

Using ChatGPT3.5 to Build Scripts in R

DATA AND DECISION SCIENCE NETWORK MEETING 2023



UNIVERSITY
OF WOLLONGONG
AUSTRALIA



THE DATA

The Pima Indians Diabetes Data is a well-known dataset in the field of healthcare and machine learning.

Dependent Variable

Diabetes Whether the participant has diabetes.

Independent Variables

gluc: Plasma glucose concentration after a 2-hour oral glucose tolerance test.

skin: Triceps skinfold thickness (mm). Insulin: 2-hour serum insulin (μ U/ml).

bmi: Body mass index (weight in kg / (height in m)²).

age: Age in years.



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Additional Variables (not being analysed)

bp: Diastolic blood pressure (mm Hg).

npreg - Number of times pregnant.

ped: A function that scores the likelihood of diabetes based on family history.



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

THE ANALYSIS

Write an R script that does the following

1. Load the data set diabetes.csv into R / RStudio on the desktop.
2. Visualise the distribution of the continuous independent variables with a histogram and boxplot comparing diabetes status.
3. Obtain a summary statistics table (valid cases, means and standard deviations) of the independent variables split by diabetes status.
4. Perform a logistic regression analysis to model the effects of the independent predictors on diabetes status.
5. Produce a summary table of coefficients from the regression.



Step 1: Load the data

PROMPT 1

BR

Write R code to import a csv file called 'diabetes.csv' on my desktop



To import a CSV file named 'diabetes.csv' from your desktop into R, you can use the `read.csv()` function. Make sure you specify the correct file path to your desktop. Here's an example of how you can do this:



```
R Copy code  
  
# Set the file path to your desktop  
desktop_path <- "~/Desktop/diabetes.csv" # Replace "~" with your actual home directory  
  
# Import the CSV file into a data frame  
diabetes_data <- read.csv(desktop_path)  
  
# Now, 'diabetes_data' contains the data from the CSV file
```

Make sure to replace `"~/Desktop/diabetes.csv"` with the actual file path to 'diabetes.csv' on your desktop. The `read.csv()` function will read the CSV file and store its contents in the `diabetes_data` data frame.

Regenerate

Step 1: Load the data

PROMPT 1



Write R code to import a csv file called 'diabetes.csv' on my desktop


Observations

- Needed to specify working directory but this was explained in output.

Step 2: Visualise the data

PROMPT 2

BR

Write code that allows me to visualise the distribution of the variables of my data set for each case of Diabetes 



To visualize the distribution of variables in your dataset for each case of diabetes, you can use various plotting libraries in R, such as `ggplot2` for creating informative and customizable plots. Here's an example of how you can create histograms for numeric variables and bar plots for categorical variables in your dataset, differentiating by the diabetes cases:

Assuming your dataset has a column named 'Diabetes' that indicates whether a case is diabetic or not (e.g., 0 for non-diabetic and 1 for diabetic):

```
R Copy code
# Load the ggplot2 library if not already loaded
if (!require(ggplot2)) {
  install.packages("ggplot2")
  library(ggplot2)
}

# Assuming your dataset is named 'diabetes_data'
# Replace 'diabetes_data' with your actual dataset name

# Create a list of numeric columns (assuming they are all numeric)
numeric_columns <- sapply(diabetes_data, is.numeric)

# Separate the dataset into two data frames: one for diabetics and one for non-diabetics
diabetics <- diabetes_data[diabetes_data$Diabetes == 1, ]
non_diabetics <- diabetes_data[diabetes_data$Diabetes == 0, ]
```

```
# Create histograms for numeric variables
for (col in names(diabetes_data)[numeric_columns]) {
  p <- ggplot(diabetes_data, aes_string(x = col)) +
    geom_histogram(binwidth = 0.5, fill = "blue", alpha = 0.5) +
    facet_grid(Diabetes ~ .) +
    labs(title = paste("Distribution of", col, "by Diabetes Status"),
         x = col, y = "Frequency") +
    theme_minimal()
  print(p)
}

# Create bar plots for categorical variables (assuming they are all factors)
categorical_columns <- sapply(diabetes_data, is.factor)

for (col in names(diabetes_data)[categorical_columns]) {
  p <- ggplot(diabetes_data, aes_string(x = col, fill = factor(Diabetes))) +
    geom_bar(position = "dodge") +
    labs(title = paste("Distribution of", col, "by Diabetes Status"),
         x = col, y = "Count") +
    theme_minimal()
  print(p)
}
```


This code first separates your dataset into two parts: one for diabetics and one for non-diabetics. Then, it creates histograms for numeric variables and bar plots for categorical variables, with different colors for diabetic and non-diabetic cases. You may need to adjust the `binwidth` parameter for histograms and customize the plots further based on your specific dataset and preferences.

