# Lesson3: explore one variable

## Pseudo-Facebook User Data

You could also use the following code to load the Pseudo Facebook data.

read.delim('pseudo\_facebook.tsv')

The read.delim() function defaults to the tab character as the separator between values and the period as the decimal character. Run ?read.csv or ?read.delim in the console for more details.

## Histogram of Users' Birthdays

**Note: The use of the scale\_x\_discrete() layer as shown in the video is depreciated as of ggplot2 version 2.0.**

**You can use scale\_x\_continuous() instead to get the break points, or use ggplot() syntax as shown below.**

Run the following code in R to get other themes.

install.packages('ggthemes', dependencies = TRUE)

library(ggthemes)

Chris is using theme\_minimal()with the font size set to 24, which is why his output might look slightly different than yours. You can set the same theme in R by running the following code, or you can set the theme to a choice of your own.

theme\_set(theme\_minimal(24))

Instead of using the qplot() function, you can also use the ggplot() function to create the histogram:

ggplot(aes(x = dob\_day), data = pf) +

geom\_histogram(binwidth = 1) +

**scale\_x\_continuous**(breaks = 1:31)

## Faceting

ggplot(data = pf, aes(x = dob\_day)) +

geom\_histogram(binwidth = 1) +

scale\_x\_continuous(breaks = 1:31) +

**facet\_wrap(~dob\_month)**

## Limiting the Axes

ggplot(aes(x = friend\_count), data = pf) +

geom\_histogram() +

scale\_x\_continuous(**limits = c(0, 1000**))

## Adjusting the Bin Width

Equivalent ggplot syntax:

ggplot(aes(x = friend\_count), data = pf) +

geom\_histogram(binwidth = 25) +

scale\_x\_continuous(limits = c(0, 1000), breaks = seq(0, 1000, 50))

qplot(x = friend\_count, data = pf, binwidth = 25) +

scale\_x\_continuous(limits = c(0, 1000), breaks = seq(0, 1000, 50))

In the alternate solution below**, the period or dot** in the formula for facet\_grid() represents all of the other variables in the data set. Essentially, this notation splits up the data by gender and produces three histograms, each having their own row.

qplot(x = friend\_count, data = pf) +

facet\_grid(gender ~ .)

Equivalent ggplot syntax:

ggplot(aes(x = friend\_count), data = pf) +

geom\_histogram() +

scale\_x\_continuous(limits = c(0, 1000), breaks = **seq(0, 1000, 50))** +

facet\_wrap(~gender)

## Omitting NA Observations

ggplot(aes(x = friend\_count), data = **subset(pf, !is.na(gender)))** +

geom\_histogram() +

scale\_x\_continuous(limits = c(0, 1000), breaks = **seq(0, 1000, 50)**) +

facet\_wrap**(~gender**)

## Statistics 'by' Gender

table (pf$gender)

**female male**

**40254 58574**

**pf$gender: female**

**Min. 1st Qu. Median Mean 3rd Qu. Max.**

**0 37 96 242 244 4923**

by(pf$friend\_count, pf$gender, summary)

**pf$gender: male**

**Min. 1st Qu. Median Mean 3rd Qu. Max.**

**0 27 74 165 182 4917**

## Tenure

The parameter color determines the color **outline** of objects in a plot.

The parameter fill determines the color of the **area** inside objects in a plot.

You might notice how the color black and the hex code color of #099DD9 (a shade of blue) are wrapped inside of I(). The I() functions stand for 'as is' and tells qplot to use them as colors.

Learn more about what you can adjust in a plot by reading the ggplot theme documentation

Equivalent ggplot syntax:

ggplot(aes(x = tenure), data = pf) +

geom\_histogram(binwidth = 30, **color** = 'black', **fill** = '#099DD9')

Equivalent ggplot syntax:

ggplot(aes(x = tenure/365), data = pf) +

geom\_histogram(binwidth = .25, **color** = 'black', **fill** = '#F79420')

## Labeling Plots

ggplot(aes(x = tenure / 365), data = pf) +

geom\_histogram(color = 'black', fill = '#F79420') +

scale\_x\_continuous(breaks = seq(1, 7, 1), limits = c(0, 7)) +

**xlab('Number of years using Facebook') +**

**ylab('Number of users in sample')**

## User Ages

ggplot(aes(x = age), data = pf) +

geom\_histogram(binwidth = 1, fill = '#5760AB') +

scale\_x\_continuous(breaks = seq(0, 113, 5))

## Transforming Data

**"Over-dispersed"** is always relative to some particular posited distribution. For example, data might be over-dispersed compared with a Poisson distribution with that mean.

