MA425/625: Applied Regression

Introduction to Software

Emmanuel Thompson

Fall 2021

Introduction to Software

Topics

- Introduction to R & RStudio
- Getting Started
  - Installing R & RStudio
  - Understanding your Working Environment
- Installing R Packages and Managing Package Dependencies
- R Data Types, Import & Export Data in R
- Introduction to R Programming
- Data Manipulation
- Data Visualization
- Data Preprocessing
- Exploratory Data Analysis
- Making Reports with RMarkdown

Introduction to R & RStudio

R

- Free, open-source, modern, and versatile statistical software.
- Programming environment for statistical computing and graphics.
- Developed by statisticians as a tool for data analysis.
  - Currently the best tool for data analysis.
- Designed to perform mathematical operations on vectors & matrices.
- Facilitates rapid development of new tools based on user demand.

RStudio

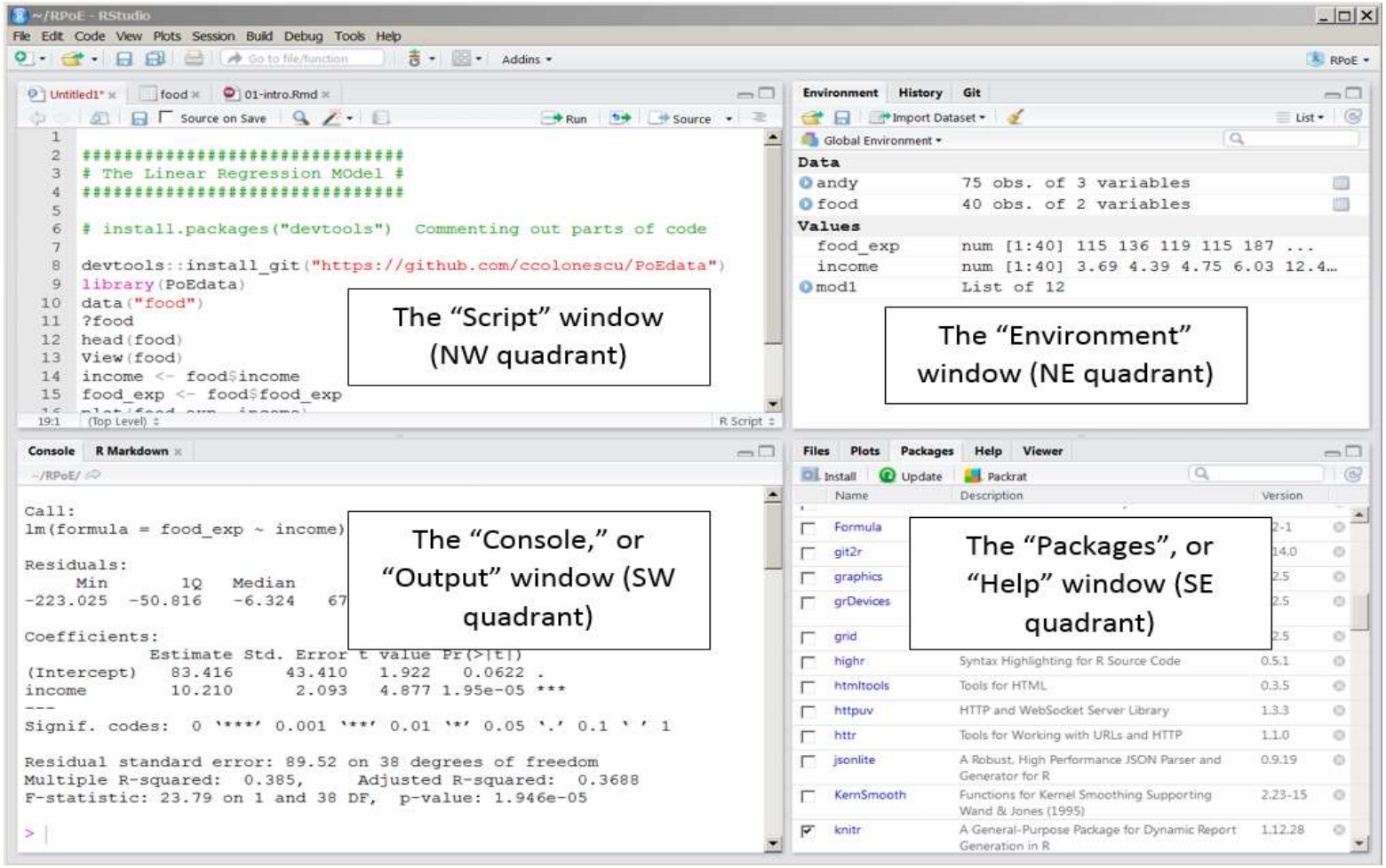
- RStudio is an integrated development environment, or IDE, for R programming.
- While R is an open source language, RStudio is a commercial product designed to make using R easier.
  - There is an open source version of RStudio that is available for free.
- Before installing RStudio, R must first be installed.
- It allows users to seamlessly import CSV, Excel, text (txt), SAS (`*.sas7bdat`), SPSS (`*.sav`), and Stata (`*.dta`) files into R without having to write the code to do so.

