Breakpoints for farm field runoff

Wesley Brooks

First read the data:

```
> setwd('c:/Users/wrbrooks/git/disco_farms')
> data_file = 'runoff_by_site.csv'
> data = read.csv(data_file, header=T, na.strings=c('NA', 'na', '', "-9"))
> #Remove NAs:
> which(is.na(data)) %% dim(data)[1]
```

```
[1] 831
```

```
> data = data[-831,]
> #Convert to metric
> data = within(data, {
+     prec <- 2.54*prec
+     I30 <- 2.54*I30
+     })
> farms = levels(data$loc)
> sites = levels(data$site)
```

The following is the R function that produces a piecewise regression model:

```
> piecewise <- function(th, x, y)
+ {
+     X <- cbind( ifelse(x>=th, 1, 0), x, pmax(0, x-th) )
+     #X <- cbind( x, pmax(0, x-th) )
+     fit = lsfit(X, y) #actually fit the model
+
+     #Counts how many observations lie above and below the breakpoint
+     upper = as.logical( ifelse(x-th>=0, 1, 0) )
+     n_upper = sum(upper)
+     n_lower = dim(X)[1] - n_upper
+
+     #Sum of squared residuals on either side of the breakpoint
+     ssr_upper = sum( fit$resid[upper]**2 )
+     ssr_lower = sum( fit$resid[!upper]**2 )
+
+     #Return the residual standard error:
+     return( n_lower*sqrt(ssr_lower/(n_lower-2)) + n_upper*sqrt(ssr_upper/(n_upper-2)
+ }
> piecewise_conts <- function(th, x, y)
+ {
+     X <- cbind( x, pmax(0, x-th) )
+     fit = lsfit(X, y) #actually fit the model
+
+     #Counts how many observations lie above and below the breakpoint
+     upper = as.logical( ifelse(x-th>=0, 1, 0) )
+     n_upper = sum(upper)
+     n_lower = dim(X)[1] - n_upper
+
+     #Sum of squared residuals on either side of the breakpoint
+     ssr_upper = sum( fit$resid[upper]**2 )
+     ssr_lower = sum( fit$resid[!upper]**2 )
+
+     #Return the residual standard error:
+     return( n_lower*sqrt(ssr_lower/(n_lower-2)) + n_upper*sqrt(ssr_upper/(n_upper-2)
+ }
> piecewise_disjoint <- function(th, x, y)
+ {
+     X <- cbind( ifelse(x>=th, 1, 0), x, pmax(0, x-th) )
+     fit = lsfit(X, y) #actually fit the model
+
+     #Counts how many observations lie above and below the breakpoint
+     upper = as.logical( ifelse(x-th>=0, 1, 0) )
+     n_upper = sum(upper)
+     n_lower = dim(X)[1] - n_upper
+
+     #Sum of squared residuals on either side of the breakpoint
+     ssr_upper = sum( fit$resid[upper]**2 )
+     ssr_lower = sum( fit$resid[!upper]**2 )
+
+     #Return the residual standard error:
+     return( n_lower*sqrt(ssr_lower/(n_lower-2)) + n_upper*sqrt(ssr_upper/(n_upper-2)
+ }
```

