Correlation and Linear Regression

Contents

1 Correlations **2**
- 1.1 Pearson's Correlation . . . 2
- 1.2 Pearson's Correlation . . . 2
  - 1.2.1 Simulate Data . . . 2
  - 1.2.2 Plot the data . . . 3
  - 1.2.3 Run Pearson's r . . . 3
  - 1.2.4 Report them in APA format . . . 3
  - 1.2.5 Pearson's correlation is scale independent! . . . 4
  - 1.2.6 Pearson's: Let visualize our result . . . 4
- 1.3 Non-Parametric Correlations . . . 5
  - 1.3.1 Spearmen's Correlation . . . 5
- 1.4 Point-by-Serial Correlation . . . 8
  - 1.4.1 Kendall's Tau . . . 9
  - 1.4.2 Calculation of Point-by Serial . . . 9

2 Regression **9**
- 2.1 Basic Regression Equation . . . 9
- 2.2 Modern Regression Equation . . . 9
- 2.3 Ice Cream example . . . 10
  - 2.3.1 Intercept . . . 10
  - 2.3.2 Slope . . . 10
  - 2.3.3 Error for this prediction? . . . 10
  - 2.3.4 Ordinary least squares (OLS) . . . 11
- 2.4 SE on the terms in the models (how good is the fit?) . . . 11
  - 2.4.1 SE on the Intercept . . . 12
  - 2.4.2 SE on the Slope . . . 12
  - 2.4.3 t-tests on slope and intercept and r^2 value . . . 12
- 2.5 Regression in ANOVA format . . . 13

3 Bootstrapped Regression **13**
- 3.1 Bootstrapped Parameters . . . 13
- 3.2 Bootstrapped R-Squared . . . 14

4 Power and Regression **16**
- 4.1 Power Calculation . . . 16
- 4.2 A Priori Power Analysis . . . 16

5 Final Notes **17**

6 References **17**

1 Correlations

Everything correlates with everything, which Paul Meehl calls the "crud factor" (aka Ambient Correlational Noise) (See Meehl, 1990ab and Lykken, 1968 cited by Meehl). Our goal today is to determine how much and we will deal with 2 variables today, but we will soon explore the problems with 3 or more variables.

A few popular correlations between two variables:

- Pearson's r (interval by interval) [for population = ρ, for sample = r]
- Spearman's rho (interval by ordinal) [for population = ρ_s, for sample = r_s]
- Kendall's tau [t] (interval by ordinal or ordinal by ordinal) like Spearman's, but more accurate with small samples
- Point-by-serial (interval by dichotomous)
- Polychoric (ordinal vs ordinal) [used more in psychometrics or factor analysis of ordinal by ordinal]
- Tetrachroic (dichotomous vs dichotomous) [same as above]

1.1 Pearson's Correlation

Most common type you will encounter and is a parametric method.

$$r_{xy} = \frac{\sum xy}{\sqrt{\sum x^2 \sum y^2}}$$

- Numerator = How much they vary together (covariance)
- Denominator = How much they vary alone (variance)
- Values is bounded between -1 and 1

1.2 Pearson's Correlation

Let's create two random normal variables that correlate with each other.

1.2.1 Simulate Data

We will use the `mvrnorm` function (multivariate normal distribution) from the *MASS* package, but to do this we need to make a **covariance** matrix with a $r = .60$ and set the mean values for each variable (which I will set to 5 for each)

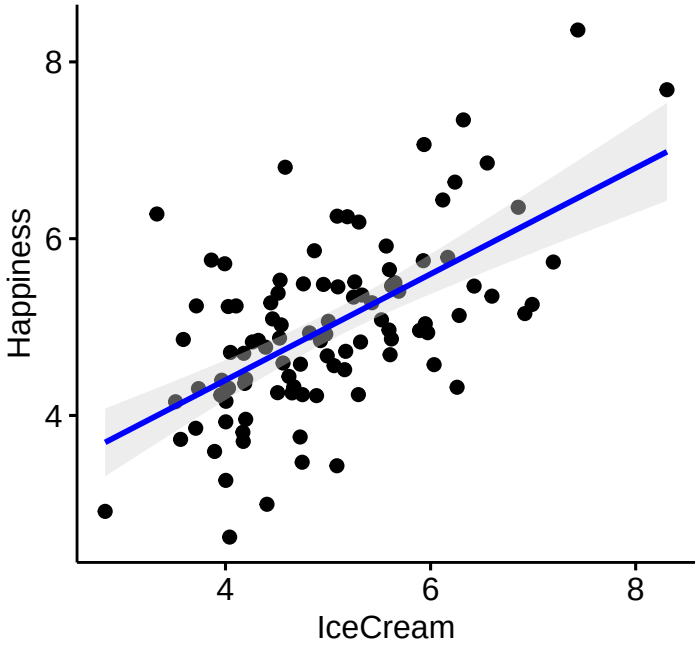
```
#Set params
Means.XY<- c(5,5) #set the means of X and Y variables
r=.6 #Correlation value
CovMatrix.XY <- matrix(c(1,r,
                         r,1),2,2) # creates the covariate matrix

# Build the correlated variables using .
# Note: empirical=TRUE means make the correlation EXACTLY r.
# empirical=FALSE, the correlation value would be normally distributed around r
library(MASS) #create data
CorrData<-mvrnorm(n=100, mu=Means.XY,Sigma=CovMatrix.XY, empirical=TRUE)
```

```r
#Convert them to a "Data.Frame", which is like SPSS data window
CorrData<-as.data.frame(CorrData)
#lets add our labels to the vectors we created
colnames(CorrData) <- c("Happiness","IceCream")
```

