MY BACKGROUND

- Marketing Analytics for Small/Medium Businesses (Google Analytics, R Programming)

- Contributor: CMSWire, VKTR, ITProToday,

- Adjunct Prof: Trine University
OVERVIEW

- Explain R Programming & Python
- The basics of RStudio (the IDE used for R as well as Python via reticulate)
- Work With Libraries & Modules functions – Tips for developing models to explore data for business applications
- The basic programming protocols for importing data
- Incorporating Data sources and ideas for creating models, from regression to sentiment analysis
- Data visualizations
- AI Assistants (ChatGPT ADA, Copilot)
- Best data model practices
Data science reflects activity.

Digital media introduced an association of activity to statistics.

Data science reveals insights and ideas on human activity associated with digital media measurements.

Most statistical relationships (time series, regressions, clustering) are first steps for analysis.

Data science is multidisciplinary - programming, statistics, data visualizations.
Data Models examine data as features relative to the value of a product, service, or business model.
EXAMPLE USES

▸ Sentiment Analysis

▸ Mapping Out Public Services and Resource Deployment

▸ Stop and Frisk: Spatial Analysis of Racial Discrepancies (Baltimore)

▸ New York Building Inspections for Fire Safety

▸ Hospital Capacity for COVID-19 cases
OTHER EXAMPLES

- Drug Crime Density
- Congressional Density
WHAT GOES INTO A DATA MODEL

- Selecting our data
- Placing data into objects
- Applying functions (via library/module)
- Inspect data relationships
- Develop visualizations and documentation to share information
WHERE TO BEGIN
R PROGRAMMING

- Open source programming language with command line interpreter based on C and Fortran
- Allows a wide array of statistical analysis
- Easier reproduction of large datasets, reading many type of data sources
- Provides Fast Computation of Vectors
- Useful for those new to programming but are more familiar with the purpose of the examined data
PYTHON

- High-level, general-purpose programming language
- Allows a wide array of computational applications
- Syntax is like Legos – meant to “snap together” parameters
- Where R has libraries for nuanced science and data sources, Python has modules for a variety of applications.
HOW OBJECTS ARE MANAGED – R PROGRAMMING

- x = c(1, 2, 3, 4)
- y = c(1, 2, 3)
- x * y = 5 12 21 20
HOW OBJECTS ARE MANAGED – PYTHON

- `x = np.array([1, 2, 3, 4])`
- `y = np.array([1, 2, 3])`
- `result = x * y`
- `print(result)  # Output: [ 1  4  9 12]`
SETTING UP YOUR BRUSH AND CANVAS..AKA YOUR IDE
OBTAINING R // PYTHON

- First step: download R here: http://cran.r-project.org/.
- Other repositories can provide R
  - University of Minnesota
  - Microsoft MRAN http://mran.microsoft.com/
- Python: https://www.python.org/
- Python Package Index - PyPl: https://pypi.org/
- Console - runs code and presents results
- Source - where files can be edited
- Environment, History
- Files, Plots, Help
SET WORKING DIRECTORY

- Type `getwd()` to verify directory
- Type the `setwd()` function at the console
  - eg. `setwd("C:/yourdirectory")`
- RStudio menu: Tools >> Global Options >> Change Working Directory
SET A GITHUB DIRECTORY

- Optional Step if GitHub Repository is a preferred for version control
  - Set up GitHub repository first
  - Navigate to GitHub settings; copy HTTPS URL
  - Navigate to New Project, then version control in RStudio
  - set URL and Project Name
GITHUB COPILOT

- Optional Step
  - Evaluates the Code of a Program and Makes Suggestions
  - Appears in the Global Options pane
  - Users can indicate the amount of time passed between the last word typed and when the Copilot kicks in to make
- Copilot Chat
LIBRARIES, PACKAGES, FUNCTIONS
PACKAGES, LIBRARIES, FUNCTIONS

- Functions = performs calculations on data
- Packages = collection or functions
- Libraries = directory of packages
WHERE TO GET R LIBRARIES

- CRAN http://cran.r-project.org/.

- MRAN (Microsoft’s version on CRAN): https://mran.microsoft.com/packages/.

