Paper 210-26

JMP and *JSL* Used to Create a Visual Application

Andy Mauromoustakos and Kevin Thompson, University of Arkansas, Fayetteville, AR

ABSTRACT

Production jobs, customization, packaging data, manipulation, simulation, and record keeping have improved with the newly released JMP Scripting Language (JSL). JSL is a very simple hierarchical language for doing calculations and sending messages to objects. A flavor of the capabilities of JSL (a hybrid between SAS Macro Language and IML that look more than C and Java code than SAS code) will be demonstrated by creating a visual application for keeping basketball statistics and graphics. The example script demonstrates potential uses of scripting the most exciting addition of the new version of JMP Version 4. The application draws the basketball court and allows the "score keeper/statistician" to enter information about the progress of the game as it develops. Information is entered about the shots taken during the game as it progresses. Relevant game and half time statistics then can be produced during breaks in the action to be used by coaches/announcers for describing the game and analyzing its outcome. The graphs produced can be used to collect real time valuable spatial information for a unique "visual summary of the game".

INTRODUCTION

At first glimpse, Version 4 of JMP looks different than Version 3, but the user will soon find that it has similar but vastly improved functionality via the addition of a scripting language. JMP Script Language (JSL) allows the researcher to code macros for performing new analysis and drawing customized interactive graphs. Handle and Mousetrap are functions available in JSL for making interactive graphs that respond to clicking and dragging. Mousetrap takes its parameters from the coordinates of just a click, without using a draggable handle. Our visual application (a JMP script) in a nutshell allows the "analyst" (assistant coach, manager, reporter or the casual fan watching a basketball game live or on the TV) to keep his own scorecard that record the location on the court from where the shot was taken its distance to the basket, who took it, and if it was successful. During any particular break in action play the user will be given the opportunity to save the shots and record the types of defenses. It provides visuals and statistics mostly using the strengths of JMP as an exploratory data analysis tool as it builds the DataTable from the shots taken by each team. The result can be used as a tool to create game strategies since it helps break down the important scoring related activity of the game. It can be used by "scouts" to provide information to a team usually not found in official Box scores since it identifies the spatial location of the shots. During the basketball season the analyst can then build an extensive and valuable database with player tendencies and effectiveness that he can market to a team or its competitors. In today's age of "big time" men college basketball and the NBA there is a need and a place for providing this kind of information database and the need for visually mining the data so that it can provide the edge that one needs to succeed. What makes the application useful is that anybody familiar with JMP and its features could learn how to use the program and maybe keep the statistics with a notebook that has JMP. It would particularly useful in any gym or little league that does not happen to have the luxury of an official scorers table and scoreboard.

Since this paper is about a specific JMP script that we wrote one may ask what is a JMP Script?

1. First and foremost is a program understood and executed only by having the JMP software
2. It provides a short cut for producing routine tasks
3. It is an explicit set of instructions that aims to achieve the same results as an interactive command.
4. It is usually for the advanced "power user" tool for achieving special tasks.
5. All of the above.

We will illustrate the script by showing most of its current features and capabilities from a recent NCAA basketball game between the University of Arkansas and the University of Alabama played on January 20, 2001, in Fayetteville AR. The author was at the game with a notebook computer that had JMP and recorded the visuals and statistics with the bball script. Due to the length of the script we will not provide the script in its entire length but will discuss the main routines in it and show its output for the example game.

BBALL.JSL SCRIPT

JSL CODE DETAILS

JSL, as a language, does not appear to be designed for large, complicated programs. An unscoped JSL variable may refer either to a JSL Global variable or to a DataTable variable depending of how and when it is first used. Global JSL variables are defined for the entire JMP session, and may cause problems if you leave JMP running between distinct projects. Local JSL variables that are protected from general JMP session are difficult to use, except when they are few in number. In this program, all variables are explicitly scoped using the :: and : operators.

A JMP JSL program is constructed as a sequence of expressions instead of statements. The semi-colon (;) in JSL glues expressions together and is not equivalent to a statement termination token. Single semi-colons at the end of an expression sequence may cause syntax errors. In order to partial emulate a statement-oriented language; many expression sequences end with the dummy expression "0".

To emulate a modular style, this program is organized as a sequence of expression and function definitions. The expression MAIN is defined at the beginning of the file and evaluated at the end of the file. All other expressions are evaluated within MAIN, by events activated by Buttons or MouseTrap clicks or within other expressions. This organization allows the sections of the program to be arranged according to program logic as opposed to execution order and to be more easily edited/replaced.

```
// BBall.JSL  A JSL program to record a
// BasketBall game
// Version 1.0   19Jan2001
// Requires:  JMP 4.0.2 or later
// Written by Kevin Thompson
//          AgriStat  AGRX 101 UofArk
//          (501)575-2448
//          kthompsn@uark.edu
//-----------------------
// Define MAIN expression
//-----------------------
::MAIN=Expr(
  // 10 is good on 800x600, 15 on 1024x800
  ::Magnification=10;

  ::TeamA=GetTeamInfo("Select the DataTable for the FIRST
team","Select the player numbers for the FIRST team" );
  ::TeamB=GetTeamInfo("Select the DataTable for the
SECOND team","Select the player numbers for the SECOND
team");
  ::TeamAScore=0;
  ::TeamBScore=0;
  ::InitializeCourtDimentions;
  ::CreateOutputDataTable;
  ::CreateSelectPlayerWindowStrings;
  ::AssignLeftRight;
  ::CreateMainWindow;
0); //End::MAIN

// Create Main Display Window (simplified)
// Called by MAIN
::CreateMainWindow=Expr(
  ::dbSureShot=NewWindow("BBall",
    VListBox(::dbLastPlay=
    TextBox("Last Play")
      ,HListBox(
          VListBox( // Left Team Name/Score
            ::dbTeamLName=TextBox(" ")
           ,::dbTeamLScore=TextBox(" ")
           ,ButtonBox("FT",
                       FreeThrowButton("L"))
           )
         ,::DrawCourt
         ,VListBox( // Right Team Name/Score
            ::dbTeamRName=TextBox(" ")
           ,::dbTeamRScore=TextBox(" ")
           ,ButtonBox("FT",
                       FreeThrowButton("R"))
           )
         )
      ,HListBox(
         ButtonBox("Swap L/R",SwapLeftRight )
        ,ButtonBox("Refresh" ,Refresh       )
        ,ButtonBox("Save Pic",SavePicture   )
        ,ButtonBox("Hide "   ,MarkBreak     )
        )
      )
  ); // End NewWindow
  ::Refresh
);//End::CreateMainWindow
```

/ Definition of other expressions */*

```
//-----------------------
// Invoke MAIN expression
//-----------------------
::MAIN;
0 // Return 0
```

The first expression executed by main is the GetTeamInfo() function. Currently, the team Player data tables are selected from the list of JMP files on disk. It would be better to allow a choice from a list of currently open data tables with an option to load a new DataTable from disk. Unfortunately, JSL does not provide a means of getting the list of active DataTables. The GetTeamInfo() function returns a list of the form {TeamName = "UA", TeamColor = 3, PlayerNumbers = {3, 4, 5}, PlayerNames = {"Charles Tatum", "Carl Baker", "TJ Cleveland"}} which is used in other expressions in the program.

