Authoring Statistics Problems in LON-CAPA using R

Marianne Huebner and Stuart Raeburn, Michigan State University

October 30, 2014

LON CAPA is a web based software for class management, exam or homework questions that can be taken online or printed for a paper test [1]. Questions for these tests can be created in the authoring space. Questions are written in PERL but can be interfaced with R. This comes in very handy for statistics and probability questions.
In this document after the general comments about organization, sample code is given for questions arising in various learning objectives for an introductory statistics course (probability, inference, regression). Some of the code may be applicable to other types of questions. The tutorial *R input and output for authoring statistics problems in LON CAPA* by M. Huebner and S. Raeburn [2] describes the different types of output (scalars or arrays) in more detail. Typical types of responses (numerical or radiobutton) are explained in the example of confidence interval and hypothesis test for a population mean in the tutorial *Response options for authoring statistics problems in LON CAPA* [3].

Getting organized

In the authoring space it is useful to have folders for different statistics topics. For example, the following folders are appropriate for an introductory course on Statistics and Probability.

- Sampling
- Probability
- Data Description
- Inference
- Regression
- Categorical Data Analysis

A number of principles should be observed when authoring new problems. First a problem should have a *meaningful name*. It is hard to remember what the problem `ex1prob1` contains weeks later, but `binomial1`, `boxplot1`, `numerical_summary`, `lin_regression` are self-explanatory.[1] Once the problem coding and writing is finished and checked, the problem is *published* to make it available for use in

---

[1]There is some discussion about underscores, dots, or capitalization in names for R with a preference for dots and no capitalization among R users. However only underscores can be used in the problem names in LON CAPA.

courses. In order to do so, **Metadata** are created consisting of author name, title, subject, key words, and class level (100,200,300, 400, graduate). Keywords are searchable. The source code is closed by default. However if the source code is selected to be "open", then other users can view your code and learn from it, and create many more problems, or correct mistakes, if there are any. The *Metadata title* is different from the *problem name* used to first create the problem. It can be helpful adding the data source to the title name. For example, instead of "one sample t", the metadata title may be "one sample t-test (aspirin)" with a specific reference to the application used in the problem.

A couple of good programming habits should be kept in mind.

- Comments save time! Comments can be added to the script portion starting with a # sign.
- Add to comments the purpose of the problem, reference, author, date the problem was created,e.g. 2014-10-29.
- Give your variables meaningful names.
- Describe variables in comments.
- Do not overwrite an already defined variable.
- Add a sentence before a function explaining what it does. One-liners are best.
- Inside a function add comments that split it into smaller tasks.

LON CAPA problem structure

A LON CAPA problem is structured with a script for the coding of the problem, a text portion where questions can be asked and a response field. The response field below is a numerical response, where students can enter their answer. It is useful to specify the format, for example `format=3f` for the number 0.405. When this LON CAPA problem is converted to multiple choice question the options will be in the same format.

```
<problem>
<script type="loncapa/perl">
# In the script portion variables are defined and answers are
calculated
</script>

<startouttext/>
# In the text portion the problem is described.
<endouttext/>

<numericalresponse answer="$correct" format="3f"  id="11">
<textline/>
</numericalresponse>
</problem>
```

Numerical response questions can be uploaded as exam problems to the class page. If the problem set is marked as "exam", questions with numerical input are converted

to multiple choice questions. However this can result in choices for probabilities larger than one or negative. In this case it will be preferable to create the problem as multiple choice (*Radiobutton response*) and code *distractors* for the incorrect choices.

Authoring problems with a numerical response has been described in detail by M. Lucas [4]. This is also a good introduction to the structure of a LON CAPA problem.

Using R for authoring LON CAPA problems

The LON CAPA script for problems is written in Perl. However it interfaces with R using `&cas`, if the output is a scalar, and `&cas_hashref`, if the output is a vector, matrix, or a list. Reading R output is described in more detail in **Topic 2 (Probability)** below. `cas` stands for computer algebra system. The format for using R in the Perl script is `&cas("R", "R programming code;");` where only the last quantity in the R programming code is saved. For example,

```
$z = &cas("R", "x<-1; y<-1; x+y;");
```

will have 2 as the output, but it does not retain the variables x and y.