- GitHub via install-github function from the devtools package, devtools::install_github("githubaccountname/packagename"). First install the devtools install.packages("devtools")
HOW TO INSTALL LIBRARIES IN R

- Installed with the install.package function
  
  Type `install.package('packagename')`

- Or conduct a search in the RStudio

- Can verify in the packages panel in R Studio

- To remove type `remove.package('packagename')`
HOW TO CALL A R LIBRARY

Call library("libraryname")
package::functionname
pacman::p_load()

- library (help="packagename") # reveals the documentation in a package
- .libpaths() # retrieves Library locations
- update.packages()
- remove.packages()
ONCE A LIBRARY IS INSTALLED

- Examine the document page for functions you are interested in
- Make sure package is latest - sometimes out of date status implies no upkeep has been done
- Can open in browser and use search feature to find libraries
Functions are intended program tasks

- syntax function_name(arg1 = value1, arg2 = value2, ...)

- R has some built-in functions; Most functions are called from a library (package)

- Manual to review functions available
A specific form of infix function

- Lets you nest formulas within one another
- Usually comes with the magrittr package
- Internal Pipes introduced in R 4.1,
SELECTING YOUR STUDIO..AKA YOUR ENVIRONMENT
R ENVIRONMENT

- A place to store name-object pairs between variable name and object assigned
- Create a hierarchy
- Functions operate in a separate environment to perform a calculation
CLEAN YOUR ENVIRONMENT

- `ls()` list global objects that exist
- `ls("package:ggplot2") [1:20]` - first twenty objects in a package
- `rm()` or `remove()` – choose what objects to remove
- `rm(list=ls())` – removes all the objects: character (0) confirms empty
USING RETICULATE

- Library for running Python or a Python environment
- To call it `repl_python()`
- To run a script `py_run_file()`
- Access an environment
- To install modules `py_install()`
VIRTUAL ENVIRONMENTS

- Designed for separate modules
- Set up at the terminal
- RStudio Can Recognize The Environments
- Reticulate has functions to access environments as well
VIRTUAL ENVIRONMENT TIPS

- Consider using a tool like virtualenvwrapper for managing virtual environments more efficiently,
  - provides commands like mkvirtualenv and workon to streamline creation and activation

- Keep your virtual environments organized and well-named for easy search identification.

- Check for outdated packages within activated environments.
  - pip list --outdated
  - conda list outdated
- Can list available modules
- Can list/enter/exit environments
CONTAINERS TO HOLD DATA
CONTAINERS FOR DATA – R

- Objects to hold data types (classes)
- Factor
- Object structure
  - List - a vector with different classes
  - Matrix - a table that combines vector & function
  - Data Frames - an object to store data tables
  - Data.tables - data frame that speeds up functions
CONTAINERS FOR DATA – PYTHON

- Object structure
  - List - a sequence of elements
  - Dictionary - key:value pairs
  - numpy - arrays, dataframe, categoricals
Objects are variables with assigned values and function.

- Created with the assign operator (`<-`, `<<-`, or `assign()`)
- Variable name can contain letter, digits, periods, and underscores
- Variable can start with a letter or period
- If it starts with a period, no number character can follow
- Case sensitive // Reserve words can not be used
## OBJECT OVERVIEWS

<table>
<thead>
<tr>
<th>Vector</th>
<th>List</th>
<th>Matrix</th>
<th>Data Frame / Data Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>

- **Vector**: 1 column 1 class
- **List**: 1 column of 1 or more classes
- **Matrix**: 2D version of a vector (Same Class in Each Column)
- **Data Frame / Data Table**: Vectors Grouped Together (Different Class In Each Column)
Arrays

- Array - c function
  - example: `z <- c(1,2,3)`
  - or `z <- c(A=1,B=2,C=3)`
VECTORS

- array objects created \texttt{as.vector()} or assign with \texttt{c(arguments)}
  - example:

- Return syntax - \texttt{[x]1} where \texttt{x} is a vector, 1 is the first element

- \texttt{length()}

- \texttt{subset()} - retains a subset of a vector

\begin{verbatim}
> n <- c(0.3, 0.9, 9.8)
> class(n)
[1] "numeric"
> o <- c("x","y","z")
> o
[1] "x" "y" "z"
> class(o)
[1] "character"
\end{verbatim}
A type of vector with different classes or object types
Matrix functions are vectors with a dimension attribute.

