The Briefest R overview, Ever

Paul Johnson[1]

[1]Center for Research Methods and Data Analysis

2019

CENTER FOR RESEARCH METHODS & DATA ANALYSIS
College of Liberal Arts & Sciences

Outline

This is Brief

- This talk introduces the vital R (R Core Team, 2017) terminology and usage necessary to get started with structural equation modeling
- Lets assume you have R and Rstudio already installed. If you don't, hurry up (Windows, Mac)
- We can't make you R experts in 1 hour, but
    - if you knew R before, you might remember
    - if you never used R before, this is an interesting way to start

Extract our Zip file

- **If you did not extract our zip folder yet, please do so now** (In Win10, one can be fooled by the File Explorer. Do please drag the folder out of the zip)
- Use File manager to look for the folder sem-2/sem-2-1-R.
- We write 1 file (sem-2-1.R.lyx). It creates
  - sem-2-1-R.pdf
  - sem-2-1-R.R

What you should do

- DO NOT LAUNCH R or Rstudio from the program/applications menu
- Instead, use your file manager, navigate to sem-2/sem-2-1-R
- Find sem-2-1-R.R, and use "open with" on that file. Choose Rstudio.
  - If you prefer another IDE (Notepad++, Emacs, Eclipse, OK!. On Windows, don't choose R)
- Our R file has "code chunks" that parallel the examples below.

All data analysis consists of 6 steps

1. Data import
2. Recoding
3. Exploration
4. Analysis
5. Export of Tables & Graphs
6. Writeup

Data Input Formats

- Base R includes importers for some data types
- Addon packages packages exist and can open various other types (with varying degrees of success)
  - SPSS, Stata, SAS
  - Excel

Check which data files we have in ”data”

- Check what files we provided for you in our data directory

```
ddir <- "data"
list.files(ddir)
```

```
[1] "affect.csv"
```

look in neighboring folder “data”

- Aside: R can create directories (`dir.create()`), copy files (`file.copy()`), etc.

Use read.table to import the csv file

```
fn <- "affect.csv"
affect <- read.table(file.path(ddir, fn), header
    = TRUE, sep = ",", stringsAsFactors = FALSE)
```

- First argument is a file name. Note, I'm using the R function `file.path` which joins together the data directory and the file name.
- 3 named arguments:
  - `header = TRUE`: use the first row as variable names
  - `sep = ","`: use the comma as the separator
  - `stringsAsFactors = FALSE`: Leave character variables as characters. Do not turn them into labeled discrete variables (R factors)

Check the result

- That thing is a data.frame object

```
str(affect)
```

```
'data.frame': 380 obs. of  19 variables:
 $ Agency1  : num  3.5 2.5 1.83 2.77 3.17 ...
 $ Agency2  : num  4 3.17 2 3.06 3.33 ...
 $ Agency3  : num  4 3 1.5 2.36 2.83 ...
 $ Intrin1  : num  4 3.21 3 3.13 3.5 ...
 $ Intrin2  : num  4 2 3 4 4 2.5 3.5 3 2 3.5 ...
 $ Intrin3  : num  4 3 2 3 4 3 4 2 3 3 ...
 $ Extrin1  : num  1 1.83 1 1.08 1.83 ...
 $ Extrin2  : num  1 2.67 1 1.17 2 ...
 $ Extrin3  : num  1.5 1.83 1 1 1.83 ...
 $ PosAFF1  : num  4 3 3.02 3 3.78 ...
 $ PosAFF2  : num  4 3.5 2.5 2.5 3.5 3 2.5 2 3 3 ...
 $ PosAFF3  : num  4 2.5 3 3 3 3 3 2.5 3.5 3 ...
 $ NegAFF1  : num  1 1.5 1 2.5 2.5 2 1 1 2 2.5 ...
 $ NegAFF2  : num  1 1.69 1 2.5 2 ...
 $ NegAFF3  : num  1 1.5 1 1.5 3 ...
 $ Sex      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ gender   : chr  "male" "male" "male" "male" ...
 $ ethnicity: chr  "Hispanic" "White" "White" "White" ...
 $ race     : chr  "Nonwhite" "White" "White" "White" ...
```

Check the result ...

- data.frame: columns can be different types of variables
  - character: character strings
  - integer: only integers, no floating point
  - numeric: floating point
- Other types we don't see here
  - logical: Coded either `TRUE` or `FALSE`, symbols that are interpreted as 1 and 0
  - factor: R's way of creating categorical variables, either nominal or ordered
  - Date: Can subtract dates to find time between
- The same information can be encoded in different ways
  - Sex is an integer
  - Gender is a character variable
- Can see in spreadsheet-like thing with the `View()` function:

```
View(affect)
```

R factor

- In R, the term "factor" is used for a categorical variable that has "internal integer values" but those values display as "labeled levels".

genderf	internal integer	label
	1	"male"
	2	"female"

- Here we create a new factor variable "affect$genderf" by pulling in affect$gender telling it which levels we want, in what order.

```
affect$genderf <- factor(affect$gender, levels =
    c("male", "female"))
```

- Key elements
  - Creates a new column inside `affect` (there are several other ways to do this)
  - The function `factor()` creates a factor

R factor ...

