R Reference Card

Help for R studio Open the cheat sheet for RStudio by selecting the Help menu -> Cheatsheets -> RStudio IDE Cheat Sheet. Note the cheatsheet will usually be downloaded in a web browser as a .pdf.

Installing Packages Use the argument dependencies=TRUE to load all other packages required by the targeted package.

Loading Packages After installing a package, we can load it into the R environment using the library() or require() functions, which more or less do the same thing.

Use library(dplyr) The package dplyr contains several easy-to-use data management functions that we will learn to use to manage our data.

Starting R and RStudio Launch R Studio, R starts automatically within RStudio

Quitting R Studio Use q() Type yes or no to confirm choice

Loading data set (CSV File) my_data <- read.csv("path/to/your/data.csv") # Load data from a CSV file**Viewing Data**: After loading your data, you can inspect it using the head() function to view the first few rows.

Building a Linear Regression Model: You can build a linear regression model using the `lm()` function. For example, let's predict miles per gallon (mpg) based on weight (wt) and horsepower (hp) using the mtcars dataset.

model <- lm(mpg ~ wt + hp, data = mtcars) # Build a linear model

Model Interpretation The summary(model) function provides key statistics, such as coefficients, p-values, R-squared value, etc., which help assess the model's fit and significance.

Understanding data

str() Displays the structure of the dataset, including variable types.

summary(): Provides summary statistics for each variable (e.g., min, max, mean, median).

head(): Displays the first few rows of the data.

Evaluating a model Residuals

residuals <- resid(model) # Get residuals from the model plot(residuals) # Plot residuals

R-squared: The **R-squared value** from the model summary indicates how well the independent variables explain the variation in the dependent variable (mpg). Higher R-squared values (close to 1) indicate a better fit.

Interacting with RStudio Console: This is where you enter R commands. It executes your commands and displays results. You can type code directly here for quick tests or experimentation.

Environment/History Pane: Displays the objects (variables, models, etc.) created in your R session and the history of previously executed commands.

Plots Pane: This is where graphical outputs (like plots and charts) are displayed. You can save plots using the "Export" button for further use.

Packages Pane: View and manage the libraries and packages installed in R. You can install new packages here and load them for use.

Interacting with R

Basic Arithmetic

3 + 2 # Addition

10 / 2 # Division

4 * 5 # Multiplication

2^3 # Exponentiation

Creating Variables: assign values to variables using the <-

x <- 10 # Assign the value 10 to x

y <- 20 # Assign the value 20 to y

Frequently used commands

summary(): Gives an overview of a dataset or model.

str(): Displays the structure of an object, such as a dataset or model.

mean() and sd(): Used to calculate the mean and standard deviation of a variable.

lm(): Fits a linear model to data.

Basic Plotting: Use plot()

References

Group, I. S. C. (n.d.-b). Introduction to R. https://stats.oarc.ucla.edu/stat/data/intro_r/intro_r_interactive.html#(15)

FOR THE R CHEAT SHEET

Basic Operation

- # this is a comment in R
- Use x <- 3 to assign a value, 3, to a variable, x
- R counts from 1, unlike many other programming languages (e.g., Python)
- length(collection) returns the number of elements contained in the variable collection
- c(value1, value2, value3) creates a vector
- container[i] selects the i'th element from the variable container

https://swcarpentry.github.io/r-novice-inflammation/reference.html#glossary

1.7 The R Zone

Getting Started with R

Comments, indents, and semicolons

```
# Anything prefaced by a pound sign (#) is a comment.
# Comments are not executed by R. Instead, they explain what the code is doing.
# Indented code (that is not a comment) will run in R as if it was on one line
# Code separated by semicolons will run as if the code was on separate lines,
# with the semicolon marking the line break
```

Open a dataset and display the data

```
# Replace "C:/…/" with the exact location of the file you want to open
cars <- read.csv(file = "C:/…/cars.txt",
stringsAsFactors = FALSE)
cars # To display the whole dataset, type the dataset name
head(cars) # Display the first few records of a dataset
names(cars) # Display variable names of a data frame, one kind of data in R
cars$weight # Look at only the weight variable within data frame cars
```

Matrices

```
# Create a matrix with three rows, two columns, and every value equal to 0.0
mat <- matrix(0.0, nrow = 3, ncol = 2); mat
colnames(mat) <- c("Var 1", "Var 2") # Give a matrix variable names
colnames(mat) # Display variable names of a matrix
```