And the next function (`peicewise_plot`) plots the piecewise regression model:

```
> piecewise_plot <- function(x,y,th, xlim=FALSE, ylim=FALSE,
+                               xlab='', ylab='', title='')
+ {
+     #define the limits of the plot
+     if( identical(xlim, FALSE) ) { xx = range(x, na.rm=TRUE) }
+     else { xx = xlim }
+
+     if( identical(ylim, FALSE) ) { yy = range(y, na.rm=TRUE) }
+     else { yy = ylim }
+
+     #put the data points on the plot
+     plot(x, y, xlab=xlab, ylab=ylab, xlim=xx, ylim=yy, pch=20,
+            bty='n', xaxt='s', yaxt='s', ann=T, main=title)
+
+     #create the piecewise regression model
+     X <- cbind(ifelse(x>=th, 1, 0), x, pmax(0, x-th))
+     #X <- cbind( x, pmax(0, x-th))
+     fit = lsfit(X, y) #actually fit the model
+
+     #extent of the lower, upper pieces:
+     xints = c( xx[1], th, xx[2] )
+
+     #draw the lower regression line
+     xints_low = xints[1:2]
+     yints_low = fit$coef[1] + (fit$coef[3]*xints_low)
+     #yints_low = fit$coef[1] + (fit$coef[2]*xints_low)
+     par(new=T, ann=F, xaxt='n', yaxt='n', bty='n')
+     plot(xints_low, yints_low, type='l', xlim=xx, ylim=yy)
+
+     #draw the upper regression line
+     xints_high = xints[2:3]
+     yints_high = fit$coef[1] + fit$coef[2] + (fit$coef[3]*xints_high) +
+                     (fit$coef[4]*(xints_high - th))
+     #yints_high = fit$coef[1] + (fit$coef[2]*xints_high) + (fit$coef[3]*(xints_high
+     par(new=T, ann=F, xaxt='n', yaxt='n')
+     plot(xints_high, yints_high, type='l', xlim=xx, ylim=yy)
+
+     #draw a vertical line to separate the two regimes
+     abline(v=th, lty=3)
+     text(x=th, y=0.6, pos=4, labels=paste("breakpoint: ", round(th,2), sep=""))
+ }
```

We identify the best breakpoint by using R's optimize function to minimize the residual standard error:

```
> #soil_moisture_limits = c(33, 43) #set the limits
> #optimize(piecewise, soil_moisture_limits, x=data$sm, y=data$rc)$minimum
```

Now, let us do the soil moisture breakpoint analysis. The breakpoints at each farm individually, and for all farms in aggregate are found by the following code (breakpoints are stored in the variable `sm_breakpoints` for use in plotting [next code chunk]):

```
> xs = seq(33, 43)
> sm_breakpoints = vector()
> breakpoints = list()
> cat("Breakpoints by farm:\n")

Breakpoints by farm:

> fit = list()
> for(site in sites)
+ {
+     farm = data[data$site==site,]
+     th <- which.min( sapply(X=xs, FUN=piecewise, x=farm$sm, y=farm$rc) ) + 32
+     #cat(paste(optimize(piecewise, soil_moisture_limits, x=farm$sm, y=farm$rc)$minim
+     cat( paste(site, ": ", th, "\n", sep=""))
+     sm_breakpoints = c(sm_breakpoints, th)
+     breakpoints[[site]] = th
+
+     X <- cbind( ifelse(farm$sm>=th, 1, 0), farm$sm, pmax(0, farm$sm-th) )
+     #X <- cbind( x, pmax(0, x-th) )
+     fit[[site]] = lsfit(x=X, y=farm$rc)
+ }

df1: 35
df3: 35
df5: 40
df7: 40
df8: 40
koepke: 35
pagel: 35
saxon: 40

>
> #Now find the breakpoint for aggregated data:
> #aggregate = data[data$site %in% farms,]
> #th <- which.min( sapply(X=xs, FUN=piecewise, x=farm$sm, y=farm$rc) ) + 32
> #cat( paste("aggregate: ", ": ", th, "\n", sep=""))
> #sm_breakpoints = c(sm_breakpoints, th)
> #breakpoints[["aggregate"]] = th
```

Then the plots are produced by calling the function `piecewise_plot` (code is listed above):

```
> layout(matrix(1:8,4,2))
> titles = c(sites)
> for( i in 1:length(sites) )
+ {
+     farm = data[data$site==sites[i],]
+     piecewise_plot(x=farm$sm, y=farm$rc, th=sm_breakpoints[i],
+                       ylim=range(data$rc, na.rm=T), xlim=range(data$sm, na.rm=T),
+                       xlab="soil moisture", ylab="runoff coefficient",
+                       title=titles[i])
+ }
>
> #aggregate = data[data$site %in% farms,]
> #piecewise_plot(x=aggregate$sm, y=aggregate$rc, th=breakpoints[["aggregate"]][i+1],
> #                   ylim=range(data$rc, na.rm=T), xlim=range(data$sm, na.rm=T),
> #                   xlab="soil moisture", ylab="runoff coefficient",
> #                   title="Aggregate")
```

