A Minimal R Cheatsheet

A couple dozen functions suffice to carry out your work in Quantitative Methods. This sheet provides the names of functions, a review of formula syntax, and some examples of use.

Help

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

Don't forget to load the `mosaic` package (and any others you might need):

```
library(mosaic)
```

Arithmetic

Basic arithmetic is very similar to a calculator. R supports all basic ops: `+ - * / ^ ( )`

```
log()    exp()  sqrt()
log10()  abs()  floor()
```

Randomization/Iteration

```
do()        # mosaic
sample()    # mosaic augmented
resample()  # with replacement
shuffle()   # mosaic
```

Graphics

```
bwplot()
xyplot()
densityplot()
histogram()
plotFun()  # mosaic
```

Numerical Summaries

The mosaic functions are able to use the *formula* interface.

```
mean()   # mosaic augmented
median() # mosaic augmented
sd()     # mosaic augmented
var()    # mosaic augmented
tally()  # mosaic
qdata()  # mosaic
pdata()  # mosaic
IQR()
```

Model Building and Inference

```
mm()  # mosaic
lm()  # linear models
glm() # for logistic models
resid()
fitted()
confint()
anova()
summary()
makeFun() # mosaic
listFun() # devel
```

Interactive

For classroom use.

```
mLM()
mLineFit()
mLinAlgebra()
mCI()
mHypTest()
mPower()
```

Formulas for Models

```
response ~ a+b # main effects
response ~ a*b # interaction, too
```

Do not use `|` or `groups`

Common forms:

```
# All cases the same:
response ~ 1
# Main effects & intercept:
response ~ X + Y
# Exclude intercept  (careful!):
response ~ -1 + X + Y
# Main effects & interaction:

response ~ X * Y
# Pure interaction:
response ~ X:Y
# Polynomial terms:
response ~ poly(X,2)
# Random model vectors:
response ~ rand(2)
```

Data and Variables

```
names()
head()
levels()
subset()
with()      # operate on data
transform() # new var in data
factor()    # categorical vars
merge()
```

Formulas for Graphs & Numerics

Plotting (e.g. `xyplot`, `densityplot`, `bwplot`) and simple numerics (e.g. `tally`, `mm`) use formulas in the following ways:

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

`y`: is y-axis variable. Leave blank for densityplots `x`: is x-axis variable `z`: conditioning variable (separate panes in graphs) `groups`: conditioning variable (overlaid in graphs)

For other things, `y ~ x | z` can usually be read as "`y` depends on `x` separately for each `z`".

Common Example Datasets

Can be used directly with `data=`:

```
Galton  # heights
CPS85   # wages
KidsFeet
Marriage
SAT
```

Read in with `fetchData()`:

```
u = fetchData("utilities.csv")
a = fetchData("alder.csv")
g = fetchData("grades.csv")
c = fetchData("courses.csv")
# Load software in development:
fetchData("m155development.R")
```

Quick Look at a Data Frame

```
cps = fetchData("CPS85")
names(cps)    nrow(cps)    summary(cps)

head(cps)     sample(cps,size=5)
```

Tallying

A simple count of the number in each level

```
tally( ~ sex, data=CPS85)
```

A two-way table of counts

```
tally( ~ sex + married, data=CPS85)
```

Conditional proportions: A `|` B means "A conditioned on B".

```
tally( ~ sex | married, data=CPS85)
```

Different from `married | sex`.

New Dataframe Variable

```
g = fetchData("Galton")
```

Add a variable named `mid`

```
g = transform(g, mid=(father+1.08*mother)/2)
names(g)  #confirm that it's there
```

Subsets

Sometimes you want only part of a data set.

```
boys = subset(KidsFeet, sex=="B")
elig = subset(CPS85, wage>10 & married=="Single")
pros = subset(CPS85, sector %in% c("prof","manag"))
```

...Random subset

```
subset(CPS85,size=4)
```

Data from Google Spreadsheets

In Google, choose `File` → `Publish to the Web`. Get link to the published data as CSV, sheet 1. Copy the link.

```
mydat = fetchGoogle("https://docs.google...")
```

Distributions

Simple distribution

```
densityplot( ~ age, data=CPS85 )
```

Overlaying two (or more) groups

```
densityplot( ~ age, groups=sex, data=CPS85, auto.key=TRUE)
```

Side-by-side plots:

```
densityplot( ~ age | sex, data=CPS85 )
```

```
bwplot( age ~ sector, data=CPS85 )
```

Scatterplots

```
xyplot( wage ~ age, data=CPS85 )
```

`groups=` and `|` work as with `densityplot()`.

Plotting Model Values

```
mod = lm(wage ~ educ+sex,data=CPS85 )
xyplot(fitted(mod) ~ educ,data=CPS85 )
xyplot(wage+fitted(mod) ~ educ,data=CPS85)
```

Extract Model Information

```
mod = lm(wage~educ+sex,data=CPS85)
coef(mod)
fitted(mod)
resid(mod)
f = makeFun(mod) # model function
```

P's and Q's

Want to find the value that separates the lower 30% from the higher 70%:

```
qdata(0.3, wage, data=CPS85 )
```

Have a value and want to find what fraction of the cases are at or below the value:

```
pdata(10, wage, data=CPS85 )
```

Randomization

Sample:

```
mysamp=sample(CPS85, size=100)
```

Resample:

```
lm(wage~educ, data=resample(CPS85))
```

Shuffle (for hypothesis testing):

```
lm(wage~shuffle(educ), data=CPS85)
```

Probability distributions:

```
rnorm(10, mean=25, sd=2)
rbinom(10, prob=.5, size=40)
rpois(10, lambda=50) # events per period
rexp(10, rate=0.01) # 1/ave time btw events
```

Confidence Intervals

... via "normal theory"

```
mod = lm(wage ~ educ, data=CPS85)
confint(mod)
```

See also `summary(mod)`.

... via bootstrapping

```
s = do(500)*lm(wage~educ, data=resample(CPS85))
sd(s) # standard error
```

See also `confint(s)`

Quantitative → Categorical

```
bwplot(height ~ factor(nkids), data=Galton)
```

Sums of Squares, Dot Products

```
sum(fitted(mod)^2)
sum(resid(mod)^2)
with(data=CPS85, sum(wage^2))
sum(fitted(mod)*resid(mod)) # dot prod
```

Something is Wrong

```
run = fetchData("repeat-runners.csv")
mean(net, data=run)
```

Some of the data was missing, thus the `NA`.

The FIX:

```
options(na.rm=TRUE)
mean( net, data=run )
```

This "cheatsheet" is based directly on teaching material from Statistical Modeling: A Fresh Approach (https://github.com/dtkaplan/Statistical-Modeling-A-Fresh-Approach) by R. Pruim & D. Kaplan. However, any questions, comments, and/or concerns should be directed to me (mailto:carsonfarmer@gmail.com).