Subset data and declare new variables

```
cars.rsub <- cars[1:50,] # Subset the data by rows
cars.csub <- cars[,1:3] # Subset by columns
cars.rcsub <- cars[c(1,3,5), c(2,4)] # Subset by specific rows and columns
cars.vsub <- cars[which(cars$mpg> 30),] # Subset by a logical condition
# To declare new variables, type the variable name, a left-arrow, then the valu
firstletter <- "a"
weight <- cars$weight
```

Display more than one figure at a time

```
par(mfrow=c(1,1)) # plots one figure; the default setting
par(mfrow=c(2,3)) # plots six figures: three in the top row, three in the botto
# Plots will fill the plot space row by row
```

Download and install an R Package

```
# Example: ggplot2, from Lesson 3. install.packages("ggplot2")
install.packages("ggplot2")
# Pick any CRAN mirror, as shown
# Open the new package
library(ggplot2)
```

2.22 The R Zone

Read in the Cars and Cars2 datasets

```
cars <- read.csv("C:/ …/cars.txt",
stringsAsFactors = FALSE)
cars2 <- read.csv("C:/ …/cars2.txt",
stringsAsFactors = FALSE)
```

Missing data

```
# Look at four variables from cars
cars.4var <- cars[, c(1, 3, 4, 8)]
head(cars.4var)
```

```
> head(cars.4var)
   mpg cubicinches  hp  brand
1 14.0         350 165     US
2 31.9          89  71 Europe
3 17.0         302 140     US
4 15.0         400 150     US
5 30.5          98  63     US
6 23.0         350 125     US
```

```
# Make certain entries missing
cars.4var[2,2] <- cars.4var[4,4] <- NA
head(cars.4var)
```

```
> head(cars.4var)
   mpg cubicinches  hp  brand
1 14.0         350 165     US
2 31.9          NA  71 Europe
3 17.0         302 140     US
4 15.0         400 150   <NA>
5 30.5          98  63     US
6 23.0         350 125     US
```

```
# Replace missing values with constants
cars.4var[2,2] <- 0
cars.4var[4,4] <- "Missing"
head(cars.4var)
```

```
> head(cars.4var)
   mpg cubicinches  hp   brand
1 14.0         350 165      US
2 31.9           0  71  Europe
3 17.0         302 140      US
4 15.0         400 150 Missing
5 30.5          98  63      US
6 23.0         350 125      US
```

```
# Replace values with mean and mode
cars.4var[2,2] <-
mean(na.omit(cars.4var$cubicinches))
our_table <- table(cars.4var$brand)
our_mode <- names(our_table)[our_table ==
max(our_table)]
cars.4var[4,4] <- our_mode
head(cars.4var)
```

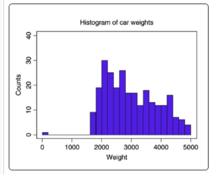
```
> head(cars.4var)
   mpg cubicinches  hp  brand
1 14.0    350.0000 165     US
2 31.9    201.5346  71 Europe
3 17.0    302.0000 140     US
4 15.0    400.0000 150     US
5 30.5     98.0000  63     US
6 23.0    350.0000 125     US
```

```
# Generate random observations
obs_brand <-
   sample(na.omit(cars.4var$brand), 1)
obs_cubicinches <-
   sample(na.omit(cars.4var$cubicinches), 1)
cars.4var[2,2] <- obs_cubicinches
cars.4var[4,4] <- obs_brand
head(cars.4var)
```

```
> head(cars.4var)
   mpg cubicinches  hp  brand
1 14.0         350 165     US
2 31.9          86  71 Europe
3 17.0         302 140     US
4 15.0         400 150 Europe
5 30.5          98  63     US
6 23.0         350 125     US
```

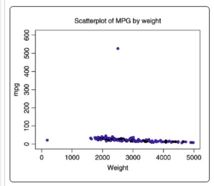
Create a Histogram

```
# Set up the plot area
par(mfrow = c(1,1))
# Create the
histogram bars
hist(cars2$weight,
   breaks = 30,
   xlim = c(0, 5000),
   col = "blue",
   border = "black",
   ylim = c(0, 40),
   xlab = "Weight",
   ylab = "Counts",
   main = "Histogram
      of Car Weights")
# Make a box around
# the plot
box(which = "plot",
   lty = "solid",
   col = "black")
```