Regenerate

Step 2: Visualise the data

PROMPT 2



Write code that allows me to visualise the distribution of the variables of my data set for each case of Diabetes 

Observations

- Prompt very general, did not specify what visualisation or which variables.
- ChatGPT worked out that a histogram and boxplot would be appropriate.
- Code contained a pointless sub-setting procedure that did not work!
- Histogram used dodged-fill.

Step 2: Visualise the data (again)

PROMPT 3 + 4

BR

Obtain a histogram and boxplot of the gluc column in R separated into cases of Diabetes

BR

repeat the above for the bmi variable





To obtain a histogram and boxplot of the 'gluc' column in R separated into cases of diabetes, you can use the `ggplot2` library for visualization. Here's how you can do it:



```
R Copy code
# Load the ggplot2 library if not already loaded
if (!require(ggplot2)) {
  install.packages("ggplot2")
  library(ggplot2)
}

# Assuming your dataset is named 'diabetes_data'
# Replace 'diabetes_data' with your actual dataset name

# Create a histogram for 'gluc' separated by Diabetes status
histogram_plot <- ggplot(diabetes_data, aes(x = gluc, fill = factor(Diabetes)
  geom_histogram(binwidth = 1, position = "dodge", alpha = 0.7) +
  labs(title = "Distribution of gluc by Diabetes Status",
        x = "gluc", y = "Frequency") +
  theme_minimal()

# Create a boxplot for 'gluc' separated by Diabetes status
boxplot_plot <- ggplot(diabetes_data, aes(x = factor(Diabetes), y = gluc, fill = factor(Diabetes)
  geom_boxplot() +
  labs(title = "Boxplot of gluc by Diabetes Status",
        x = "Diabetes Status", y = "gluc") +
  theme_minimal()
```

```
# Combine both plots
library(gridExtra)
combined_plots <- grid.arrange(histogram_plot, boxplot_plot, ncol = 2)

# Display the combined plots
print(combined_plots)
```

This code will create a histogram and a boxplot of the 'gluc' column, with data separated into cases of diabetes (0 for non-diabetic and 1 for diabetic). You can adjust the `binwidth` parameter for the histogram and customize the appearance of the plots as needed.

Step 2: Visualise the data (again)

PROMPT 3 + 4

BR

Obtain a histogram and boxplot of the gluc column in R separated into cases of Diabetes

Observations

- Prompt much more specific.
 - Output more easily interpretable for a novice.
 - Histogram still used dodged-fill.
-

BR

repeat the above for the bmi variable



Step 3: Descriptive Statistics

PROMPT 5

BR

Create a table of descriptive statistics, namely the number of non-missing observations, mean, and standard deviation of the bmi, gluc, skin, and age variables separated into cases of Diabetes





You can use the `dplyr` library in R to calculate descriptive statistics for the 'bmi', 'gluc', 'skin', and 'age' variables separated into cases of diabetes (0 for non-diabetic and 1 for diabetic). Here's how you can create a table of these statistics:



R

Copy code

```
# Load the dplyr library if not already loaded
if (!require(dplyr)) {
  install.packages("dplyr")
  library(dplyr)
}

# Assuming your dataset is named 'diabetes_data'
# Replace 'diabetes_data' with your actual dataset name

# Filter data for non-diabetic cases (Diabetes = 0)
non_diabetic_stats <- diabetes_data %>%
  filter(Diabetes == 0) %>%
  select(bmi, gluc, skin, age) %>%
  summarise(
    N = n(),
    Mean_bmi = mean(bmi, na.rm = TRUE),
    SD_bmi = sd(bmi, na.rm = TRUE),
    Mean_gluc = mean(gluc, na.rm = TRUE),
    SD_gluc = sd(gluc, na.rm = TRUE),
    Mean_skin = mean(skin, na.rm = TRUE),
    SD_skin = sd(skin, na.rm = TRUE),
    Mean_age = mean(age, na.rm = TRUE),
    SD_age = sd(age, na.rm = TRUE)
  )
```

```
# Filter data for diabetic cases (Diabetes = 1)
diabetic_stats <- diabetes_data %>%
  filter(Diabetes == 1) %>%
  select(bmi, gluc, skin, age) %>%
  summarise(
    N = n(),
    Mean_bmi = mean(bmi, na.rm = TRUE),
    SD_bmi = sd(bmi, na.rm = TRUE),
    Mean_gluc = mean(gluc, na.rm = TRUE),
    SD_gluc = sd(gluc, na.rm = TRUE),
    Mean_skin = mean(skin, na.rm = TRUE),
    SD_skin = sd(skin, na.rm = TRUE),
    Mean_age = mean(age, na.rm = TRUE),
    SD_age = sd(age, na.rm = TRUE)
  )

# Create a table combining the statistics for non-diabetic and diabetic cases
statistics_table <- rbind(non_diabetic_stats, diabetic_stats)
rownames(statistics_table) <- c("Non-Diabetic", "Diabetic")

# Print the table
print(statistics_table)
```

This code will create a table that includes the number of non-missing observations, mean, and standard deviation for the 'bmi', 'gluc', 'skin', and 'age' variables separated into cases of diabetes (non-diabetic and diabetic). Adjust the variable names and dataset name as needed for your specific dataset.

Regenerate

Step 3: Descriptive Statistics

PROMPT 5

BR

Create a table of descriptive statistics, namely the number of non-missing observations, mean, and standard deviation of the bmi, gluc, skin, and age variables separated into cases of Diabetes



Observations

- Code did not run!
- Code assumed Diabetes to have numeric coding 0 and 1.
- Code could have been easily accomplished with a `group_by()` command.

Step 3: Descriptive Statistics (Brad's Version)

IT COULD HAVE BEEN MORE SIMPLE

```
#Brad's Version

n_mean_sd <- function(x) c(`Valid N`=sum(!is.na(x)),
                          Mean=mean(x,na.rm=T),
                          `Std Dev`=sd(x,na.rm=T))

descriptive_stats <- diabetes_data %>%
  group_by(Diabetes) %>%
  summarise_at(c("bmi","gluc","skin","age"),n_mean_sd) %>%
  mutate(Statistic = c("Valid N","Mean", "Standard Deviation")) %>%
  relocate(Diabetes,Statistic)

####
```

Step 4: Fit Logistic Regression Model

PROMPT 5

BR

Fit a logistic regression model in R modelling Diabetes with respect to the variables gluc, bmi, skin and age



You can fit a logistic regression model in R to model diabetes with respect to the variables 'gluc', 'bmi', 'skin', and 'age' using the `glm()` function. Here's how you can do it:



```
R Copy code  
  
# Assuming your dataset is named 'diabetes_data'  
# Replace 'diabetes_data' with your actual dataset name  
  
# Fit a logistic regression model  
logistic_model <- glm(Diabetes ~ gluc + bmi + skin + age, data = diabetes_da  
  
# Summarize the logistic regression model  
summary(logistic_model)
```

In this code:

- `Diabetes` is the binary dependent variable (0 for non-diabetic and 1 for diabetic).
- `gluc`, `bmi`, `skin`, and `age` are the predictor variables.
- `data = diabetes_data` specifies the dataset.
- `family = "binomial"` specifies the logistic regression model.

The `summary(logistic_model)` function call will provide a summary of the logistic regression model, including coefficients, standard errors, p-values, and other relevant statistics.

Please make sure to replace `'diabetes_data'` with the actual name of your dataset containing the relevant variables.

Regenerate

Step 4: Fit Logistic Regression Model

PROMPT 5

BR



Fit a logistic regression model in R modelling Diabetes with respect to the variables gluc, bmi, skin and age

Observations

- Code did not run!
- Code again assumed Diabetes to have numeric coding 0 and 1.
- Requires interim step to convert data type.

Step 5: Extract Coefficients Table

PROMPT 6

- ›  Obtain a table of statistics from the logistic regression analysis which includes the estimates, standard errors, z-values, and p-values for each predictor. 