Now get the I30 breakpoints:

```
df1: 2.3
df3: 0.7
df5: 2.2
df7: 2.1
df8: 3.4
koepke: 1.5
pagel: 2.3
saxon: 2.1
```

When we put the events in bins based on their antecedent soil moisture (SM: high, medium, and low), the following are the I30 breakpoints (units are centimeters of rain per hour):

```
Intensity breakpoints at koepke when binned by soil moisture:
-Inf <= SM < 30: 3.3
30 <= SM < 35: 1.8
35 <= SM < Inf: 1.6

Intensity breakpoints at pagel when binned by soil moisture:
-Inf <= SM < 30: 1.3
30 <= SM < 35: 0.6
35 <= SM < Inf: 0.6
```

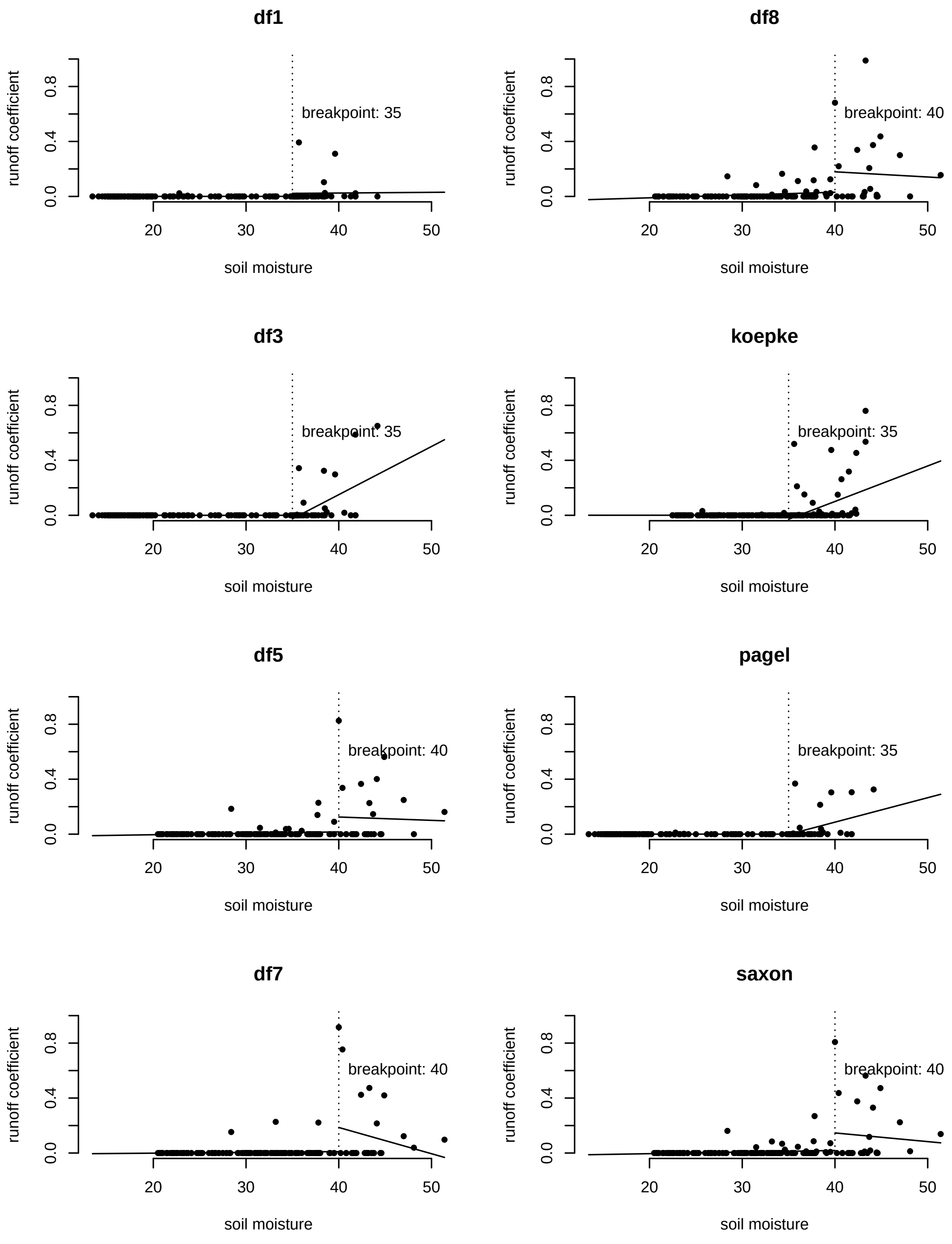

Figure 1: caption.

```
Intensity breakpoints at saxon when binned by soil moisture:
-Inf <= SM < 35: 1.8
35 <= SM < 40: 1.2
40 <= SM < Inf: 2
```

When we put the events in bins based on their antecedent soil moisture (SM: high, medium, and low), the following are the precipitation breakpoints (units are centimeters of rain):

```
Precipitation breakpoints at koepke when binned by soil moisture:
-Inf <= SM < 30: 2.8
30 <= SM < 35: 1.72
35 <= SM < Inf: 1.58

Precipitation breakpoints at pagel when binned by soil moisture:
-Inf <= SM < 30: 2.63
30 <= SM < 35: 1.17
35 <= SM < Inf: 2.03

Precipitation breakpoints at saxon when binned by soil moisture:
-Inf <= SM < 35: 2.51
35 <= SM < 40: 1.18
40 <= SM < Inf: 2.26
```

Rather than three bins, let's try two (above vs. below the SM breakpoint):

```
Intensity breakpoints at koepke when binned by soil moisture:
-Inf <= SM < 35: 1.9
35 <= SM < Inf: 1.6

Intensity breakpoints at pagel when binned by soil moisture:
-Inf <= SM < 35: 1.3
35 <= SM < Inf: 0.6

Intensity breakpoints at saxon when binned by soil moisture:
-Inf <= SM < 40: 0.9
40 <= SM < Inf: 2
```

When we put the events in bins based on their antecedent soil moisture (SM: high or low), the following are the precipitation breakpoints (units are centimeters of rain):

```
Precipitation breakpoints at koepke when binned by soil moisture:
```

