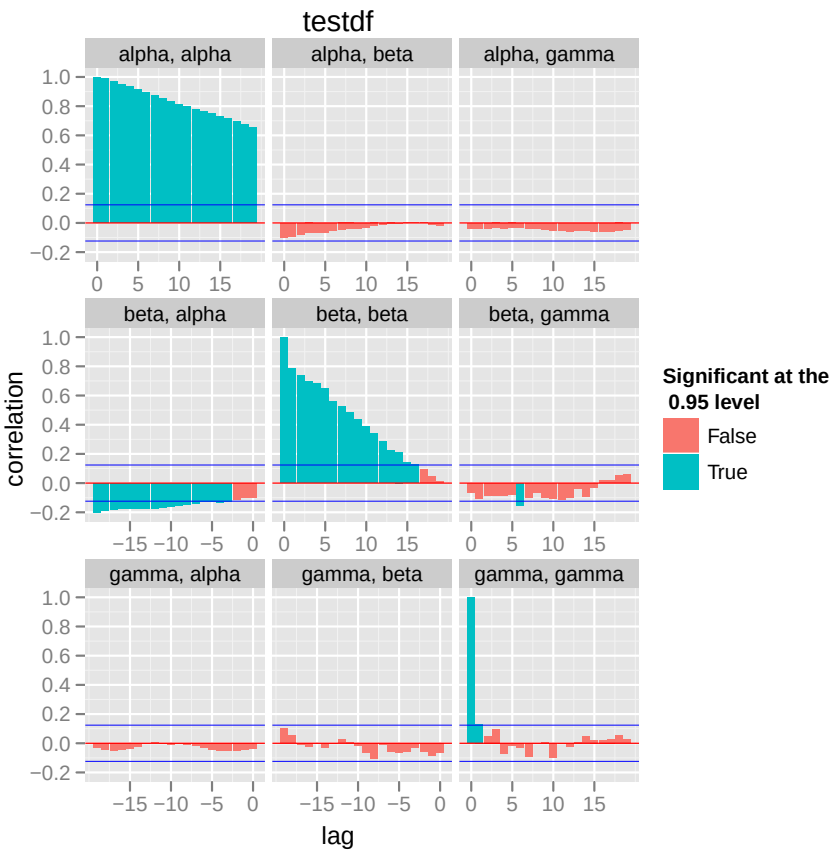



```
> acf(testdf)
```

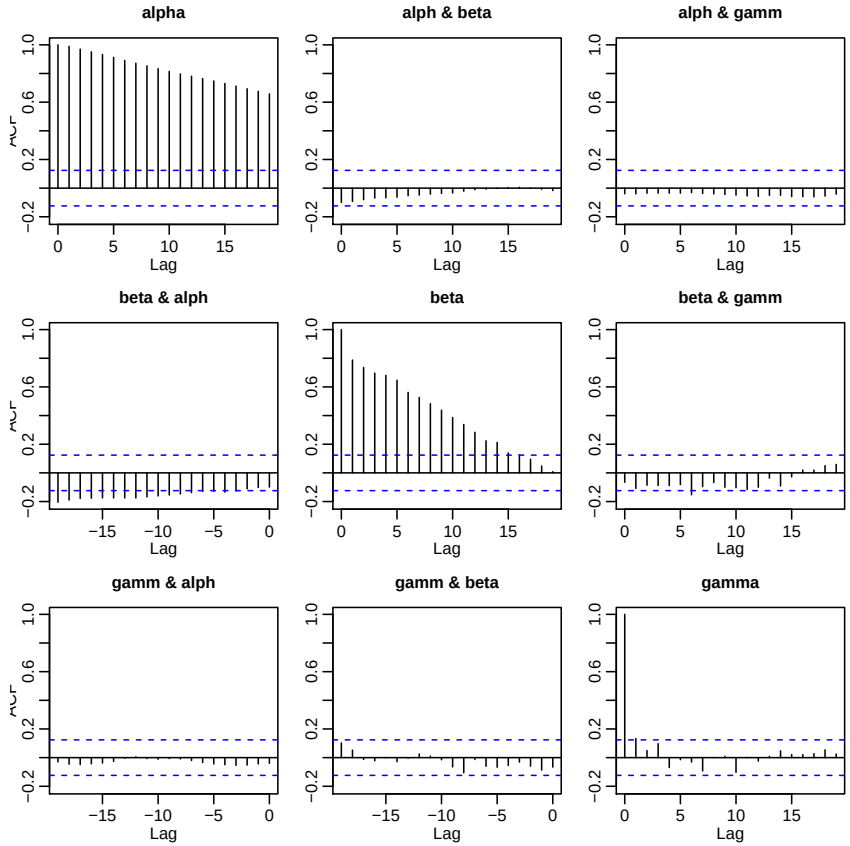


More Complex Examples

```
> p <- qacf(testdf, show.sig = TRUE)
> print(p)
```

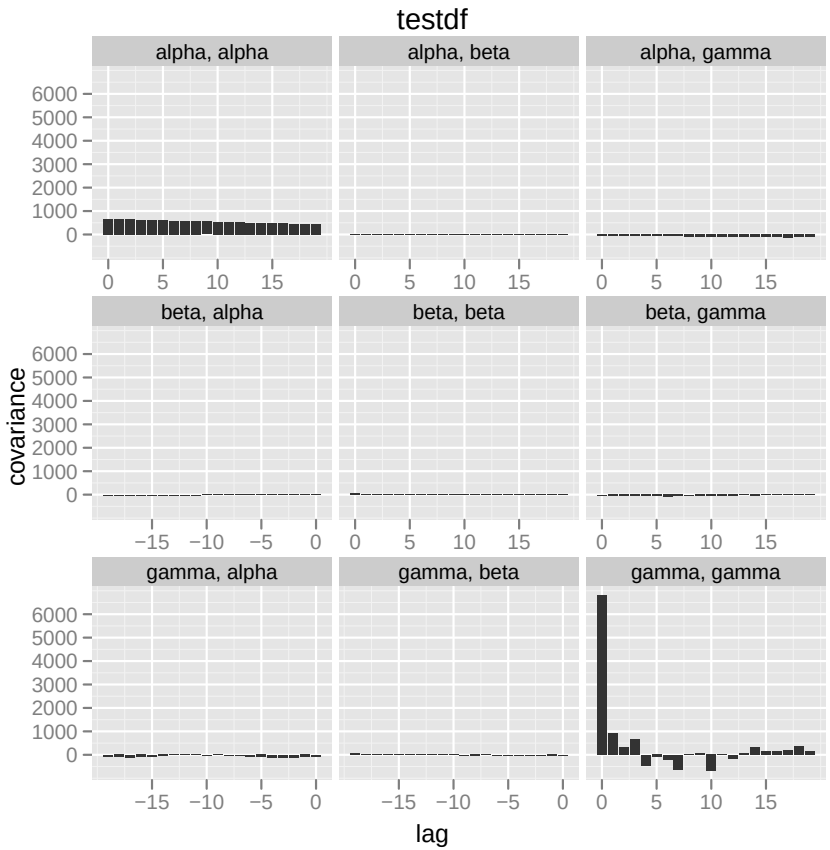