Combines vector & function syntax.
MATRX EXAMPLE

- \( m <- \text{matrix} (c(2,5,17,1,9,13), \text{nrow}=2 \text{ byrow}=\text{TRUE}) \)
- Yields matrix 2x3 of dbl type

- \( m <- \text{matrix} (c(2,5,17,1,9,13), \text{nrow}=2 \text{ byrow}=\text{FALSE}) \)

- \( \text{colnames}(m) <- c(“\text{Column1}”, “\text{Column2}”) \)

- \( \text{rownames}(m) <- c(“\text{Row1}”, “\text{Row2}”) \)

- \( \text{rbind}() \) adds new rows, \( \text{cbind}() \) adds new column
rectangular objects for storing data tables (kind of array)

\[ n = \{2, 3, 5\} \]

\[ df = \text{data.frame}(n, s, b) \]

\[ \text{example.df} = \{2, 3, 4, 5\} \]

\[ \text{as.data.frame} => \text{confirms a data frame object} \]
DATA FRAME (CONTINUED)

- `nrow(mtcars)`
- `ncol(mtcars)`
- Typing `mtcars[1,2]` calls first row, second column in a table
  - Returns `[1] 6`
- Or `mtcars["mazda Rx4","cyl"]` also returns `[1] 6`
- Data.table – data frame extension
DATA FRAME (CONTINUED)

- Can add columns but they must be of equal length across the dataframe
- value type must be the same within list (column)
- value type can be different across columns
Matrix VS Data Frames

- Matrix solve different problems than data frames. They are meant for
  - linear transformation
  - different math language
  - represents a larger set of information (eigenvectors / values)
DATA SLICE

- A slice is a row or column
- Can call by number or column label
- Data slice can call multiple columns: `mtcars[c("col1", "col2")]

```r
> iris[20,]
      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
20       5.1      3.8       1.5       0.3  setosa

> iris[1]
   Sepal.Length
1      5.1
2      4.9
3      4.7
4      4.6
5      5.0
6      5.4
7      4.6
8      5.0
```
DATA SLICE FROM DATAFRAMES / DATATABLES / MATRIX

- subset()
  - [ ] - for subsets from a vector, matrix, or array
  - [[ ]] – for extractions
  - $extraction by name (works the same as [[]])
GETTING USE TO OBJECTS

- Call up the dataset mtcars or gtcars from the library gt
- With mtcars, create a data slice of the mpg & hp columns by number and name
- With gtcars create a data slice of the
DID YOU GET THE FOLLOWING?

- mtcars[c("hp","mpg")]

```r
> mtcars[c("hp","mpg")]

       hp mpg
Mazda RX4  110 21.0
Mazda RX4 Wag 110 21.0
Datsun 710   93 22.8
Hornet 4 Drive 110 21.4
Hornet Sportabout 175 18.7
Valiant      105 18.1
Duster 360   245 14.3
Merc 240D    62 24.4
Merc 230     95 22.8
Merc 280     123 19.2
Merc 280C    123 17.8
Merc 450SE   180 16.4
Merc 450SL   180 17.3
Merc 450SLC  180 15.2
Cadillac Fleetwood 205 10.4
Lincoln Continental 215 10.4
```
WHAT TO DO TO CAPTURE DATA
EXAMINE HOW DATA IS STRUCTURED

- How Is This Data Table Structure Maintained - file? API? URL?
- What types of data exists within the fields?
- Does a header exist in the desired table?
- How can I maintain cleaning standards?
PRACTICE DATA SETS WITHIN R

- type `data()`, then type XXX to see available sets
- Can use a dataset from another library (gt, ml3rddata, torch)
- You can import a practice dataset from a spreadsheet
- Can even create a library
REPOSITORIES FOR DATASETS

- ourworldindata.org
- Kaggle
- data.gov
- healthdata.gov
- World Bank
- Pew Institute
- Data.world
- City Data Portal
READ FUNCTIONS TO IMPORT DATASHEETS