1.2.2 Plot the data

```r
#make the scatter plot
library(ggpubr) #graph data
ggscatter(CorrData, x = "IceCream", y = "Happiness",
   add = "reg.line",  # Add regressin line
   add.params = list(color = "blue", fill = "lightgray"), # Customize reg. line
   conf.int = TRUE, # Add confidence interval
   cor.coef = FALSE, # Add correlation coefficient. see ?stat_cor
   )
```



1.2.3 Run Pearson's r

The `cor.test` function runs the Pearson's correlation.

```r
Corr.Result.1<-cor.test(CorrData$Happiness, CorrData$IceCream,
         method = c("pearson"))
```

1.2.4 Report them in APA format

The `cor_apa` function in the APA package will report it in APA format for you.

```r
library(apa)
cor_apa(Corr.Result.1,format ="text")
```

```
## r(98) = .60, p < .001
```

1.2.5 Pearson's correlation is scale independent!

No matter the mean differences or range of scores, the Pearson's r will give the same results. We can also z-score the data and get the same result. However, if they are scaled non-linearly (sqrt, ^2, log,...) the correlation will change.

1.2.5.1 Lets add (change the mean)

```
Happiness.big<-CorrData$Happiness+1000
IceCream<-CorrData$IceCream
cor_apa(cor.test(Happiness.big, IceCream,
         method = c("pearson")),format ="text")
```

```
## r(98) = .60, p < .001
```

1.2.5.2 Z-scored

```
Happiness.z<-scale(CorrData$Happiness)
IceCream.z<-scale(CorrData$IceCream)
cor_apa(cor.test(Happiness.z, IceCream.z,
         method = c("pearson")),format ="text")
```

```
## r(98) = .60, p < .001
```

1.2.5.3 What happens if I LINEARLY scale them differently?

```
cor_apa(cor.test(Happiness.big, IceCream.z,
         method = c("pearson")),format ="text")
```

```
## r(98) = .60, p < .001
```

1.2.5.4 What happens if I NON-LINEARLY scale them differently?

```
Happiness<-CorrData$Happiness # orginal
IceCream.sq4<-(CorrData$IceCream)^4 #Non-linear
cor_apa(cor.test(Happiness, IceCream.sq4,
         method = c("pearson")),format ="text")
```

```
## r(98) = .60, p < .001
```

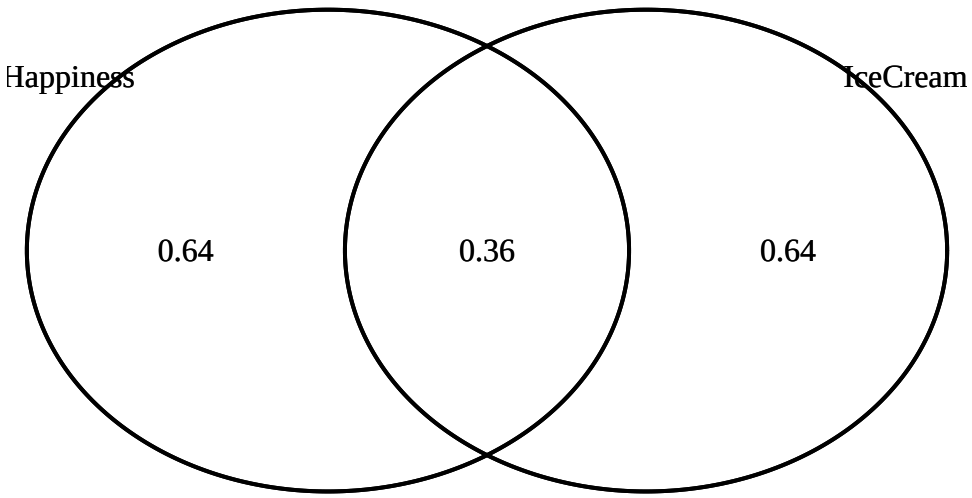
1.2.6 Pearson's: Let visualize our result