**install.packages('gridExtra')**

**library(gridExtra)**

install.packages('gridExtra')

library(gridExtra)

p1 <- ggplot (pf, aes(x= friend\_count)) +

geom\_histogram()

p2 <- p1 + **scale\_x\_log10()**

p3 <- p1 + **scale\_x\_sqrt()**

**grid.arrange**(p1,p2,p3,ncol=1)

## Add a Scaling Layer

ggplot(aes(x = friend\_count), data = pf) + scale\_x\_log10()

## Frequency Polygons

the shape of the frequency polygon depends on how our bins are set up - the height of the lines are the same as the bars in individual histograms, but the lines are easier to make a comparison with since they are on the same axis.

Equivalent ggplot syntax:

ggplot(aes(x = friend\_count, y = ..count../sum(..count..)), data = subset(pf, !is.na(gender))) +

**geom\_freqpoly(aes**(color = gender), binwidth=10) +

scale\_x\_continuous(limits = c(0, 1000), breaks = seq(0, 1000, 50)) +

xlab('Friend Count') +

ylab('Percentage of users with that friend count')

ggplot(aes(x = www\_likes), data = subset(pf, !is.na(gender))) +

geom\_freqpoly(aes(color = gender)) +

**scale\_x\_log10()**

## Box Plots

ggplot (data = subset(pf,!is.na(gender)), aes(x=gender , y= friend\_count)) +

geom\_boxplot() +

scale\_y\_continuous(limits= c(1,1000))

OR THIS EQUIVILANT CODE

ggplot (data = subset(pf,!is.na(gender)), aes(x=gender , y= friend\_count)) +

geom\_boxplot() +

**coord\_cartesian(ylim = c(1,1000))**

summary(pf$friend\_count)

## Getting Logical

sum(pf$mobile\_check\_in)/ length(as.numeric(pf$mobile\_check\_in))

# PS3:

## Price by Cut

ggplot(diamonds, aes(x= price))+

geom\_histogram(col='black', fill='blue', binwidth=10)+

scale\_x\_continuous(limits = c(0,18823), breaks= seq(326, 18823, 100))+

coord\_cartesian(xlim = c(326,1000)) +

facet\_wrap(~cut)

**by(diamonds$price, diamonds$cut, summary, digits = max(getOption('digits')))**

## Scales and Multiple Histograms

# In the two last exercises, we looked at

# the distribution for diamonds by cut.

# Run the code below in R Studio to generate

# the histogram as a reminder.

# ===============================================================

qplot(x = price, data = diamonds) + facet\_wrap(~cut)

# ===============================================================

# In the last exercise, we looked at the summary statistics

# for diamond price by cut. If we look at the output table, the

# the median and quartiles are reasonably close to each other.

# diamonds$cut: Fair

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 337 2050 3282 4359 5206 18570

# ------------------------------------------------------------------------

# diamonds$cut: Good

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 327 1145 3050 3929 5028 18790

# ------------------------------------------------------------------------

# diamonds$cut: Very Good

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 336 912 2648 3982 5373 18820

# ------------------------------------------------------------------------

# diamonds$cut: Premium

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 326 1046 3185 4584 6296 18820

# ------------------------------------------------------------------------

# diamonds$cut: Ideal

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 326 878 1810 3458 4678 18810

# This means the distributions should be somewhat similar,

# but the histograms we created don't show that.

# The 'Fair' and 'Good' diamonds appear to have

# different distributions compared to the better

# cut diamonds. They seem somewhat uniform

# on the left with long tails on the right.

ggplot(diamonds, aes(x= price))+

geom\_histogram(col='black', fill='blue', binwidth=10)+

scale\_x\_continuous(limits = c(0,18823), breaks= seq(326, 18823, 100))+

coord\_cartesian(xlim = c(326,1000)) +

**facet\_grid(cut ~ price, scales="free\_y")**

# lesson4:

## Scatterplots

You can also read in the data using the following code:

read.delim('pseudo\_facebook.tsv')

The equivalent ggplot syntax for the scatterplot:

ggplot(aes(x = age, y = friend\_count), data = pf) +

geom\_point()

## Overplotting

ggplot(pf, aes(x= age, y= friend\_count)) +

**geom\_jitter(alpha = 1/20)** +

**xlim(13, 90)**

## Coord\_trans()

ggplot(pf, aes(x= age, y= friend\_count)) +

geom\_point(alpha = 1/20, **position = position\_jitter(height = 0)**) +

xlim(13, 90) +

**coord\_trans(y= 'sqrt')**

## Conditional Means

Another warning: Version 0.4.0 of dplyr has a bug when using the median function on the summarize layer, depending on the nature of the data being summarized. You may need to cast the data as a numeric (float) type when using it on your local machine, e.g. median(as.numeric(var)).