```
> acf(testdf)
```

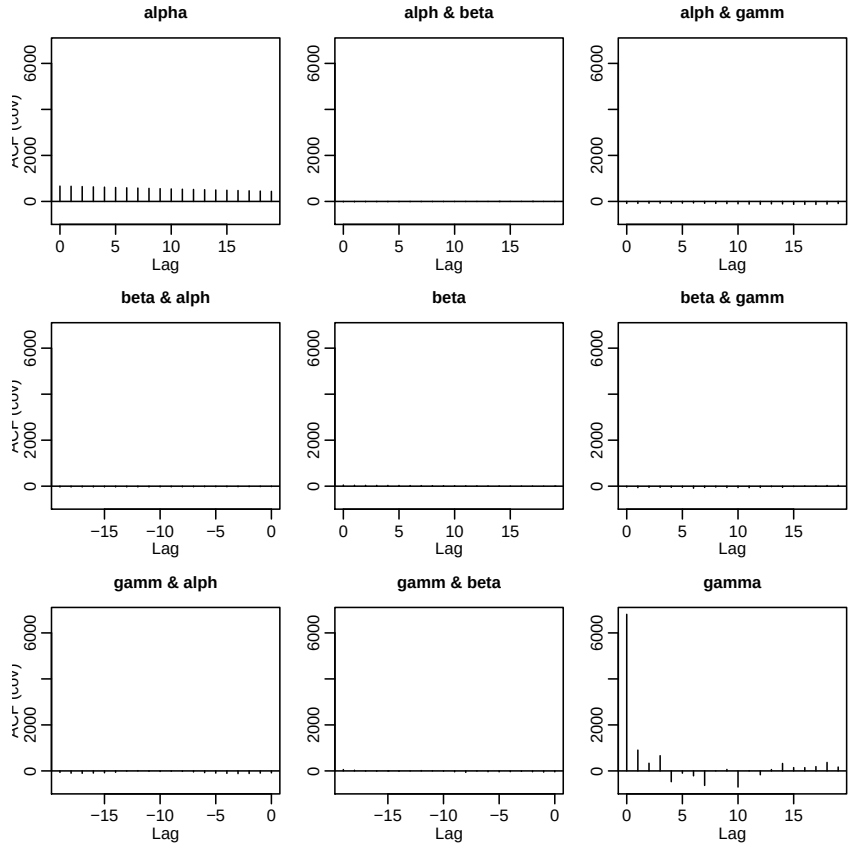


More Complex Examples

```
> p <- qacf(testdf, type = "covariance")
> print(p)
```

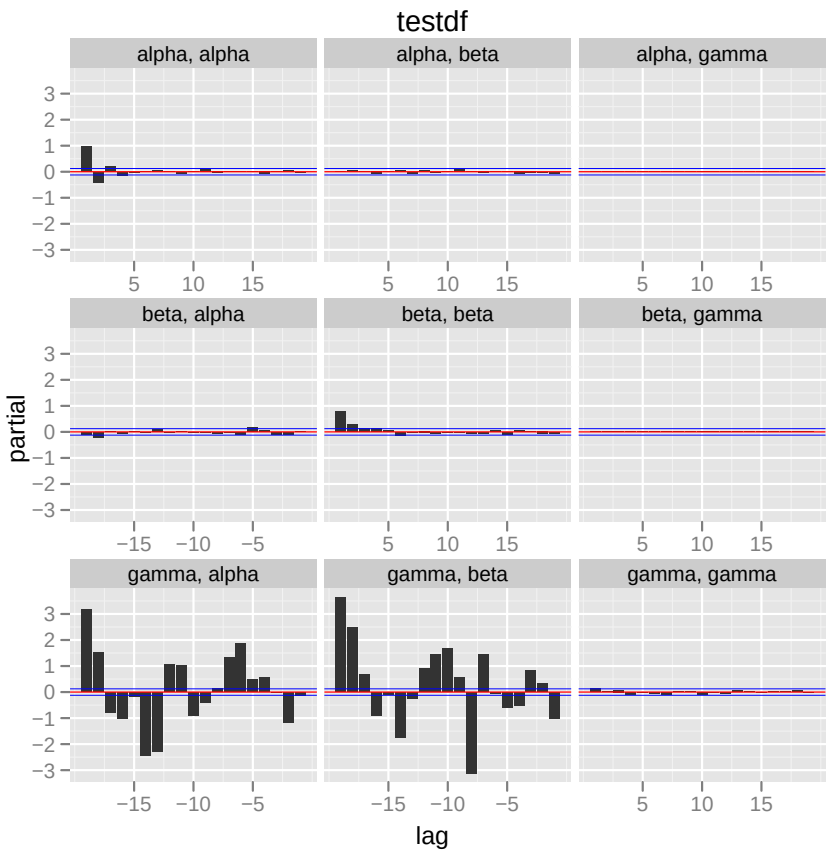



```
> acf(testdf, type = "covariance")
```