- Overlap between the two variables is defined by r^2

```
 # lets us plot our results (like the book)
library(VennDiagram)
# calculate r-squared
```

```
overlap=r^2

Simple.Corr.Venn<-draw.pairwise.venn(1, 1, overlap, c("Happiness", "IceCream"))
grid.draw(Simple.Corr.Venn)
```



1.3 Non-Parametric Correlations

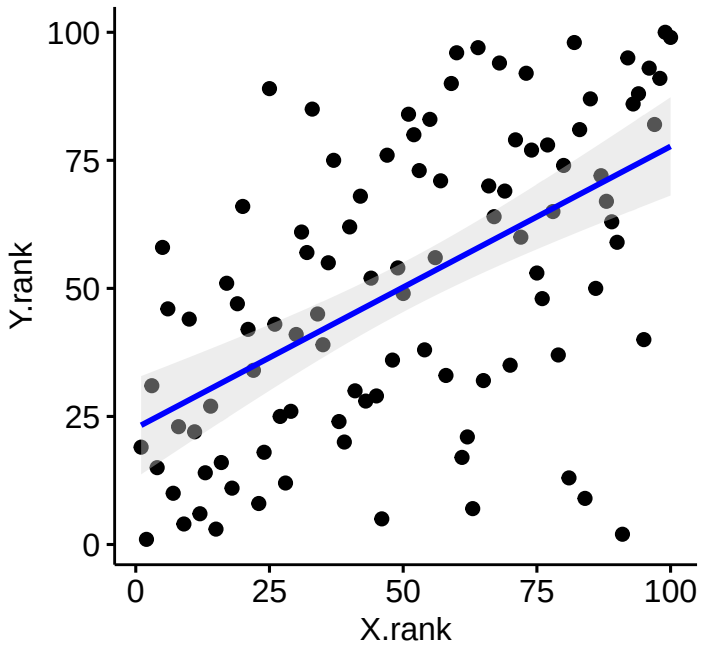
Spearmen and Kendall correlation can be used for ordinal data, but should be used if you have a "bend" (non-linear relationship) between variables.

1.3.1 Spearmen's Correlation

Spearman is basically a Pearson Correlation on rank-ordered data. Let's rank order our random correlated data. You must rank each variable independently first (where ties are averaged).

```
CorrData$X.rank<-rank(CorrData$Happiness)
CorrData$Y.rank<-rank(CorrData$IceCream)

ggscatter(CorrData, x = "X.rank", y = "Y.rank",
   add = "reg.line",  # Add regressin line
   add.params = list(color = "blue", fill = "lightgray"), # Customize reg. line
   conf.int = TRUE, # Add confidence interval
   )
```




```
cor_apa(cor.test(CorrData$Y.rank, CorrData$X.rank, method = c("pearson")))
```

```
## r(98) = .55, p < .001
```

You should use the use the built-in Spearman correlation (`cor.test`, but pass **method = c("spearman")**) because the pvalues are calculated differently and ranks the raw data automatically.

```
# APA format (note the S should be subscript)
cor_apa(cor.test(CorrData$Y, CorrData$X,
         method = c("spearman")),format ="text")
```