- Check that gender and genderf are different things, but represent same information
  The table function can create a quick cross-tabulation:

```
table("genderf" = affect$genderf, "gender" =
    affect$gender)
```

```
        gender
genderf  female male
  male        0  195
  female    185    0
```

Notice the table output is more sparse if we don't include names for the arguments:

```
table(affect$genderf, affect$gender)
```

```
         female male
  male        0  195
  female    185    0
```

R factor ...

- Can use jazzier names if you like

```
table("gender as factor" = affect$genderf, "Sex
    as an integer" = affect$Sex)
```

```
                  Sex as an integer
gender as factor    1   2
          male    195   0
          female    0 185
```

- I'll also need an ethnicity factor variable in a later section:

```
affect$ethnicityf <- factor(affect$ethnicity)
```

I allowed R to create the levels in alphabetical order, as we see here:

```
table("ethnicity factor" = affect$ethnicityf,
    "ethnicity" = affect$ethnicity)
```

R factor ...

```
                ethnicity
ethnicity factor Asian Black Hispanic White
        Asian       38     0        0     0
        Black        0    19        0     0
        Hispanic     0     0       67     0
        White        0     0        0   256
```

- Worth mentioning: R is case sensitive
    - If I create variables, they always start with small letters
    - This input data used capital and small letters, regrettably.

About packages

- R is a computational engine
  - to which packages are attached
- The R distribution includes
  - 15 base packages (incl. base, datasets, stats, stats4, graphics)
  - 15 recommended packages (incl. foreign, MASS, mgcv, nlme, survival)
- The Comprehensive R Archive Network (CRAN) has 12K other "contributed" packages.

Please install a couple of packages

```
CRAN <- "http://rweb.crmda.ku.edu/cran"
KRAN <- "http://rweb.crmda.ku.edu/kran"
options(repos = c(KRAN, CRAN))
install.packages(c("kutils", "rockchalk"), dep =
    c("Depends", "Imports", "LinkingTo"))
```

This specifies CRMDA server KRAN first, so if we have updates they are found, but then it also looks on the more general CRAN network.

- When you run `install.packages()`, R may ask if you if it can create a personal package repository for you. Generally, the answer is "yes".
- About the packages
  - kutils: data management functions created at CRMDA
  - rockchalk: regression functions

Package function example: Data Descriptions

```
summary(affect)
```

```
    Agency1          Agency2          Agency3          Intrin1
 Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
 1st Qu.:2.052   1st Qu.:2.167   1st Qu.:2.167   1st Qu.:2.500
 Median :2.494   Median :2.500   Median :2.500   Median :3.000
 Mean   :2.442   Mean   :2.550   Mean   :2.544   Mean   :3.002
 3rd Qu.:2.833   3rd Qu.:2.898   3rd Qu.:2.833   3rd Qu.:3.500
 Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000
    Intrin2          Intrin3          Extrin1           Extrin2
 Min.   :1.000   Min.   :1.000   Min.   :0.9717   Min.   :1.000
 1st Qu.:2.500   1st Qu.:2.500   1st Qu.:1.3190   1st Qu.:1.185
 Median :3.000   Median :3.000   Median :1.5000   Median :1.538
 Mean   :2.987   Mean   :3.080   Mean   :1.6151   Mean   :1.686
 3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:1.8333   3rd Qu.:2.000
 Max.   :4.025   Max.   :4.077   Max.   :3.5215   Max.   :3.500
    Extrin3           PosAFF1          PosAFF2          PosAFF3
 Min.   :0.9548   Min.   :1.000   Min.   :1.000   Min.   :1.000
 1st Qu.:1.1667   1st Qu.:2.744   1st Qu.:2.500   1st Qu.:2.500
 Median :1.5000   Median :3.023   Median :3.000   Median :3.000
 Mean   :1.6333   Mean   :3.136   Mean   :2.991   Mean   :3.069
 3rd Qu.:1.9320   3rd Qu.:3.500   3rd Qu.:3.500   3rd Qu.:3.500
 Max.   :3.8397   Max.   :4.000   Max.   :4.000   Max.   :4.000
    NegAFF1           NegAFF2          NegAFF3             Sex
```