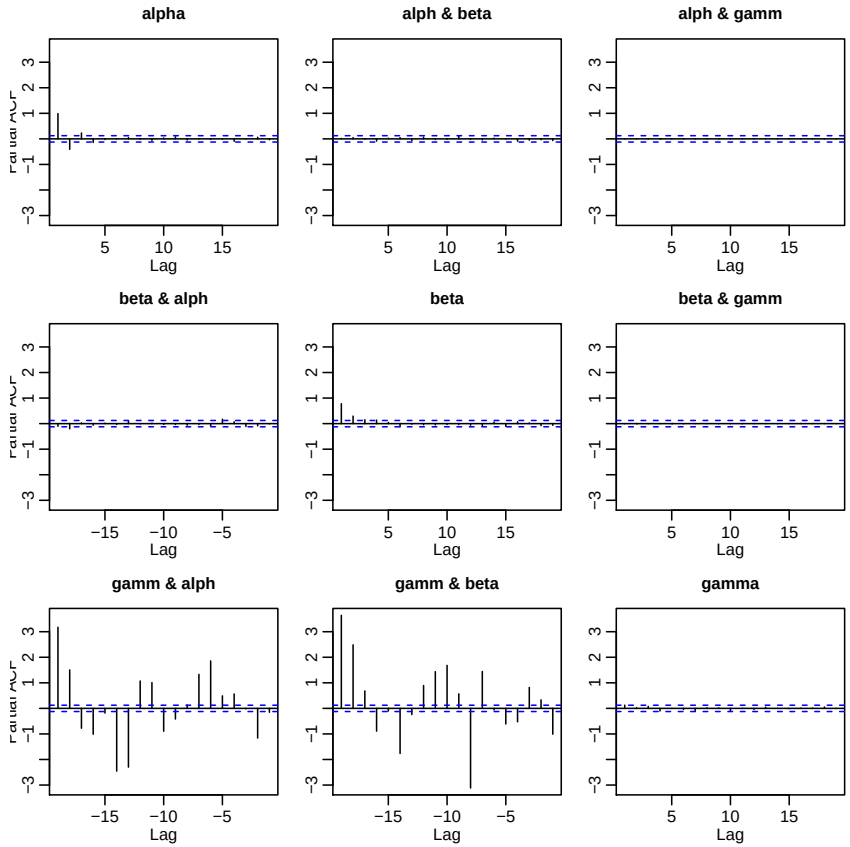


More Complex Examples

```
> p <- qacf(testdf, type = "partial")
> print(p)
```

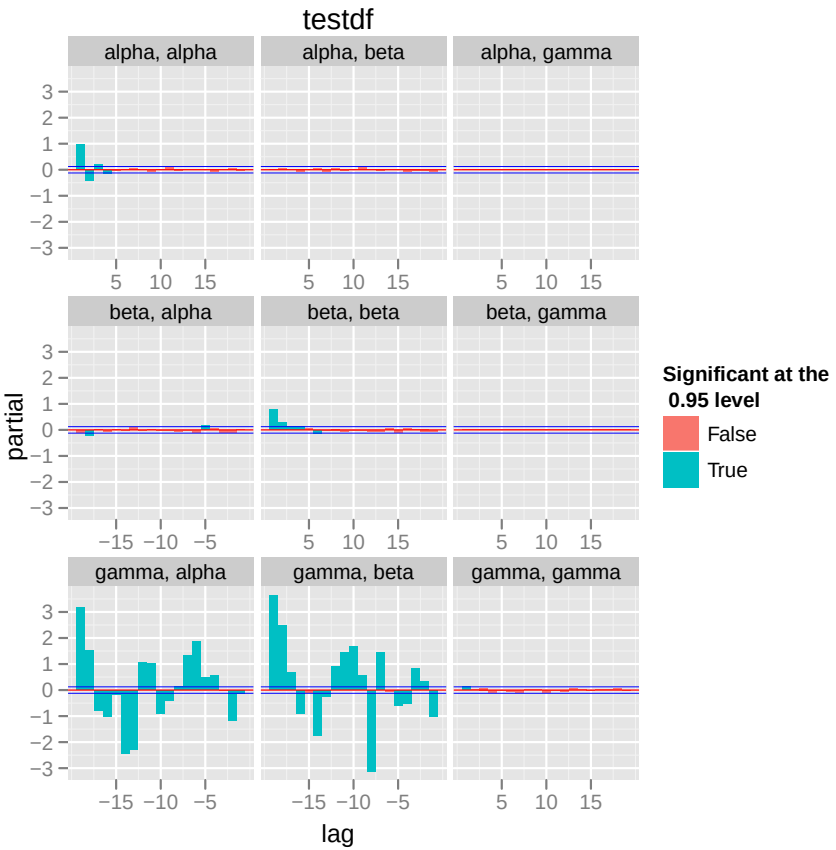



```
> acf(testdf, type = "partial")
```