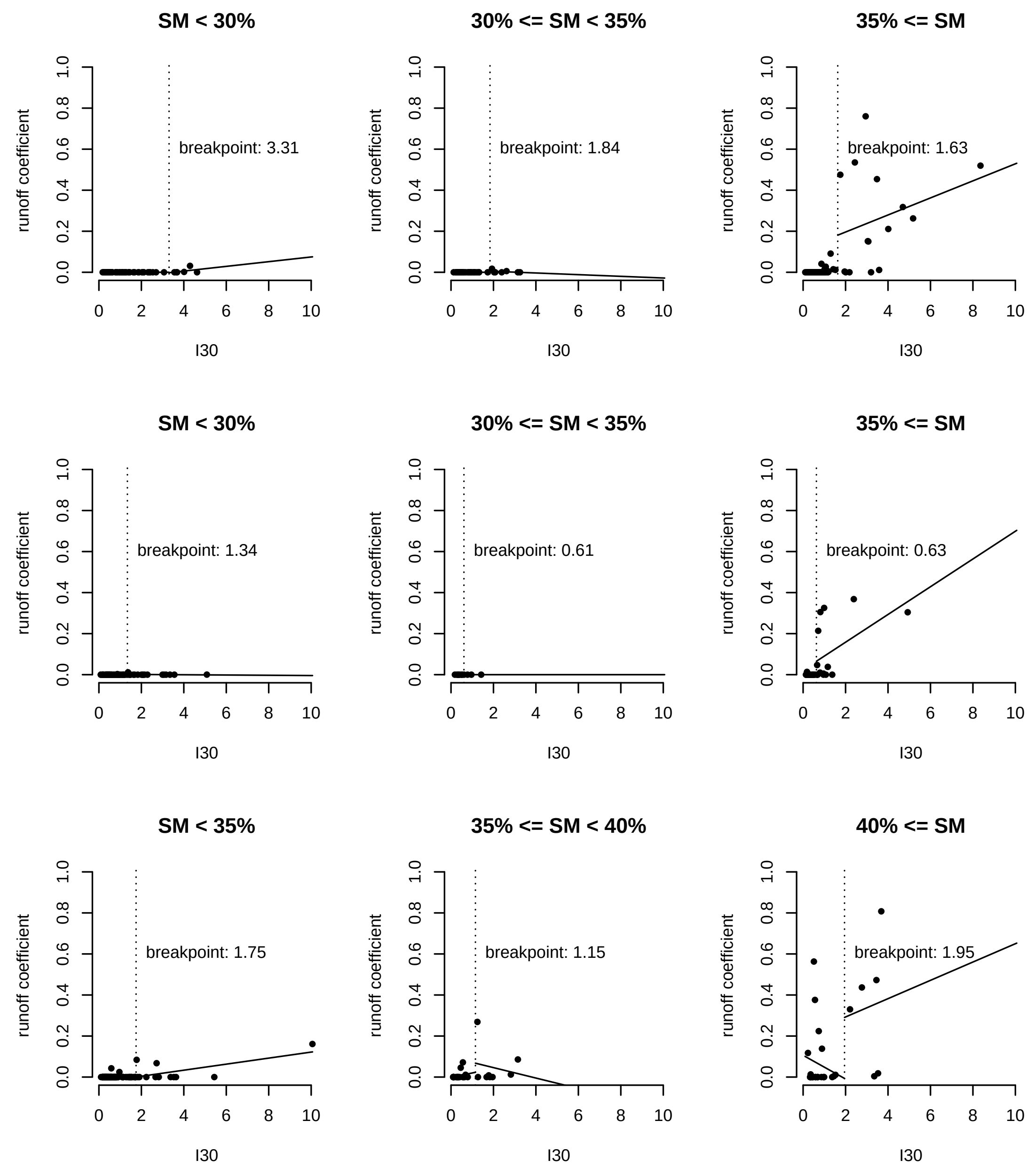

Figure 2: Top row: Koepke, middle row: Pagel, bottom row: Saxon.