Create a Scatterplot

```
plot(cars2$weight, cars2$mpg,
xlim = c(0, 5000),
ylim = c(0, 600),
xlab = "Weight",
ylab = "MPG",
main = "Scatterplot
of MPG by
Weight",
type = "p",
pch = 16,
col = "blue")
#Add open black # circles
points(cars2$weight,
cars2$mpg,
type = "p",
col = "black")
```



Descriptive Statistics

```
mean(cars$weight) # Mean
median(cars$weight) # Median
length(cars$weight) # Number of observations
sd(cars$weight) # Standard deviation
summary(cars$weight) # Min, Q1, Median, Mean, Q3, Max
```

Transformations

```
# Min-max normalization
summary(cars$weight)
mi <- min(cars$weight)
ma <- max(cars$weight)
minmax.weight <- (cars$weight - mi)/(ma - mi)
minmax.weight
```

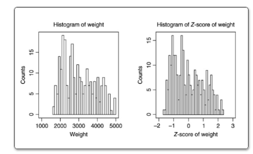
```
# Z-score standarization
m <- mean(cars$weight); s <- s
z.weight <- (cars$weight - m)/
z.weight
length(cars$weight)
```

```
# Decimal scaling
max(abs(cars$weight)) # 4 digits
d.weight <- cars$weight/(10^4); d.weight
```

Side-by-Side Histograms

```
par(mfrow = c(1,2))
# Create two histograms
hist(cars$weight, breaks = 20,
xlim = c(1000, 5000),
main = "Histogram of Weight",
xlab = "Weight",
ylab = "Counts")
box(which = "plot",
lty = "solid",
col = "black")
```

```
hist(z.weight,
breaks = 20,
xlim = c(-2, 3),
main = "Histogram of Z-
score of Weight",
xlab = "Z-score of Weight",
ylab = "Counts")
box(which = "plot",
lty = "solid",
col = "black")
```



Skewness

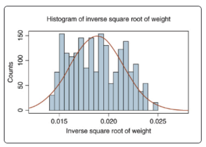
```
(3*(mean(cars$weight) - median(cars$weight)))/sd(cars$weight)
(3*(mean(z.weight) - median(z.weight)))/sd(z.weight)
```

Transformations for Normality

```
sqrt.weight <- sqrt(cars$weight) # Square root
sqrt.weight_skew <- (3*(mean(sqrt.weight) - median(sqrt.weight))) / sd(sqrt.we
ln.weight <- log(cars$weight) # Natural log
ln.weight_skew <- (3*(mean(ln.weight) - median(ln.weight))) / sd(ln.weight)
invsqrt.weight <- 1 / sqrt(cars$weight) # Inverse square root
invsqrt.weight_skew <- (3*(mean(invsqrt.weight) - median(invsqrt.weight))) /sd
```

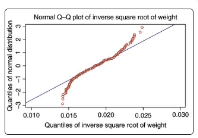
Histogram with Normal Distribution Overlay

```
par(mfrow=c(1,1))
x <- rnorm(1000000,
mean = mean (invsqrt.weight),
sd = sd(invsqrt.weight))
hist(invsqrt.weight,
breaks = 30,
xlim = c(0.0125, 0.0275),
col = "lightblue",
prob = TRUE,
border = "black",
xlab = "Inverse Square
Root of Weight",
ylab = "Counts",
main = "Histogram of  Inverse Square Root
of Weight")
box(which = "plot",
lty = "solid",
col="black")
# Overlay with  Normal density
lines(density(x), col = "red")
```



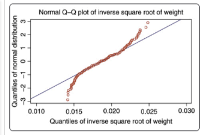
Normal Q-Q Plot

```
qqnorm(invsqrt.weight,
datax = TRUE,
col = "red",
ylim = c(0.01, 0.03),
main = "Normal Q-Q Plot of Inverse
Square Root of Weight")
qqline(invsqrt.weight,
col = "blue",
datax = TRUE)
```



Normal Q-Q Plot

```
qqnorm(invsqrt.weight,
datax = TRUE,
col = "red",
ylim = c(0.01, 0.03),
main = "Normal Q-Q Plot of Inverse
Square Root of Weight")
qqline(invsqrt.weight,
col = "blue",
datax = TRUE)
```

De-transform data

```
# Transform x using y = 1 / sqrt(x)
x <- cars$weight[1]; y <- 1 / sqrt(x)
# Detransform x using x = 1 / (y)^2
detransformedx <- 1 / y^2
x; y; detransformedx
```

```
> x; y; detransformedx
[1] 4209
[1] 0.01541383
[1] 4209
```