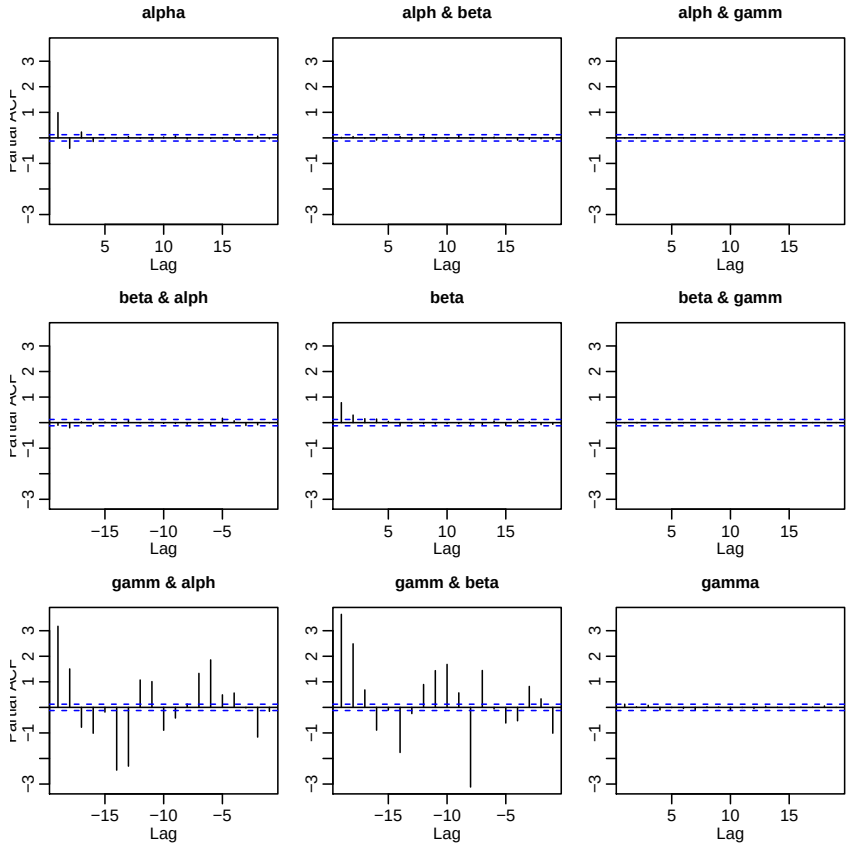


More Complex Examples

```
> p <- qacf(testdf, type = "partial", show.sig = TRUE)
> print(p)
```




```
> acf(testdf, type = "partial")
```

