Minimal R for Intro Stats

Randall Pruim, Project MOSAIC

July 19, 2017

"Less volume, more creativity."

Mike McCarthy, Head Coach, Green Bay Packers

Mike McCarthy had signs proclaiming his "Less volume, more creativity" mantra hung on the office walls of all of his coordinators during one off-season. When asked about it, he said, "A lot of times you end up putting in a lot more volume, because you are teaching fundamentals and you are teaching concepts that you need to put in, but you may not necessarily use because they are building blocks for other concepts and variations that will come off of that . . . In the offseason you have a chance to take a step back and tailor it more specifically towards your team and towards your players."

Statistics instructors using R face a similar dilemma. R is capable of so much that it is tempting to include this, and then that, and then the other, and then one more thing. Vectors and lists and recycling and coercion and functions and . . . It all seems so fundamental to the way R works. And when mastered, these concepts do become building blocks for other concepts and variations.

But when looking back at the end of a term, we have to admit that some of these things really aren't necessary to get the job done, and may do more harm than good for beginners. We too need to take a step back and tailor things toward our students and their abilities and needs. The colored commands on the next page are sufficient for an Introductory Statistics course that includes ANOVA, regression, and resampling techniques. The others are optional extras. This is followed by a 1-page sampler showing usage examples for some of the functions.

Note: These pages are intended as a guide for instructors, not as a reference card for students. Although they may also be useful for students, they would need supplementing with additional details.

The list of functions we present are not the only sufficient set of functions, but they were carefully chosen to fit as much as possible into a small number of paradigms. In particular,

1. We make use of the "formula interface" whenever possible.

   Students will need the formula interface to do regression an ANOVA. Since we are going to teach it anyway, we use formulas as consistently and as often as we can. In some cases, my colleagues and I have written new functions or expanded the use of existing functions to serve this end. These functions are available in the `mosaic` package and are indicated in the comments in our palette. Some of the data sets are from the `mosaicData` package.

2. We use `lattice` graphics.

   R has three separate high level plotting libraries (base, `lattice`, and `ggplot2`). Each has its advantages, but we choose `lattice` because it uses the same formula interface used elsewhere and because it encourages students to things about disaggregating data according to the values of covariates by making this very easy to do.

   Recently, we have introduced another package, `ggformula`, that provides a similar formula interface to `ggplot2`. We anticipate that this will be our graphics solution going forward. A separate vignette demonstrates how to use `ggformula` in place of `lattice`.

Whether you use this list or some other list, we encourage you to make a complete list of the commands you want your students to learn over the course of a semester. Organize them by topic. Organize them again by syntactic structure. Ask yourself how they look as a whole. Have you chosen a set of functions that fit well together? And most importantly: What is your creativity to volume quotient?

Help

```
apropos()
?
??
example()
```

Basic Calculations

Basic calculation works like a calculator.

```
# basic ops: + - * / ^ ( )
log(); exp(); sqrt()
log10(); abs(); choose()
```

Formula Interface

The following syntax (often with some parts omitted) is used for graphical summaries, numerical summaries, and inference procedures.

```
goal(y ~ x | z, data=...,
                groups=...)
```

For plots:

- `y`: is y-axis variable
- `x`: is x-axis variable
- `z`: conditioning variable (separate panels)
- `groups`: conditioning variable (overlaid graphs)

For other things:

'`y ~ x | z`' can usually be read '`y` is modeled by (or depends on) `x` differently for each `z`'.

See the sampler for examples.

Numerical Summaries

These functions have a formula interface to match plotting.

```
favstats()   # mosaic
tally()      # mosaic
mean()       # mosaic augmented
median()     # mosaic augmented
sd()         # mosaic augmented
var()        # mosaic augmented
diffmean()   # mosaic
```

```
quantile()   # mosaic augmented
prop()       # mosaic
perc()       # mosaic
rank()
IQR()        # mosaic augmented
min(); max() # mosaic augmented
```

Graphics (mostly lattice)

```
bwplot()
xyplot()
histogram() # mosaic augmented
densityplot()
freqpolygon() # mosaic
qqmath()
makeFun()   # mosaic
plotFun()   # mosaic
```

```
ladd()      # mosaic
dotPlot()   # mosaic
bargraph()  # mosaic
xqqmath()   # mosaic
```

```
mplot(HELPrct)
```

Randomization/Simulation

```
rflip()     # mosaic
do()        # mosaic
sample()    # mosaic augmented
resample()  # with replacement
shuffle()   # mosaic
rbinom()
rnorm()     # etc, if needed
```

Distributions

```
pbinom(); pnorm();
xpnorm()    # mosaic augmented
pchisq(); pt()
qbinom(); qnorm();
qchisq(); qt()
plotDist()  # mosaic
```