- \texttt{read\_\texttt{* * *}} = csv, delim, tsv
- \texttt{read\_csv(f, col\_name=FALSE or "A", "B", "C")}
- \texttt{readxl} – library for reading Excel files
- \texttt{readr} - reads text files
READING APIS W/ HTTR

```r
8 query <- "https://pokeapi.co/api/v2/type/3/"
9 out <- GET(url=query)
10 out
```

- `httr()` - connects APIs, allows data to be imported into a container
- `out <- GET(query=list(search = ""))`
- `content(out)` to view what content appears from the API
- `header(out)` - headers that appear
- `str(content(out))`
SORT AND ARRANGE DATA
EXPLORATORY DATA ANALYSIS (EDA)

- Involves reviewing statistical characteristics of data
- Check observations for NAs, Unusual Values, Extreme Outliers
- Adjusting Rows and Columns as needed to prep for an analysis
- Spot any potential data anomalies to avoid feeding damaged data to an advanced data model (APIs)
- Highlights and documents data parameters to make connections to a planned data model
EDA CONSTRUCTS A BRIDGE FOR YOUR DATA

- Data is information that underlie products, be it by API, user input, or gathered data from a source.

- Data Models examine data as features relative to the value of a product, service, or business model.
EDA puts data in a format to be modeled

- Explore data to see if it can be a model for regressions, clusterings, and other statistical analysis
- Feature Engineering reveals how the model information relates to each other, discovering clues for valuable decisions and benefits
INSPECT THE DATA YOU HAVE

- #print mydata variable
- head(mydata, n=10) first ten rows are displayed
- tail(mydata, n=8) last eight rows are displayed
IDENTIFY DATA TYPES FROM YOUR DATASETS

- **categorical**
- **continuous**
- `dplyr::glimpse()` - observations & variables, integer vs character
- `dplyr::summary()` - returns quartiles
- `skimr::skim()` - returns number of column type, missing values, empty values, quartiles
- `SmartEDA::ExpData()` - returns descriptions of observations & variables, missing cases
FINDING MISSING OBSERVATIONS

- `table(is.na(x))` — determines any missing values
- `colSums(is.na(x))` — check which variables are missing
- `grep(x)` — evaluate a string for pattern matching (from the stringr library)
- `skimr::skim` — displays missing values & more quantile functions
CAN ADJUST CHARACTERS USING STRINGR FUNCTIONS

- `stringr::str_length("a b c")` -> [1] 3
- `stringr::str_sub()` -> access characters
- `stringr::str_trim()` -> remove whitespace
- `stringr::str_pad()` -> add whitespace
- `get()` - evaluate a string
ANOTHER DESCRIBE FUNCTION

- psych::describe() function
  - mean
  - median
  - min
  - max
  - range
  - skew
  - kertosis
STATS WORK ASSIGNMENT

- Find the dataset trees, women, cars
- Create a file to show the min, max, mean, and range (library hint: psych)
  - women: height
  - trees: girth
  - mtcars: mpg
DPLYR PACKAGE

- Provides simple “verb” functions for common data manipulation tasks
  - `glimpse()` (like a summarize)
  - returns #rows, #columns, datatype of columns, first 10 entries
  - similar to python df.describe()
PANDA DESCRIBE

- Similar to glimpse() function in R (dplyr)
- Displays numeric by default
- non-numeric columns specified by the include='all' argument
DPLYR PACKAGE – FILTER()

- filter() - subsets datasets/ a clean syntax
  - can use & or |
  - similar to python df[df[‘country’]==‘France’]
**DPLYR PACKAGE – ARRANGE()**

- `arrange()` - shuffles columns in ascending/descending order
  - similar to python `df.sort_values()`
**DPLYR PACKAGE – SELECT()**

- `select()` - snapshot of columns you want for an original dataset
  - similar to python
OTHER DPLYR FUNCTIONS

- `mutate()` (and `transmute()`)
- `summarise()` - sum, mean, medium, standard dev, variation
- `group_by()`
- `inner_join()`
- `count()` - number of appearances in a column
STANDARD R PLOT

```r
plot(mtcars$disp, mtcars$mpg)
```