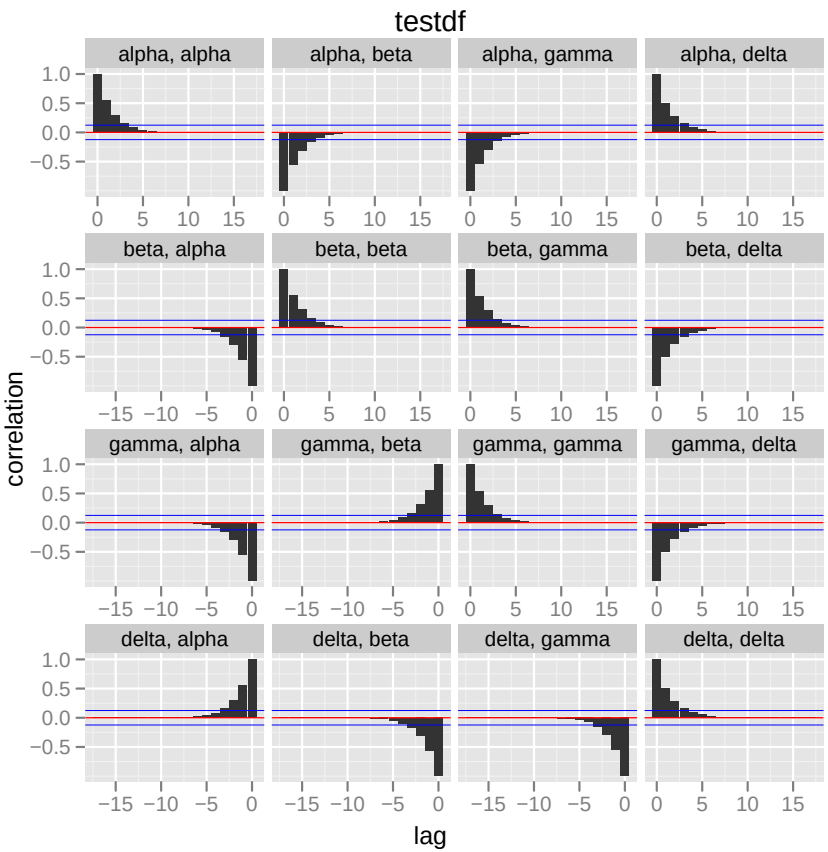


4 Series Example

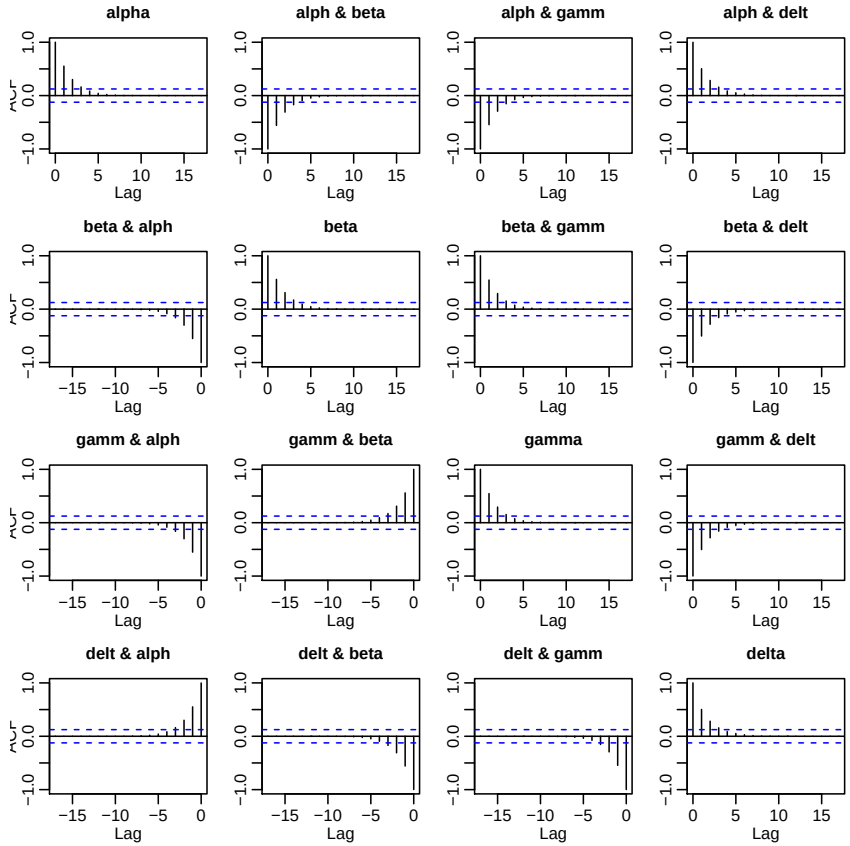
```
> n <- 250
> alpha <- c(5)
> beta <- c(5)
> gamma <- c(5)
> delta <- c(5)
> Z.1 <- rnorm(n, 0, 1)
> Z.2 <- rnorm(n, 0, 2)
> Z.3 <- rnorm(n, 0, 5)
> for (i in 2:n) {
+     alpha[i] <- alpha[i - 1] + Z.1[i] - Z.1[i - 1] + delta[i -
+         1] - beta[i - 1]
+     beta[i] <- beta[i - 1] - 2 * Z.2[i] + Z.2[i - 1] - delta[i -
+         1]
+     gamma[i] <- gamma[i - 1] + beta[i - 1] + 0.2 * Z.3[i] + Z.3[i -
+         1]
+     delta[i] <- delta[i - 1] + runif(1, 0.5, 1.5) * delta[i -
+         1]
+ }
> testdf <- data.frame(alpha, beta, gamma, delta)
```

