mosaicManip: Interactive Applets for Teaching with R

Andrew Rich,* Daniel Kaplan,* Randall Pruim,+ Nicholas J. Horton,× & J.J. Allaire°

Macalester College*, Calvin College+, Smith College×, and RStudio°

Objective

Even for those of us who favor a command-line interface for teaching, there are undeniably situations where the ability to interact graphically has profound advantages. Instructors often have ideas for simple interactive applications to help students develop a stronger understanding and challenge their misconceptions. Regrettably, the construction of such "applets" is beyond the programming skills of many otherwise creative instructors.

The `manipulate` package brings the ability to construct interactive applets in R to users with mainstream capabilities in programming.

Here we report on applets constructed with `manipulate` during a Google Summer of Code project. In addition to providing applets that instructors can use in teaching statistics and calculus, we hope to inspire instructors to imagine new applets and provide examples of programming styles that can be used to implement them.

Basic Manipulate Functionality

The `manipulate` package implements a simple paradigm for a graphical user interface:

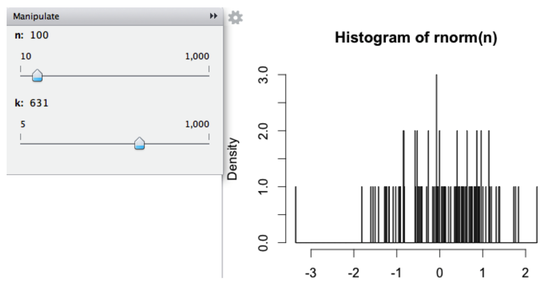
- The `manipulate( )` function creates an environment.
- GUI controls — sliders, checkboxes, pickers — set values of variables in the environment.
- An R expression is evaluated each time the GUI controls are set. Typically, the R expression produces graphics.

To illustrate, consider a simple R graphics command to draw a histogram of the output of a random number generator:

```
> hist( rnorm(200), breaks=10, freq=FALSE )
```

The `manipulate( )` function wraps this expression and allows fixed parameters — sample size of $n = 200$ and $k = 10$ breaks in the histogram — to be altered interactively.

```
> manipulate(
    hist(rnorm(n),breaks=k,freq=FALSE),
    n = slider(10,1000,step=10,initial=100),
    k = slider(5,1000,step=1,initial=10))
```

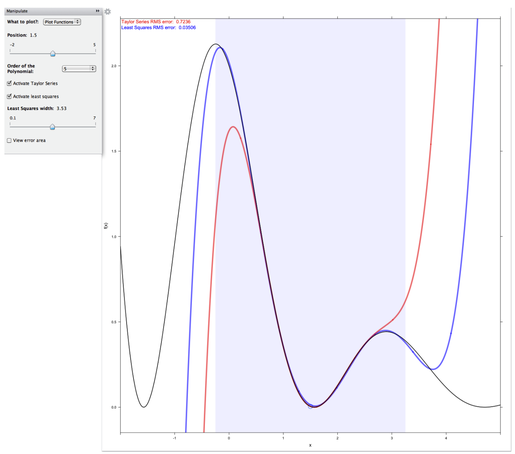


The `mosaicManip` Package

The `mosaicManip` package is currently in alpha testing — based on experiences with students next semester we will revise the applets before release as a CRAN package.

Project MOSAIC seeks to connect modeling, statistics, computation and calculus in the undergraduate mathematics curriculum. This goal is well illustrated by the `mTaylor` applet. The applet shows how a mathematical function can be approximated by a Taylor series; the user chooses the origin and the order of the series. To link calculus to statistics and modeling, `mTaylor` compares the Taylor series to a least-squares polynomial approximation over a finite region and displays the RMS error of both approximations. The goal is to develop concepts of smoothness, different ways to measure quality of approximation, and to introduce least-squares into the mainstream calculus curriculum.

```
> mTaylor( (1+cos(2*x))*exp(-x/2) ~ x )
```



The related `m2Fit` applet, shown to the right, allows students to visualize a 2nd-order polynomial least-squares approximation to a function of two variables.

The Authors:

The authors gratefully acknowledge support:

- Google Summer of Code (to A. Rich)
- U.S. National Science Foundation (DUE-0920350) for Project MOSAIC
- W.M. Keck Foundation (to D. Kaplan)
- Howard Hughes Medical Institute (to D. Kaplan, R. Pruim, and N. J. Horton)

A.J. Rich D.T. Kaplan R. Pruim N.J. Horton J.J. Allaire

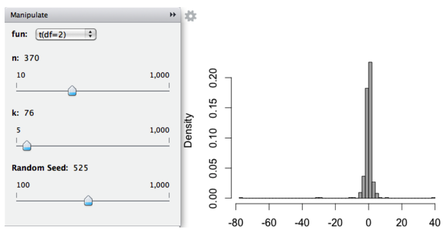
Contact Info:

- `www.mosaic-web.org`
- `rpruim@calvin.edu` & `kaplan@macalester.edu`
- `www.rstudio.org`

Example: A Simple `manipulate( )` Applet

Simple applets can often be implemented with one-line expressions coupled to GUI controls by `manipulate( )`. Often, initial success with a simple applet leads immediately to ideas for expanding its functionality. Here, we modify the "one-liner" graphics command from the prototype `hist` applet to produce an applet that allows the user to select from a set of random number generators, to maintain a random seed from one invocation to the next, and to allow the user to vary the seed.

```
# Create a function to implement the applet
histPlay = function(){
  # The drawing function takes inputs.
  plotfun = function(n=10,k=10,
                     seed=25,fun=rnorm) {
    set.seed(seed)
    hist(fun(n),breaks=k,freq=FALSE,
         col="gray",main="",xlab="x")
  }
  # Convenience definitions
  t2 = function(n){rt(n,df=2)}
  F2.3 = function(n){rf(n,2,3)}
  # Setting up the controls as a list.
  controls = list(
    fun=picker("Normal"=rnorm,
               "Exp"=rexp,
               "t(df=2)"=t2,
               "F(2,3)"=F2.3,
               "Log Normal"=rlnorm),
    n = slider(10,1000,step=10,initial=100),
    k = slider(5,1000,step=1,initial=10),
    seed = slider(100,1000,step=1,
                  initial=round(runif(1,120,980)),
                  label="Random Seed"))
  manipulate( # Invoke manipulate
       plotfun(n=n,k=k,seed=seed,fun=fun),
       controls)  }
```



This is certainly not a one-liner. But the structure is simple enough to be imitated for other applets or to be extended.

CAN YOU SEE HOW? Modify the `histPlay( )` applet to:

- Add a poisson($\lambda = 2$) density to the list of choices.
- Let the user set the color of the histogram.

`manipulate( )` is a feature of the RStudio interface. The `rpanel` package provides similar functionality via Tcl/Tk.

The `mosaic` Package

Project MOSAIC is a U.S. NSF-sponsored initiative to help strengthen ties among modeling, statistics, computation, and calculus in the undergraduate curriculum. The `mosaic` package for R seeks to enhance student development of sophisticated computing skills by providing a smooth path between elementary computations and more advanced, modeling-oriented computing. The goal — not yet completely realized — is to provide access to mathematical operations, including randomization and iteration, with a simple syntax that is faithful to mainstream R.

- Extends the syntax to make stronger connections between models and "basic statistics." Example:

```
> mean( wage ~ sex, data=CPS)
  sex        S   N Missing
1   F 7.878857 245       0
2   M 9.994913 289       0
```

- Adds simple operators to support sampling, simulations, randomization, bootstrapping. Example: a bootstrap standard error

```
> trials = do(500)*mean( wage ~ sex,
                         data=resample(CPS))
> sd( S ~ sex, data=trials)
  sex         S   N Missing
1   F 0.3134989 500       0
2   M 0.3034850 500       0
```

- Extending some common operators to provide pedagogical and graphical enhancements.
- Adding basic calculus functionality, e.g., differentiation

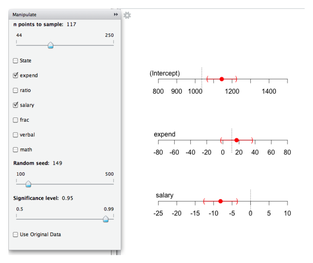
```
> f = D( a*x^k+2*y-b*x*y ~ x, a=3,k=7,b=2 )
> f(x=1,y=2)
[1] 17
> f
function (x, a = 3, k = 7, y, b = 2)
a * (x^(k - 1) * k) - b * y
```

and integration with symbolic-like properties

```
> g = antiD( dnorm(x, mean, sd) ~ x )
> g( from=-2,to=2,mean=0,sd=1)
[1] 0.9544997
> g( from=-Inf, to=5, mean=5, sd=19 )
[1] 0.5
> gprime = D( g(x, mean=a, sd=b) ~ x)
> gprime( 1, a=0, b=1 )
[1] 0.2419707
> dnorm( 1, 0, 1)
[1] 0.2419707
```

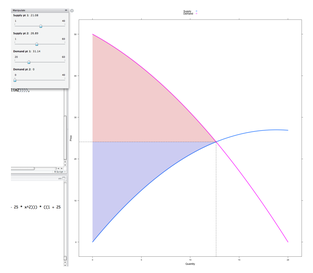
By using R syntax, by providing calculus operations that apply transparently to "novel" functions such as smoothers, and by adding randomization and iteration to the basic operators that students see, the `mosaic` package should help students see the connections between calculus and statistics.

The Google Summer of Code Applets in the `mosaicManip` package ...