Package function example: Data Descriptions ...

```
 Min.   :0.8845   Min.   :0.864   Min.   :0.9186   Min.   :1.000
 1st Qu.:1.0000   1st Qu.:1.000   1st Qu.:1.0000   1st Qu.:1.000
 Median :1.5000   Median :1.495   Median :1.5000   Median :1.000
 Mean   :1.7007   Mean   :1.527   Mean   :1.5448   Mean   :1.487
 3rd Qu.:2.0000   3rd Qu.:2.000   3rd Qu.:2.0000   3rd Qu.:2.000
 Max.   :4.0000   Max.   :4.000   Max.   :4.0000   Max.   :2.000
    gender             ethnicity             race
 Length:380         Length:380         Length:380
 Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character



   genderf        ethnicityf
 male  :195    Asian    : 38
 female:185    Black    : 19
               Hispanic: 67
               White    :256
```

Data Descriptions

```
library(rockchalk)
summarize(affect)
```

```
Numeric variables
         Agency1  Agency2  Agency3  Intrin1  Intrin2  Intrin3  Extrin1  Extrin2  Extrin3
min       1        1        1        1        1        1         0.972   1         0.955
med       2.494    2.500    2.500    3        3        3         1.500   1.538     1.500
max       4        4        4        4        4.025    4.077     3.522   3.500     3.840
mean      2.442    2.550    2.544    3.002    2.987    3.080     1.615   1.686     1.633
sd        0.516    0.527    0.546    0.769    0.852    0.770     0.481   0.577     0.571
skewness  0.141   -0.036    0.068   -0.299   -0.464   -0.460     0.988   0.925     1.211
kurtosis  0.017    0.174    0.207   -0.770   -0.659   -0.537     0.753   0.360     1.559
nobs     380      380      380      380      380      380       380     380       380
nmissing  0        0        0        0        0        0         0       0         0
         PosAFF1  PosAFF2  PosAFF3  NegAFF1  NegAFF2  NegAFF3    Sex
min       1        1        1        0.884    0.864    0.919     1
med       3.023    3        3        1.500    1.495    1.500     1
max       4        4        4        4        4        4         2
mean      3.136    2.991    3.069    1.701    1.527    1.545     1.487
sd        0.668    0.685    0.707    0.714    0.663    0.653     0.500
skewness -0.410   -0.234   -0.447    1.246    1.507    1.409     0.052
kurtosis -0.504   -0.455   -0.358    1.457    2.091    1.686    -2.002
nobs     380      380      380      380      380      380       380
nmissing  0        0        0        0        0        0         0

Nonnumeric variables
          gender            ethnicity                    race               genderf
  female: 185          Asian   : 38          Nonwhite: 124          male  : 195
  male  : 195          Black   : 19          White   : 256          female: 185
                       Hispanic: 67
                       White   : 256
```

Data Descriptions ...

```
nobs          : 380    nobs          : 380.000 nobs          : 380.000 nobs          : 380
nmiss         :   0    nmiss         :   0.000 nmiss         :   0.000 nmiss         :   0
entropy       :   1    entropy       :   1.374 entropy       :   0.911 entropy       :   1
normedEntropy:   1    normedEntropy:   0.687 normedEntropy:   0.911 normedEntropy:   1
              ethnicityf
Asian    : 38
Black    : 19
Hispanic: 67
White    : 256
nobs          : 380.000
nmiss         :   0.000
entropy       :   1.374
normedEntropy:   0.687
```

Outline

The lm function

- Linear regression uses lm()

```
summary(lm(PosAFF1 ~ genderf, data = affect))
```

```
Call:
lm(formula = PosAFF1 ~ genderf, data = affect)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1522 -0.4087 -0.1134  0.3804  0.8804

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.11964    0.04786  65.178   <2e-16 ***
genderffemale  0.03260    0.06860   0.475    0.635
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6684 on 378 degrees of freedom
Multiple R-squared:  0.0005973, Adjusted R-squared:  -0.002047
F-statistic: 0.2259 on 1 and 378 DF,  p-value: 0.6349
```

- Did you see some output whir past you?

The lm function ...

- Instead, we save the output into an object named “m1” (our choice)

```
m1 <- lm(PosAFF1 ~ genderf, data = affect)
summary(m1)
```

```
Call:
lm(formula = PosAFF1 ~ genderf, data = affect)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1522 -0.4087 -0.1134  0.3804  0.8804

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.11964    0.04786  65.178   <2e-16 ***
genderffemale  0.03260    0.06860   0.475    0.635
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6684 on 378 degrees of freedom
Multiple R-squared:  0.0005973, Adjusted R-squared:  -0.002047
F-statistic: 0.2259 on 1 and 378 DF,  p-value: 0.6349
```

The lm function ...