Create indicator variables

```
north_flag <- east_flag <- south_flag <-
c(rep(NA, 10))
region <- c(rep(c("north", "south", "east",
"west"),2), "north", "south")
# Change the region variable to indicators
for (i in 1:length(region)) {
if(region[i] == "north") north_flag[i] = 1
else north_flag[i] = 0
if(region[i] == "east") east_flag[i] = 1
else east_flag[i] = 0
if(region[i] == "south") south_flag[i] = 1
else south_flag[i] = 0
}
north_flag; east_flag; south_flag
```

```
> north_flag; east_flag; south_flag
[1] 1 0 0 0 1 0 0 0 1 0
[1] 0 0 1 0 0 0 1 0 0 0
[1] 0 1 0 0 0 1 0 0 0 1
```

Index fields

```
# Data frames have an index field;
# the left-most column
cars
cars[order(cars$mpg),]
```

```
# For vectors or matrices,
# add a column to act as an index field
x <- c(1,1,3:1,1:4,3); y <- c(9,9:1)
z <- c(2,1:9)
matrix <- t(rbind(x,y,z)); matrix
indexed_m <- cbind(c(1:length(x)), matrix
indexed_m[order(z),]
```

Duplicate records

```
er of duplicate records, use anyDuplicated
ted(cars)
ne each record, use Duplicated
(cars)
record is a duplicate,
 record is not a duplicate
```

```
# Let's duplicate the first record
new.cars <- rbind(cars, cars[1,])
# Check for duplicates
anyDuplicated(new.cars)
# The 262nd record is a duplicate
duplicated(new.cars)
```

CHAPTER 3

3.13 The R Zone

Read in the Churn data set

```
churn <- read.csv(file =
        "C:/…/churn.txt",
        stringsAsFactors=TRUE)
# Show the first ten records
churn[1:10,]
```

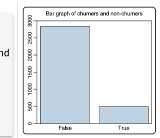
```
> churn[1:10,]
   State Account.Length Area.Code    Phone Int.l.Plan
1     KS            128       415 382-4657         no
2     OH            107       415 371-7191         no
3     NJ            137       415 358-1921         no
4     OH             84       408 375-9999        yes
5     OK             75       415 330-6626        yes
6     AL            118       510 391-8027        yes
7     MA            121       510 355-9993         no
8     MO            147       415 329-9001        yes
9     LA            117       408 335-4719         no
10    WV            141       415 330-8173        yes
```

```
# Summarize the Churn variable
sum.churn <- summary(churn$Churn)
sum.churn
```

```
# Calculate proportion of churners
prop.churn <- sum(churn$Churn ==
        "True") / length(churn$Churn)
prop.churn
```

Bar chart of variable Churn

```
barplot(sum.churn,
        ylim = c(0, 3000),
        main = "Bar Graph of Churners and
        Non-Churners",
        col = "lightblue")
box(which = "plot",
        lty = "solid",
        col="black")
```



Create a table of proportions over rows

```
row.margin <- round(prop.table(counts,
                margin = 1),
        4)*100
row.margin
```

```
> row.margin
       International Plan
Churn      no   yes
  False 93.47  6.53
  True  71.64 28.36
```

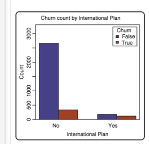
Create a table of proportions over columns

```
col.margin <- round(prop.table(counts,
                margin = 2),
        4)*100
col.margin
```

```
> col.margin
       International Plan
Churn      no   yes
  False 88.50 57.59
  True  11.50 42.41
```

Clustered Bar Chart, with legend

```
barplot(counts,
        col = c("blue", "red"),
        ylim = c(0, 3300),
        ylab = "Count",
        xlab = "International Plan",
        main = "Churn Count by
                International Plan",
        beside = TRUE)
legend("topright",
        c(rownames(counts)),
        col = c("blue", "red"),
        pch = 15,
        title = "Churn")
box(which = "plot",
        lty = "solid",
        col="black")
```



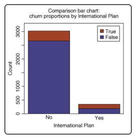
Make a table for counts of Churn and International Plan

```r
counts <- table(churn$Churn,
   churn$Int.l.Plan,
       dnn=c("Churn", "International Plan"))
counts
```

```
> counts
       International Plan
Churn    no  yes
  False 2664  186
  True   346  137
```