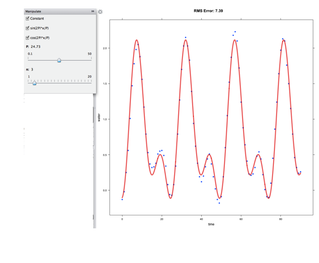
mBias: Demonstrates how confidence intervals cover "true" values and how model misspecification can create bias.

```
> mBias(sat ~ expend, data=Sat) )
```



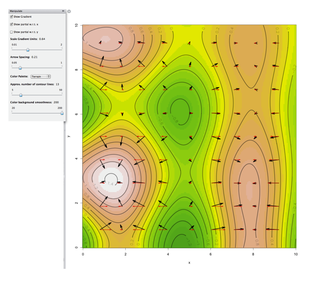
mEcon: Displays supply and demand curves and shows consumer and producer surplus. User can vary the shape of the curves.

```
> mEcon()
```



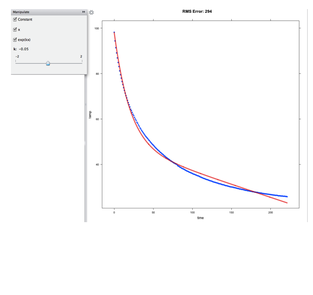
mFitSines: Fitting periodic functions to data. User controls the period of the fundamental and the number of harmonics.

```
> mFitSine(water ~ time, fetchData("hawaii.csv") )
```



mGrad: Displays partial derivatives and gradients of a function of two variables.

```
> mGrad(sin(x)*exp(-y/10)-cos(y)*exp(-x/4)~x&y)
```

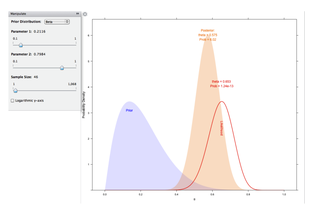


mFitExp: Fits exponential functions. User controls exponential constant.

```
> mFit( temp~time, fetchData("stan-data.csv") )
```

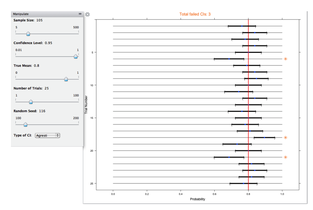
mSDGame: Displays a distribution selected at random. User guesses standard deviation. The guess is compared to the actual standard deviation.

```
> mSDGame()
```



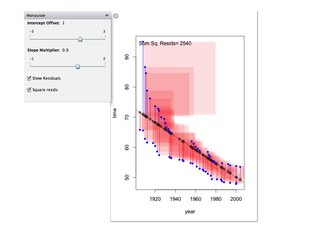
mBayes: Bayesian estimation of the probability in a Bernoulli trial. User supplies success/fail data, and sets prior using the controls. Demonstrates the relationship between the prior, likelihood, and posterior.

```
> mBayes( rbinom(50, p=.7, size=1) )
```



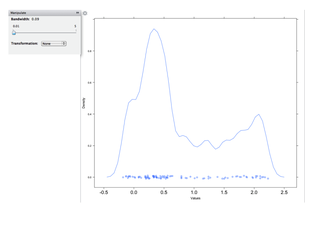
mCI: Construction of confidence intervals of sample proportion and interpretation in terms of coverage. User controls sample size and confidence level. The coverage is easily visualized by displaying multiple trials.

```
> mCI()
```



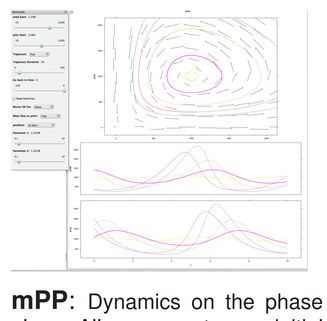
mLineFit: Illustrates the principle of least squares in fitting a line. User adjusts the slope and intercept; sum of square residuals are shown numerically and visually (as the amount of red ink).

```
> mLineFit( time ~ year, fetchData("swim100m.csv") )
```



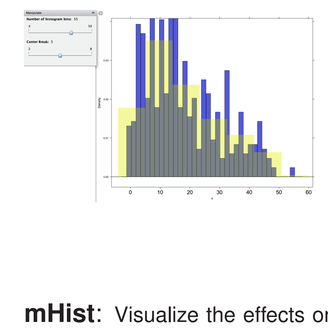
mDensity: Illustrates how the bandwidth parameter controls the appearance of a density estimator.

```
> mDensity(hawaii$water)
```



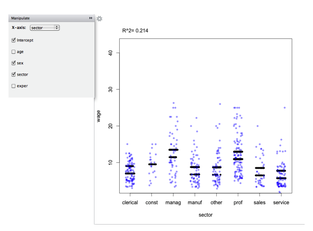
mPP: Dynamics on the phase plane. Allows users to vary initial points and view trajectories, solutions, nullclines, and flow fields.

```
> mPP( )
```