There are other ways to work with data and create new data frames without using the dplyr package. Learn about the R functions lapply, tapply, and split in a blog post.

For more on geom\_line(), you can check the documentation here.

#install.packages('dplyr')

#library('dplyr')

#library('ggplot2')

pf.fc\_by\_age <- **group\_by** (pf, age) %>%

**summarise**(friend\_count\_mean = mean(friend\_count),friend\_count\_median= median(friend\_count),n = n()) %>%

**arrange**(age)

head(pf.fc\_by\_age)

ggplot(pf.fc\_by\_age, aes(x=age ,y=friend\_count\_mean)) +

**geom\_line()**

## Overlaying Summaries with Raw Data

**Note**: ggplot 2.0.0 changes the syntax for parameter arguments to functions when using stat = 'summary'. To denote parameters that are being set on the function specified by fun.y, use the fun.args argument, e.g.:

geom\_line(stat = 'summary', fun.y = quantile, fun.args = list(probs = .9), ... )

To zoom in, the code should use the coord\_cartesian(xlim = c(13, 90)) layer rather than xlim(13, 90) layer.

Look up documentation for coord\_cartesian() and quantile() if you're unfamiliar with them.

ggplot(pf, aes(age, friend\_count)) +

geom\_point(position = position\_jitter(height = 0),

col = 'orange',

alpha = 0.05) +

coord\_cartesian(**xlim** = c(13,70), **ylim** = c(9,1000))+

geom\_line(**stat = 'summary', fun.y = mean**)+

geom\_line(**stat = 'summary', fun.y = "quantile", fun.args = list(probs =*0.1* ),** col='blue', lty=2)+

geom\_line(**stat = 'summary', fun.y = "quantile", fun.args = list(probs =*0.5* ),** col='blue')+

geom\_line(**stat = 'summary', fun.y = "quantile", fun.args = list(probs =*0.9* ),** col='blue', lty=2 )

## Correlation

**cor**(pf$friend\_count,pf$age , **method** = 'pearson')

or

**cor.test**(pf$friend\_count,pf$age , **method** = 'pearson')

or

**with(pf, cor.test**(friend\_count, age , **method** = 'pearson'))

## Strong Correlations

ggplot (pf, aes(x=www\_likes\_received, y= likes\_received)) +

ggplot (pf, aes(x=www\_likes\_received, y= likes\_received)) +

geom\_point() +

**#this line belo DOES NOT WORK**

**#coord\_cartesian(xlim = c(0,quantile(pf$www\_likes\_received, 0.95)), ylim = quantile(pf$likes\_received, 0.95))**

xlim(0,quantile(pf$www\_likes\_received, 0.95))+

**ylim**(0,**quantile** (pf$likes\_received , 0.95))+

**geom\_smooth**(method = **'lm'**, col= **'red'**)

## Noisy Scatterplots

install.packages(alr3)

library(alr3)

data(Mitchell)

## Making Sense of Data

ggplot (Mitchell, aes(x= Month, y=Temp )) +

geom\_point()+

**scale\_x\_continuous(breaks = seq(0, 203,12))**

## A New Perspective

ggplot (Mitchell, **aes(x= Month, y=Temp )**) +

geom\_point()+

scale\_x\_continuous(breaks = seq(0, 203,12))

OR

ggplot (Mitchell, **aes(x=(Month%%12), y=Temp )**) +

geom\_point()+

scale\_x\_continuous(breaks = seq(0, 203,12))

## Understanding Noise: Age to Age Months

pf$age\_with\_months <- pf$age + (1 - pf$dob\_month / 12)

## Age with Months Means

pf.fc\_by\_age\_months <- pf %>%

group\_by(age\_with\_months) %>%

summarise(

friend\_count\_mean= mean(friend\_count),

friend\_count\_median= median(friend\_count),

n= n()) %>%

arrange(age\_with\_months)