#Overlayed bar chart

```r
barplot(counts,
       legend = rownames(counts),
       col = c("blue", "red"),
       ylim = c(0, 3300),
       ylab = "Count",
       xlab = "International Plan",
       main = "Comparison Bar Chart:
           Churn Proportions by
           International Plan")
box(which = "plot",
       lty = "solid",
       col="black")
```



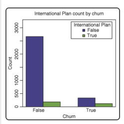
Create a table with sums for both variables

```r
sumtable <- addmargins(counts,
       FUN = sum)
sumtable
```

```
> sumtable
       International Plan
Churn    no  yes  sum
  False 2664  186 2850
  True   346  137  483
  sum   3010  323 3333
```

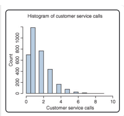
Clustered Bar Chart of Churn and International Plan with legend

```r
barplot(t(counts),
       col = c("blue", "green"),
       ylim = c(0, 3300),
       ylab = "Counts",
       xlab = "Churn",
       main = "International Plan Count by
           Churn",
       beside = TRUE)
legend("topright",
       c(rownames(counts)),
       col = c("blue", "green"),
       pch = 15,
       title = "Int'l Plan")
box(which = "plot",
       lty = "solid",
       col="black")
```



Histogram of non-overlayed Customer Service Calls

```r
hist(churn$CustServ.Calls,
       xlim = c(0,10),
       col = "lightblue",
       ylab = "Count",
       xlab = "Customer Service Calls",
       main = "Histogram of Customer Service
           Calls")
```



Two-sample T-Test for Int'l Calls

```r
# Partition data
churn.false <- subset(churn,
       churn$Churn ==
           "False")
churn.true <- subset(churn,
       churn$Churn ==
           "True")
# Run the test
t.test(churn.false$Intl.Calls,
       churn.true$Intl.Calls)
```

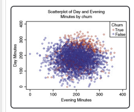
```
> t.test(churn.false$Intl.Calls,
+        churn.true$Intl.Calls)

 Welch Two Sample t-test

data:  churn.false$Intl.Calls and churn.true$Intl.Calls
t = 2.9604, df = 640.643, p-value = 0.003186
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
 0.1243807 0.6144620
sample estimates:
mean of x mean of y
 4.532982  4.163561
```

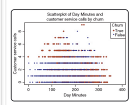
Scatterplot of Evening Minutes and Day Minutes, colored by Churn

```r
plot(churn$Eve.Mins,
       churn$Day.Mins,
       xlim = c(0, 400),
       ylim = c(0, 400),
       xlab = "Evening Minutes",
       ylab = "Day Minutes",
       main = "Scatterplot of Day
           and Evening Minutes by
           Churn",
       col = ifelse(churn$Churn==
           "True",
           "red",
           "blue"))
legend("topright",
           c("True",
           "False"),
       col = c("red",
           "blue"),
       pch = 1,
       title = "Churn")
```



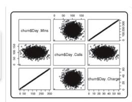
Scatterplot of Day Minutes and Customer Service Calls, colored by Churn

```r
plot(churn$Day.Mins,
       churn$CustServ.Calls,
       xlim = c(0, 400),
       xlab = "Day Minutes",
       ylab = "Customer Service Calls",
       main = "Scatterplot of Day Minutes and
           Customer Service Calls by Churn",
       col = ifelse(churn$Churn=="True",
           "red",
           "blue"),
       pch = ifelse(churn$Churn=="True",
           16, 20))
legend("topright",
       c("True",
           "False"),
       col = c("red",
           "blue"),
       pch = c(16, 20),
       title = "Churn")
```



Scatterplot matrix

```r
pairs(~churn$Day.Mins+
       churn$Day.Calls+
       churn$Day.Charge)
```