Getting Started

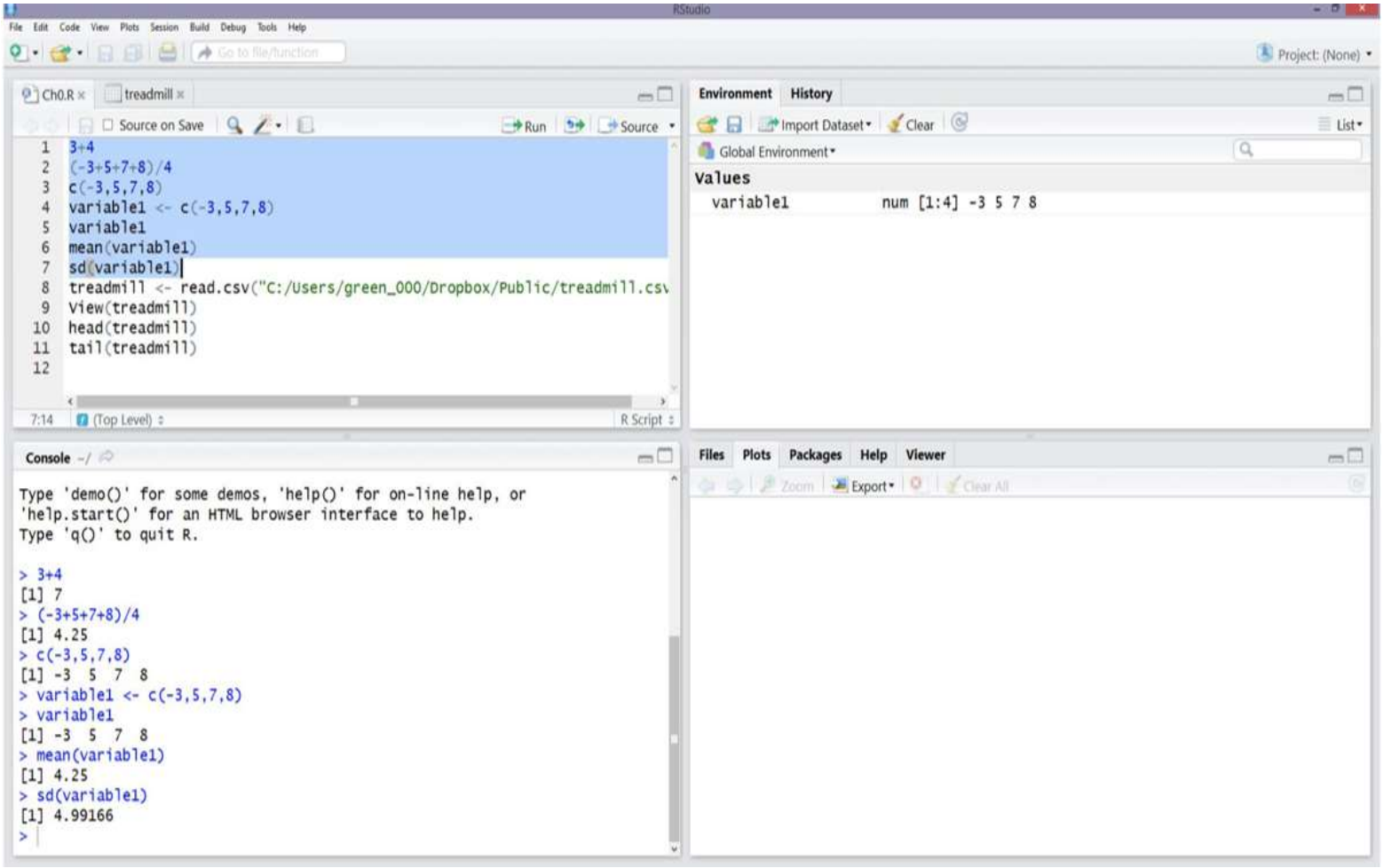
Installing R & RStudio

- To install R and RStudio on any computer, download the software from their associated websites.
  - Click ⇒ The R Project for Statistical Computing.
  - Click ⇒ RStudio.
- RStudio Cloud is a hosted version of RStudio in the cloud.
  - Click ⇒ RStudio Cloud.