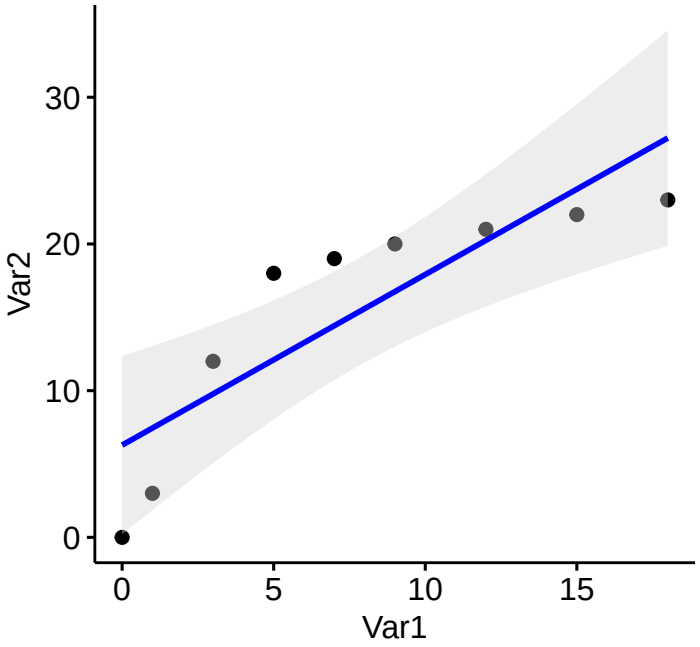
```
## r_s = .55, p < .001
```

1.3.1.1 Pearson vs Spearmen's Correlation for slight nonlinearity

Let's say you get some data and clearly there is a slight nonlinearity in the relationship between the two variables. Pearson is designed for linear relationships and we can see the problem in our fitted line below.

```
CorrNL<-data.frame(Var1=c(0,1,3,5,7,9,12,15,18),
                   Var2=c(0,3,12,18,19,20,21,22,23))

ggscatter(CorrNL, x = "Var1", y = "Var2",
   add = "reg.line",  # Add regressin line
   add.params = list(color = "blue", fill = "lightgray"), # Customize reg. line
   conf.int = TRUE, # Add confidence interval
   )
```

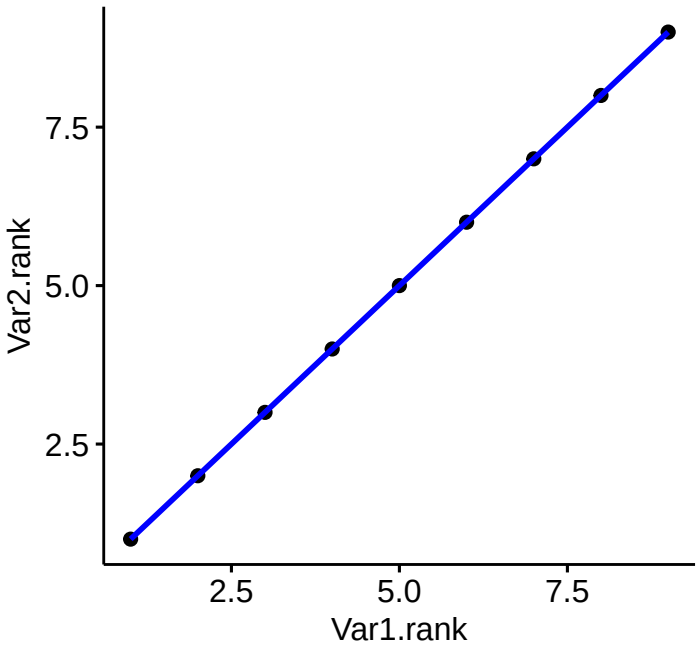



```
## r(7) = .86, p = .003
```

If we switch to a Spearman correlation the data are converted to ranks and the "bump" is now gone and our correlation gets stronger. S

```
CorrNL$Var1.rank<-rank(CorrNL$Var1)
CorrNL$Var2.rank<-rank(CorrNL$Var2)

ggscatter(CorrNL, x = "Var1.rank", y = "Var2.rank",
   add = "reg.line",  # Add regressin line
   add.params = list(color = "blue", fill = "lightgray"), # Customize reg. line
   conf.int = TRUE, # Add confidence interval
   )
```




```
## r_s = 1.00, p < .001
```