## Noise in Conditional Means + Smoothing Conditional Means

p1 <- ggplot (data= subset(pf.fc\_by\_age, age <= 70), aes (x= age, y= friend\_count\_mean)) +

geom\_line()+

geom\_smooth()

p2 <- ggplot (data= subset(pf.fc\_by\_age\_months, age\_with\_months <= 70), aes (x= age\_with\_months, y= friend\_count\_mean)) +

geom\_line() +

geom\_smooth()

p3 <- ggplot (data= subset(pf, age <= 70), aes (x= round(age / 5)\* 5 , y= friend\_count)) +

geom\_line(stat = 'summary', fun.args= mean)

# install.packages('gridExtra')

# library('gridExtra')

**grid.arrange(p2, p1, p3, ncol=1)**

# Lesson 5

## Third Qualitative Variable

# pf <- read.delim('C:\\Users\\Ziyad\\Dropbox\\Nano Degree\\4\\4eda-course-materials\\lesson3\\pseudo\_facebook.tsv')

pf.fc\_by\_age\_gender <- pf %>%

**filter** (!is.na(gender)) %>%

group\_by(age, gender) %>%

summarise(mean\_friend\_count = mean(friend\_count),

median\_friend\_count = mean(friend\_count),

n= n()) %>%

**ungroup**() %>%

**ungroup**(**age**)

## Plotting Conditional Summaries

ggplot (data= pf.fc\_by\_age\_gender, aes(x= age,y= median\_**friend**\_count)) +

geom\_line(aes**(col=** **gender**))

#old code

ggplot (data= pf.fc\_by\_age\_gender, aes(x= age,y= friend\_count)) +

**geom\_line**(aes(col= gender), **stat** = **'summary'**, **fun.args** = **mean**)

## Wide and Long Format

install.packages("tidyr")

# only need to run this once library(tidyr) spread(subset(pf.fc\_by\_age\_gender, select = c('gender', 'age', 'median\_friend\_count')), gender, median\_friend\_count)

## Ratio Plot

ggplot (data= pf.fc\_by\_age\_gender.wide, aes(x= age,**y= female / male**)) +

geom\_line()+

**geom\_hline**(**yintercept** = 1, **lty**= 2, **alpha** = 0.3)

## Third Quantitative Variable}

pf$year\_joined <- **floor**(2014 - pf$tenure/365)

## Plotting it All Together}

ggplot (subset(pf, !is.na(year\_joined.bucket)), aes(x= age, y= friend\_count))+

**geom\_line**(**aes(col= year\_joined.bucket**), **stat = 'summary'**, **fun.args= median**)

## Plot the Grand Mean

ggplot (subset(pf, !is.na(year\_joined.bucket)), aes(x= age, y= friend\_count))+

geom\_line(aes(col= year\_joined.bucket), stat = 'summary', fun.args= mean) +

**geom\_line(stat = 'summary', fun.args = mean, lty= 2)**

## Friending Rate

**with (subset(pf, tenure >0 ), summary(friend\_count/ tenure))**

## Bias Variance Trade off Revisited

NOTE: The code changing the binning is substituting x = tenure in the plotting expressions with x = 7 \* round(tenure / 7), etc., binning values by the denominator in the round function and then transforming back to the natural scale with the constant in front.

Number of Purchases

yo$all.purchases <- yo$strawberry + yo$blueberry + yo$pina.colada + yo$plain + yo$mixed.berry

#the below code line is equla to the above code line

# yo <- **transform**(**yo**, **all.purchases** = strawberry + blueberry + pina.colada + plain + mixed.berry)

**Note**: x %in% y returns a logical (boolean) vector the same length as x that says whether each entry in x appears in y. That is, for each entry in x, it checks to see whether it is in y.

This allows us to subset the data so we get all the purchases occasions for the households in the sample. Then, we create scatterplots of price vs. time and facet by the sample id.

Use the pch or shape parameter to specify the symbol when plotting points. Scroll down to 'Plotting Points' on QuickR's Graphical Parameters.

Looking at Sample of Households}

**set.seed(4230)**

**sample.id <- sample (levels(yo$id), 16)**

ggplot (data= subset(yo, **id%in% sample.id**), aes(x= time,y= price))+

facet\_wrap(~id)+

geom\_line()+

geom\_point(aes(size= all.purchases), **pch**= 1)

## Scatterplot Matrices

You may also find that variable labels are on the outer edges of the scatterplot matrix, rather than on the diagonal. If you want labels in the diagonal, you can set the axisLabels = 'internal' argument in your ggpairs command.