Parameters defining court dimensions and markers are specified separately from drawing the court so that a user could customize the court for high school, NCAA or NBA.

```
// Dimentions of BasketBall court, measurements in ft
// Called by MAIN
::InitializeCourtDimentions=Expr(
 ::CourtLength=94;
   //NCAA:94' NBA:94' NFHS:84'
 ::CourtWidth =50;
 ::CenterLineColor="Black";
 ::CenterTextColor="Orange";
 ::CenterText     ="NCAA";
 ::JumpCircleColor="Gray";
 ::JumpCircleRadius1=2;
 ::JumpCircleRadius2=6;
 ::LaneColor ="Gray";
 ::LaneLength=18 + 10/12;
 ::LaneWidth =12;
 ::FreeThrowCenter =19;
 ::LeftBasketColor ="Black";
 ::RightBasketColor="Black";
 ::BasketCenter=63/12;
 ::BasketRadius= 9/12;
 ::BackboardCenter=4 ;
 ::BackboardWidth=6  ;
 ::BackboardCenter=4 ;
 ::_3ptColor="Gray"  ;
 ::_3ptRadius=19 + 9/12;
   //NCAA:19'9" NBA:22' NFHS:19'9"
0); //End::InitializeCourtDimentions
```

The court itself is drawn in a graph box that contains a graphic script that draws the court reference lines. The graph box also enables MouseTrap scripts that execute when you click on the court.

```
// Create Graph Window of BasketBall Court
// Called by CreateMainWindow
::DrawCourt=Expr(
 ::dbCourt=Graph(
   FrameSize(Magnification*CourtLength/2,
             Magnification*CourtWidth/2)
  ,XScale(-CourtLength/2,CourtLength/2)
  ,YScale(-CourtWidth/2 ,CourtWidth/2 )
  ,Xaxis(Min(-CourtLength/2),
         Max(CourtLength/2),
         Inc(CourtLength),
         ShowMajorTicks(0),
         ShowLabels(0) )
  ,Yaxis(Min(-CourtWidth/2 ),
         Max(CourtWidth/2 ),
         Inc(CourtWidth ),
         ShowMajorTicks(0),
         ShowLabels(0) )
  ,XName(" ")
  ,YName(" ")
  ,DoubleBuffer
  , //Graph Scripts
    ::DrawCourtReferenceLines;
    MouseTrap(::MouseTrapDown,
              ::MouseTrapUp);
    ::DisplayShotsOnCourt;
   0) // EndGraph
);// End::DrawCourt

// Draw Reference lines on BasketBall Court
// Called by DrawCourt
::DrawCourtReferenceLines=Expr(
  // Court Border
  pen color(CenterLineColor);
  rect({-CourtLength/2,-CourtWidth/2},
       {CourtLength/2,CourtWidth/2});
  // HalfCourt Line
  pen color(CenterLineColor);
  vline(0, -CourtWidth/2, CourtWidth/2 );
  // Text at CenterCourt
```

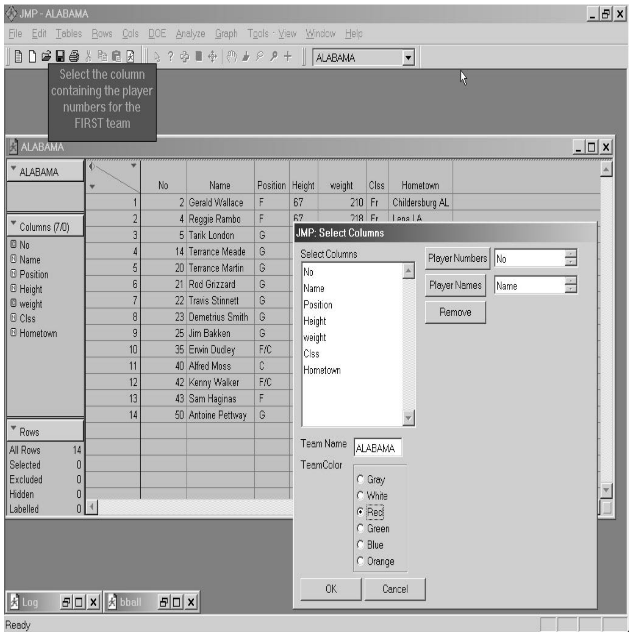
```
    text color(CenterTextColor);
    text (Center Justified, {0,0}, CenterText);
    //Jump Circles
    pen color(JumpCircleColor);
    circle({0,0},JumpCircleRadius1,
             JumpCircleRadius2);
    //FreeThrow Circle and Lane
    pen color(LaneColor);
    //Left
      rect({-CourtLength/2, LaneWidth/2},
          {-CourtLength/2+LaneLength,
                    -LaneWidth/2});
      circle({-CourtLength/2
               +FreeThrowCenter,0},
            LaneWidth/2);
    //Right
      rect({ CourtLength/2,-LaneWidth/2},
          { CourtLength/2
              -LaneLength, LaneWidth/2});
      circle({ CourtLength/2
               -FreeThrowCenter,0},
             LaneWidth/2);
    //3Point Circle
    pen color(_3ptColor);
    //Left
      line( {-CourtLength/2, _3ptRadius},
           {-CourtLength/2 +BasketCenter,
              _3ptRadius});
      arc( -CourtLength/2 +BasketCenter
                   -_3ptRadius,
          -_3ptRadius,
          -CourtLength/2 +BasketCenter
                    +_3ptRadius,
          _3ptRadius,
          0, 180);
      line( {-CourtLength/2,-_3ptRadius},
           {-CourtLength/2 +BasketCenter,
               -_3ptRadius});
    //Right
      line( { CourtLength/2,-_3ptRadius},
           { CourtLength/2-BasketCenter,-        _3ptRadius});
      arc(  CourtLength/2-BasketCenter+_3ptRadius,
           _3ptRadius,
           CourtLength/2-BasketCenter-_3ptRadius,
           -_3ptRadius,
           0, 180);
      line( { CourtLength/2, _3ptRadius},
           { CourtLength/2-BasketCenter, _3ptRadius});
    //Left Basket and BackBoard
    pen color(LeftBasketColor);
    circle({-CourtLength/2+BasketCenter,0},BasketRadius);
    vline(-CourtLength/2+BackboardCenter,
        -BackboardWidth/2, BackboardWidth/2);
    //Right Basket and BackBoard
    pen color(RightBasketColor);
    circle({ CourtLength/2-BasketCenter,0},BasketRadius);
    vline( CourtLength/2-BackboardCenter,
         BackboardWidth/2,-BackboardWidth/2);
   0); //End::DrawCourtReferenceLines
```

When the MouseTrap script is executed it displays a window to allow the user to indicate the player taking the shot and whether was good. In order to include team and player information within the window, it is first constructed as a character string and then parsed into a JSL expression.

```
// MouseTrap scripts, invoked by MouseClicks
// Called by mouseclick on basketball court (DrawCourt)
::MouseTrapDown={}; // Empty script
::MouseTrapUp=Expr(
  ShotLocation=EvalList({"Court",X,Y});
  ::CloseSelectPlayerWindow;
  ::DisplaySelectPlayerWindow;
0); //End::MouseTrapUp

// Create DisplaySelectPlayerWindow from the character strings
// Called by AssignLeftRight, SwapLeftRight
// DisplaySelectPlayerWindow is called by MouseTrap on
(DrawCourt)
::CreateSelectPlayerWindow=Expr(
  ::DisplaySelectPlayerWindow=
     Parse(::chrSelectPlayerStart
          || ::chrSelectPlayerTeamA
          || ::chrSelectPlayerMiddle
          || ::chrSelectPlayerTeamB
          || ::chrSelectPlayerEnd ) ;
0);


// Create character strings used to construct SelectPlayer Pop-
up Window
// Called by MAIN
::CreateSelectPlayerWindowStrings=Expr(
  // SelectPlayerString Start
  ::chrSelectPlayerStart=
    "::dbSelectPlayer=NewWindow(\!"Select Player\!",  "
    || "HListBox(" ;

  // SelectPlayerString TeamA VListBoxB(TeamName,
PlayerList)
  ::chrSelectPlayerTeamA="VListBox(TextBox(\!""
                 || Char(::TeamA["TeamName"])
                 || "\!") ";
  For(i=1, i<=NItems(::TeamA["PlayerNumbers"]), i++,
     ::chrSelectPlayerTeamA = ::chrSelectPlayerTeamA
      || ",HListBox(
           ButtonBox(\!"X\!",
            ::SelectPlayerButton({MakeBasket=1,
              TeamName=\!""
              || Char(::TeamA["TeamName"])
              || "\!", PlayerNumber=\!""
              || Char(::TeamA["PlayerNumbers"][i])
              || "\!", PlayerName=\!""
              || Char(::TeamA["PlayerNames"][i])
              || "\!"}) ) "
      ||  ",TextBox(\!""
              || Char(::TeamA["PlayerNumbers"][i])
              ||"\!") "
      ||  ",ButtonBox(\!"O\!",
            ::SelectPlayerButton({MakeBasket=0,
              TeamName=\!""
              || Char(::TeamA["TeamName"])
              || "\!", PlayerNumber=\!""
              || Char(::TeamA["PlayerNumbers"][i])
              || "\!", PlayerName=\!""
              || Char(::TeamA["PlayerNames"][i])
              || "\!"}) ) "
      ||  ",TextBox(\!""
              || Char(::TeamA["PlayerNames"][i])
              ||"\!") "
      ||  ")"  // End HLB
     );
    ::chrSelectPlayerTeamA=::chrSelectPlayerTeamA  || ")";

    // SelectPlayerString Middle
    ::chrSelectPlayerMiddle=",TextBox(\!"    \!"),"  ;

    // SelectPlayerString TeamB VListBox(TeamName, PlayerList)

    /* same as for TeamA */

    // SelectPlayerString End
    ::chrSelectPlayerEnd= ")"
      || ",TextBox(\!"    \!")"
      || ",HListBox(TextBox(\!"  \!")"
      ||          ",TextBox(\!"  \!")"
      ||          ",ButtonBox(\!"Cancel\!",
                          SelectPlayerButton({}))"
      ||          ")"
      || ")" ;

  0); // End::CreateSelectPlayerWindowStrings
```