- summary() is a generic function, there are different "implementations" customized to the different types of inputs
- What other follow-up functions might be used?

  anova Stat tests to compare models (F, or χ^2)

  predict obtain predicted values, either for observed cases or hypothetical inputs

  resid display residuals

  plot regression diagnostics

Add more predictors

```
m2 <- lm(PosAFF1 ~ genderf + Agency1, data =
    affect)
```

"+" sign serves obvious role

```
summary(m2)
```

```
Call:
lm(formula = PosAFF1 ~ genderf + Agency1, data = affect)

Residuals:
     Min       1Q   Median       3Q      Max
-2.21823 -0.40443  0.00502  0.51699  1.27670

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.45163    0.16694  14.686  < 2e-16 ***
genderffemale 0.04212    0.06720   0.627    0.531
Agency1       0.27167    0.06516   4.169  3.8e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6544 on 377 degrees of freedom
```

Add more predictors ...

```
Multiple R-squared:  0.04464, Adjusted R-squared:  0.03958
F-statistic: 8.809 on 2 and 377 DF,  p-value: 0.0001824
```

Other follow ups you might try

- R uses function `anova()` as a general purpose comparison tool. Confusing to people who expect it means ANOVA but it does not.
  - anova behaves differently if we supply just one model

```
anova(m2)
```

```
Analysis of Variance Table

Response: PosAFF1
           Df  Sum Sq Mean Sq F value    Pr(>F)
genderf     1   0.101  0.1009  0.2357    0.6276
Agency1     1   7.443  7.4425 17.3820 3.796e-05 ***
Residuals 377 161.422  0.4282
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(m1, m2)
```

Other follow ups you might try ...

```
Analysis of Variance Table

Model 1: PosAFF1 ~ genderf
Model 2: PosAFF1 ~ genderf + Agency1
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    378 168.86
2    377 161.42  1    7.4425 17.382 3.796e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- or two models

```
anova(m1, m2)
```

```
Analysis of Variance Table

Model 1: PosAFF1 ~ genderf
Model 2: PosAFF1 ~ genderf + Agency1
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    378 168.86
2    377 161.42  1    7.4425 17.382 3.796e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Other follow ups you might try ...

- Regression diagnostics using influence measures

```
m2.inf <- influence.measures(m2)
summary(m2.inf)
```

```
Potentially influential observations of
   lm(formula = PosAFF1 ~ genderf + Agency1, data = affect) :

    dfb.1_ dfb.gndr dfb.Agn1 dffit   cov.r   cook.d hat
26   0.32  -0.11    -0.29     0.33_*  1.00    0.04   0.03_*
38   0.06  -0.03    -0.05     0.06    1.02_*  0.00   0.02
95   0.10   0.04    -0.12    -0.13    1.03_*  0.01   0.03_*
124 -0.01  -0.01     0.01     0.01    1.03_*  0.00   0.02
136  0.14  -0.05    -0.12     0.14    1.03_*  0.01   0.03_*
146 -0.09  -0.03     0.11     0.12    1.03_*  0.01   0.03_*
177  0.00   0.13    -0.05    -0.19    0.96_*  0.01   0.01
193  0.22   0.10    -0.27    -0.32_*  0.99    0.03   0.02
194  0.22   0.10    -0.27    -0.32_*  0.99    0.03   0.02
219  0.06  -0.12    -0.06    -0.17    0.98_*  0.01   0.01
245  0.08   0.03    -0.09     0.10    1.03_*  0.00   0.02
252 -0.10   0.04     0.10     0.12    1.03_*  0.00   0.03_*
274 -0.11  -0.12     0.12    -0.21    0.97_*  0.01   0.01
275  0.02  -0.14    -0.02    -0.19    0.96_*  0.01   0.01
336  0.08  -0.18    -0.08    -0.27_*  0.92_*  0.02   0.01
```

Other follow ups you might try ...

```
365  0.02   0.01     -0.02      0.03     1.03_*  0.00   0.02
376  0.02  -0.14     -0.02     -0.19     0.96_*  0.01   0.01
380 -0.13  -0.12      0.14     -0.22     0.98_*  0.02   0.01
```

High Level plot functions in R Base

- functions provided with base R

create a "device"

plot	hist	barplot
plot.default	boxplot	dotchart
matplot	coplot	

- Run "example(hist)", "example(barplot)", and so forth
- Run "demo(graphics)"