- `plot(file, type=)` # simple line graph
- `barplot( )` # simple bar chart
- `boxplot( )` # simple box plot
USE DOCUMENTS TO SHARE INSIGHTS

- In-program Documents
- DataExplorer::create_report()
- Quarto (R & Python)
- Display Multiple Graphs
- Patchwork
- pairs() // ggpairs()
- Summarize What Data Has Been Updated
- Identify What Forecasts Can Be Developed

Regression Code Starts Here

Insert the libraries funModeling and corrplot to create the correlation matrix. The regression is based on the mtcars dataset.

```r
library(funModeling)
library(corrplot)
b <- lm(mtcars$mpg ~ mtcars$disp, mtcars)
```
Popular graphing library by Hadley Wickham

Geom is a function to graph data field

aes is the aesthetics (usually data source in basic form, but also control appearance choices)

Each returns a layer to form the graph shape.

Requires tidy data frame
GGPLOT - GEOM

- Discrete and continuous data,
- one variable, two variables, three variables
- geom values include "point", "smooth", "boxplot", "line", "histogram", "density", "bar", and "jitter".
PYTHON VISUALIZATION

- Matplotlib
- seaborn (built on matplotlib)
  - pair plot function – sns.pairplot().
REGRESSION

- Examining statistical relationship among data columns:  \( y = ax + b \)
- Simple example: \( y = \text{total cost} \)  \( a = \text{variable cost} \), \( b = \text{fixed cost} \)
- Linear or Nonlinear
- Usually calculate MSPE, MSAE, or R squared for correlation
- Balance of data: too few - biased estimate, while too many leads to less precision
INSPECT CORRELATION AMONG THE DATA COLUMNS

- `cor()` function `cor(yourdataframe, method="Pearson")`
- `Hmisc::rcorr()`
- data frame must be used in the function
- Pearson is the default method; Spearman can also be used
- linear correlation between -1 and +1, with 0 as no correlation
Regression data <- data.frame(x1, x2, x3, x4)

lm (y ~ ., data = Regression_data)

Period means “to include all the independent variables”
CREATE A REGRESSION EXAMPLE

- Using gtcars in the gt library, create a regression with price being the dependent variable
- Using
WHAT A REGRESSION MODEL CAN LOOK LIKE

carmodel <- lm(msrp ~ hp + trq + mpg_c + mpg_h + bdy_style, data = lmchallenge)
TIME SERIES: DECOMPOSITION

- Decomposition: transforms a series into seasonal, trend, and observed

  Place a time series into a decompose function

  Then plot the object
EXAMINE DATES

- Adjust dataset dates into a recognizable object format
- POSIX or characters
- R: lubridate
- Python: datetime
**TIME SERIES**

- `ts()` converts a numeric vector into a R time series object
  - `ts(data, start, end, frequency)`
  - `data = vector or matrix with data`
  - `start = start time for first observation`
  - `end = end time for last observation`
  - `ts(vector, start= , end= , frequency= )`
TIME SERIES: DECOMPOSITION

Create a time series data frame

Place a time series into decompose function

Then plot the decompose object
Advertise 1. Examine once every two weeks.

Algorithms

1. Lower bids on high-performing campaigns. These are campaigns with high ACOS. Examine once every two weeks.
2. Raise bids on keywords with high ACOS. Examine once every two weeks.

Advertise cost of sales ACOS: 

The lower the ACOS the better your budget.

Amazon Ads

Advertise: You can use Amazon Ads for targeting the more relevant keywords. You can run automatic campaigns for 2 to 3 weeks. You don’t have broad match and then narrow your match to search phrases.