RUNNING BBALL

The user needs to have version 4.02 or later of JMP and have knowledge about the rules of basketball "to play". Also he needs to provide before the start of the game as "input" the two teams in a form of JMP data that has the player number and player name. For NCAA and NBA game this task can be accomplished from Internet sites (such as ESPN CNNSI and many others) where all the user has to do is essentially cut and paste the roster of each team into a JMP datable as seen in the capture below. The user is allowed to modify the team name and assign a team color.


JMP - ALABAMA
File Edit Tables Rows Cols DOE Analyze Graph Tools View Window Help
ALABAMA
Select the column containing the player numbers for the FIRST team
ALABAMA
Log
bball
Ready

ALABAMA

Columns (7/0): No, Name, Position, Height, weight, Clss, Hometown

Rows: All Rows 14, Selected 0, Excluded 0, Hidden 0, Labelled 0

	No	Name	Position	Height	weight	Clss	Hometown
1	2	Gerald Wallace	F	67	210	Fr	Childersburg AL
2	4	Reggie Rambo	F	67	218	Fr	Lena LA
3	5	Tarik London	G				
4	14	Terrance Meade	G				
5	20	Terrance Martin	G				
6	21	Rod Grizzard	G				
7	22	Travis Stinnett	G				
8	23	Demetrius Smith	G				
9	25	Jim Bakken	G				
10	35	Erwin Dudley	F/C				
11	40	Alfred Moss	C				
12	42	Kenny Walker	F/C				
13	43	Sam Haginas	F				
14	50	Antoine Pettway	G				

JMP: Select Columns

Select Columns: No, Name, Position, Height, weight, Clss, Hometown

Player Numbers: No

Player Names: Name

Remove

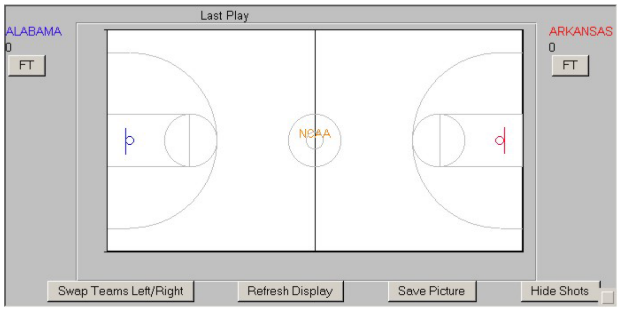
Team Name: ALABAMA

TeamColor: Gray / White / [✓ Red] / Green / Blue / Orange

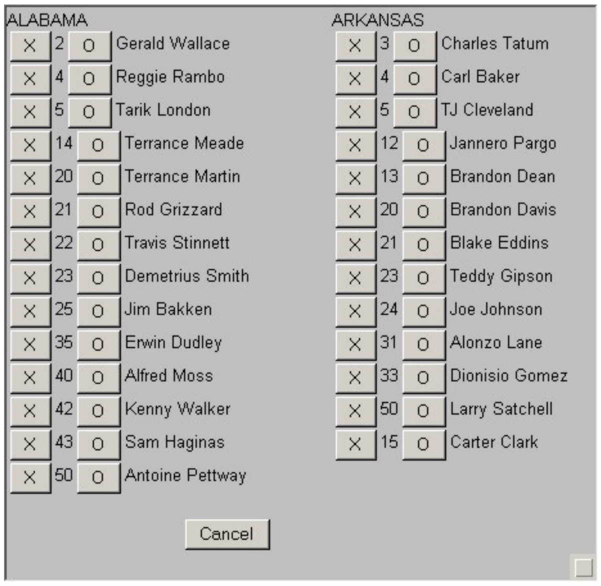
OK Cancel

After the user repeats the process above with the second team, the script draws the court and the program is ready to start recording the action in the order of shots taken using a mouse. It is worth noting here that the basketball court has dimensions for college court with the 3-point line extending to almost 20 feet instead of the 3-point line used by the NBA. The user can make these kinds of adjustments fairly easily in the section of the code that draws the outline of the court mentioned above. Adjustments for high schools that usually play on an 84 feet long court can be easily made also in one place of the script code that sets the parameters for the court dimensions. Information on typical court dimensions (for high school, NCAA, NBA) can be found on various websites such as http://www.ask-the-ref.com/ or http://www.nba.com/basics/rules/RULES.html.

All distances are calculated and recorded as X,Y coordinates from the (0,0) point that is the center of the court. Distances from the position of the shot taken to the basket (not the backboard) are calculated for each shot. Each team is colored coded (we used blue for Alabama and red for Arkansas to help the visual instead of keeping close to the actual team uniforms. Each team player is color-coded and each player is supposed to shoot to the basket (that is assigned to his team) as shown below. In the background it will open a DataTable (empty at the start) that will trap and record all the mouse clicks one for each shot taken.



We will review later some of the additional buttons at the bottom and the side of the court. Now we are ready to record the first shot taken by either team. To trap and display a field goal attempt FGA (Field Goal Attempt) we click with the mouse approximately where the shot was taken and the code open up a dialog box with both teams and player information for us to select by selecting the (X or O buttons on either side of each player number) that indicates a made or missed shot respectively as shown below.



ALABAMA				ARKANSAS			
X	2	O	Gerald Wallace	X	3	O	Charles Tatum
X	4	O	Reggie Rambo	X	4	O	Carl Baker
X	5	O	Tarik London	X	5	O	TJ Cleveland
X	14	O	Terrance Meade	X	12	O	Jannero Pargo
X	20	O	Terrance Martin	X	13	O	Brandon Dean
X	21	O	Rod Grizzard	X	20	O	Brandon Davis
X	22	O	Travis Stinnett	X	21	O	Blake Eddins
X	23	O	Demetrius Smith	X	23	O	Teddy Gipson
X	25	O	Jim Bakken	X	24	O	Joe Johnson
X	35	O	Erwin Dudley	X	31	O	Alonzo Lane
X	40	O	Alfred Moss	X	33	O	Dionisio Gomez
X	42	O	Kenny Walker	X	50	O	Larry Satchell
X	43	O	Sam Haginas	X	15	O	Carter Clark
X	50	O	Antoine Pettway				

Cancel

The Cancel button cancels the action taken without creating an observation in cases of accidental clicks. The code shown below traps the information that is created and stores it in a JMP DataTable for exploratory analysis during breaks in action. Free-Throws are entered by clicking on the FT button below each team name "outside the court". This action opens the same dialog and prompts the user to select player and if the FTA (FreeThrow Attempt) was made or missed. Since all FTA's are taking from the same distance and count as one point they are hidden so that they do not clutter the visual but are properly accounted. The distances to the basket (outside the 3 point collegiate line) is used to account for the 2point vs 3point shot and are used to calculate the variables Points by the formulas below:

```
Match( :Type, "2point", 2, "3point", 3, "FreeThrow", 1, 0) * (
:HitMiss == 1)
```