1.4 Point-by-Serial Correlation

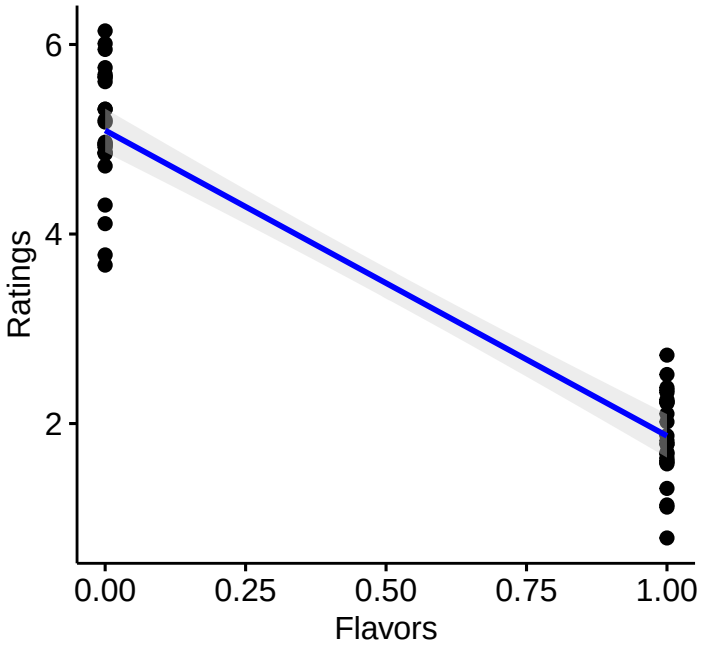
This time lets make up some data on the fly

```
set.seed(42)
Ratings<-c(rnorm(25,mean=5,sd = .5),rnorm(25,mean=2,sd = .5))
Flavors<-c(rep(0,25),c(rep(1,25)))
FlavorNames<-c(rep("Cookie Dough",25),c(rep("Rum-Raisin",25)))

#Build data frame
Ice.Cream.Data<-data.frame(
  Ratings = Ratings,
  Flavors = Flavors,
  Names = FlavorNames)
head(Ice.Cream.Data)
```

```
##    Ratings Flavors        Names
## 1 5.685479       0 Cookie Dough
## 2 4.717651       0 Cookie Dough
## 3 5.181564       0 Cookie Dough
## 4 5.316431       0 Cookie Dough
## 5 5.202134       0 Cookie Dough
## 6 4.946938       0 Cookie Dough
```

```
ggscatter(Ice.Cream.Data, x = "Flavors", y = "Ratings",
   add = "reg.line",  # Add regression line
   add.params = list(color = "blue", fill = "lightgray"), # Customize reg. line
   conf.int = TRUE, # Add confidence interval
   )
```



1.4.1 Kendall's Tau

Kendall tau will always be more conservative than spearman correlation and is generally more robust (`cor.test`, but pass **method = c("kendall")**). It is safer to use but less widely known.

```
# APA format (note the S should be subscript)
cor_apa(cor.test(CorrData$Y, CorrData$X,
         method = c("kendall")),format ="text")
```

```
## r_tau = .39, p < .001
```

1.4.2 Calculation of Point-by Serial

using the `polycor` package, we can run **polyserial** function using maximum-likelihood estimation (generally more accurate).

```
library(polycor) #Advanced Correlations
with(Ice.Cream.Data,
     polyserial(Flavors,Ratings, ML=TRUE))
```

```
## [1] -0.8091302
```

2 Regression

- Correlation and regression are similar
- Correlation determines the standardized relationship between X and Y
- Linear regression = 1 DV and 1 IV, where the relationship is a straight line
- Linear regression determines how X predicts Y
- Multiple (linear) regression = 1 DV and 2+ IV (also straight lines)
- Multiple regression determines how X,z, and etc, predict Y [next week]

2.1 Basic Regression Equation

- Linear Regression equation you learned when younger was probably $y = MX + b$
- y = predict value
- M = slope
- X = Variable used to predict Y
- b = intercept

2.2 Modern Regression Equation

- $Y = B_{YX}X + B_0 + e$
- Y = predict value
- B_{YX} = slope
- B_0 = intercept
- e = error term (observed - predicted). Also called the residual.

2.3 Ice Cream example

- Specify the model with the lm function.
- We are going to predict happiness scores from ice cream!

```
Happy.Model.1<-lm(Happiness~IceCream,data = CorrData)
summary(Happy.Model.1)
```

```
##
## Call:
## lm(formula = Happiness ~ IceCream, data = CorrData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.79983 -0.54079 -0.03726  0.44039  2.28066
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.00000    0.41198   4.855 4.56e-06 ***
## IceCream     0.60000    0.08081   7.425 4.19e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8041 on 98 degrees of freedom
## Multiple R-squared:   0.36,  Adjusted R-squared:  0.3535
## F-statistic: 55.12 on 1 and 98 DF,  p-value: 4.193e-11
```

2.3.1 Intercept

- 2 is where the line hit the y-intercept (when happiness = 0).

2.3.2 Slope

- 0.6 is **the rise over run**
- for each 0.6 change in ice cream value, there is a corresponding change in happiness!
- so we can predict happiness from ice cream score:

  (0.6 * 5 spoons of ice cream + baseline happiness intercept: 2) = 5

- This is your *predicted* happiness score if you had 5 spoons of ice cream

2.3.3 Error for this prediction?

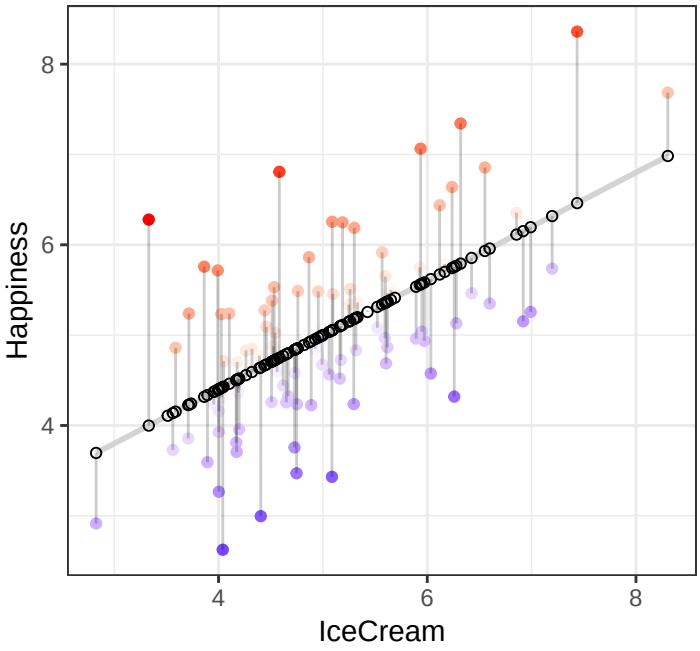
R will do all the prediction for us for each value of ice cream residuals = **observed - predicted**

- Red dots = **observed** *above* predictor line
- Blue dots = **observed** *below* predictor line
- the stronger the color, the more an impact that point has in pulling the line in its direction
- Hollow dots = **predicted**
- The gray lines are the distance between **observed** and **predicted** values!

What should the mean of the residuals equal?

```
CorrData$predicted <- predict(Happy.Model.1)   # Save the predicted values with our real data
CorrData$residuals <- residuals(Happy.Model.1) # Save the residual values

library(ggplot2)
ggplot(data = CorrData, aes(x = IceCream, y = Happiness)) +
  geom_smooth(method = "lm", se = FALSE, color = "lightgrey") +  # Plot regression slope
  geom_point(aes(color = residuals)) +  # Color mapped here
  scale_color_gradient2(low = "blue", mid = "white", high = "red") +  # Colors to use here
  guides(color = FALSE) +
  geom_segment(aes(xend = IceCream, yend = predicted), alpha = .2) +  # alpha to fade lines
  geom_point(aes(y = predicted), shape = 1) +
  theme_bw()  # Add theme for cleaner look
```



2.3.4 Ordinary least squares (OLS)

- Linear regression finds the best fit line by trying to minimize the sum of the squares of the differences between the observed responses those predicted by the line.
- OLS computationally simple to get the slope value, but is inaccurate

$$B_{YX} = \frac{\sum XY - \frac{1}{n}\sum X \sum Y}{\sum x^2 - \frac{1}{n}\sum x^2} = \frac{Cov_{XY}}{var_x}$$

- Modern methods use an alternative (ML, REML) we will examine later when we get to GLM

2.4 SE on the terms in the models (how good is the fit?)

- Residual Standard error $= \sqrt{\frac{\sum e^2}{n-2}}$
- in R language:

```r
n=length(CorrData$residuals)

RSE = sqrt(sum(CorrData$residuals^2) / (n-2))
RSE
```

```
## [1] 0.8040713
```

- So, our error on the prediction is 0.8040713 happiness points based on our model.

2.4.1 SE on the Intercept

- Intercept Standard error = $RSE\sqrt{\frac{1}{n} + \frac{M_x^2}{(n-1)var_x}}$
- in R language:

```r
ISE = RSE*(sqrt( 1 / n + mean(CorrData$IceCream)^2 / (n - 1)*var(CorrData$IceCream)))
ISE
```

```
## [1] 0.4119838
```

2.4.2 SE on the Slope

- Slope Standard error = $\frac{sd_y}{sd_x}\sqrt{\frac{1-r_Y X^2}{n-2}}$
- in R language:

```r
#lets extract the r2 from the model
r2.model<-summary(Happy.Model.1)$r.squared

SSE = sd(CorrData$Happiness)/sd(CorrData$IceCream) * sqrt((1- r2.model)/ (n - 2))
SSE
```

```
## [1] 0.0808122
```