Using datasets or R functions that are only available in specific R packages is possible, when loading the corresponding library. For example, the 1840's dataset by physicist JD Forbes consisting of barometric pressure and boiling point data for different locations is available in the `library(MASS)` in R. The boiling point for the first locale is (degrees Fahrenheit):

```
$x = &cas("R","data(forbes); x<-forbes; x[1,1];", "MASS");
```

In general the format for the R-Perl interface is

```
&cas("R","R programming code", "package name");
```

The following R libraries are available as of 2014:

```
'alr3','boot','car','class','cluster','datasets','Hmisc','KernSmooth','
leaps','lmtest','MASS','methods','mgcv','nlme','nnet','qAnalyst',
'quadprog','rpart','SuppDists','spatial','splines','stats','stats4',
'survival','tseries','zoo'
```

Sampling and random seeds

When sampling from probability distributions in R, it is important to set the seed. This will make sure that students have the same problem with each try. If the random seed is not fixed, a new randomization is generated each time the student submits an answer.

```
# set seed
$seed = &random(1,200000000,1);

# generate uniformly distributed random numbers from Uniform(25,100).
```

```
$nsample= &roundto(&cas("R", "set.seed($seed);runif(1,min=25,
max=100);"), 0);
```

The seed can be incremented by 1 before the next call with `$seed ++` to avoid generating identical values.

Topics for introductory statistics courses include data description, probability, statistical inference, linear regression, categorical data analysis, and sampling design. Examples for each of these topics are described below. Sampling design is not included, since this involves generating random numbers for quantities such as sample size, parameter, or statistic. Generating random numbers is shown in some of the other examples and does not necessarily involve R.

1. Data description

Learning Objective: Students should be able to describe and present data and information (e.g. mean, median, standard deviation, plots and graphs).

Figures such as histograms or boxplots can be incorporated into a problem with questions about graphical data description. To do so one needs to first create the figure outside LON CAPA, and then upload the figure in a subfolder of the data description folder or a separate folder.

Including figures in a problem

In the script portion of the `<problem>` one can define the figure with the path to the directory where the figure is saved.

```
<script type="loncapa/perl">
$pic='/res/msu/YOURNAME/Figures/fig.png'
</script>
```

The the figure can be shown in the text as follows:

```
<startouttext />
Consider the following Figure:
<p>
<img align="middle" src="$pic" encrypturl="no" />
</p>
<endouttext />
```

Before the end of the problem `</problem>` it is necessary to point to the directory where the figure is stored:

```
<allow src="/res/msu/YOURNAME/Figures//$pic" />
```

Randomly selecting one figure from a group of figures

To create a selection of problems for individual students one can randomly choose a figure from a set of figures, and give the corresponding descriptors as choices in a

multiple choice question. For example, if four histograms are saved in the Figures subfolder that are of different shapes, the following code defines an array of four histograms and an array of four associated descriptors.

```
<script type="loncapa/perl">
$seed = 563293971;
$r = &random(0,3,1);
$idx = $r + 1;

@hist = ('hist1.png','hist2.png','hist3.png','hist4.png');
@types = ('unimodal',  'bimodal',  'skewed to the right',  'bimodal and
skewed to the left');

#Use &choose() to select image filename and corresponding description
$pic = '/res/msu/YOURNAME/Figures/'.&choose($idx,@histfigs);
$type = '&choose($idx,@types)' ;

# Remove the chosen set. The remaining options are incorrect.
@typewrongs = @types;
push(@typewrongs);
splice(@typewrongs,$r,1);
</script>
```

The selected figure is then incorporated into the problem with $pic. The radio button responses describing the shapes will then be $type (correct) and $typewrongs[0], $typewrongs[1],$typewrongs[2] (incorrect).