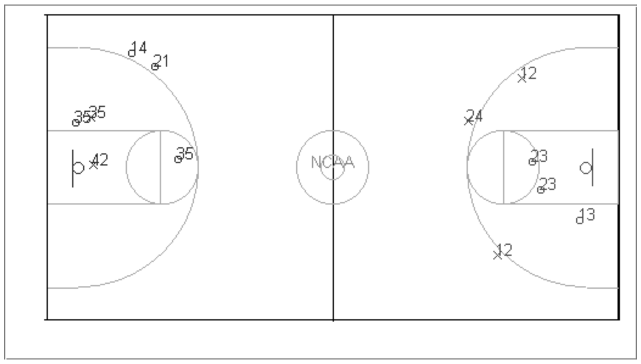
We also calculate PointsMissed by replacing in the formula above :HitMiss=1 with :HitMiss=0. The Points variable is used to calculate the variables that contain each the update current Scoreboard shown on the graphic using formulas such as the one below for each team that accumulates point for each team.

```
If(Row() == 1,  :Points * ( :TeamName == "ARKANSAS"),
:Points * ( :TeamName == "ARKANSAS") + Lag( :ARKANSAS
Score, 1))
```

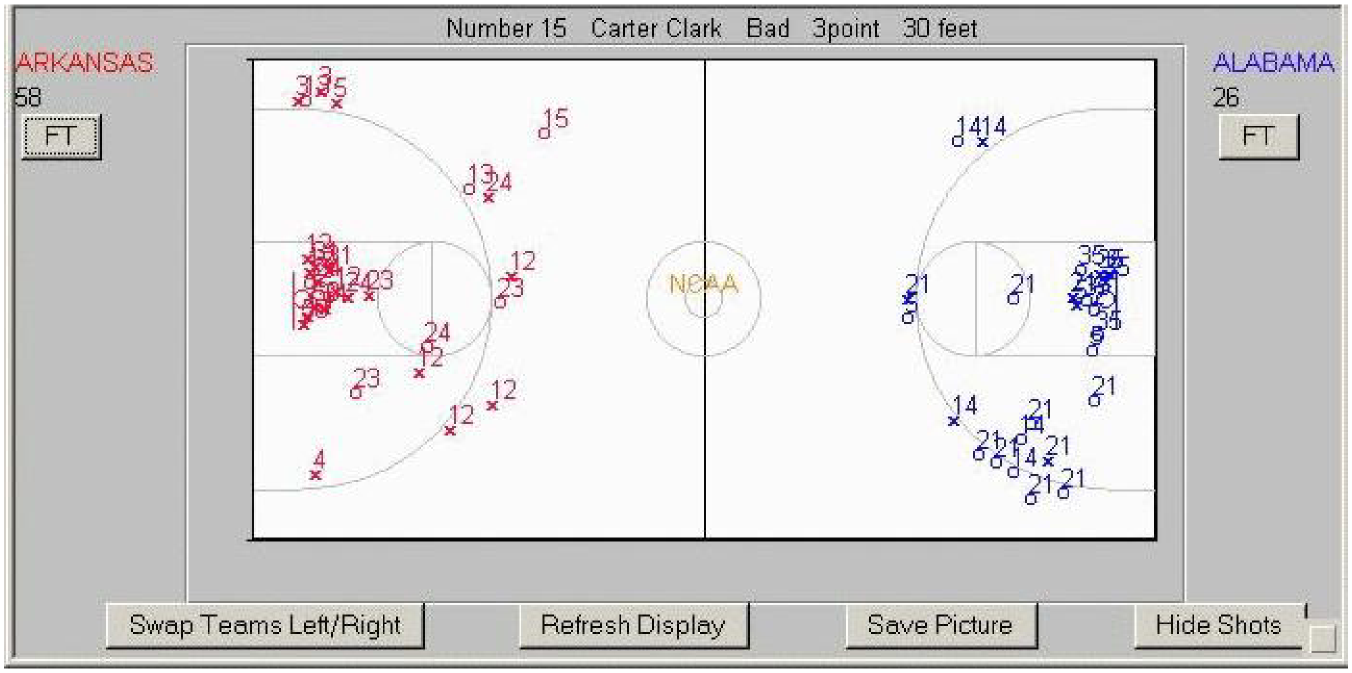
A computer time stamp is also saved with each shot and the graphic (and DataTable) are both updated with each shot until there is a break in action (such as timeouts or due to users choice). The user can save the graphic with all the shots clicking on the SavePicture button in the bottom right of the court. That action saves both the JMP DataTable and the graph (in rtf format). The last play entered is recorded in print on the top of the court and (see caption below) and is replayed in audio by the program from the notebook speakers. This feature allows the user to verify the information he entered with the announcement that usually follows the play during the game either by the game or by the TV announcers. The HideShots button allows the user to hide the previous shots and put a marker with his own comments (text) that perhaps pertain to defense, real clock time, etc. The defense stamp will allow the user to select and display a visual of the shots that a team is allowing under a specified defense and also it will allow to user to evaluate team offensive and defensive effectiveness. More on that in the section for future improvements for future upcoming versions of the program. Markers such as these that record defense types and possessions can be used to evaluate scoring efficiencies (scores/possessions) for different types of defenses. Of course these types of enhancements will require that the user will recognize and immediate record the changes in the defense even during the same possession. We anticipate that these buttons that record possessions and defenses will appear at the side of the graphic visual below each FT box. A visual trail of nine pictures (rtf files) for each half of play from the Alabama and Arkansas game were saved usually attributed to timeouts. We show here the first frame (up to the first timeout when both teams were primarily on a man-to-man defense). The information recorded in print and by the speaker pertains to the last shot for that frame.

Number 13 Brandon Dean Bad 2point 9 feet

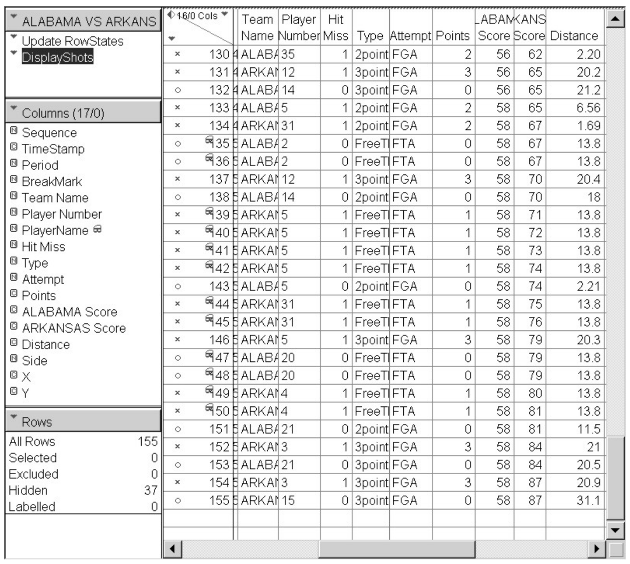
ALABAMA	ARKANSAS
MAN defense	Man Defense
3	8



The Swap Teams Left/Right button is used in the beginning of the second half to make it easier to keep the statistics for the analyst/scorer from the same “spot on the court”. We also provide a screen captured for the entire second half that Arkansas played on a very effective matchup-zone defense that gave Alabama trouble especially since they could score from the outside. On the other hand Arkansas was extremely proficient both inside and outside and firing away seven 3 pointers from the outside. The score shown pertains to that half only and not the entire game. We believe that graphs can provide the fan with a unique visual summary of the game and its story. We can replay the graph as animation displaying shots taken made or missed by who one at the time with a half second delay so that the viewer could “replay the action per half in less than 2 minutes.


Number 15 Carter Clark Bad 3point 30 feet

A view of the information saved in the DataTable at the end of a game is shown below. We can see that all the 37 total (FTA for the game) taken by either team have a hidden row state assigned to them. These datasets provides more than the usual scoring information and can be used to provide a unique insight about the outcome of the game using the excellent JMP exploratory and analytical features.