Low Level plotting functions

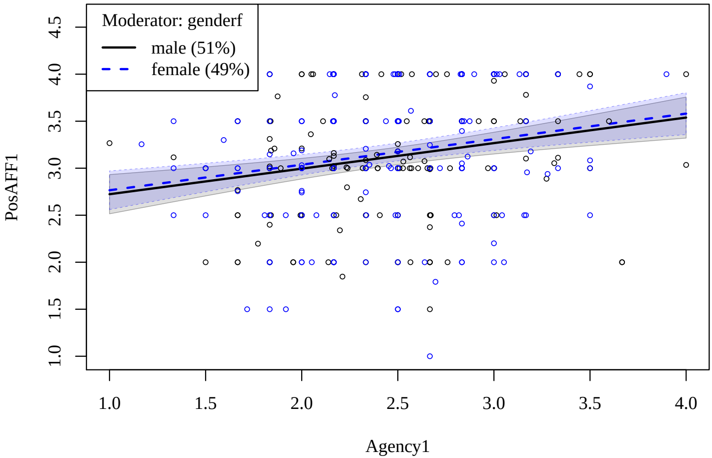
- High level functions create basic plot framework, coordinates
- Low Level functions: added accents or features

text	points	lines	box
arrows	segments	mtext	abline
axis	legend	title	polygon
rect			

Regression plot from the rockchalk package

```
library(rockchalk)
plotSlopes(m2, plotx = "Agency1", modx =
    "genderf", interval = "confidence")
```

Regression plot from the rockchalk package ...

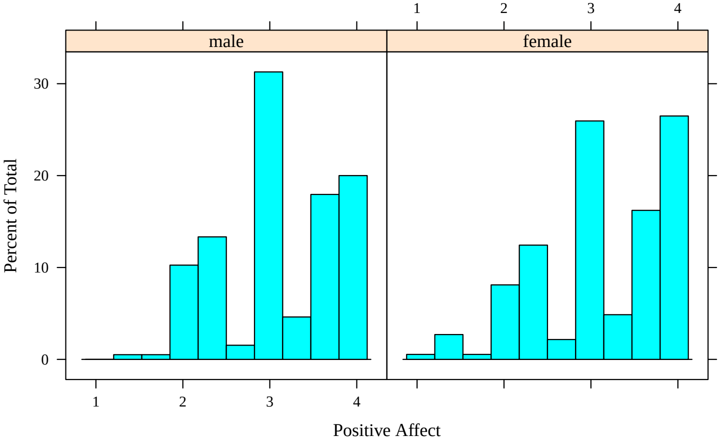


Many plot-oriented packages

- In R's recommended set, the package lattice is intended to produce polished "trellis" plots
- The separate sections are referred to as "panels", which allow intricate customizations
- The formula uses the pipe "|" to signify subgroups

```
library(lattice)
histogram( ~ PosAFF1 | genderf, data = affect,
    xlab = "Positive Affect")
```

Many plot-oriented packages ...



- A popular package ggplot2 offers similar output under the guise of "facets".

Outline

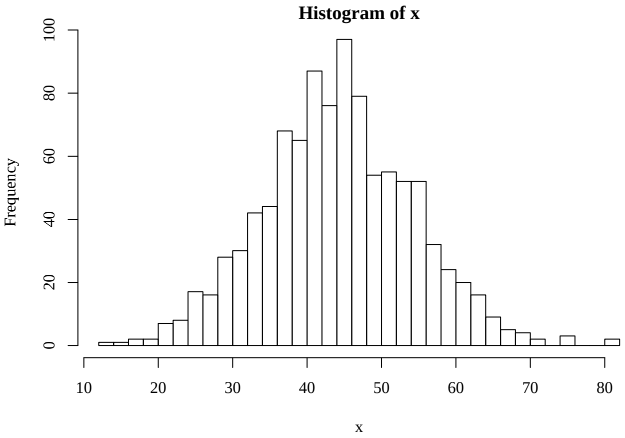
Normal jargon

- In SEM, you can't turn a page without somebody writing "assuming the data is multivariate normally distributed ..." or $\mathbf{X} \sim MVN(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. They also talk about Wishart distribution likelihood. A lot!
- That's mystifying to most social scientists
- R offers some ways to explore the gaps in our understanding.
- We start with 1 Normal variable, then look at multivariate Normal, then Wishart.

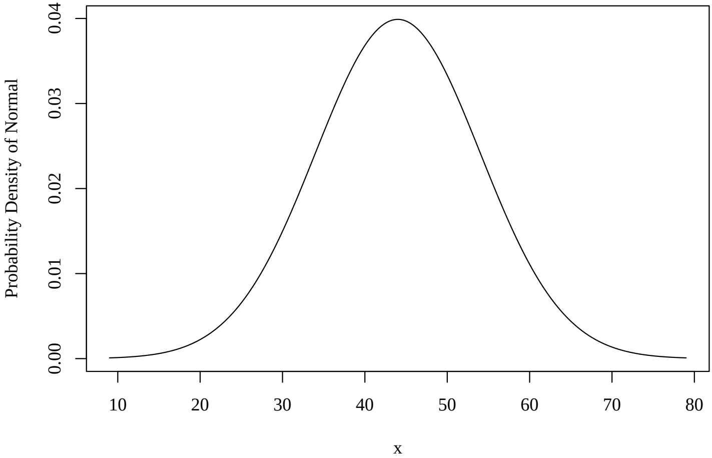
Pull one Normal sample

```
set.seed(234234)
N <- 1000
mu <- 44
sigma <- 10
x <- rnorm(N, m = mu, s = sigma)
hist(x, breaks = 30)
```