For numerical summaries of data, one can simulate a data set and then calculate mean, standard deviation, quartiles. Beware, when sampling data from a uniform distribution, the sample mean and sample median can be very close! To avoid this, one can sample from a skewed distribution. For example, for a list of integers with different means and medians, one can sample from two or more uniform distributions. Below $num1 data points are sampled from a Uniform distribution in the interval from $min1 to $max1, where $num1, $min1, $max1 are predefined.

```
($data1,$dump1) = &cas_hashref("R","set.seed($seed);list<-
round(sort(runif($num1,min=$min1,max=$max1)),0);");
@sample1 = &cas_hashref_array($data1);
```

Several such sample data sets are then combined and numerical summaries are calculated in R.

```
$datalist = join(',', @sample1, @sample2, @sample3);
$mean = &cas("R","x<-c($datalist); round(mean(x),1);");
```

In the text of the problem this list can then be called as follows.

```
<startouttext />
  Consider the following list of n=$n numbers:  
```

```
    $datalist
<endouttext />
```

Note that `$dump1` is a version of `$data1`, which can be used for debugging or for figuring out the variable names. A detailed example is shown in **Topic 3 (Inference).**

2. Probability

Learning Objective: Students should be able to use probability in modeling uncertainty (e.g. probability rules, random variables, expectation, variance).

Probability problems can span the understanding of events and combinatorics, conditional probability, (discrete) random variables, or working with probability distributions.

For conditional probabilities and events one can upload tree diagrams or Venn diagrams and ask questions about these. How to include figures is described in **Topic 1 (Data Description)**. Probability rules can be applied when a text problem with different probabilities is given. For example, $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$ can be calculated from $P(A), P(B)$, and $P(A \text{ and } B)$. No R is needed.

```
$pA  = &random(0.2,0.5, 0.01);    # probability of event A
$pB  = &random(0.7,0.9,0.01)-$pA;  # probability of event B
$pAB = &random(0.1,0.3,0.01);     # probability of event A and B
```

Importing vectors of numbers into R and calculations within R

A discrete probability distribution is imported into R and the expectation of the random variable is calculated.

```
$probdist = "$p1,$p2,$p3,$p4,$p5,$p6, $p7";
$xvalues = "1,2,3,4,5,6,7";
$mu = &roundto(&cas("R", "x<-c($xvalues); p<-c($probdist);
sum(x*p)"),2);
```

Reading R output: scalars or vectors

After reading the probability density function of a discrete random variable into R the cumulative distribution function F is calculated. The result is an array of values F(1), F(2), F(3), etc.

For the given probability distribution the cumulative distribution function is:

```
p<-c(0.1,0.2,0.3,0.05,0.35)
cumsum(p)

## [1] 0.10 0.30 0.60 0.65 1.00
```

The corresponding script in LON CAPA is:

```
$probdist = (0.1,0.2,0.3,0.05,0.35);
($cdfdata, $cdfdump) = &cas_hashref("R", "p<-c($probdist); cumsum(p)");
@cdfdata = &cas_hashref_array($cdfdata);
$cdf = join(',',@cdfdata);
```

The following code extracts the value F(3):

```
$z = &cas_hashref_entry($cdfdata,"2");
```

Or one can save all values of the cumulative distribution function and then choose F(3). Note the first element of the array is `$zz[0]` and corresponds to F(1).

```
@zz = &cas_hashref_array($cdfdata);
$x = 2;
$F3 = $zz[$x];
```

Learning Objective: Students should be able to use probability distributions in calculating probabilities (e.g. binomial, Poisson, normal, exponential, t distributions).

For a problem with the normal distribution one can randomly choose a mean and standard deviation and appropriate limits, and then calculate answers to different questions.

```
# generate mean and standard deviation of a normal distribution
$mu = &roundto(&cas("R", "set.seed($seed);runif(1,min=115, max=125);"),
0);
$sd = &roundto(&cas("R", "set.seed($seed);runif(1,min=15, max=25);"),
0);

# generate percentile to use for inverse normal
$pctile = &roundto(&cas("R", "set.seed($seed);runif(1,min=0.05,
max=0.2);"), 2);

# generate lower and upper bounds. Random lengths of intervals are
needed so that answers are different.
$cons = &roundto(&cas("R", "set.seed($seed);runif(1,min=1.2,
max=1.7);"), 1);
$low = $mu - $cons*$sd;
$high = $mu + $cons*$sd;

# lower tail probability P(X< $high)
$prob1 = &roundto(&cas("R", "pnorm($high, mean=$mu, sd=$sd)"),2);

# inverse normal P(X> $quant) = $pctile
$quant = &roundto(&cas("R", "qnorm(1-$pctile, mean=$mu, sd=$sd)"),0);
```

The answer for a normal distribution with randomly generated mean 123.72 and standard deviation 19and randomly generated upper limit is $P(X < 152.66) = 0.936$, For a randomly generated upper tail probability the value x for $P(X > x) = 0.16$ is $x = 142.27$.