ALABAMA VS ARKANS
Update RowStates
DisplayShots

Columns (17/0)
Sequence
TimeStamp
Period
BreakMark
Team Name
Player Number
PlayerName
Hit Miss
Type
Attempt
Points
ALABAMA Score
ARKANSAS Score
Distance
Side
X
Y

Rows

All Rows	155
Selected	0
Excluded	0
Hidden	37
Labelled	0

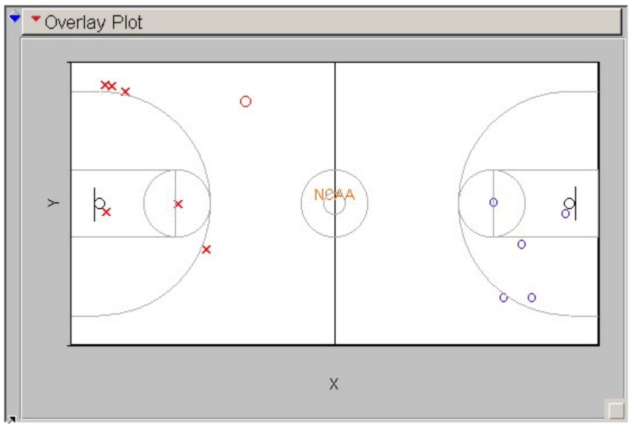
16/0 Cols

			Team Name	Player Number	Hit Miss	Type	Attempt	Points	LABAM Score	KANS Score	Distance
×	130	4	ALABA	35	1	2point	FGA	2	56	62	2.20
×	131	4	ARKAN	12	1	3point	FGA	3	56	65	20.2
○	132	4	ALABA	14	0	3point	FGA	0	56	65	21.2
×	133	4	ALABA	5	1	2point	FGA	2	58	65	6.56
×	134	4	ARKAN	31	1	2point	FGA	2	58	67	1.69
○	135	5	ALABA	2	0	FreeT	FTA	0	58	67	13.8
○	136	5	ALABA	2	0	FreeT	FTA	0	58	67	13.8
×	137	5	ARKAN	12	1	3point	FGA	3	58	70	20.4
○	138	5	ALABA	14	0	2point	FGA	0	58	70	18
×	139	5	ARKAN	5	1	FreeT	FTA	1	58	71	13.8
×	140	5	ARKAN	5	1	FreeT	FTA	1	58	72	13.8
×	141	5	ARKAN	5	1	FreeT	FTA	1	58	73	13.8
×	142	5	ARKAN	5	1	FreeT	FTA	1	58	74	13.8
○	143	5	ALABA	5	0	2point	FGA	0	58	74	2.21
×	144	5	ARKAN	31	1	FreeT	FTA	1	58	75	13.8
×	145	5	ARKAN	31	1	FreeT	FTA	1	58	76	13.8
×	146	5	ARKAN	5	1	3point	FGA	3	58	79	20.3
○	147	5	ALABA	20	0	FreeT	FTA	0	58	79	13.8
○	148	5	ALABA	20	0	FreeT	FTA	0	58	79	13.8
×	149	5	ARKAN	4	1	FreeT	FTA	1	58	80	13.8
×	150	5	ARKAN	4	1	FreeT	FTA	1	58	81	13.8
○	151	5	ALABA	21	0	2point	FGA	0	58	81	11.5
×	152	5	ARKAN	3	1	3point	FGA	3	58	84	21
○	153	5	ALABA	21	0	3point	FGA	0	58	84	20.5
×	154	5	ARKAN	3	1	3point	FGA	3	58	87	20.9
○	155	5	ARKAN	15	0	3point	FGA	0	58	87	31.1

The highlighted (blue above) DisplayShoots table property is a script embedded in the JMP DataTable by the BBall.JSL script that recreates an image of the court with "user selected shots" for display. The script itself is a similar code as the one to the part of the program mentioned in the code section. The script that creates the table property that displays selected shots looks as follows.

```
DisplayShotsString=Char( " "
 || "dtShots<<SetProperty(\!"DisplayShots\!","
 ||    "::Magnification=" || Char(::Magnification) || ";"
 ||    Char(NameExpr(InitializeCourtDimentions))   || ";"
 ||    "::dbPlot=Overlay Plot(X(:X), Y( Y), Separate Axes(1));"
 ||    "::rptPlot=::dbPlot<<Report; "
 ||    "::rptPlot[OutlineBox(1)][PictureBox(1)][AxisBox(1)]<<
          AxisSettings(Min(-CourtWidth/2 ),
                       Max(CourtWidth/2 ),
                       Inc(CourtWidth ),
                       ShowMajorTicks(0),
                       ShowLabels(0)); "
 ||    "::rptPlot[OutlineBox(1)][PictureBox(1)][AxisBox(2)]<<
          AxisSettings(Min(-CourtLength/2),
                       Max(CourtLength/2),
                       Inc(CourtLength),
                       ShowMajorTicks(0),
                       ShowLabels(0)); "
 ||    "::rptPlot[OutlineBox(1)][PictureBox(1)][FrameBox(1)]<<
          FrameSize(Magnification*CourtLength/2,
                    Magnification*CourtWidth/2);"
 ||    "::rptPlot[OutlineBox(1)][PictureBox(1)][FrameBox(1)]<<
         AddGraphicsScript("
 ||      Char(NameExpr(DrawCourtReferenceLines))
 ||      ");"
 || ")" );
Eval(Parse( DisplayShotsString ));
0);//End::CreateOutputDataTable
```

For the purpose of illustrating the important storyline that appeared in every article about the game the following day on the news we selected and hid all observations up to the last four minutes of the game. Then by running the DisplayShots script one can provide the graphic that illustrates the impressive Arkansas 22-0 run to end the game and turn a 7-point game into a rout. Below we display the last 34 shot sequence where Alabama missed all their FG/FT and Arkansas did not miss a shot except the last 30 foot shot at he end.



We would like to concentrate the rest of the paper on the visual exploration of the data from the data collection that could provide insight and the scripts that can help provide these types of analyses/graphics.

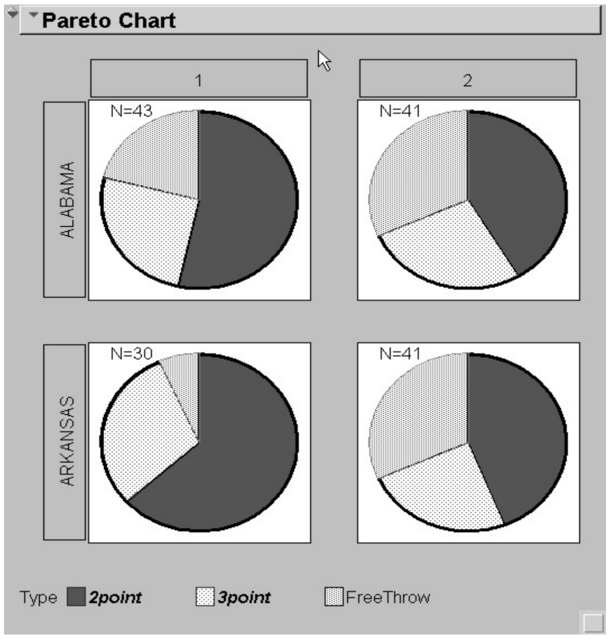
GAME STATISTICS GRAPHS AND SCRIPTS

JMP GRAPH PLATFORMS

GRAPH→PARETO

Team shot type breakdown by half can be provided by the following script and the graphic that follows.