Inference

```
t.test()      # mosaic augmented
binom.test()  # mosaic augmented
prop.test()   # mosaic augmented
xchisq.test() # mosaic
fisher.test()
pval()        # mosaic
model <- lm() # linear models
summary(model)
coef(model)
confint(model) # mosaic augmented
anova(model)
makeFun(model) # mosaic
resid(model); fitted(model)
mplot(model)   # mosaic
```

```
mplot(TukeyHSD(model))
model <- glm() # logistic reg.
```

Data

```
nrow(); ncol(); dim()
inspect()             # mosaic
names()
head(); tail()
```

```
read.file()           # mosaic
with()
summary()
glimpse()             # dplyr
ntiles()              # mosaic
cut()
c()
cbind(); rbind()
colnames()
rownames()
relevel()
reorder()
```

```
rep()
seq()
sort()
rank()
```

Data Transformation

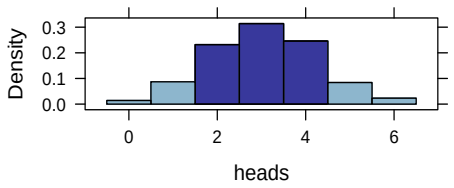
Even if students don't use these in a first course, instructors may use them to prepare data for student use.

```
select()      # dplyr
mutate()      # dplyr
filter()      # dplyr
arrange()     # dplyr
summarise()   # dplyr
group_by()    # dplyr
left_join()   # dplyr
inner_join()  # dplyr
```

```
rflip(6)


Flipping 6 coins [ Prob(Heads) = 0.5 ] ...

T H T H H T

Number of Heads: 3 [Proportion Heads: 0.5]

do(2) * rflip(6)

  n heads tails   prop
1 6     2     4 0.3333
2 6     1     5 0.1667

coins <- do(1000) * rflip(6)
tally(~heads, data = coins)

heads
  0   1   2   3   4   5   6
 14  87 232 314 246  84  23
```

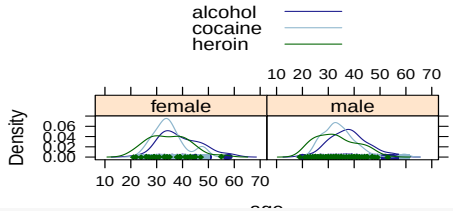
```
tally(~heads, data = coins, format = "perc")

heads
   0    1    2    3    4    5    6
 1.4  8.7 23.2 31.4 24.6  8.4  2.3

tally(~(heads >= 5 | heads <= 1), data = coins)

(heads >= 5 | heads <= 1)
 TRUE FALSE
  208   792
```

```
histogram(~heads, data = coins, width = 1,
    groups = (heads >= 5 | heads <= 1))
```

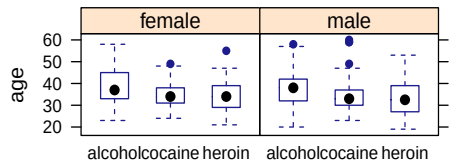



```
tally(sex ~ substance, data = HELPrct)

        substance
sex      alcohol cocaine heroin
  female      36      41     30
  male       141     111     94

mean(age ~ sex, data = HELPrct)

female   male
 36.25  35.47

diffmean(age ~ sex, data = HELPrct)

diffmean
 -0.7841

favstats(age ~ sex, data = HELPrct)

     sex min Q1 median   Q3 max  mean
1 female  21 31     35 40.5  58 36.25
2   male  19 30     35 40.0  60 35.47
     sd   n missing
1 7.585 107       0
2 7.750 346       0
```

```
densityplot(~age | sex, groups = substance,
    data = HELPrct, auto.key = TRUE)
```

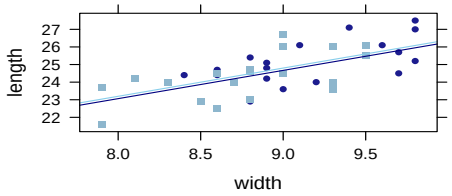



```
bwplot(age ~ substance | sex, data = HELPrct)
```




```
pval(binom.test(~sex, data = HELPrct))

  p.value
1.932e-30

confint(t.test(~age, data = HELPrct))

  mean of x lower upper level
1     35.65 34.94 36.37  0.95
```

```
model <- lm(length ~ width + sex,
            data=KidsFeet)
lmfunc <- makeFun(model)
lmfunc(width=8.25, sex="B")

   1
23.6

xyplot(length ~ width, groups=sex,
       data=KidsFeet)
plotFun(lmfunc(w, sex="B") ~ w,
        add=TRUE, col="skyblue")
plotFun(lmfunc(w,sex="G") ~ w,
        add=TRUE, col="navy")
# or plotModel(model)
```




```
plotDist("chisq", df = 4)
```