2.4.3 t-tests on slope and intercept and r^2 value

- Values are tested against 0, so its all one sample t-tests
- slope: $t = \frac{B_{YX} - H_0}{SE_{B_{YX}}}$
- intercept: $t = \frac{B_0 - H_0}{SE_{B_0}}$

r^2 is a little different as its a correlation value

- correlations are not normally distributed
- Fisher created a conversion for r to make it a z (called Fishers' r to Z)
- r^2: $t = \frac{r_{XY}\sqrt{n-2} - H_0}{\sqrt{1-r_{XY}^2}}$, where $df = n - 2$
- its often given for as an F value, remember $t^2 = F$

```r
#intercept
t.I= Happy.Model.1$coefficients[1]/ISE
t.I
```

```
## (Intercept)
##     4.85456
```

```
#Slope
t.S= Happy.Model.1$coefficients[2]/SSE
t.S
```

```
## IceCream
## 7.424621
```

```
# For r-squared
t.r2xy = r2.model^.5*sqrt(n-2)/sqrt(1-r2.model)
F.r2xy = t.r2xy^2
F.r2xy
```

```
## [1] 55.125
```

Note: We are testing null hypothesis value for slope, i.e., null = 0. But it's a terrible guess. Everything correlates with everything, so it's important to keep this in mind moving forward. So that would be the NILL hypothesis. *NILL can be tested better with bootstrapping.*

2.5 Regression in ANOVA format

- You can also report the results of all the predictors (if you have multiple) in ANOVA style format (F-test we calculated above on r^2)
- This is useful in multiple regression as it tell your if your overall set of predictors is significant

```
anova(Happy.Model.1)
```

```
## Analysis of Variance Table
##
## Response: Happiness
##           Df Sum Sq Mean Sq F value    Pr(>F)
## IceCream   1  35.64  35.640  55.125 4.193e-11 ***
## Residuals 98  63.36   0.647
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3 Bootstrapped Regression

We can bootstrap the parameters (Intercept and slope and overall R^2)

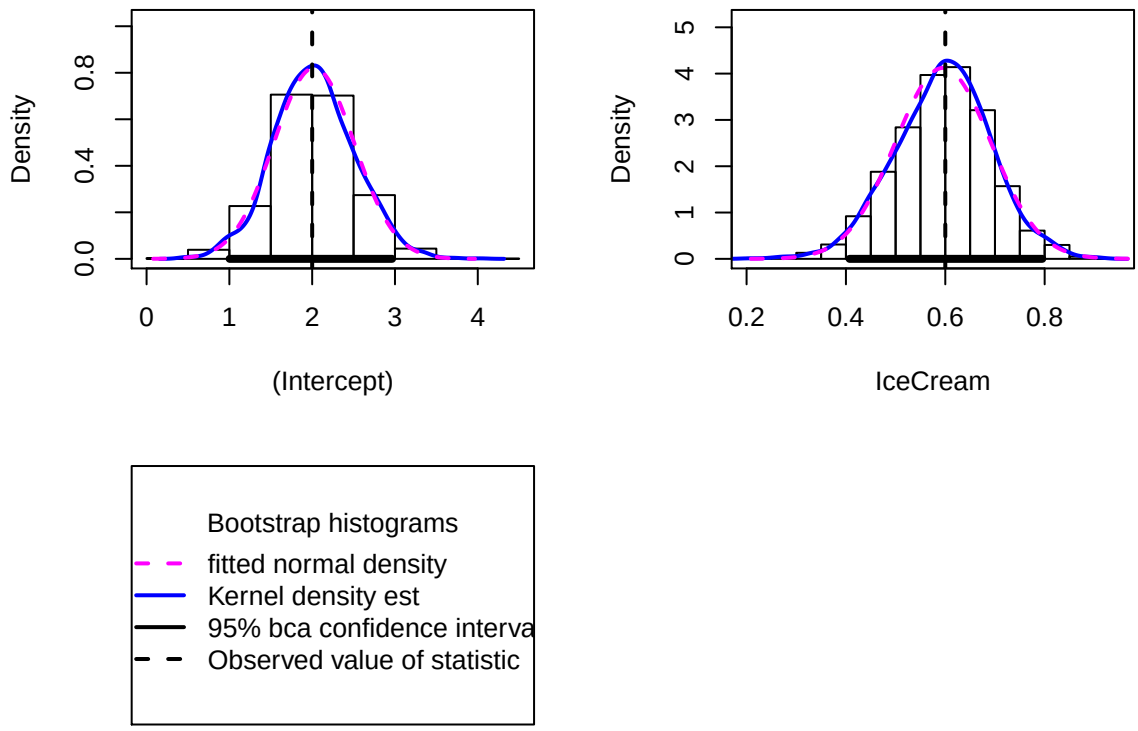
3.1 Bootstrapped Parameters

We will use the `car` (which calls the `boot` package) and it will give Bca confidence intervals. see https://socialsciences.mcmaster.ca/jfox/Books/Companion/appendix/Appendix-Bootstrapping.pdf

```
set.seed(666)
library(boot)
library(car)
```

```
# Boot the model fit, 2K times
BootParms <-Boot(Happy.Model.1, R = 2000)