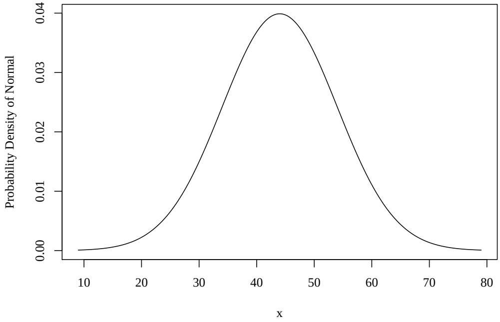
Pull one Normal sample ...



The Normal probability model



The Normal probability model



That's the probability density,

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \, e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

In my mind, it is written like this (anticipates multivariate Normal)

$$\frac{1}{(2\pi)^{1/2}\sigma} \, e^{-\frac{1}{2}(x-\mu)\frac{1}{\sigma^2}(x-\mu)}$$

Outline

Draw an MVN sample

- First, we need to create a mean vector `mu`

```
library(rockchalk)
mu <- c(3, 1, 44, 19) # numbers from top of my
    head
mu
```

```
[1]  3  1 44 19
```

- and a covariance matrix `sigma`

```
rho <- lazyCor(c(0.5, 0.6, 0.7, -0.1, 0.1, 0.2))
sd <- c(1, 2, 7, 4)
Sigma <- lazyCov(rho, sd)
Sigma
```

Draw an MVN sample ...

```
     [,1] [,2] [,3] [,4]
[1,]  1.0  1.0  4.2  2.8
[2,]  1.0  4.0 -1.4  0.8
[3,]  4.2 -1.4 49.0  5.6
[4,]  2.8  0.8  5.6 16.0
```

- Note I'm using "Sigma" here as name for covariance matrix, to signify it stands for Σ, not "sigma" σ.
- Ask for one random sample from that MVN generator

```
N <- 1
mvrnorm(N, m = mu, S = Sigma)
```

```
[1]  2.647835 -1.278373 51.686729 15.517961
```

- What did you get? Is that "4 people-worth" of data, or "1 person's data"?
- Ask for 5 cases from that generator

Draw an MVN sample ...

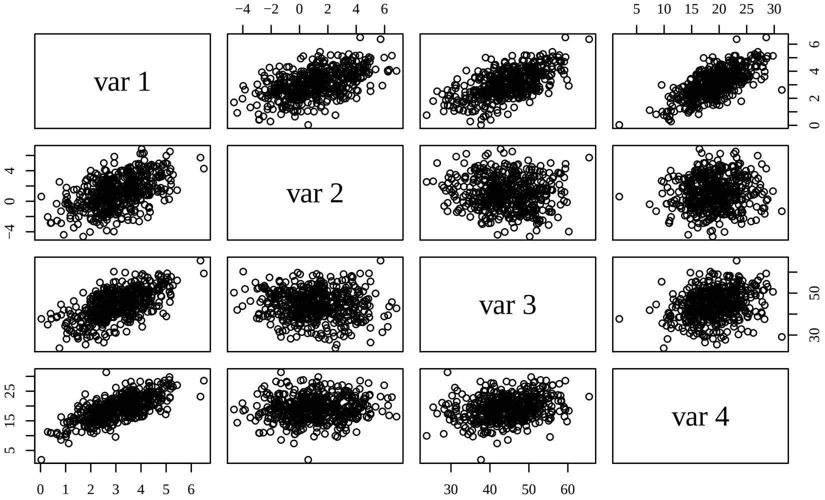
```
N <- 5
mvrnorm(N, m = mu, S = Sigma)
```

```
          [,1]       [,2]     [,3]      [,4]
[1,]  2.2850417  2.3721830 35.21399 17.396727
[2,]  2.7031800  1.6026484 42.71234 17.042282
[3,] -0.5003047 -2.8547621 37.42284  6.372255
[4,]  4.1630361 -0.6135541 58.45679 19.259787
[5,]  2.9469960  3.9712670 40.34506 14.036906
```

- Dial up the sample size to 500. Call the result `X`, get a pair plot

```
N <- 500
X <- mvrnorm(N, m = mu, S = Sigma)
pairs(X)
```

Draw an MVN sample ...