3. Inference

Learning Objective: Students should be able to use data generated by random sampling to make inferences about populations using confidence interval and hypothesis testing methods and check the appropriateness of the methods.

Confidence intervals and hypothesis tests (one sample t procedure)

We can use the t test function in R to calculate a confidence interval and perform a hypothesis test. Here we need to be careful about extracting the output from an array. The R output is as follows.

```
##
##  One Sample t-test
##
## data:  x
## t = -3.672, df = 9, p-value = 0.005139
## alternative hypothesis: true mean is not equal to 1
## 90 percent confidence interval:
##  -1.1279  0.2892
## sample estimates:
## mean of x
##   -0.4194
```

For answers to various question about confidence intervals and hypotheses tests we may need to extract p-value, test statistic, degrees of freedom, or the value of the null hypothesis. The confidence level is predefined.

```
$clevel = 0.9;
($ttestdata,$ttestdump) = &cas_hashref("R","x<-c($xvalues); t.test(x,
mu=1, conf.level=$clevel);");
```

The output $ttestdata contains the results of the R calculations and $ttestdump is a readable version of $ttestdata, which can be written temporarily in the output text, e.g. `<startouttext />` $ttestdump `<endouttext />`, for debugging or identifying variable names.

```
$VAR1 = { 'statistic' => \{ 't' => '-6.45842' }, 'parameter' => \{ 'df'
=> '9' }, 'p.value' => '0.0001169849', 'conf.int' => \{ '1' => '-
0.9719675', '2' => '-0.1000325' }, 'estimate' => \{ 'mean of x' => '-
0.536' }, 'null.value' => \{ 'mean' => '1' }, 'alternative' =>
'two.sided', 'method' => 'One Sample t-test', 'data.name' => 'x' };
```

This helps to identify the structure and the variable names of the results of the t-test. Namely, this is a two-sided t-procedure with

- null hypothesis $H_0: \mu = 1$

- sample mean $\bar{x} = -0.536$

- test statistic `t=-6.45842`

- degrees of freedom `df=9`

- p value `p.value=0.0001169849`

- confidence interval $(-0.9719675, -0.1000325)$

We do not see the confidence level, so it is best to define it externally as shown in the code above.

```
$pval = &roundto( &cas_hashref_entry($ttestdata,"p.value"),4) ;
$tstat = &roundto(&cas_hashref_entry($ttestdata,"statistic", "t"),3);
$df = &cas_hashref_entry($ttestdata,"parameter", "df");
$mu0 = &cas_hashref_entry($ttestdata,"null.value", "mean");
```

There are two ways to extract the confidence interval limits. The upper and lower confidence limits can be extracted as scalars, or the interval can be saved as an array.

```
#scalars
$lcl = &roundto(&cas_hashref_entry($ttestdata,"conf.int", "1"),2);
$ucl = &roundto(&cas_hashref_entry($ttestdata,"conf.int", "2"),2);

#array
@CI = &cas_hashref_array($ttestdata,"conf.int");
```

If the confidence limits were saved in an array `@CI`, then the lower and upper confidence limits can be called in the text with LCL=`$CI[0]` and UCL=`$CI[1]`.

4. Regression

Learning Objective: Students should be able to examine relationships among variables using regression techniques and check the appropriateness of the regression models.

Simple linear regression

If the data are fixed, they can be read into R as described in **Topic 2 (Probability)**. However in the example below we generate explanatory and response variables for regression modeling.

```
# Initialize random seed
$seed = &random(1,200000000,1);

# Choose the number of data points to be generated
$nsample= &roundto(&cas("R", "set.seed($seed);runif(1,min=25,
max=100);"), 0);

# Set the mean and standard deviation of the normally distributed
random variables to be generated.
$mean= 0;
$sd= 1;
```

```
# generate x  values
$seed ++;
@x = ();
($xdata,$xdump)=&cas_hashref("R","set.seed($seed);list<-rnorm($nsample,
mean=$mean, sd=$sd);");
@x = &cas_hashref_array($xdata);
$xvalues = join(',',@x);

# Generate regression line y = a + b*x +error, with a positive or a
negative slope
$intcpt0=2;
$residualsd=0.2;
$slope0=&cas("R", "set.seed($seed);sample(c(-0.05,0.05),1);")

$seed ++;
@y = ();
($ydata,$ydump)=&cas_hashref("R","set.seed($seed);list<-
$intcpt0+$slope0*c($xvalues)+rnorm($nsample, mean=0,
sd=$residualsd);");
@y = &cas_hashref_array($ydata);
$yvalues = join(',',@y);
```