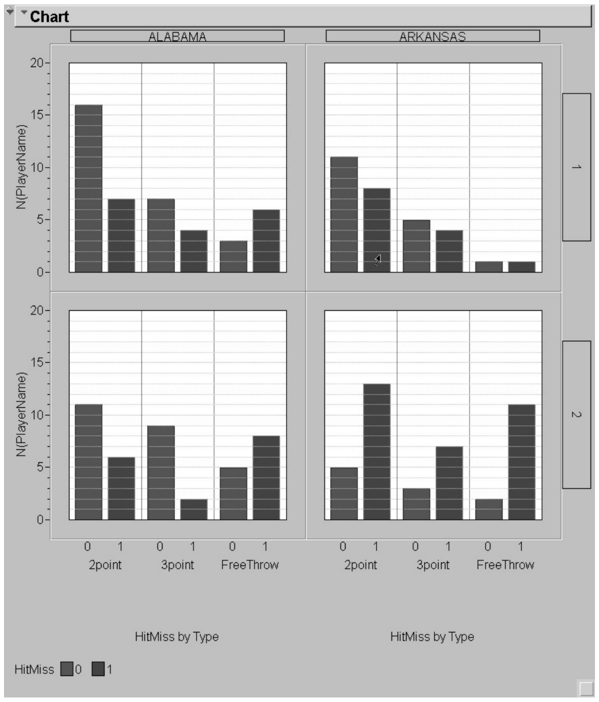
```
Pareto Plot(Cause( :Type), X( :TeamName,  :Period), Pie
Chart(1), N Legend(1), Colors(Blue), cause[1] << Markers(8),
cause[2] << Markers(8))
```



GRAPH→CHART

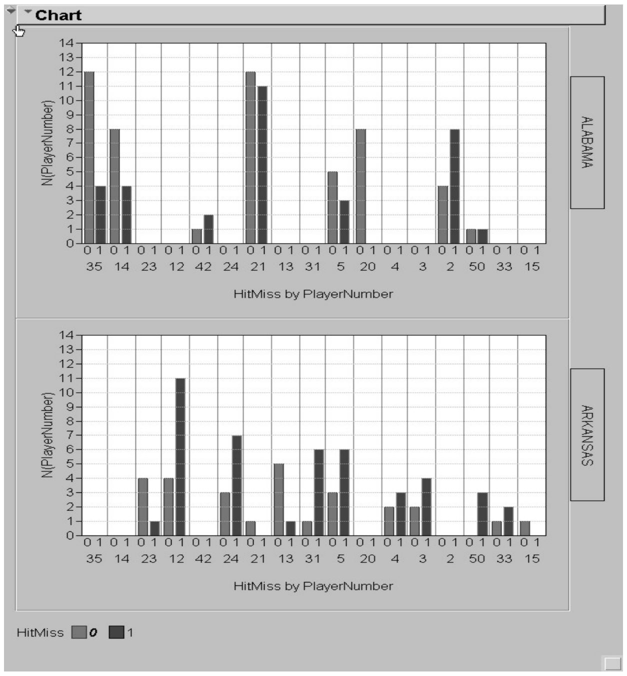
The JSL script for a visual team summary of the hits ("made shots=1", "missed shot=0" by team vertically and for each half is given and shown below.

Chart(Grouping(:Period, :TeamName), X(:Type, :HitMiss), Y(N(:PlayerName)), Bar Chart(1))

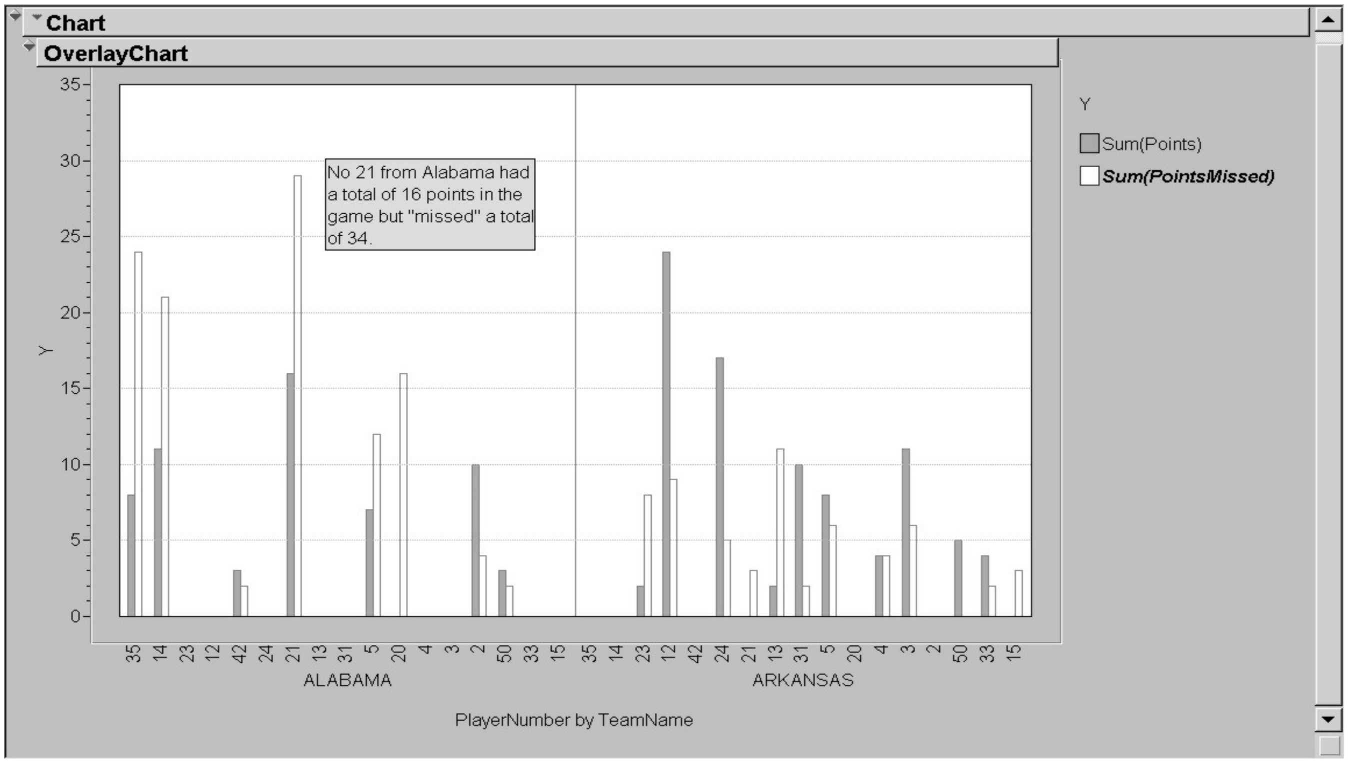


Individual player statistics of shot attempts (identified as HitMiss=1, for made and HitMiss=0 for missed shots) for the entire game are provided by the following script and the graphic that follows where the darkest color identifies shots made and the lighter colors are shots missed.

Chart(Grouping(:TeamName), X(:PlayerNumber, :HitMiss), Y(N(:PlayerNumber)), Separate Axes(1), Level[1] << Colors(28), Bar Chart(1), Y[1] << Overlay Pattern(4))



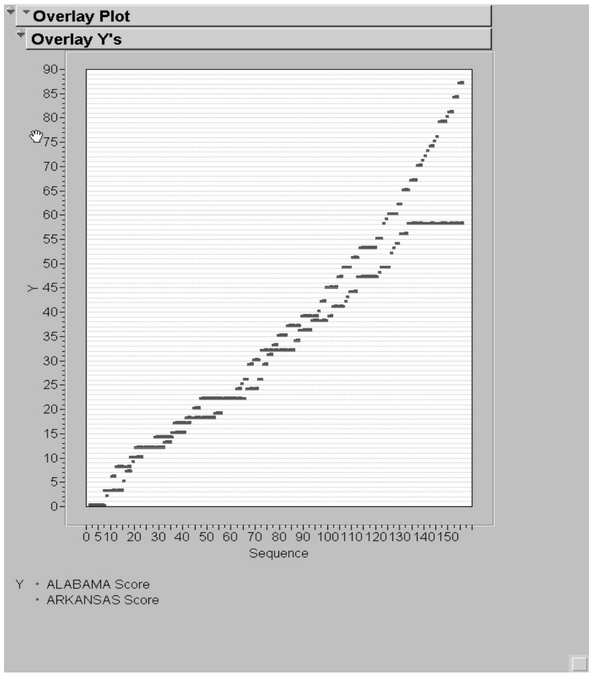
Individual player scoring statistical summary can be made to display not only the total points scored but the total points missed that gives a better idea about selecting players that had a good scoring night. On the graph below you can easily identified and contrast the two team leading scores: Number 12 for ARKANSAS (for scoring 24 point and missing 9 points as having a good scoring output) as opposed to Number 21 of Alabama (that scored 16 points but missed a total of 34 points).