Purchase Ads in Amazon: Amazon Marketing Services

A+ text:

Premium A+ mobile: Amazon – Amazon Marketing Services

Advertise cost of sales ACOS: 

The lower the ACOS the better your budget.
Type # to comment

Call out each step so that another data practitioner can pick up where you left off

Document what each library provides

```r
#call libraries for twitter. requires "ROAuth" for access
library(twitteR)
library(devtools)
library(ROAuth)
#call Twitter with OAuth via ROAuth
setup_twitter_oauth("SSAEOWJ20I5LT7tDUPjeP96v","bpyHAXEQu9"
#call a Timeline
userTimeline("zimanaanalytics")
#create a variable for searching Twitter for a
#searchterm. In this case I am using hashtags
#I've used on the Zimana account.
analytics_tweets = searchTwitter("#analytics")
marketing_tweets = searchTwitter("#marketing")
#print results as a check.
```
Definition: a variation of a Markdown file .QMD file to hold text and graphics alongside code

Code can be shown or run to demo a chart with the text

Creates YAML Header which matches document to HTML elements

Create Parameters to reuse documents with different data
QUARTO

- 3 documents - YAML header, R code chunk, text mixed with simple text
- Chunks - to insert add chunk command or typed 3marks{r} and 3marks{python}
- Can also use rpubs.com (free) or ShinyApps.io for hosting
CREATE QUARTO DOCUMENT

- library(quarto)
- create a chunk for an R programming and one for a Python
SHINY

- Apps that run an R program
- Has its own extension
- Template with 2 components - User interface, Server Instructions
- Python Shiny (new) – Discord group
TO SHARE SHINY APPS

- use library rsconnect

- Go to shiny apps.io and click log-in (can use GitHub or create user/pwd)

- Retrieve token from setting, use it within rsconnect to create connection
Are data objects for share resources outside of a traditional database (files < 500 MB)

- Can export to Amazon S3, Dropbox, Google Drive
- Can add metadata & versioning
- Python version
First, place objects in a board

Then write to the board, creating the pin

Read the pin via pin_read()

Makes code reproducible without changing file path

```
> library(pins)
> board <- board_temp()
> mtcars <- tibble::as_tibble(mtcars)
> board %>% pin_write(mtcars, "mtcars")
Guessing `type = 'rds`
Creating new version '20230510T140216Z-cc5fd'
Writing to pin 'mtcars'
> board %>% pin_read("mtcars")
# A tibble: 32 × 11

   mpg   cyl disp  hp
  <dbl> <dbl> <dbl> <dbl>
1    21     6 160 110
2    21     6 160 110
3   22.8    4 108  93
4   21.4    6 258 110
5    18.7    8 360 175
```
AI IN DATA MODELING

- Can add comments to a given code, inserting via clean code standards
- Can write a documentation draft
- Create a README file
- Plus feature ($20/mo)
- Simplifies Data Revisions
USE AI TO PLAN DATA STRUCTURE

Example: Bard (footnote - can export to Colab)

Create a table of products and services sold in a department store. Each column should contain product, retail price in dollars, SKU, and units sold. Fill each observation with a random number.

Sure, here is a table of products and services sold in a department store, with each column containing product, retail price in dollars, SKU, and units sold. The values in each column are random numbers.

<table>
<thead>
<tr>
<th>Product</th>
<th>Retail Price</th>
<th>SKU</th>
<th>Units Sold</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-Shirt</td>
<td>20</td>
<td>12345</td>
<td>100</td>
</tr>
<tr>
<td>Jeans</td>
<td>50</td>
<td>67890</td>
<td>50</td>
</tr>
<tr>
<td>Shoes</td>
<td>100</td>
<td>34567</td>
<td>25</td>
</tr>
<tr>
<td>Dress</td>
<td>75</td>
<td>98765</td>
<td>100</td>
</tr>
<tr>
<td>Hat</td>
<td>25</td>
<td>43210</td>
<td>50</td>
</tr>
<tr>
<td>Scarf</td>
<td>15</td>
<td>56789</td>
<td>25</td>
</tr>
<tr>
<td>Bag</td>
<td>50</td>
<td>78901</td>
<td>50</td>
</tr>
</tbody>
</table>

Create the same table using R programming

Sure, here is the same table created using R programming:

```r
# Generate random numbers
retail_prices <- sample(10:100, 8)
skus <- sample(10000:99999, 8)
units_sold <- sample(1:100, 8)

# Create table
table <- data.frame(
  product_name = product_names, 
  retail_price = retail_prices, 
  sku = skus, 
  units_sold = units_sold)
```
USE AI TO PLAN VISUALIZATIONS

Example: ChatGPT Plus w/ Advanced Data Analysis
GITHUB COPILOT

- Subscription AI: makes suggestions for comments and syntax
- Can provide more context for your IDE environment
- Visual Studio Code, RStudio (new)
EXPECT UNEXPECTED ITERATIONS WITH AI
TIPS TO KEEP IN MIND ABOUT DATA MODELING WITH AI

- Can hallucinate concepts together – always vet
- Give context rather than a one-shot question
- Discord community - latest news & tips (Propensity, Bard)
- AI helps with zeroing in on concept, but can not do the whole job for you
HOW DO I KNOW I HAVE A GOOD DATA MODEL?