To generate the regression output we need to fit a linear regression model:

```
# generate regression output with the same x and y variables as above
($regdata,$regdump) = &cas_hashref("R","x<-c($xvalues);y<-
c($yvalues);fit<-lm(y~x);summary(fit)");

# And the correlation coefficient
$cor =&cas("R","x<-c($xvalues); y<-c($yvalues); round(cor(x,y),2);");
```

First we can use these data to make a scatter plot and overlay the estimated regression line. Here is the R output similar to what *gnuplot* produces below.
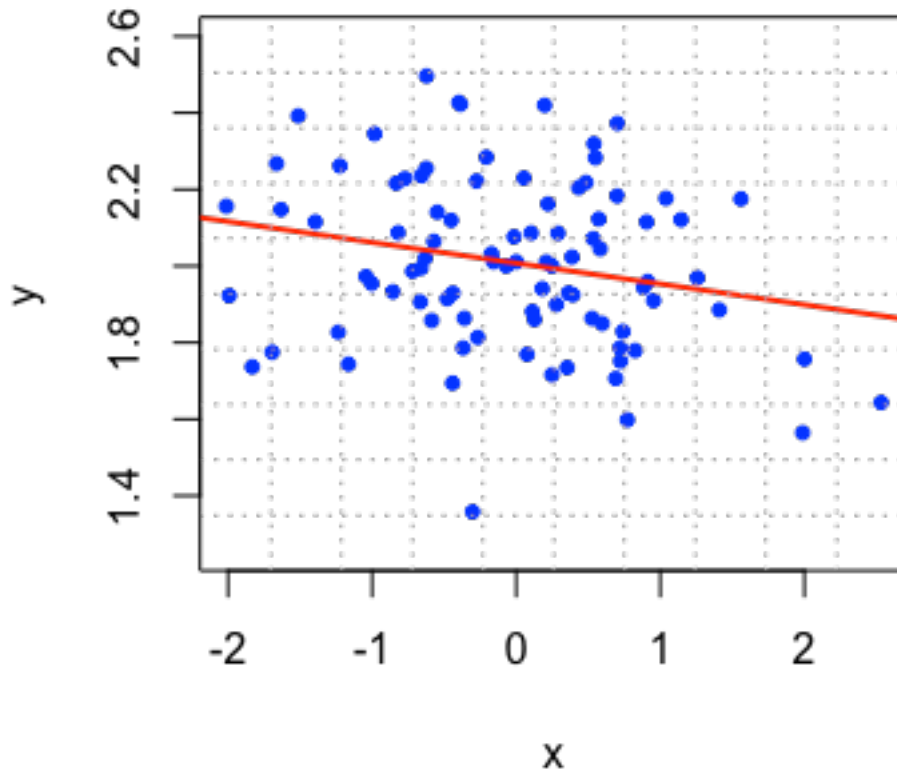
```
# generate regression line for plotting
($datalinreg,$dumplinreg) = &cas_hashref("R","x<-c($xvalues);y<-
c($yvalues);lm(y~x);");
@xx = &cas_hashref_array($datalinreg,"model","x");
@yy = &cas_hashref_array($datalinreg,"model", "y");

$xmin= &min(@xx);
$xmax= &max(@xx);
$len= 100;
$steps= ($xmax-$xmin)/100;

 for ($x = $xmin; $x<$xmax; $x+= 0.01) {
     push @xline, $x;
     push @yline, $slope*$x + $intcpt;
 }
```