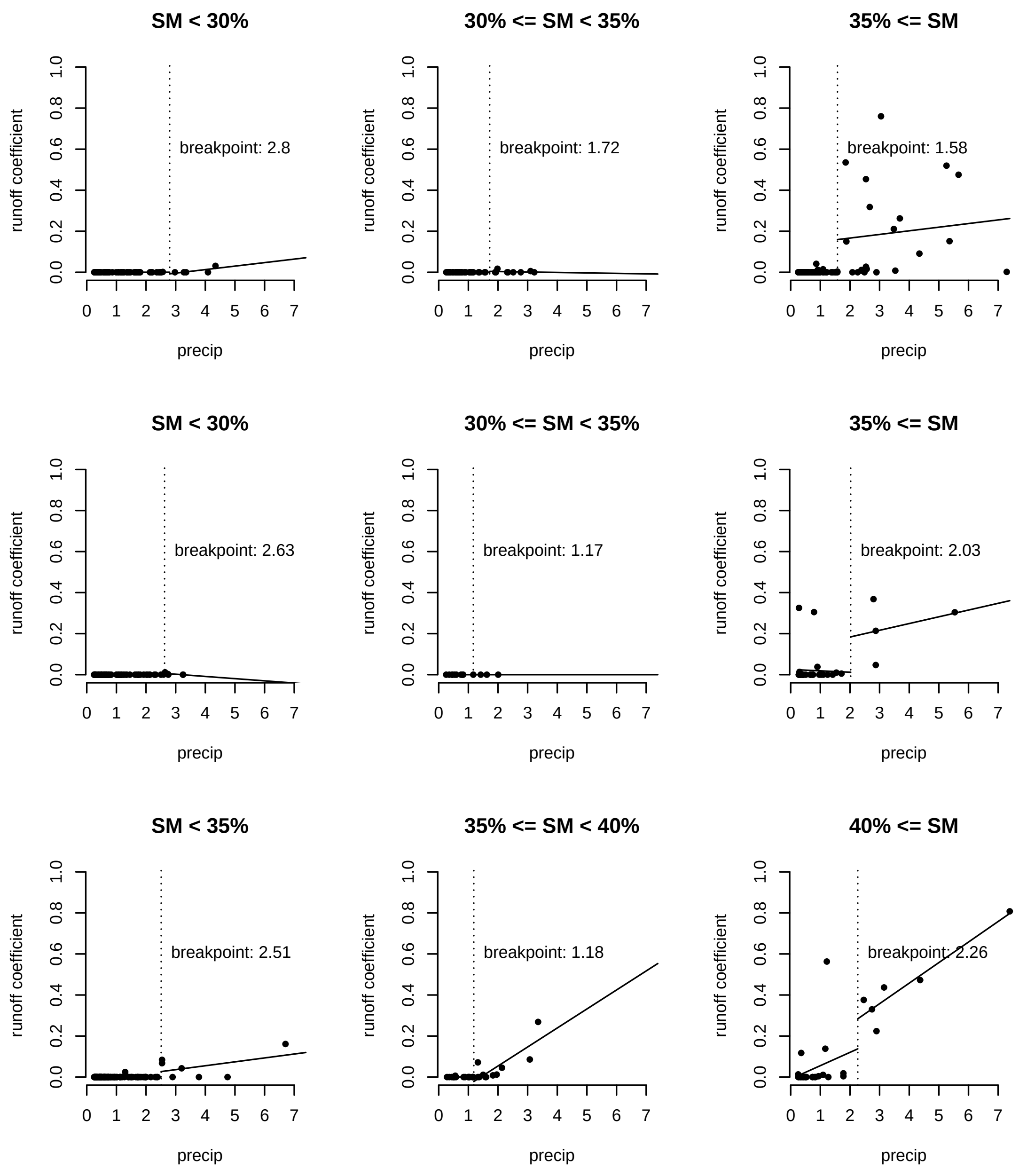

Figure 3: Top row: Koepke, middle row: Pagel, bottom row: Saxon.

