Statistics and Data Science I (ABC) CHEAT SHEET

CourseKata

Basics

```
print("Hello world!")

# assign value to object
myNumber <- 5

# combine values into vector
myVector <- c(1, 2, 3)

# first element in vector
myVector[1]

# orders values or cases
sort(myVector)

# arithmetic operations
sum(1, 2, 100), +, -, *, /
sqrt(157)
abs(data_set$Y)

# logical operations
>, <, >=, <=, ==, !=, |, &

# results in a variable with values
# of TRUE or FALSE
data_set$C <- data_set$A > data_set$B
```

Probability Distribution

```
# calculate the probability area
xpnorm(65.1, data_set$mean, data_set$sd)

zscore(data_set$Y)

# returns t at this probability
qt(.975, df = 999)
# returns F at this probability
qf(.95, df1 = 1, df2 = 100)

# CI using t distribution
confint(empty_model)


# calculate p-value using F-distribution
xpf(sample_F, df1 = 2 , df2 = 10)
```

Summary Tables

```
# compute five-number summary
favstats(~ Y, data = data_set)

# create frequency table
tally(data_set$Y)
tally(~ Y, data = data_set)

# tally by condition
tally(~ Y < 1900, data = data_set)

# two-way frequency table
tally(Y ~ X, data = data_set, margin = TRUE,
format = "proportion")
```

Simple Statistics

```
mean(data_set$Y)
var(data_set$Y)
sd(data_set$Y)

cohensD(Y ~ X, data = data_set)
cor(Y ~ X, data = data_set)

b1(Y ~ X, data = data_set)
b1(one_model)

pre(Y ~ X, data = data_set)
f(Y ~ X, data = data_set)
```

Data Frame

```
# structure of data frame
str(data_set)

# view first/last six rows
head(data_set)
tail(data_set)

# select multiple variables
select(data_set, Y1, Y2)

# first six rows of selected variables
head(select(data_set, Y1, Y2))


# select variable (a column)
data_set$Y

# find rows that meet condition
data_set[data_set$Y > 40]
filter(data_set, Y > 300)
filter(data_set, Y != "NA")
```

```
# arrange rows by variable
arrange(data_set, Y)

# creates data frame from csv file
data_set <- read.csv("file_name", header = TRUE)

# convert quantitative variable
# to categorical
factor(data_set$Y)
factor(data_set$Y, levels = c(1,2), labels =
c("A", "B"))

# transform values
recode(data_set$Y, "0" = 0, "1" = 50, "2" = 100)

# creates two equal sized groups
ntile(data_set$Y, 2)

# convert categorical variable
# to quantitative
as.numeric(data_set$Y)
```

Simulation

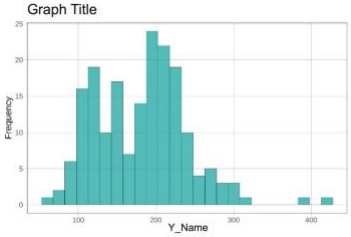
```
# sample without replacement
sample(data_set, 6)

# sample with replacement
resample(data_set, 10)

do(3) * resample (data_set, 10)

# mixes up values in a variable
shuffle(data_set$Y)

# simulate sampling 10000 Ys
# from normal distribution
sim_Y <- rnorm(10000, Y_stats$mean,
Y_stats$sd)

# put simulated Ys into dataframe
data_set<- data.frame(sim_Y)

# simulate sampling distribution of
means
sim_SDoM <- do(10000) * mean(rnorm(157,
Y_stats$mean, Y_stats$sd))

# bootstrap sampling distribution of
means
bootSDoM <- do(10000) *
mean(resample(data_set$Y, 157))

# randomize sampling distribution
# of b1s, centered on 0
sdob1 <- do(1000) *
  b1(shuffle(Y) ~ X, data = data_set)

# bootstrap sampling distribution of b1s,
# centered on sample b1
sdob1_boot <- do(1000) *
  b1(Y ~ X, data = resample(data_set))

# count the number of b1s at the upper
# and lower extreme
tally(sdob1$b1 > sample_b1 |
sdob1$b1 < -sample_b1)


# return TRUE for middle 95% of distribution
middle(sdob1$b1, .95)

# randomize sampling distribution of PREs
sdoPRE <- do(1000) * PRE(shuffle(Y) ~ X,
data = data_set)

# randomize sampling distribution of Fs
sdoF <- do(1000) *
  fVal(shuffle(Y) ~ X, data = data_set)

# counts extreme Fs
tally(~fVal > sample_F, data = sdoF)
```

Fitting and Evaluating Models

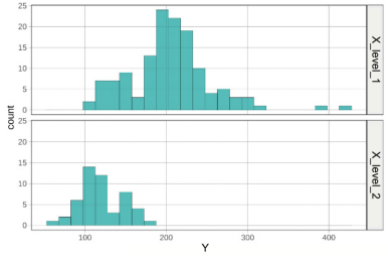
```
# empty model
empty_model <- lm(Y ~ NULL,
data = data_set)

# use one expanatory variable
one_model <- lm(Y ~ X, data = data_set)

# create a function from a formula
one_model_fun <- makeFun(one_model)

one_model_fun(x_level_1)

# model predictions and residuals
data_set$empty_predict <- predict(empty_model)
data_set$empty_resid <- resid(empty_model)

# produce ANOVA table
anova(empty_model)
supernova(one_model)

# t-test, using pooled variance
t.test(Tip ~ Condition, data = data_set,
var.equal=TRUE)

# pairwise comparison
# corrections: "Bonferroni" or "none"
pairwise(one_model, correction = "none")
```