#the below code will take a long ime around 1 hour to run

install.packages("GGally")

library(GGally)

theme\_set(theme\_minimal(20))

set.seed(1836)

pf\_subset <- pf[,c(2:15)]

names(pf\_subset)

ggpairs (pf\_[sample.int(pf\_subset), 1000],)

## Scatterplot Review

# library(ggplot2)

data(diamonds)

ggplot (diamonds, aes(x=carat ,y= price)) +

geom\_point()+

**scale\_x\_continuous(lim = c(0,quantile(diamonds$carat,0.99)))+**

**scale\_y\_continuous(lim = c(0,quantile(diamonds$price,0.99)))**

# PS5:

## Price Histograms with Facet and Color

ggplot(diamonds, aes(x= price, col=cut)) +

geom\_histogram()+

facet\_wrap(~color)+

**scale\_fill\_brewer(type = 'qual')**

## Price vs. Table Colored by Cut

ggplot(diamonds, aes(x= table ,y= price ,col= cut))+

geom\_point()+

**scale\_color\_brewer(type = 'qual')**

# Lesson 6

## ggpairs Function

The "params" argument to the ggpairs function are there to change the shape of the plotted points in the plot matrix, to make them easier to see. GGally 1.0 changes the syntax of these plotting parameters to no longer be part of a params argument, and instead can be specified as follows:

ggpairs(diamond\_samp,

lower = list(continuous = wrap("points", shape = I('.'))),

upper = list(combo = wrap("box", outlier.shape = I('.'))))

```{r ggpairs Function}

# install these if necessary

install.packages('GGally')

install.packages('scales')

install.packages('memisc')

install.packages('lattice')

install.packages('MASS')

install.packages('car')

install.packages('reshape')

install.packages('plyr')

# load the ggplot graphics package and the others

library(ggplot2)

library(GGally)

library(scales)

library(memisc)

# sample 10,000 diamonds from the data set

set.seed(20022012)

diamond\_samp <- diamonds[sample(1:length(diamonds$price), 10000), ]

ggpairs(diamond\_samp, params = c(shape = I('.'), outlier.shape = I('.')))

## The Demand of Diamonds

library(gridExtra)

plot1 <- qplot(data =diamonds, x= price, binwidth= 100, **fill**= **I**('blue')) +

ggtitle('Price')

plot2 <- qplot(data =diamonds, x= price , binwidth= 0.01, **fill**= **I**('orange')) +

ggtitle('Price (log10)')+

**scale\_x\_log10()**

**grid.arrange(plot1, plot2, ncol= 2)**

## Scatterplot Transformation

qplot(data =diamonds, x= carat, y= price) +

**scale\_y\_continuous(trans = log10\_trans())+**

**ggtitle**('Price (log10) by carat')

## Use cuberoot\_trans

ggplot(aes(carat, price), data = diamonds) +

geom\_point() +

scale\_x\_continuous(**trans = cuberoot\_trans(),** **limits = c(0.2, 3),**

breaks = c(0.2, 0.5, 1, 2, 3)) +

scale\_y\_continuous(**trans = log10\_trans(),** **limits = c(350, 15000),**

breaks = c(350, 1000, 5000, 10000, 15000)) +

ggtitle('Price (log10) by Cube-Root of Carat')

## Overplotting Revisited

ggplot(aes(carat, price), data = diamonds) +

geom\_point(**position** = **'jitter'**, alpha= 1/2, size= 3/4, )+

scale\_x\_continuous(trans = cuberoot\_trans(), limits = c(0.2, 3),

breaks = c(0.2, 0.5, 1, 2, 3)) +

scale\_y\_continuous(trans = log10\_trans(), limits = c(350, 15000),

breaks = c(350, 1000, 5000, 10000, 15000)) +

ggtitle('Price (log10) by Cube-Root of Carat')

## Building the Linear Model

m1 <- lm(I(log(price)) ~ I(carat^(1/3)), data = diamonds)

m2 <- update(m1, ~ . + carat)

m3 <- update(m2, ~ . + cut)

m4 <- update(m3, ~ . + color)

m5 <- update(m4, ~ . + clarity)

mtable(m1, m2, m3, m4, m5, sdigits = 3)

## Model Problems

This vid is full of recources, C H E C K I T O U T