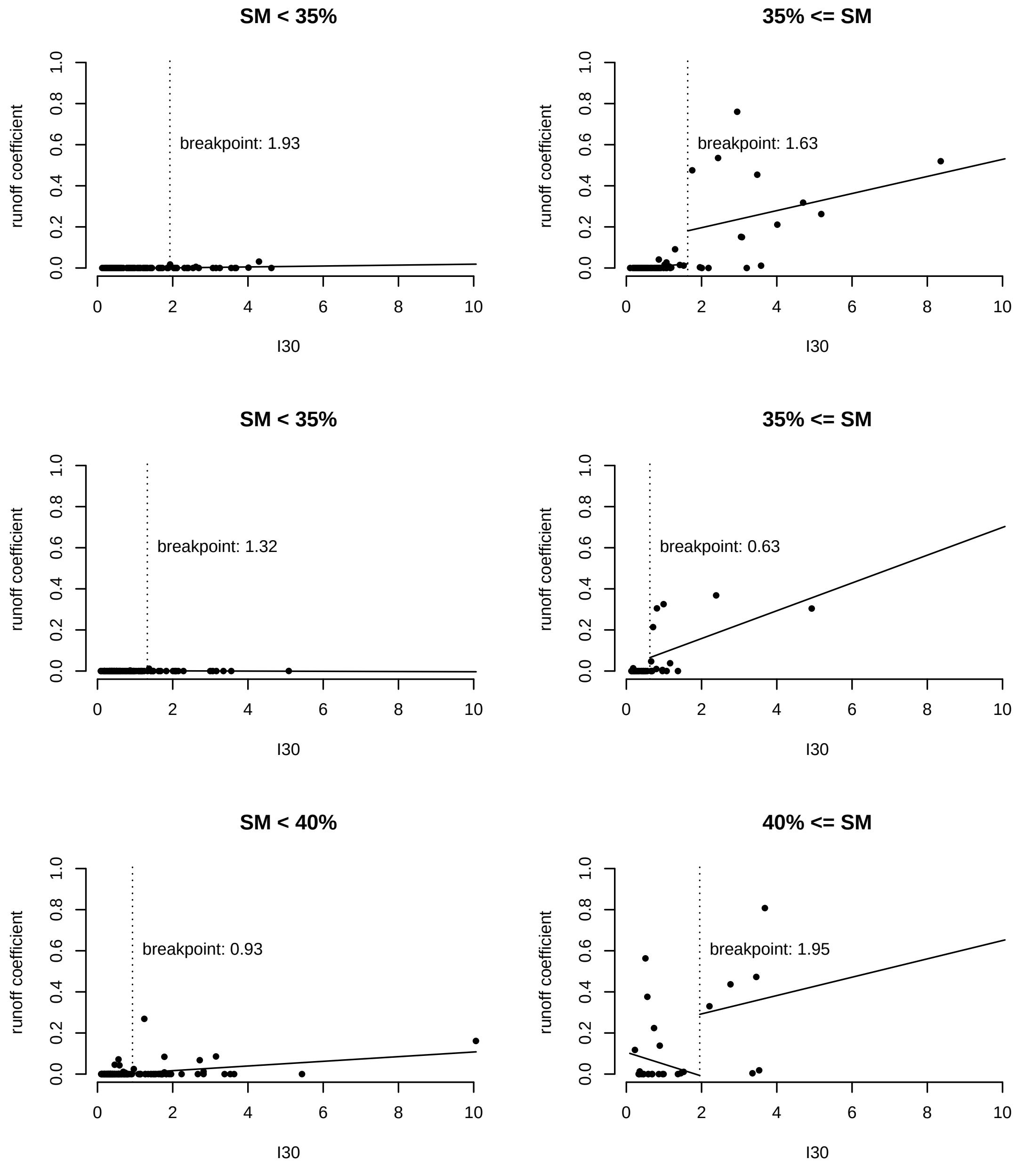

Figure 4: Top row: Koepke, middle row: Pagel, bottom row: Saxon.

```
-Inf <= SM < 35: 1.97
35 <= SM < Inf: 1.58

Precipitation breakpoints at pagel when binned by soil moisture:
-Inf <= SM < 35: 2.62
35 <= SM < Inf: 2.03

Precipitation breakpoints at saxon when binned by soil moisture:
-Inf <= SM < 40: 2.01
40 <= SM < Inf: 2.26

>     anova.bp <- function(data, split.on, site=NA, loc=NA, conts=FALSE, discrete.xran
+     {
+         if(!conts)
+         {
+             if(discrete.xrange)
+                 th <- which.min( sapply(X=xrange, FUN=piecewise_disjoint, x=data[dat
+             else
+             {
+                 lower_lim = sort( data[data$site==site,split.on] )[4]
+                 upper_lim = sort(data[data$site==site,split.on], decreasing=T )[4]
+                 th <- optimize(piecewise_disjoint, upper=upper_lim, lower=lower_lim,
+             }
+
+             X.bp = cbind( ifelse(data[data$site==site,split.on]>=th, 1, 0), data[dat
+         }
+         else
+         {
+             if(discrete.xrange)
+                 th <- which.min( sapply(X=xrange, FUN=piecewise_conts, x=data[data$s
+             else
+             {
+                 lower_lim = sort( data[data$site==site,split.on] )[4]
+                 upper_lim = sort(data[data$site==site,split.on], decreasing=T )[4]
+                 th <- optimize(piecewise_conts, upper=upper_lim, lower=lower_lim, x=
+             }
+
+             X.bp = cbind( data[data$site==site,split.on], max(0, data[data$site==sit
+         }
+
+         X.line = data[data$site==site,split.on]
+         y = data[data$site==site,]$rc
+
+           fit.bp = lsfit(x=X.bp, y=y)
+           fit.line = lsfit(x=X.line, y=y)
+
+           cat(paste("theta= ", th, "\n", sep=""))
+
+           if(!conts)
+               1-pf(q=(sum(fit.line$resid**2) - sum(fit.bp$resid**2)) / 3 / (sum(fit.bp
+           else
+               1-pf(q=(sum(fit.line$resid**2) - sum(fit.bp$resid**2)) / 2 / (sum(fit.bp
+       }
```