GRAPH→OVERLAY

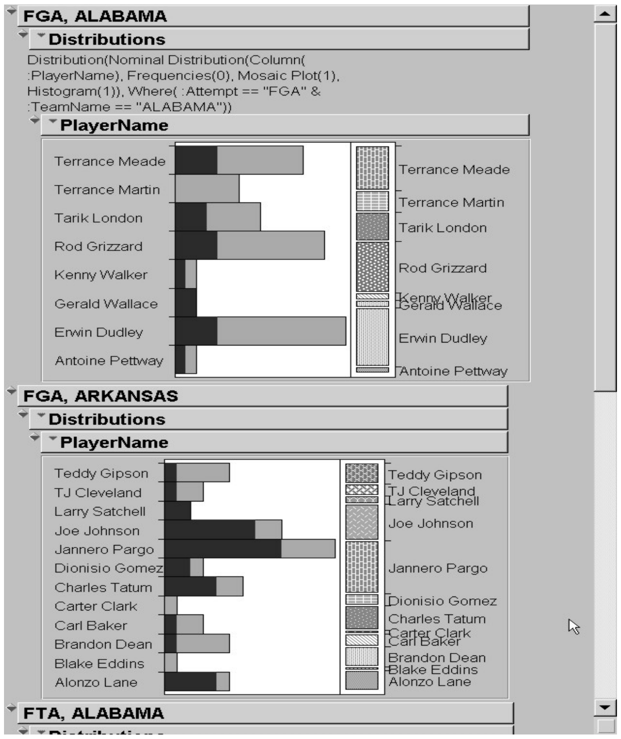
The visual of the scoring for each team as it progress in time can be shown by utilizing the pointing and selecting the overlay graph in JMP or by the script below that produces the following graph.

```
Overlay Plot(X( :Sequence), Y( :ALABAMA Score,
:ARKANSAS Score),  :ALABAMA Score(Connect Color(53),
Overlay Marker(0)),  :ARKANSAS Score(Connect Color(3),
Overlay Marker(0)))
```



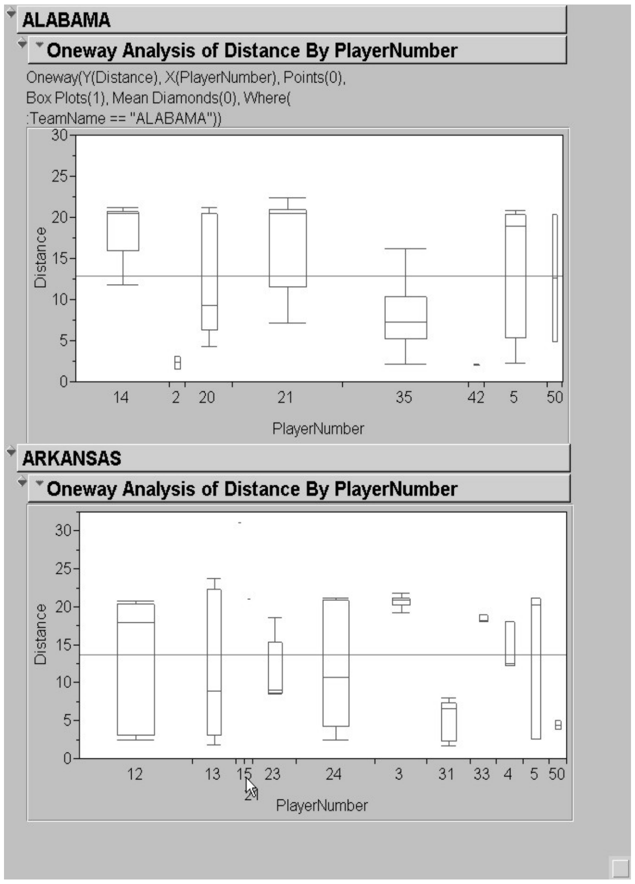
JMP ANALYSIS PLATFORMS

By selecting all rows that correspond to a made shot (1) the distribution of players by team provides us the following graph.



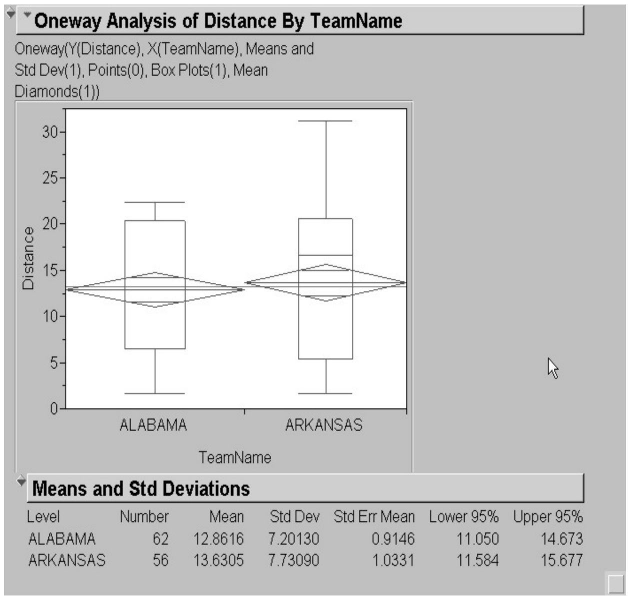
```
Distribution(Nominal Distribution(Column(
:PlayerName), Frequencies(0), Mosaic Plot(1),
Histogram(1)), Where( :Attempt == "FGA" &
:TeamName == "ALABAMA"))
```

Since the data are spatially referenced one can calculate and compare the average distance for all FGA taken by each player and team since this statistic is never given in the official box scores for each player and team as shown below. FTA are excluded.



```
Oneway(Y(Distance), X(PlayerNumber), Points(0),
Box Plots(1), Mean Diamonds(0), Where(
:TeamName == "ALABAMA"))
```

Again with FTA excluded we calculated the average distance from where each team shot the ball from the field. We can see in the graph below that ARKANSAS one average shot from about a little less than a foot father away from the basket.



```
Oneway(Y(Distance), X(TeamName), Means and
Std Dev(1), Points(0), Box Plots(1), Mean
Diamonds(1))
```

Means and Std Deviations

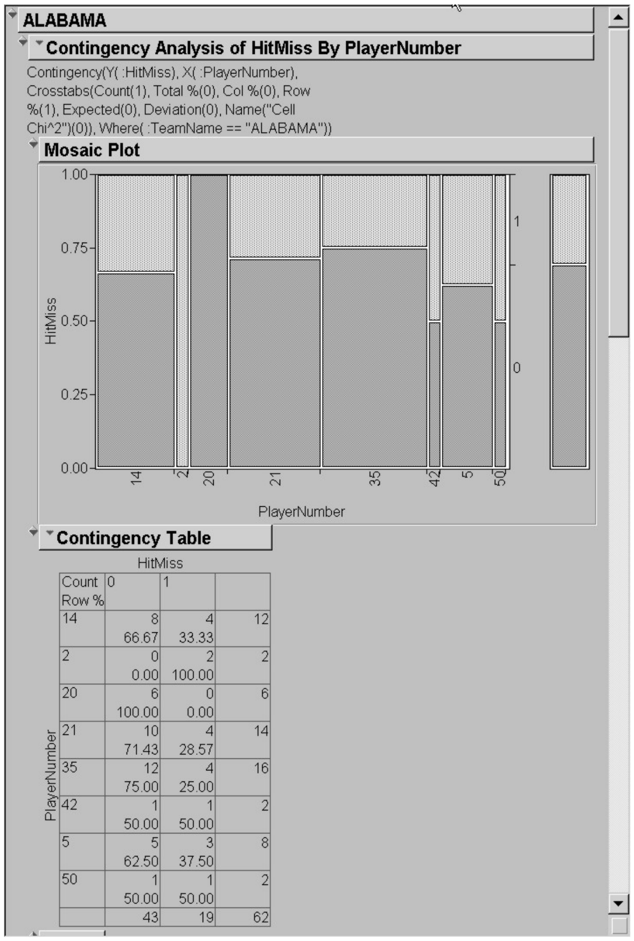
Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
ALABAMA	62	12.8616	7.20130	0.9146	11.050	14.673
ARKANSAS	56	13.6305	7.73090	1.0331	11.584	15.677