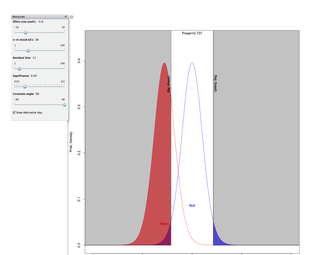
mHist: Visualize the effects on shapes of histograms due to the bin spacing and location.

```
> mHist( CPS$exper )
```



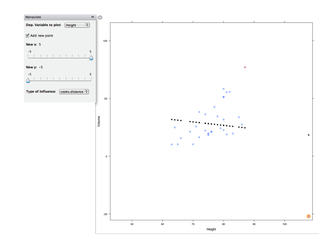
mLM: Shows the form of model values as different terms are included in a model. Changing the *x* variable on the display helps to develop a better understanding of relationships involving both categorical and quantitative variables.

```
> mLM( wage ~ sex + exper + sector, data=CPS ))
```



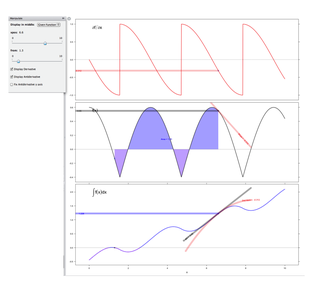
mHypTest: Shows how the power depends on the effect size (under the alternative hypothesis), the sample size, the size of residuals, and collinearity. Both an F-test and a t-test mode.

```
> mHypTest( )
```



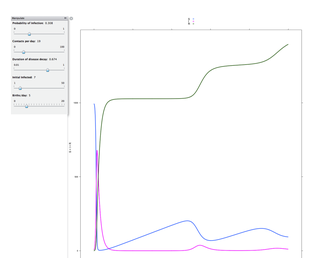
mInfluence: Explore how outliers can influence fits and how transformations can be used.

```
> mFit( temp~time, fetchData("stan-data.csv") )
```



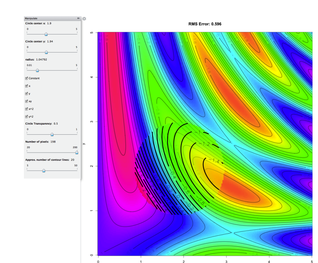
mDerivs: Demonstrates the relationship between a function, its integral, and its first few orders of derivative

```
> mDerivs(abs(cos(x))-.4 ~ x)
```



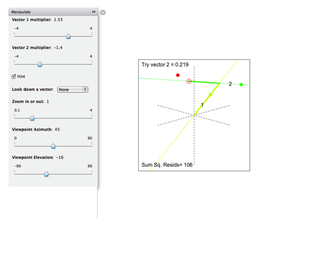
mSIR: The "tipping point" of an epidemic can be seen by adjusting transmission and duration parameters, as well as the introduction of new susceptibles.

```
> mSIR( )
```



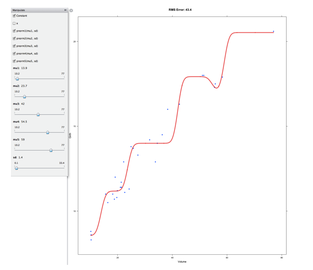
m2Fit: Demonstrates local fit using 2nd-order polynomial to a function of two variables. User selects location and radius of fitting region as well as the polynomial terms to include. The quality of the approximation is shown by the overlap of contours as well as the RMS error.

```
> m2Fit( sin(x*y)+cos(x) ~ x&y, xlim=c(0,5), ylim=c(0,5) )
```



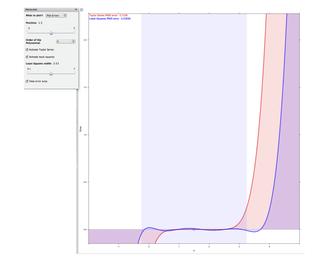
mLinAlgebra: Illustrates least squares fitting from a vector-space perspective. The user controls the multiplier on each coefficient and attempts to reach the "target."

```
> mLinAlgebra( c(1,2,3), c(4,0,-1), target=c(2,-5,4) )
```



mSigmoidal: Fitting sums of sigmoidal functions to data. User controls the number of sigmoids (overfitted here!) as well as transition slope and position (`pnorm` is the sigmoidal with parameters mean and sd).

```
> mSigmoidal( Girth  Volume, data=trees )
```



mTaylor: Compares least squares and Taylor series polynomial approximation. This view shows how the error depends on *x*. Note that Taylor is extremely good near the point, while least squares is better over a region.

```
> mTaylor( (1+cos(2*x))*exp(-x/2) ~ x )
```