Understanding your Working Environment



Variable & Assignment Operator



Installing R Packages and Managing Package Dependencies

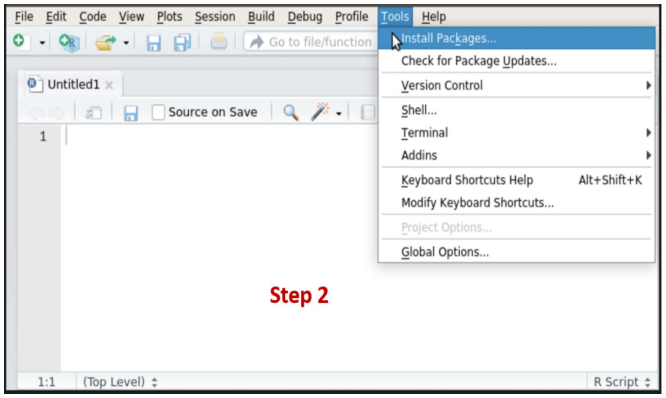
- Packages are collection of functions and datasets that add new features to R.
- Most R packages are available from `CRAN`, the official R repository.
  - `CRAN` is a network of servers (called mirrors) around the world.
- Packages on CRAN are checked before they are published, to make sure they don't contain malicious components.

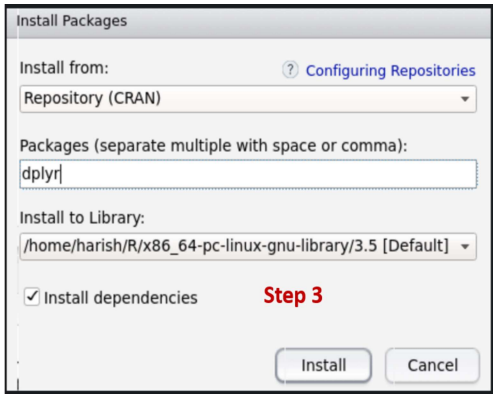
Installing R packages from the Terminal

- To install R packages from the Terminal, follow these steps:
  1. Open the Terminal.
  2. Type and run the following command. Make sure to replace package_name1 with an actual package name, such as `ggplot2`:
  - `install.packages("package_name1")`.
- You can also install multiple packages at the same time with both the R command line and RStudio. Just separate the individual packages with commas:
  - `install.packages("package_name1", "package_name2")`.

Installing R packages from inside RStudio

- `To install R packages from RStudio, follow these steps:`
  1. Open RStudio.
  2. Click on `Tools` from the menu bar and then click on `Install Packages`...:
  3. In the Install Packages dialog box, type in the package name in the `Packages text box` and click on the `Install button`:





There is also a list of common problems when installing packages available on the RStudio support page at https://support.rstudio.com/hc/en-us/articles/200554786-Problem-Installing-Packages.

Introduction to tidyverse

- The power of R comes from numerous packages available for you to use.
- `tidyverse` is a collection of R packages designed for data science.
  - All packages share an underlying design philosophy, grammar, and data structures.
- Install all the packages in the `tidyverse` by running:

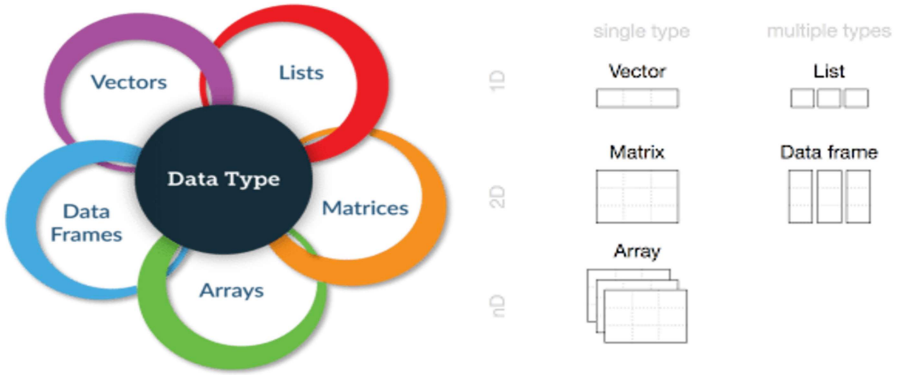
```
install.packages("tidyverse")
```

- To load the core `tidyverse` and make it available in your current R session run:

```
library(tidyverse)
```