You can obtain a table of statistics, including estimates, standard errors, z-values, and p-values for each predictor in the logistic regression analysis using the `summary()` function on your logistic regression model. Here's how you can do it:



R

Copy code

```
# Assuming you've already fitted the logistic regression model as 'logistic_model'
# Replace 'logistic_model' with the actual name of your logistic regression model

# Obtain a table of statistics
logistic_summary <- summary(logistic_model)

# Extract coefficients, standard errors, z-values, and p-values for each predictor
coefficients <- coef(logistic_summary)
std_errors <- coef(summary(logistic_model))[, "Std. Error"]
z_values <- coefficients / std_errors
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Create a table
statistics_table <- data.frame(
  Predictor = names(coefficients),
  Estimate = coefficients,
  Std.Error = std_errors,
  Z.Value = z_values,
  P.Value = p_values
)

# Print the table
print(statistics_table)
```

TRY AGAIN!




To obtain a table of statistics including the estimates, standard errors, z-values, and p-values for each predictor in the logistic regression model, you can use the `coef(summary(logistic_model))` command. Here's how you can do it:

```
R Copy code  
  
# Assuming you have already fit the logistic regression model as 'logistic_m  
  
# Obtain a table of statistics  
coef_table <- coef(summary(logistic_model))  
  
# Print the table  
print(coef_table)
```

This code will give you a table with estimates (coefficients), standard errors, z-values (Wald statistics), and p-values for each predictor variable in the logistic regression model. The table will include both the intercept and the coefficients for 'gluc', 'bmi', 'skin', and 'age'.

Step 5: Extract Coefficients Table

PROMPT 6

- BR Obtain a table of statistics from the logistic regression analysis which includes the estimates, standard errors, z-values, and p-values for each predictor. 

Observations

- Original code attempted to do quite a complex routine to obtain output.
- Task can be performed with one simple command:

```
coef(summary(logistic_model))
```
- Required two generations to achieve.

Writing a Complete Script

CAN WE BETTER PROMPT THE OUTPUT?

BR

I'm working in R with the dataset diabetes with the following variables:
Diabetes Whether the participant has diabetes - "No Diabetes", "Has Diabetes"
gluc: Plasma glucose concentration after a 2-hour oral glucose tolerance test.
skin: Triceps skinfold thickness (mm).
bmi: Body mass index (weight in kg / (height in m)^2).
age: Age in years.

Write an R script that does the following

1. Load the data set diabetes.csv into R / RStudio on the dekstop.
2. Visualise the distribution of the continuous independent variables with a histogram and boxplot comparing diabetes status.
3. Obtain a summary statistics table (valid cases, means and standard deviations) of the independent variables split by diabetes status.
4. Perform a logistic regression analysis to model the effects of the independent predictors on diabetes status.
5. Produce a summary table of coefficients from the regression.



You can achieve these tasks in R using various packages and functions. Here's an R script that accomplishes each of the steps you mentioned:

```
R Copy code  
  
# Step 1: Load the dataset  
diabetes_data <- read.csv("diabetes.csv")  
  
# Step 2: Visualize the distribution of continuous variables by diabetes status  
library(ggplot2)  
  
# Histograms  
ggplot(diabetes_data, aes(x = gluc, fill = Diabetes)) +  
  geom_histogram(binwidth = 10, position = "dodge", alpha = 0.7) +  
  labs(title = "Plasma Glucose Distribution by Diabetes Status", x = "Plasma Glucose", y = "Count") +  
  theme_minimal()  
  
ggplot(diabetes_data, aes(x = skin, fill = Diabetes)) +  
  geom_histogram(binwidth = 5, position = "dodge", alpha = 0.7) +  
  labs(title = "Triceps Skinfold Thickness Distribution by Diabetes Status", x = "Triceps Skinfold Thickness", y = "Count") +  
  theme_minimal()  
  
ggplot(diabetes_data, aes(x = bmi, fill = Diabetes)) +  
  geom_histogram(binwidth = 2, position = "dodge", alpha = 0.7) +  
  labs(title = "BMI Distribution by Diabetes Status", x = "BMI", y = "Count") +  
  theme_minimal()
```

```
# Boxplots  
ggplot(diabetes_data, aes(x = Diabetes, y = gluc, fill = Diabetes)) +  
  geom_boxplot() +  
  labs(title = "Plasma Glucose Boxplot by Diabetes Status", x = "Diabetes Status", y = "Plasma Glucose") +  
  theme_minimal()  
  
ggplot(diabetes_data, aes(x = Diabetes, y = skin, fill = Diabetes)) +  
  geom_boxplot() +  
  labs(title = "Triceps Skinfold Thickness Boxplot by Diabetes Status", x = "Diabetes Status", y = "Triceps Skinfold Thickness") +  
  theme_minimal()  
  
ggplot(diabetes_data, aes(x = Diabetes, y = bmi, fill = Diabetes)) +  
  geom_boxplot() +  
  labs(title = "BMI Boxplot by Diabetes Status", x = "Diabetes Status", y = "BMI") +  
  theme_minimal()  
  
ggplot(diabetes_data, aes(x = Diabetes, y = age, fill = Diabetes)) +  
  geom_boxplot() +  
  labs(title = "Age Boxplot by Diabetes Status", x = "Diabetes Status", y = "Age") +  
  theme_minimal()  
  
# Step 3: Summary statistics table  
summary_table <- aggregate(. ~ Diabetes, data = diabetes_data, FUN = function(x) {  
  print(x)  
})  
print(summary_table)  
  
# Step 4: Logistic regression analysis  
model <- glm(Diabetes ~ gluc + skin + bmi + age, data = diabetes_data, family = "binomial")  
summary(model)  
  
# Step 5: Summary table of coefficients  
coef_table <- coef(summary(model))  
print(coef_table)
```