Visualizations

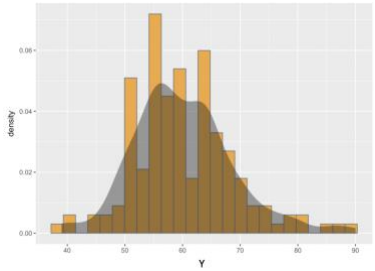
```
gf_histogram(~ Y, data = data_set) %>%
# change labels
  gf_labs(title = "Graph Title", x = "Y_Name",
y = "Frequency")
```



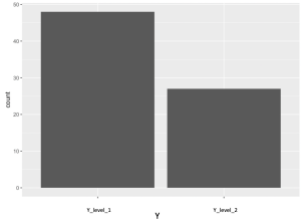

```
# faceted grid of histograms
gf_histogram(~ Y, data = data_set) %>%
  gf_facet_grid(X ~ .)
```



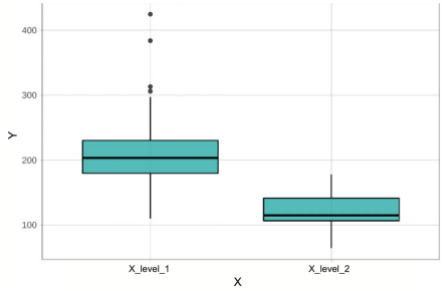

```
gf_dhistogram(~ Y, data = data_set, fill =
"orange", color = "slategray") %>%
  gf_density()
```



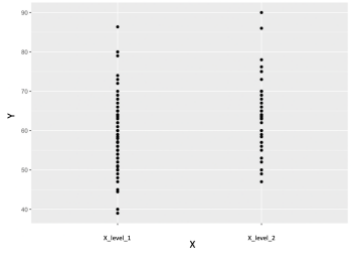

```
gf_bar( ~ Y, data = data_set)
```



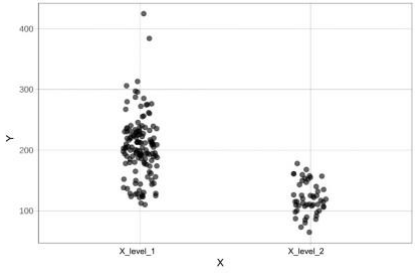

```
gf_boxplot(Y ~ X, data = data_set)
```



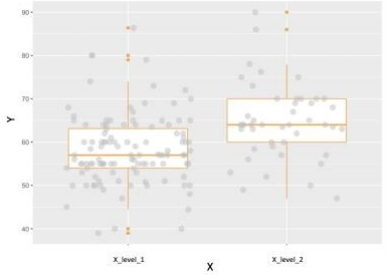

```
gf_point(Y ~ X, data = data_set)
```



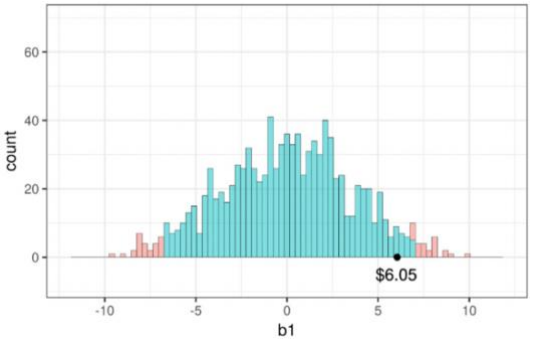

```
gf_jitter(Y ~ X, data = data_set)
```



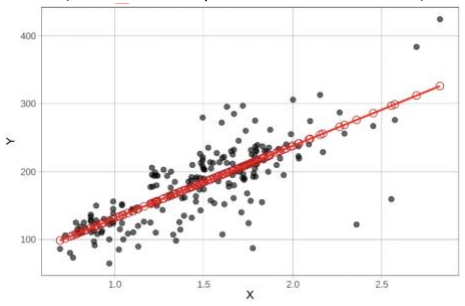

```
gf_boxplot(Y ~ X, data = data_set,
color = "orange") %>%
 gf_jitter(height = 0, color = "grey"),
alpha = .5, size = 3)
```




```
# sampling distribution of b1
gf_histogram(~b1, data = sdob1,
 fill = ~middle(b1, .95)) %>%
 # modify the limits on x- and y-axes
 gf_lims(x = c(-12, 12), y = c(0, 70))
```




```
gf_point(Y ~ X, data = data_set) %>%
 # add model predictions as red points
 gf_point(Y ~ X , shape = 1, size = 3,
color = "firebrick") %>%
 # add best fitting model as a red line
 gf_model(one_model, color = "red")
```




```
pairwise(one_model, plot = TRUE)
```