- A good feature from a model solves a problem with as few data as possible
- Is the differentiation among model variables reasonably clear?
- Model data must be statistically and technically reliable
- Model expresses minimal amount of data - collect data only as you need
DEBUG IN R PROGRAMMING

- group tasks together and comment with #
- click a line in RStudio to add or remove a breakpoint
- debug()
- browser()
- traceback()
- Rlang::last_trace()
DEBUG IN PYTHON

- Use the `timeit` module for measure how long functions work
- `unittest`
- `pytest`
- `nose`
- `doctest`
KEEP DATA SOURCES IN MIND WHILE DEVELOPING A MODEL

- Most regulations focuses on length of time data has been held
- Be aware of permission processes when accessing data
- Use communication tools that permit a conversation thread to note how data is being managed
Use R and Python to help you sort through data into an object

Libraries and Modules can be applied for creating statistical models

For Exploratory Data Analysis, import data, review observations, then add statistical functions to describe the data

Can use AI to narrow down syntax options, but vet accuracy of content and be aware of limitations
THANK YOU!

TWITTER: @ZIMANAANLYTICS
IG: ZIMANAANALYTICS
MAIN@ZIMANA.COM
RESOURCES

- inside-r.org
- Data Science Central (@Analyticsbridge)
- Data Science Plus
- ODSC - Open Data Science
- R-Bloggers offers up to date news on packages and examples from other data scientists
- rdocumentation.org - offers updates on packages
RESOURCES PART DEUX

- datasociety.net
- datakind.org
- R for Public health.blogspot.com
- KDNuggets
- r-consortium.org
RESOURCES PART TROIS

- UC Business Analytics
- RStudio Community rstd.io/community
DATA REPOSITORIES

- Kaggle
- data.gov
- UC Irvine Machine Learning Repository
- World Bank
- dataportals.org
- Quora: where I can find large datasets open to the public
- Wikipedia: list of machine learning data
TECHNIQUES AND TIP SOURCES – R PROGRAMMING

- Crantastic
- R-Journal
- Use Meetup.com for local R user groups
- #rstats hashtag (Twitter) - programmer community
- StackExchange & Medium
- Rmetrics - finance-oriented models
- Bioconductor - projects for bio
X PROFILES FOR INFO

- Women in ML/DS @wimlds
- @rOpenSci
- One R Tip A Day - @RLangTip
- @MicrosoftR
- @RLadiesGlobal
- @RJobListings
GOOD R ADVOCATES

- David Robinson - @drob
- Emily Robinson - @robinson_es
- Julie Silge - @juliasilge
- Mara Averick - @dataandme
- Angela Li - @CivicAngela
- Hadley Wickham - @hadleywickham
GOOGLE ANALYTICS LIBRARIES

- RGA
- rga
- RGoogleAnalytics
LIBRARIES FOR WORKING WITH EXCEL

- xlsx
- Readxl
- Gdata
- XLConnect bridge between R and Excel, can conduct functions in between.
PACKAGE CREATION

- usethis::create_package(C:/users/__/Desktop/pkgname) – opens a new RStudio session
- file >> new project
- usethis::usegithub
- collection of file with folder structure
DATABASE SOURCE

- RODBC
- RJDBC
- ROracle
- dplyr
- dbplyr
- RSQL
DATABASE SOURCE

- dplyr
- DBI
- odbc
- keyring
- pool
LIBRARIES

- ggfortify
- mlbench
- caret
- equationmatic
- tidyverse
- caTools - for machine learning for train & test split data
DEBUG PROCESSING

- `system.time()` - useful for expression with a few seconds of more, also for signal garbage collection (user, system, elapsed)