Regenera

Writing a Complete Script

CAN WE BETTER PROMPT THE OUTPUT?

BR

I'm working in R with the dataset diabetes with the following variables:

Diabetes Whether the participant has diabetes - "No Diabetes", "Has Diabetes"

gluc: Plasma glucose concentration after a 2-hour oral glucose tolerance test.

skin: Triceps skinfold thickness (mm).

bmi: Body mass index (weight in kg / (height in m)²).

age: Age in years.

Write an R script that does the following

1. Load the data set diabetes.csv into R / RStudio on the desktop.
2. Visualise the distribution of the continuous independent variables with a histogram and boxplot comparing diabetes status.
3. Obtain a summary statistics table (valid cases, means and standard deviations) of the independent variables split by diabetes status.
4. Perform a logistic regression analysis to model the effects of the independent predictors on diabetes status.
5. Produce a summary table of coefficients from the regression.

Observations

- By better initialising ChatGPT3.5 we can get simpler and better output.
- ChatGPT no longer has to write code that is completely generic and hard to understand.
- Consistent code practice throughout.
- Still have issues with code though....



The Statistical Consulting Centre



UNIVERSITY
OF WOLLONGONG
AUSTRALIA



A BIT ABOUT US ...

Marijka Batterham

• **Director**

Brad Wakefield

• **Statistical Consultant**

Aim

The service aims to improve the statistical content of research carried out by members of the University. Researchers from all disciplines may use the Centre. Priority is currently given to staff members and postgraduate students undertaking research for Doctor of Philosophy or Masters' degrees.

How we can help

Currently the Statistical Consulting Centre provides each academic or post-graduate student with a free initial consultation. Up to ten hours per calendar year of consulting time is provided without charge if research funding is not available. When researchers require more consulting time, or receive external funding, a service charge may be necessary.