More Complex Examples

```
> p <- qacf(testdf)
> print(p)
```

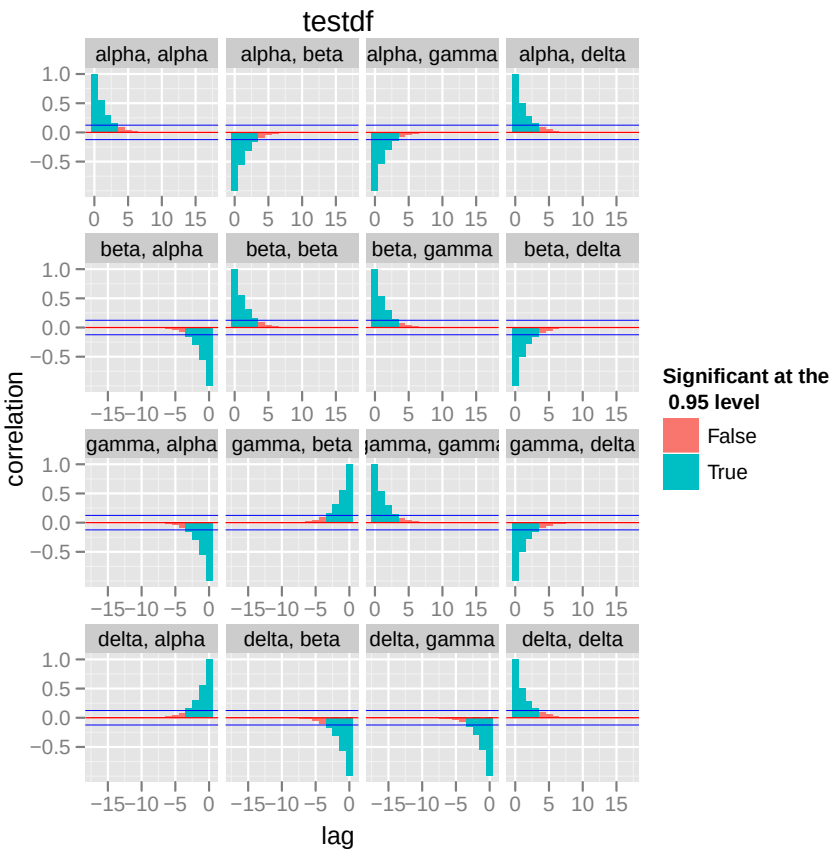



```
> acf(testdf)
```

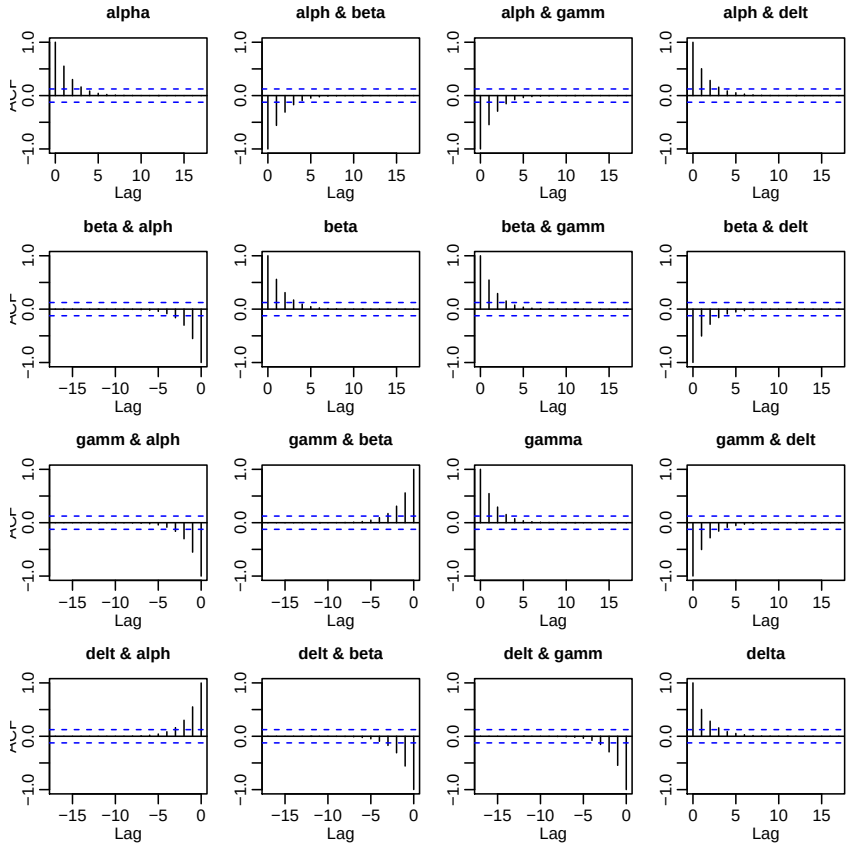


More Complex Examples

```
> p <- qacf(testdf, show.sig = TRUE)
> print(p)
```

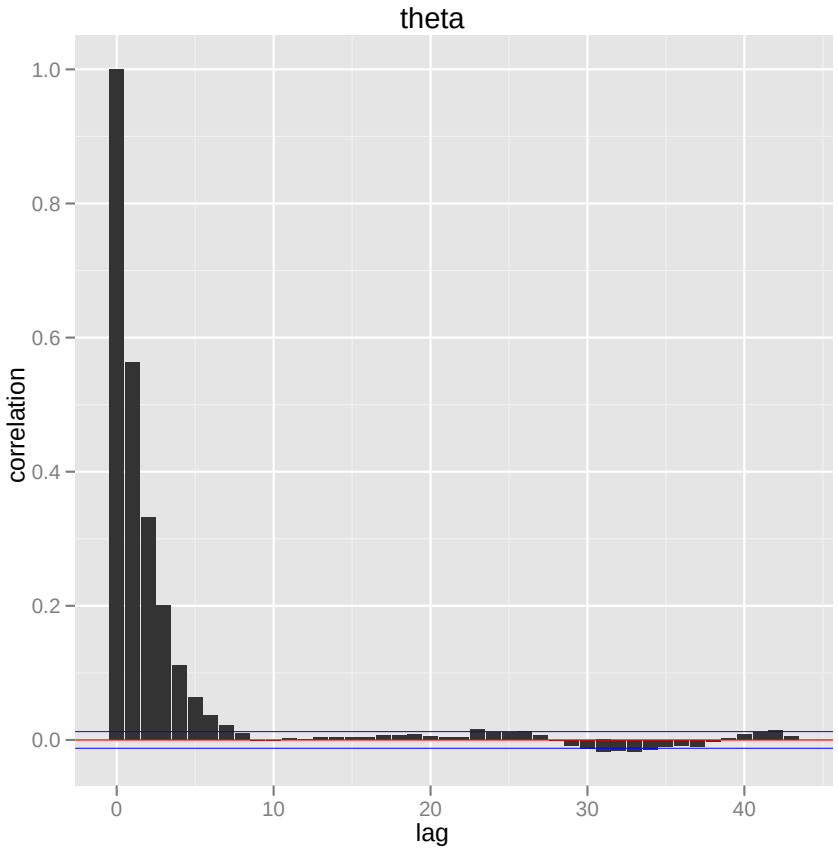



```
> acf(testdf)
```

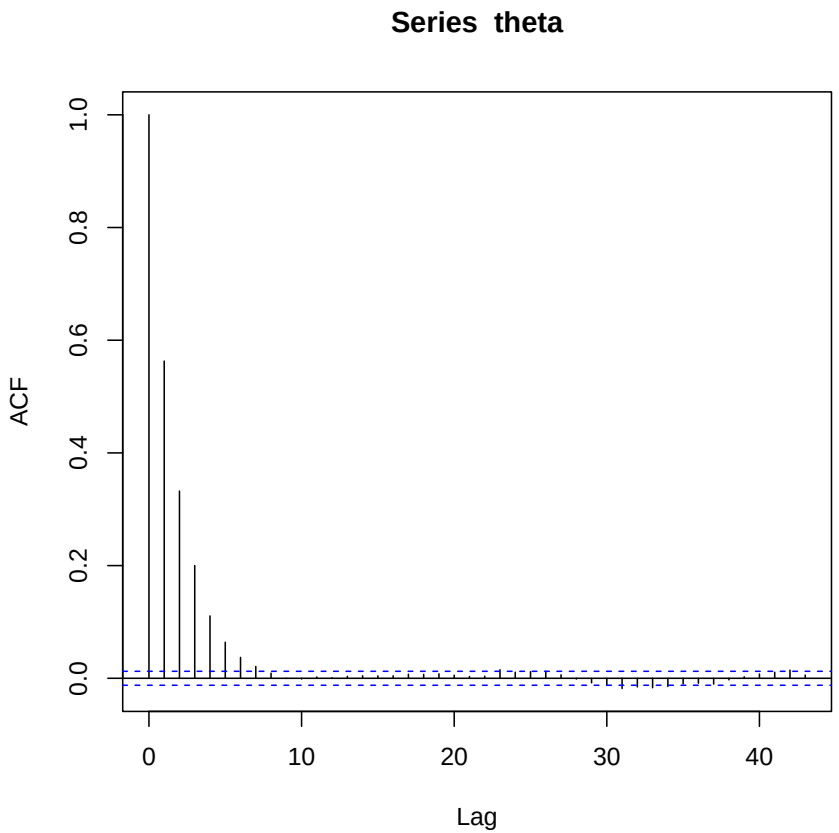


One Variable Plots
Autocorrelation

```
> p <- with(params, qacf(theta))
> print(p)
```