FIT Y BY X (CONTIGENCY)

The contingency table produce on the variable HitMiss as Y (changed into a nominal variable) by PlayerNumber as X can produce a visual that is usually found on the official box score for the game that identifies numbers of made FG to FGA and FT to FTA (not shown) as frequencies and as percentages. But again it is the Mosaic display that helps to tell the story of the numbers graphically since that is easier to interpret. Darker boxes indicate missed FGA and lighter boxes made FGA. The overall width of each player's box represents the percent of the team FGA taken by that particular player. For instance it is easy to identify player number 20 for Alabama since he missed all FGA and player number 35 as one missing the highest percentages (≥75%) of his FGA. The contingency script is given below

```
Contingency(Y( :HitMiss), X( :PlayerNumber),
Crosstabs(Count(1), Total %(0), Col %(0), Row %(1),
Expected(0), Deviation(0), Name("Cell Chi^2")(0)), Where(
:TeamName == "ALABAMA"))
```

A screen caption to the side displays parts of the output when executing this script. The same script can produce the ARKANSAS statistics when replacing the TeamName by ARKANSAS in the code above (output not shown).


ALABAMA

Contingency Analysis of HitMiss By PlayerNumber

```
Contingency(Y( :HitMiss), X( :PlayerNumber),
Crosstabs(Count(1), Total %(0), Col %(0), Row
%(1), Expected(0), Deviation(0), Name("Cell
Chi^2")(0)), Where( :TeamName == "ALABAMA"))
```

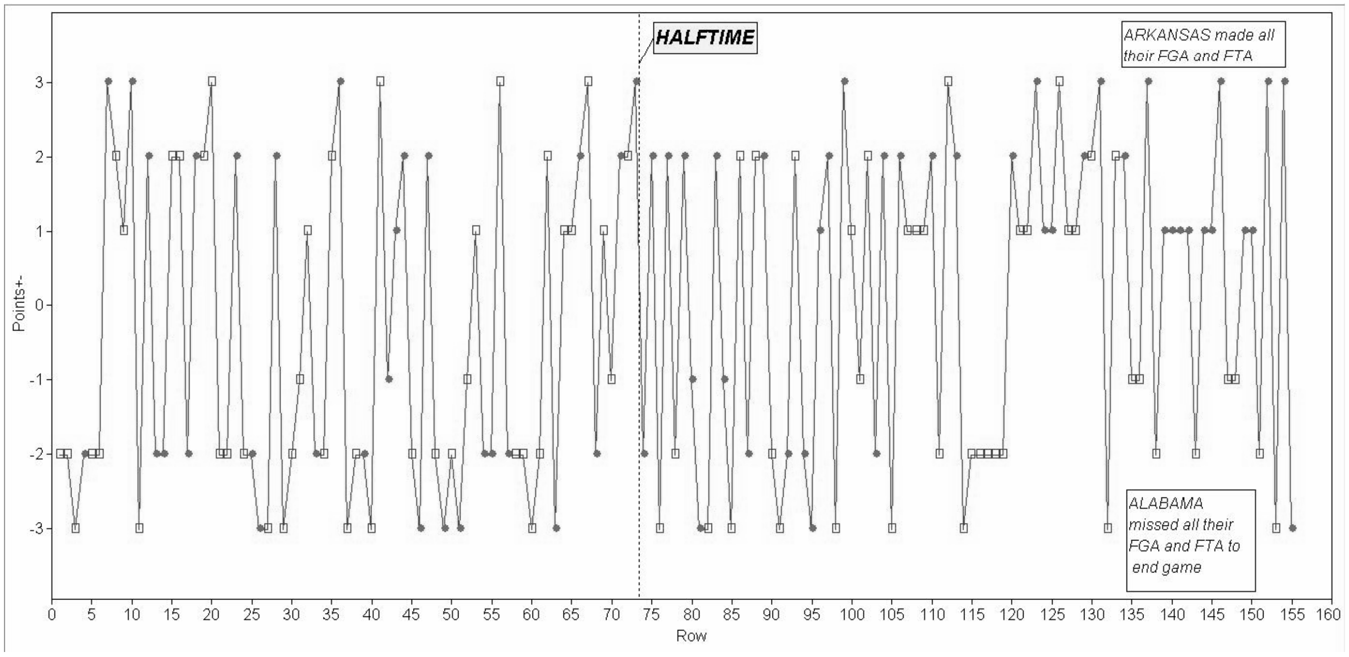
Mosaic Plot

Contingency Table

PlayerNumber	Count Row %	HitMiss 0	1	
	14	8 66.67	4 33.33	12
	2	0 0.00	2 100.00	2
	20	6 100.00	0 0.00	6
	21	10 71.43	4 28.57	14
	35	12 75.00	4 25.00	16
	42	1 50.00	1 50.00	2
	5	5 62.50	3 37.50	8
	50	1 50.00	1 50.00	2
		43	19	62

TIMES SERIES PLATFORM

A new variable Point+- was created to be Points + (-PointsMissed) and as such it takes values 3, 2, 1 and –3, -2, -1 for made and missed FGA and FTA respectively. The Time Series platform can be used to plot the entire game shots sequence to easily identify periods of offensive and defensive effectiveness/ineffectiveness. The time series plot below was annotated to show the halftime break and the periods where ARKANSAS ("filled circles") was very efficient offensively making all their FGA and FTA and at the same time ALABAMA ("empty squares") was missing all their FGA and FTA. We have changed the markers used and assigned by the bball script to create a better visual for the paper.



FUTURE UPDATES (VERSION 2)

Currently we are working on future enhancements of bball. One obvious enhancement will be to account for possession and defenses by populating the graphic on the side bellow its team name and score with button for Possessions and several on button options that identify the team who is defending that basket as one being: (M) for man-to-man defense, (Z) for zone, (P) for press and (T) for trap and by allowing various combination of these. We believe that a single user will still be able to collect that data while watching the game. The reason that we did not include other meaningful statistics that usually appear in the official box score such as rebounds assists steals and turnovers is that one individual could not keep all these in real time. Implementing these statistics can be fairly easy programming wise but the only way that the user could record all that information it that if he is watching the game on tape where he is able to slow it down, stop and review the play to record all these statistics.

CONCLUSION

JMP 4 has come a long way from just being an excellent exploratory discovery tool as seen by the exhaustive list of significant improvements listed in the introduction. The software has changed for the better not only in its functionality that is already impressive but also in its capability. We applaud the inclusion of JSL as a tool that can provide added functionality to the power user. We had fun learning and playing with JSL that we will help us in the future to build customized statistical programs not routinely found in JMP. The best approach to build such an application will be to use Visual Basic to program the application and interface it with JMP for calculating the various summaries and graphs.

REFERENCES

JMP Scripting Guide, Version 4, SAS Institute Inc. Cary, NC.

JMP Scripting Guide, Version 4, SAS Institute Inc. Cary, NC.

ACKNOWLEDGMENTS

Acknowledgments to Dr. H. Don Scott, University Professor of CSES at the University of Arkansas for conversations and ideas for future improvements.

CONTACT INFORMATION

Contact the authors at:

Author Name: Andy Mauromoustakos
Company: University of Arkansas
Address: Agricultural Statistics Laboratory, 101 AGRX
City State ZIP: Fayetteville, AR 72701
Work Phone: (501)575-5678 Fax: (501) 575-8643
Email: andym@uark.edu

To get the newest version of the bball.jsl:

Author Name: Kevin Thompson
Company: University of Arkansas
Address: Agricultural Statistics Laboratory, 101 AGRX
City State ZIP: Fayetteville, AR 72701
Work Phone: (501)575-2448 Fax: (501) 575-8643
Email: kthompsn@uark.edu