In the text portion of the problem we can show the scatter plot with the regression line using *gnuplot*.

```
<gnuplot width="400" solid="0" plotcolor="monochrome" gridlayer="off"
bmargin="default" font="9" alttag="regression" bgcolor="xffffff"
texfont="22" transparent="off" plottype="Cartesian" rmargin="default"
gridtype="Cartesian" minor_ticscale="0.5" fontface="sans-serif"
grid="on" align="left" texwidth="93" height="300" border="on"
samples="100" fgcolor="x000000" major_ticscale="1" tmargin="default"
lmargin="default" fillstyle="empty">
    <title>Scatter plot with regression line</title>
    <curve linestyle="points" linetype="solid" color="x000080"
pointtype="7" limit="closed" pointsize="1" linewidth="2">
        <data>@xx</data>
        <data>@yy</data>
    </curve>
 <curve linestyle="lines" linetype="solid" color="xff0000"
pointtype="7" limit="closed" pointsize="1" linewidth="2">
        <data>@xline</data>
        <data>@yline</data>
    </curve>
    <xlabel>X</xlabel>
    <ylabel>Y</ylabel>
</gnuplot>
```

The computer output from a regression model includes the coefficients (intercept and slope), standard errors, t statistics, and p values for the coefficients. Below is the result in R.

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6661 -0.1393  0.0017  0.1681  0.4552
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.0075     0.0225   89.15   <2e-16 ***
## x            -0.0543     0.0250   -2.17    0.032 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.215 on 89 degrees of freedom
## Multiple R-squared:  0.0504, Adjusted R-squared:  0.0397
## F-statistic: 4.72 on 1 and 89 DF,  p-value: 0.0325
```

We can extract the relevant output as follows:

```
$slope=&cas_hashref_entry($regdata, "coefficients", "2", "1");
$slopeSE=&cas_hashref_entry($regdata, "coefficients", "2", "2");
$slopeTstat=&cas_hashref_entry($regdata, "coefficients", "2", "3");
$slopePval=&cas_hashref_entry($regdata, "coefficients", "2", "4");
```

```
$intcpt=&cas_hashref_entry($regdata, "coefficients", "1", "1");
$intcptSE=&cas_hashref_entry($regdata, "coefficients", "1", "2");
$intcptTstat=&cas_hashref_entry($regdata, "coefficients", "1", "3");
$intcptPval=&cas_hashref_entry($regdata, "coefficients", "1", "4");

$fstatistic=&cas_hashref_entry($regdata, "fstatistic", "value");
$fstatDF1=&cas_hashref_entry($regdata, "fstatistic", "numdf");
$fstatDF2=&cas_hashref_entry($regdata, "fstatistic", "dendf");

$rsquared = &cas_hashref_entry($regdata,"r.squared");
$sigma = &cas_hashref_entry($regdata,"sigma");
$df = &cas_hashref_entry($regdata,"df", "2");
```

The output can then be written in an html table to create a problem about reading computer output from a regression analysis.

5. Categorical Data Analysis

Learning Objective: Students should be able to evaluate associations between two categorical variables and interpret contingency tables.

In the following script cell counts for a 2 by 2 contingency table are generated according to a specified distribution, but with a randomly chosen sample size.

```
# Generate 2 by 2 table entries according to specified proportions
$t=&random(200,220,1);  #sample size
# Specified probabilities
$p11=0.2; $p12=0.25; $p21=0.15; $p22=0.4;
#generate cell counts
$a11=&roundto($t*$p11,0);
$a12=&roundto($t*$p12,0);
$a21=&roundto($t*$p21,0);
$a22=&roundto($t*$p22,0);
#due to possible rounding errors recalculate the sample size
$n=$a11+$a12+$a21+$a22;
```

Then an html table can be written in the problem statement with cell counts $a_{11}, a_{12}, a_{21}, a_{22}$.

Conclusion

An introductory statistics course has learning objectives for data description, probability, statistical inference, simple linear regression, and categorical data analysis. In this document we describe commonly arising questions how to interface R and Perl in authoring statistics problems for such topics.

References

[1] LON CAPA Documentations: http://www.lon-capa.org/documentation.html

[2] M. Huebner and S Raeburn. R input and output for authoring statistics problems in LON CAPA (2014). http://www.lon-capa.org/presentations/loncapa_dataR.pdf

[3] M. Huebner and S Raeburn. Response options for authoring statistics problems in LON CAPA (2014).

http://www.lon-capa.org/presentations/loncapa_responseR.pdf

[4] M. Lucas. Authoring Numerical Problems in LON CAPA: Step-By-Step. 2014. http://www.lon-capa.org/conf2014/AuthoringNumericalProblemsTutorialv0.2.pdf