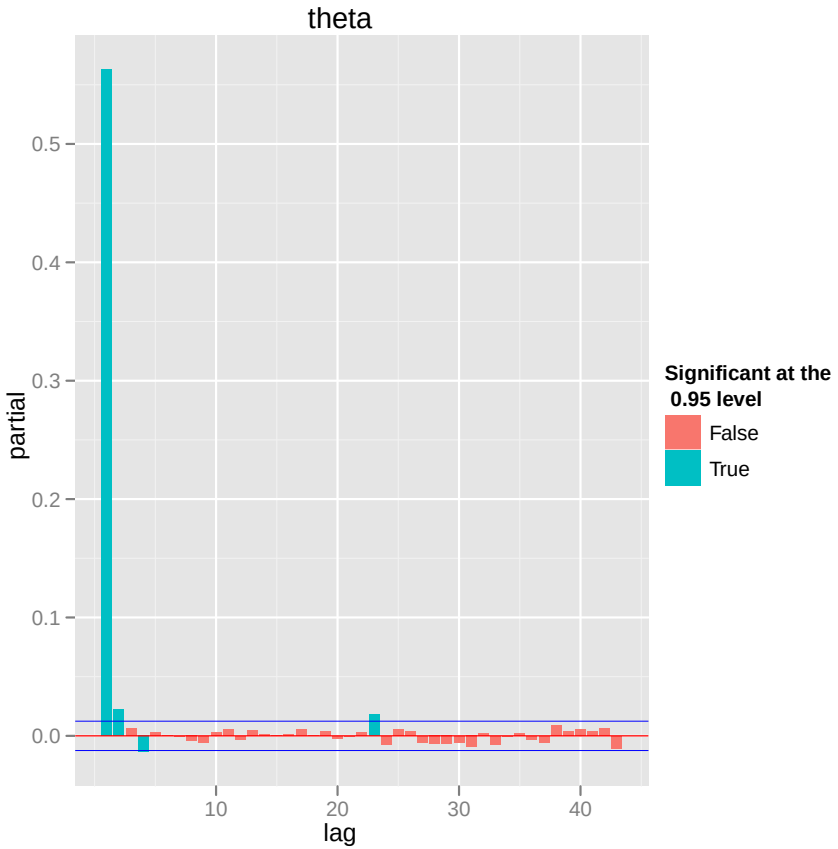

```
> with(params, acf(theta))
```

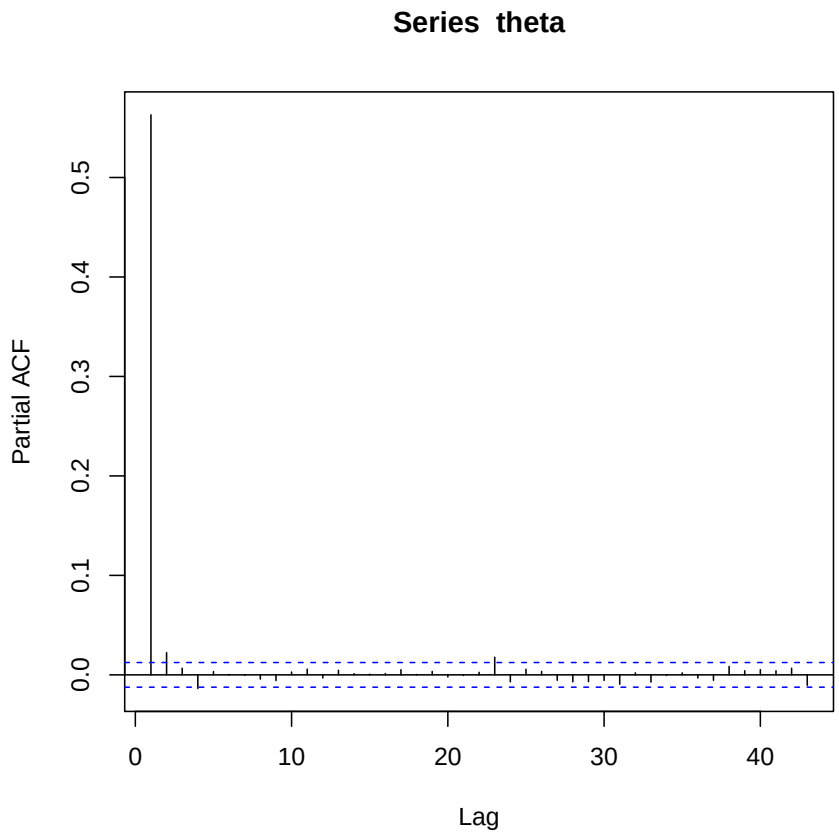


One Variable Plots
Partial Autocorrelation

```
> p <- with(params, qacf(theta, type = "partial", show.sig = TRUE))
> print(p)
```

```
> with(params, acf(theta, type = "partial
```