Sample versus Sigma

The sample variance/covariance matrix

```
var(X)
```

```
          [,1]       [,2]      [,3]       [,4]
[1,] 1.057213  1.1005485  4.383614  2.9438276
[2,] 1.100549  4.3144473 -1.067406  0.8533109
[3,] 4.383614 -1.0674059 47.564156  6.9088575
[4,] 2.943828  0.8533109  6.908857 16.1618587
```

is not exactly the same as Sigma.

- But it hovers around Sigma, doesn't it? Check for yourself.

```
X <- mvrnorm(N, m = mu, S = Sigma)
var(X)
```

```
          [,1]       [,2]      [,3]       [,4]
[1,] 0.9428358  0.8792096  4.156707  2.6206476
[2,] 0.8792096  3.9736975 -1.956495  0.3235383
[3,] 4.1567068 -1.9564948 50.783077  5.1520706
[4,] 2.6206476  0.3235383  5.152071 15.6659116
```

Sample versus Sigma ...

- Pull another sample, calculate the variance matrix again

```
X <- mvrnorm(N, m = mu, S = Sigma)
var(X)
```

```
         [,1]      [,2]      [,3]      [,4]
[1,] 1.106521  1.100028  4.846117  2.965107
[2,] 1.100028  4.146812 -1.390915  1.240546
[3,] 4.846117 -1.390915 53.874826  7.121330
[4,] 2.965107  1.240546  7.121330 15.287877
```

- Again (again, again, again)

```
X <- mvrnorm(N, m = mu, S = Sigma)
var(X)
```

```
         [,1]       [,2]       [,3]      [,4]
[1,] 1.059179  1.1419708  4.4572142  2.867251
[2,] 1.141971  4.1243364 -0.6975967  1.136091
[3,] 4.457214 -0.6975967 49.6711047  6.161464
[4,] 2.867251  1.1360906  6.1614637 15.743375
```

Sample versus Sigma ...

- Notice: The covariance matrix changes a little bit from one sample to another

Digression: The MVN Density Formula

- A vector of means μ="mu" and a covariance matrix Σ="Sigma"

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \sim MVN(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = MVN\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} & & \sigma_{1p} \\ \sigma_{12} & \sigma_2^2 & & \sigma_{2p} \\ & & \ddots & \\ \sigma_{1p} & \sigma_{2p} & & \sigma_p^2 \end{bmatrix} \right)$$

- The multivariate PDF looks similar to the second way I wrote the Normal pdf for one variables

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} e^{\frac{-1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

where p is the number of elements in μ and $|\boldsymbol{\Sigma}|$ is the determinant of Σ.

- If you need to create Sigma, it is easiest do do that by starting with standard deviations and a correlation matrix

$$SD \times Corr \times SD$$

Digression: The MVN Density Formula ...

$$Sigma = \begin{bmatrix} \sigma_{x1} & 0 & 0 & 0 & 0 \\ 0 & \sigma_{x2} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{x3} & 0 & 0 \\ 0 & 0 & 0 & \sigma_{x4} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{x5} \end{bmatrix} \times \begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \dots & \rho_{1p} \\ \rho_{21} & 1 & \rho_{23} & & \rho_{2p} \\ \rho_{31} & \ddots & 1 & & \rho_{3p} \\ \vdots & 11 & \rho_{11} & \ddots & \\ \rho_{p1} & \rho_{11} & \rho_{11} & & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} \sigma_{x1} & 0 & 0 & 0 & 0 \\ 0 & \sigma_{x2} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{x3} & 0 & 0 \\ 0 & 0 & 0 & \sigma_{x4} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{x5} \end{bmatrix} \quad (1)$$

Outline

The Wishart

- The Wishart has special meaning for structural equation modelers.
- It is the distribution underlying the classic LISREL model and Maximum Likelihood estimation of SEM.

Where do Wishart Draws Come From?

- Draw `X` from `mvrnorm`. Suppose it has 1000 rows and 4 variables.
- We need the expected values to be 0, so re-set `mu`

```
mu <- c(0, 0, 0, 0)
N <- 1000
X <- mvrnorm(N, m = mu, S = Sigma)
```

- Calculate the covariance matrix of $\mathbf{X}$. That is 4x4

```
var(X)
```

```
          [,1]       [,2]      [,3]      [,4]
[1,] 0.9422976  0.8845368  4.166469  2.572029
[2,] 0.8845368  3.9215809 -1.616552  0.201467
[3,] 4.1664686 -1.6165519 46.590962  6.564746
[4,] 2.5720286  0.2014670  6.564746 14.987849
```

Where do Wishart Draws Come From? ...