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

To book an appointment, simply go to our website and select

MAKE AN APPOINTMENT



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

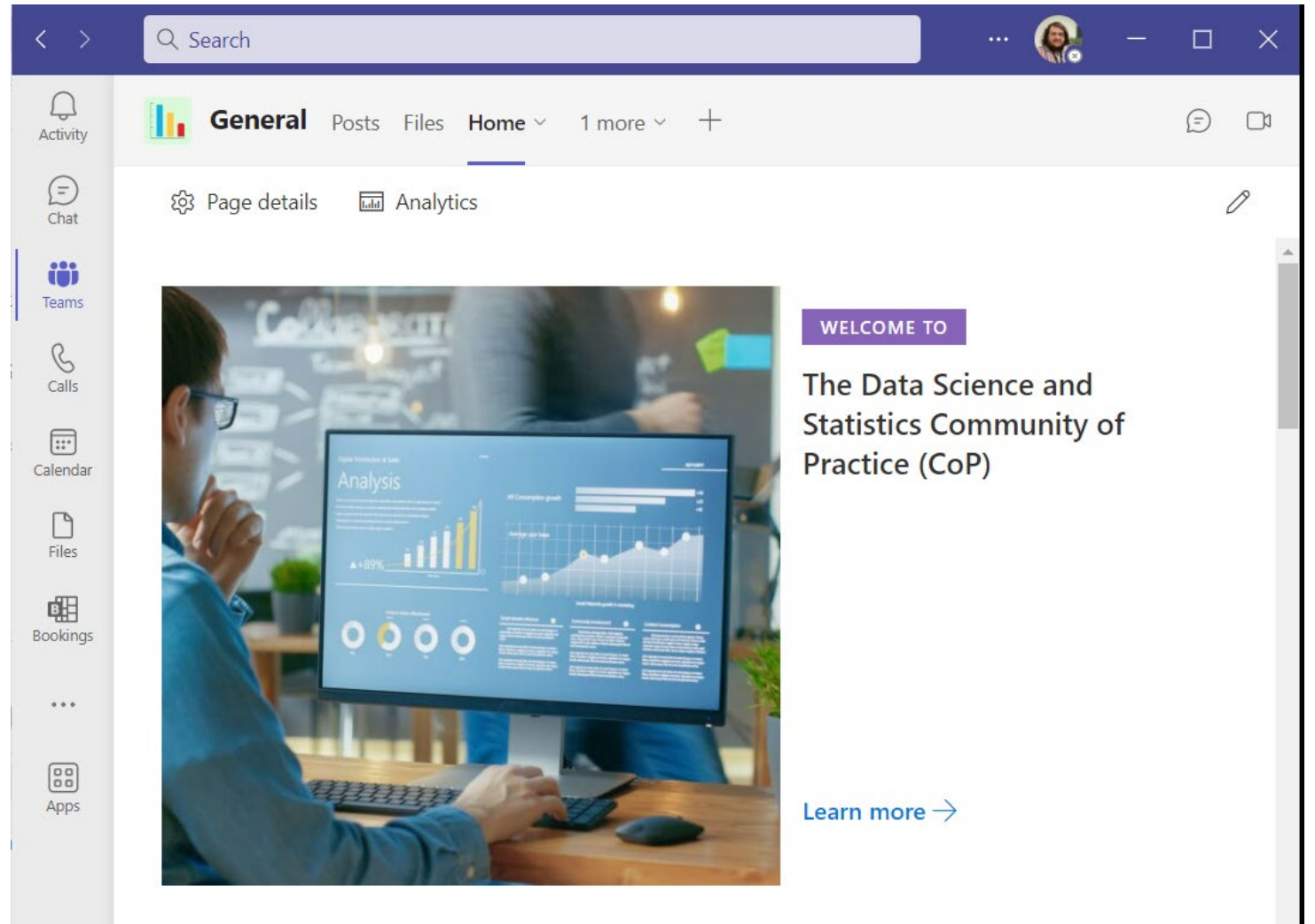
We are starting a **Data Science and Statistics Community of Practice**

An online space to:

- Foster Knowledge Sharing
- Access Resources
- Collaborate with Peers.
- Get Data Science and Stats Support.
- Learn about training opportunities.



UNIVERSITY
OF WOLLONGONG
AUSTRALIA



[Link in the Chat](#)



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

How to Create Graphs & Figures for Publication

DATA VISUALISATION WORKSHOP

**RUNNING ONLINE
OVER TWO HALF DAYS**

Monday 16th October 2023

Tuesday 17th October 2023

9:30AM – 12:30PM



AVAILABLE FOR



**IBM SPSS RSTUDIO
PICK YOUR STREAM**

**RUN BY THE UOW STATISTICAL
CONSULTING CENTRE NIASRA**

Workshop cost is \$110 or \$100 when
paid via internal transfer.

**SCAN FOR
MORE DETAILS**



Did you know all academics and HDR students have access to statistical consulting?

Check out the Statistical Consulting Centre's website to learn how to book an appointment.

www.uow.edu.au/niasra/our-research/statistical-consulting-centre/



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

UOW.EDU.AU #THISISUOW



NIASRA
NATIONAL INSTITUTE FOR APPLIED
STATISTICS RESEARCH AUSTRALIA



UOW
AUSTRALIA



UNIVERSITY
OF WOLLONGONG
AUSTRALIA