- microbenchmark library - runs an expression several times

- `profvis("insert code")` - flame graph waterfall of memory consumed in miliseconds; each block call to a function multiple calls to the same function processed

- `usethis::use_testthat()` – automated testing
DATA CLEANING

- `janitor::get_dupes()` - finds duplicates observations in rows
- `janitor::remove_constant()` - removes columns with constant values (not good for N/As)
- `tidy::gather()`, `separate()`, `spread()`
STACKING DATA

- can subset w/ slice
- can use bind_rows to stack vertically
  - mpg1 <- mpg %>% slice(1) %>% select(manufacturer, model, hwy, city) %>% mutate(dataset = 1)
  - mpg2 <- mpg %>% slice(44:45) %>% select(manufacturer, model, hwy, city) %>% mutate(dataset = 2)
  - stack_vert <- mpg1 %>% bind_rows(mpg2)
- To stack horizontally use bond_col
library(DBI)

con <- dbConnect(RMySQL::MySQAL(), dbname='mydb',
host = "https://predictivehacks.com/", port = 3306, user
= "george", password="predictive hacks")

# next establish an object

tables <- dbListTables(con)

# check the structure str(tables)
DBPLYR EXAMPLE

- mtcars_db %>%
  - mutate(mpg_rnd = round(mpg)) %>%
  - select(mpg_rnd = hp) %>%
  - show_query()
TRANSFORM

- tidy::pivot_wider()
  
  - use a tibble
  
  - use the pivot_wider() with functions names_from and values_from
  
  - pivot_wider(names_from = c(title one, title two), values_from = column w/ values)
COUNT

- `count()` → counts number of instances
  - `sort = TRUE` (ascending order)
  - `wt = total` (sum up a column)
  - `name = column_name` (names a column “column_name”)
- `dplyr::add_count(column)` is the same as `group_by(column) %>>%mutate(n = n()) %>>%ungroup()`
  - `launches%>%add_count(type)%>>%filter(n >= 20)`
- `summarize()` to create a list column
  - `restaurants%>%group_by(cuisine) %>%summarize(avg_score=mean(aug_score))`
DATA EXPLORER

- plot_correlation()
- plot_str()
- introduce()
- plot_missing()
- plot_bar()
- plot_histogram()
RANDOM NUMBERS

- `runif()` - range 0 to 1 uniform distribution
  - `runif(5)` - a vector of 5 numbers between 0 and 1
  - `runif(4, min=0, max=100)` vector of 4 numbers between 0 and 100
- `sample(vector elements, number of items, replace=FALSE, prob=NULL)`
  - `replace` = should sampling be with replacement?
  - `probability of weights for obtaining the elements of the vectors`
  - `sample(1:100, 4 , replace=TRUE)` vector of 4 numbers between 0 and 100
SET SEED

- `set.seed(n)` - initial values of the random number seed used by random number functions like `rnorm()`, `runif()`, `sample()`

- `n` can be any number between 0 and 2 to the 31st power
OTHER IMPORT LIBRARIES

- rvtest - HTML (web scraping)
- jsonlite - JSON
- plumber - R code becomes API
- rapier - convert R code to a web API
CLASS

- character = text
- numeric = number
- INT = integers
- Complex x <- 2 + 4i
- Logical (Boolean) TRUE or FALSE
NATURE OF A CLASS

str()

class()

mode = numeric, character, complex, logical  mode(x)

length = number of elements in an object length(x)
**UNIQUE CLASS**

- \( \exp(x) - e^{^x} \)
- \( \text{abs()} - \) absolute
- \( \text{typeof()} \)
- \( \text{table()} - \) level count
- \( \text{levels()} - \) name of level
- \( \text{nlevels()} - \# \) of level
UNSURE OF A SYMBOL OR FUNCTION? TYPE THE FOLLOWING:

- `? 'functionname'`
- `help(functionname)`
- `? '~'`
- `help.start()`

- Information appears in the help panel
- For arguments on a function type `args(functionname)`
ASKING FOR HELP

- Unsure of a symbol or function? Type the following
  
  help.search(“your phrase”)

- Help information appears in the help panel

  Go to Help panel in R studio