- Recall formula for covariance when $\mu = (0,0,0,0)^T$ is

$$\frac{1}{N-1}\mathbf{X}^T\mathbf{X}$$

- The variation of $\mathbf{X}^T\mathbf{X}$ from one-sample to another has a mathematical law, Wishart's distribution.
- Recall that our $\boldsymbol{\Sigma}$ is

```
Sigma
```

```
      [,1] [,2] [,3] [,4]
[1,]   1.0  1.0  4.2  2.8
[2,]   1.0  4.0 -1.4  0.8
[3,]   4.2 -1.4 49.0  5.6
[4,]   2.8  0.8  5.6 16.0
```

For a given N, the Wishart value will be something hovering around

```
(N-1) * Sigma
```

Where do Wishart Draws Come From? ...

```
        [,1]    [,2]    [,3]    [,4]
[1,]  999.0   999.0  4195.8  2797.2
[2,]  999.0  3996.0 -1398.6   799.2
[3,] 4195.8 -1398.6 48951.0  5594.4
[4,] 2797.2   799.2  5594.4 15984.0
```

The Wishart

- R provides a simulator for the Wishart distribution, it requires the parameters for the degrees of freedom (N-1) and the covariance matrix.

```
rWishart(1, df = N - 1, Sigma = Sigma)
```

```
, , 1

          [,1]       [,2]      [,3]       [,4]
[1,]  963.5219   904.8346  4077.600  2689.9585
[2,]  904.8346  3905.8417 -1584.001   455.8215
[3,] 4077.6003 -1584.0012 49089.373  4401.6478
[4,] 2689.9585   455.8215  4401.648 16211.4643
```

- Draw another one

```
rWishart(1, df = N - 1, Sigma = Sigma)
```

The Wishart ...

```
, , 1

          [,1]       [,2]      [,3]       [,4]
[1,] 1027.3948   993.9768  4263.393  2906.1633
[2,]  993.9768  3872.0724 -1459.308   937.0922
[3,] 4263.3930 -1459.3083 48596.057  6467.7721
[4,] 2906.1633   937.0922  6467.772 15680.1347
```

A Sample of Covariance Matrices?

- Draw 100 Wishart matrices

```
lotsofwishes <- rWishart(100, df = N - 1, Sigma =
    Sigma)
```

- They are an R "array", can get first matrix like this

```
lotsofwishes[ , , 1]
```

```
           [,1]       [,2]      [,3]       [,4]
[1,] 1018.6397   979.5887  4285.956  2850.5592
[2,]  979.5887  3842.1253 -1539.122   972.9656
[3,] 4285.9557 -1539.1221 49115.957  6045.3696
[4,] 2850.5592   972.9656  6045.370 15556.5362
```

- Or the 53rd like this:

```
lotsofwishes[ , , 53]
```

A Sample of Covariance Matrices? ...

```
            [,1]      [,2]       [,3]      [,4]
[1,] 1061.316 1125.1713  4414.3902  2904.322
[2,] 1125.171 3933.0465  -640.3095  1421.105
[3,] 4414.390 -640.3095 49214.9351  5448.250
[4,] 2904.322 1421.1049  5448.2496 15957.901
```

- Here's where the structural equation modeling part comes back into play
- SEM compares a model/theoretical covariance with a sample covariance.
- The model/theoretical covariance, which is a matrix full of coefficients (loadings, etc) that represent our theoretical parameters.
- Estimator shifts the parameters to try to make the theoretical covariance similar to the observed covariance as possible.
- Even when we have N observations in a data set, the SEM calculations are based on distilled information, the $p \times p$ covariance matrix. SEM is, in a sense, based on 1 data point, which is a matrix.

4 Basic Goals Achieved

1. Import data
2. Revise data
3. Do analysis (fit models)
4. Create plots

If you ever need help

- Ask in the r-help email list, or in https://stackoverflow.com/questions/tagged/r
- Save some time: Ask your question with
  - code you ran
  - copy/pasted output & error messages
  - Output from `sessionInfo()`

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 19.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
      LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
      LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
     LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] lattice_0.20-38   rockchalk_1.8.144
```

Session ...

```
loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1     MASS_7.3-51.4  grid_3.6.0     plyr_1.8.4
      nlme_3.1-140   xtable_1.8-4
 [7] stats4_3.6.0   zip_2.0.2      carData_3.0-2  minqa_1.2.4
      nloptr_1.2.1   Matrix_1.2-17
[13] pbivnorm_0.6.0 boot_1.3-22    openxlsx_4.1.0 splines_3.6.0
     lme4_1.1-21    tools_3.6.0
[19] foreign_0.8-71 kutils_1.69    compiler_3.6.0 mnormt_1.5-5
     lavaan_0.6-3
```