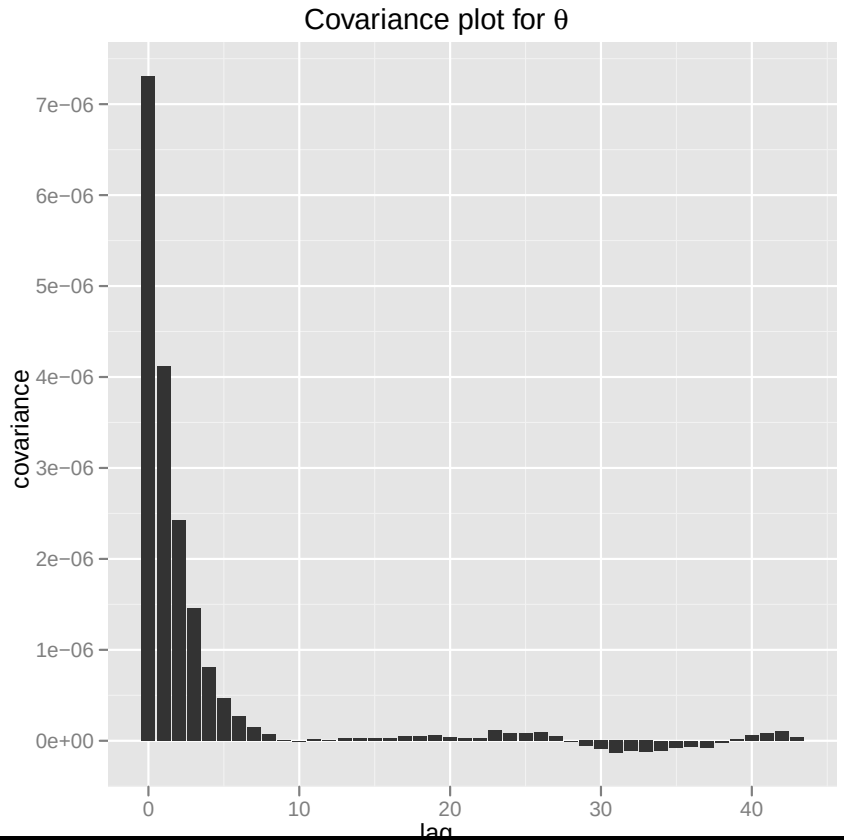


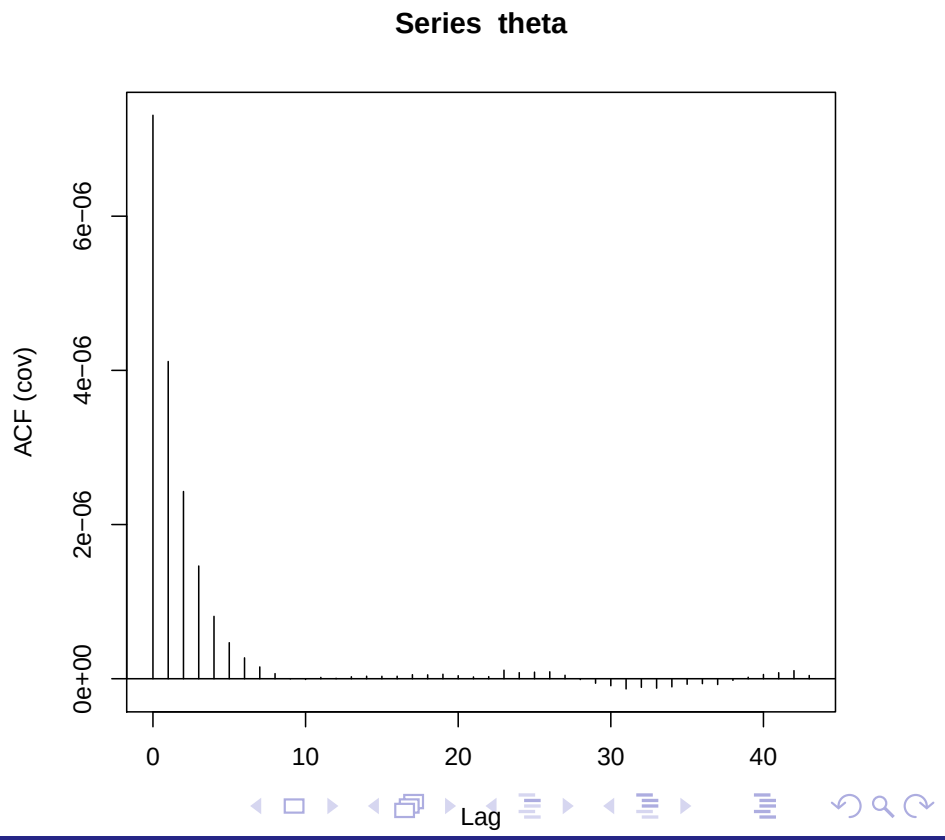


One Variable Plots
Covariance

```
> p <- with(params, qacf(theta, type = "covariance"))
> p <- p + opts(title = expression(paste("Covariance plot for ",
+     theta)))
> print(p)
```

```
> with(params, acf(theta, type = "covaria
```





- Add error handling
- Add control on the facet labels - this is functionality needed in `ggplot2` before it can be added to my `qacf()` function.
- Suppress the "Using as id variables" message generated from the `melt()` function.

Some Resources for ggplot2 and Sweave

- For ggplot2
  - ggplot2 Google Group: groups.google.com/group/ggplot2
  - Hadley Wickham's website: http://had.co.nz/ggplot2
  - "ggplot2 Elegant Graphics for Data Analysis", by Hadley Wickham, Springer, 2009
- For Sweave
  - User manual is actually very helpful for learning the options.
  - Best to see examples from other users and trial and error.
  - Remember that when using Sweave the code chunks will not work in many of the different environments in LaTeXunless you have made the environment fragile.
  - Great example on R-Bloggers: http://pineda-krch.com/2011/01/17/the-joy-of-sweave