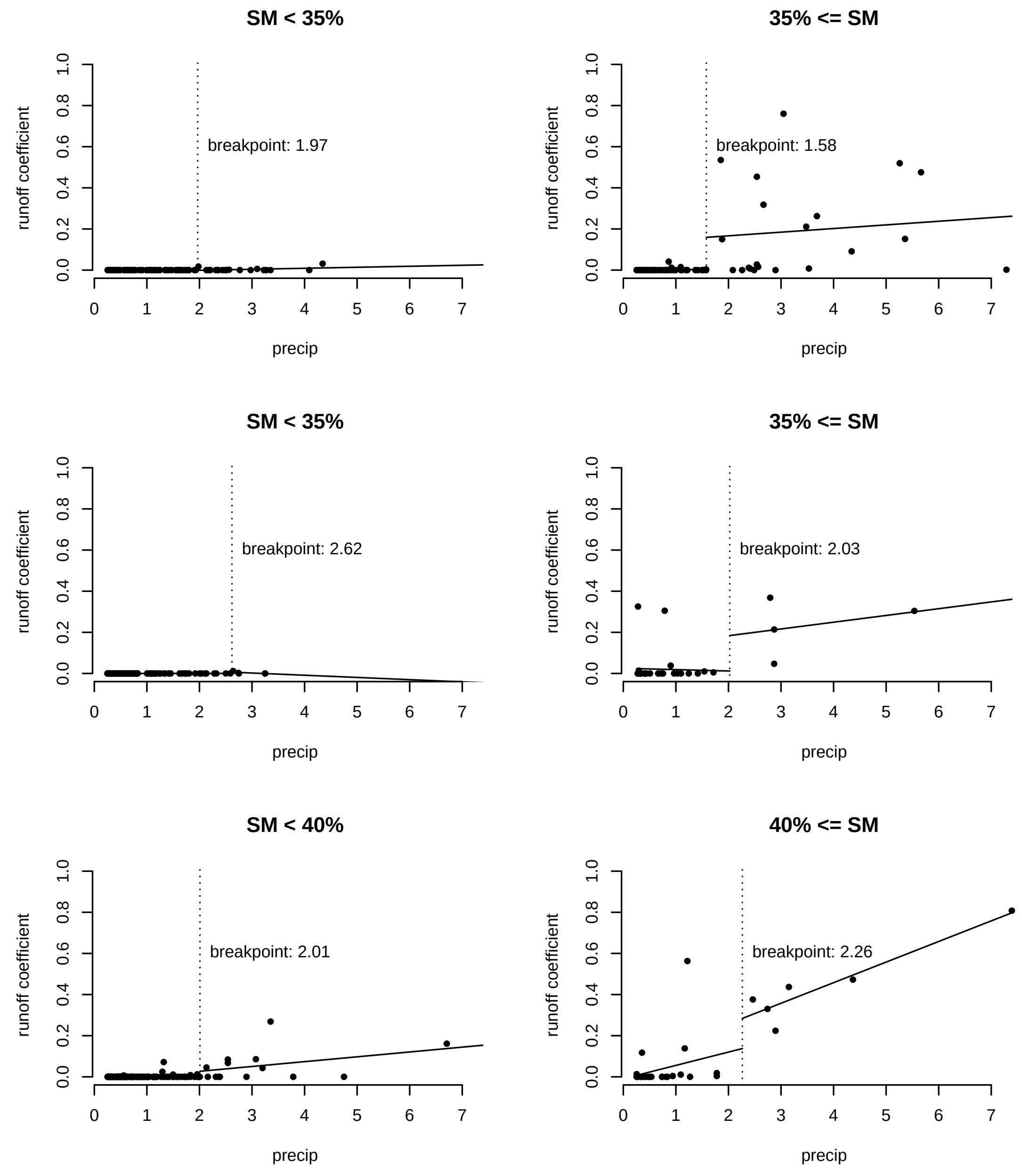

Figure 5: Top row: Koepke, middle row: Pagel, bottom row: Saxon.