# View results
hist(BootParms, legend="separate")
```




```
# get 95% confidence interval
confint(BootParms, level=.95)
```

```
## Bootstrap bca confidence intervals
##
##                 2.5 %    97.5 %
## (Intercept) 1.0027550 2.9654668
## IceCream    0.4072358 0.7952822
```

3.2 Bootstrapped R-Squared

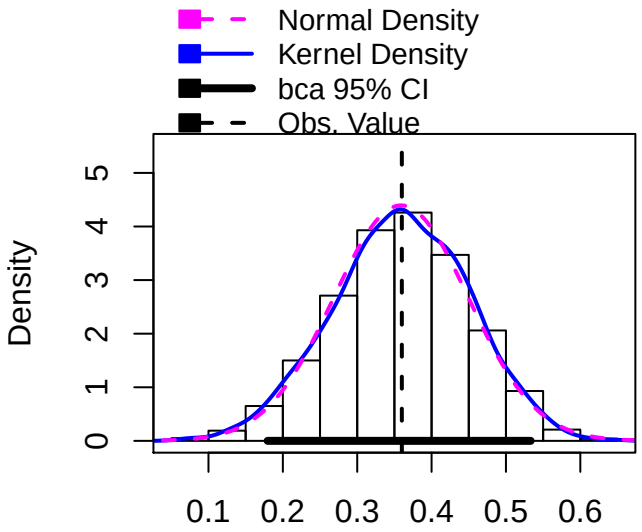
To get R^2 is a little harder. We must write a function to extract R^2 from the analysis. See https://www.statmethods.net/advstats/bootstrapping.html

```
set.seed(666)
# function to obtain R-Squared from the data
rsq <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
```

```
  fit <- lm(formula, data=d)
  return(summary(fit)$r.square)
}

# bootstrapping with 2000 replications
BootRsq <- boot(data=CorrData, statistic=rsq,
    R=2000, formula=Happiness~IceCream)

# view results
hist(BootRsq)
```




```
# get 95% confidence interval
boot.ci(BootRsq)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = BootRsq)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.1831,  0.5390 )   ( 0.1891,  0.5429 )
##
## Level     Percentile            BCa
## 95%   ( 0.1771,  0.5309 )   ( 0.1795,  0.5332 )
## Calculations and Intervals on Original Scale
```

4 Power and Regression

For regression, we will need to convert our r^2 into cohen's f^2

$$f^2 = \frac{r^2}{1 - r^2}$$

4.1 Power Calculation

We will use the `pwr` package.

```
library(pwr) #power analysis
#power for GLM
# u  = degrees of freedom for numerator
# v = degrees of freedom for denominator
# f2 = effect size
# sig.level= (Type I error probability)
# power = (1 minus Type II error probability)
f2.icecream <- r^2 / (1-r^2)

pwr.f2.test(u = 1, v = n-2, f2 = f2.icecream, sig.level = 0.05, power = NULL)
```

```
##
##      Multiple regression power calculation
##
##               u = 1
##               v = 98
##              f2 = 0.5625
##       sig.level = 0.05
##           power = 1
```

So we had a power of basically 1 given this sample size and our true effect size of 0.6

4.2 A Priori Power Analysis

- What sample size do I need given a specific f^2
- Note: Gpower might use f, not f^2

```
#power for GLM
# u  = degrees of freedom for numerator
# v = degrees of freedom for denominator
# f2 = effect size
# sig.level= (Type I error probability)
# power = (1 minus Type II error probability)

pwr.f2.test(u = 1, v = NULL, f2 = f2.icecream, sig.level = 0.05, power = .80)
```

```
##
##      Multiple regression power calculation
##
```

```
##               u = 1
##               v = 14.12059
##              f2 = 0.5625
##       sig.level = 0.05
##           power = 0.8
```

5 Final Notes

- Testing between correlations using old fashion Fisher's test is old fashion (Cohen et al., p. 49). The modern approach is the bootstrap, as the old method is underpowered.
- Your book is a little out of date: CIs are better but bootstrapped CIs are becoming more standard. Use the code above to bootstrapped CIs when reporting results.

6 References

Lykken, D. T. (1968). Statistical significance in psychological research. *Psychological bulletin*, 70(3p1), 151.

Meehl, P. E. (1990a). Appraising and amending theories: The strategy of Lakatosian defense and two principles that warrant it. *Psychological inquiry*, 1(2), 108-141.

Meehl, P. E. (1990b). Why summaries of research on psychological theories are often uninterpretable. *Psychological reports*, 66(1), 195-244.