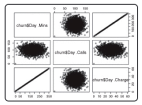
Correlation values and p-values in matrix form

```r
# Collect variables of interest
corrdata <-
       cbind(churn$Account.Length,
           churn$VMail.Message,
           churn$Day.Mins,
           churn$Day.Calls,
           churn$CustServ.Calls)
# Declare the matrix
corrpvalues <- matrix(rep(0, 25),
       ncol = 5)
# Fill the matrix with correlations
for (i in 1:4) {
       for (j in (i+1):5) {
           corrpvalues[i,j] <-
           corrpvalues[j,i] <-
               round(cor.test(corrdata[,i],
                   corrdata[,j])$p.value,
                   4)
       }
}
round(cor(corrdata), 4)
corrpvalues
```

```
> round(cor(corrdata), 4)
        [,1]    [,2]    [,3]    [,4]    [,5]
[1,]  1.0000 -0.0046  0.0062  0.0385 -0.0038
[2,] -0.0046  1.0000  0.0008 -0.0095 -0.0133
[3,]  0.0062  0.0008  1.0000  0.0068 -0.0134
[4,]  0.0385 -0.0095  0.0068  1.0000 -0.0189
[5,] -0.0038 -0.0133 -0.0134 -0.0189  1.0000
> corrpvalues
       [,1]   [,2]   [,3]   [,4]   [,5]
[1,] 0.0000 0.7894 0.7198 0.0264 0.8266
[2,] 0.7894 0.0000 0.9642 0.5816 0.4440
[3,] 0.7198 0.9642 0.0000 0.6969 0.4385
[4,] 0.0264 0.5816 0.6969 0.0000 0.2743
[5,] 0.8266 0.4440 0.4385 0.2743 0.0000
```

Scatterplot matrix

```r
pairs(~churn$Day.Mins+
       churn$Day.Calls+
       churn$Day.Charge)
```



Regression of Day Charge vs Day Minutes

```r
fit <- lm(churn$Day.Charge ~
              churn$Day.Mins)
summary(fit)
```

```
> summary(fit)

Call:
lm(formula = churn$Day.Charge ~ churn$Day.Mins)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0045935 -0.0025391  0.0004326  0.0024587  0.0045224

Coefficients:
                Estimate Std. Error   t value Pr(>|t|)
(Intercept)    6.134e-04  1.711e-04 3.585e+00 0.000341 ***
churn$Day.Mins 1.700e-01  9.108e-07 1.866e+05  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.002864 on 3331 degrees of freedom
Multiple R-squared:     1, Adjusted R-squared:     1
F-statistic: 3.484e+10 on 1 and 3331 DF,  p-value: < 2.2e-16
```

Correlation values, with p-values

```r
days <- cbind(churn$Day.Mins,
       churn$Day.Calls,
       churn$Day.Charge)
MinsCallsTest <- cor.test(churn$Day.Mins,
       churn$Day.Calls)
MinsChargeTest <- cor.test(churn$Day.Mins,
       churn$Day.Charge)
CallsChargeTest <- cor.test(churn$Day.Calls,
       churn$Day.Charge)
round(cor(days),
       4)
MinsCallsTest$p.value
MinsChargeTest$p.value
CallsChargeTest$p.value
```

```
> round(cor(days), 4)
       [,1]   [,2]   [,3]
[1,] 1.0000 0.0068 1.0000
[2,] 0.0068 1.0000 0.0068
[3,] 1.0000 0.0068 1.0000
> MinsCallsTest$p.value
[1] 0.6968515
> MinsChargeTest$p.value
[1] 0
> CallsChargeTest$p.value
[1] 0.6967428
```

Download and install the R Package *ggplot2*

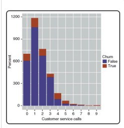
```r
install.packages("ggplot2")
# Pick any CRAN mirror
# (see example image)
# Open the new package
library(ggplot2)
```

```
79: USA (CA 1)                80: USA (CA 2)
81: USA (IA)                  82: USA (IN)
83: USA (KS)                  84: USA (MD)
85: USA (MI)                  86: USA (MO)
87: USA (OH)                  88: USA (OR)
89: USA (PA 1)                90: USA (PA 2)
91: USA (TN)                  92: USA (TX 1)
93: USA (WA 1)                94: USA (WA 2)
95: Venezuela                 96: Vietnam

Selection: 79
```

Overlayed bar charts

```r
ggplot() +
      geom_bar(data = churn,
      aes(x = factor(churn$CustServ.Calls),
      fill = factor(churn$Churn)),
      position = "stack") +
      scale_x_discrete("Customer Service Calls") +
      scale_y_continuous("Percent") +
      guides(fill=guide_legend(title="Churn")) +
      scale_fill_manual(values=c("blue", "red"))
```



```r
ggplot() +
      geom_bar(data=churn,
      aes(x = factor(churn$CustServ.Calls),
      fill = factor(churn$Churn)),
      position = "fill") +
      scale_x_discrete("Customer Service Calls") +
      scale_y_continuous("Percent") +
      guides(fill=guide_legend(title="Churn")) +
      scale_fill_manual(values=c("blue", "red"))
```