- Learn more about the `tidyverse` package at https://www.tidyverse.org/.
  - Core `tidyverse` packages include: `ggplot2, dplyr, tidyr, readr, purr, tibble, stringr, and forcats`.

R Data Types, Import & Export Data in R




```
# Demonstrate in class!
a <- c(12,3.9,6,-2,-14.4) # numeric vector
b <- c("chris","danny","kevin") # character vector
c <- c(FALSE,TRUE,TRUE,FALSE,TRUE,TRUE) #logical vector
y <- matrix(1:20, nrow=5,ncol=4) # generates 5 x 4 numeric matrix
dat <- data.frame(id=c(1:4), Color=c("red", "white", "red", NA), # data frame
                  Passed=c(TRUE,TRUE,TRUE,FALSE))
mylist <- list(myvector_n=a, myvector_c=b, mymatrix=y, mydataframe=dat)
gender <- c(rep("male",20), rep("female", 30))
gender <- factor(gender)
```

- To explore data types in R, try this free interactive introduction to R course: https://www.datacamp.com/courses/free-introduction-to-r/

Importing Data into R

Working Directory

- To set working directory via point-and-click:
  1. Session|Set Working Directory|Choose Directory. In the dialog, highlight the directory and click Open.
  2. To set working directory with R code, use `setwd()` function; path must be in quotes.

```
# setwd("path")
setwd("C:/Users/ethom/Dropbox/MA425_625/Fall_2021/tools")
```

Loading Built-in Datasets in R

- To see the list of pre-loaded data, type the function `data()`:

```
data()
```

- Load and print data as follow:

```
data(Seatbelts)
head(Seatbelts, 5)
```

```
##      DriversKilled drivers front rear   kms PetrolPrice VanKilled law
## [1,]           107    1687   867  269  9059   0.1029718        12   0
## [2,]            97    1508   825  265  7685   0.1023630         6   0
## [3,]           102    1507   806  319  9963   0.1020625        12   0
## [4,]            87    1385   814  407 10955   0.1008733         8   0
## [5,]           119    1632   991  454 11823   0.1010197        10   0
```

```
data("iris")
head(iris, 5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

R Base Functions for Importing Data

```
# Read tabular data into R
read.table(file, header = FALSE, sep = "", dec = ".")
# Read "comma separated value" files (".csv")
read.csv(file, header = TRUE, sep = ",", dec = ".", ...)
# Or use read.csv2: variant used in countries that
# use a comma as decimal point and a semicolon as field separator.
read.csv2(file, header = TRUE, sep = ";", dec = ",", ...)
# Read TAB delimited files
read.delim(file, header = TRUE, sep = "\t", dec = ".", ...)
read.delim2(file, header = TRUE, sep = "\t", dec = ",", ...)
```

- file: the path to the file containing the data to be imported into R.
- sep: the field separator character. "\t" is used for tab-delimited file.
- header: logical value. If TRUE, read.table() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.
- dec: the character used in the file for decimal points.

Importing Local .txt or a .csv File

```
my_data <- read.csv("credit.csv", header = TRUE,  stringsAsFactor = FALSE)
head(my_data, 4)
```

```
##   checking_balance months_loan_duration credit_history              purpose
## 1           < 0 DM                    6       critical furniture/appliances
## 2       1 - 200 DM                   48           good furniture/appliances
## 3          unknown                   12       critical            education
## 4           < 0 DM                   42           good furniture/appliances
```

Importing Data from the Internet

- You can use the functions `read.delim()`, `read.csv()` and `read.table()` to import data from the internet.

```
test.txt <- read.table("https://stats.idre.ucla.edu/wp-content/uploads/2016/02/test.txt", header=T)
head(test.txt, 3)
```

```
##   make   model mpg weight price
## 1  AMC Concord  22   2930  4099
## 2  AMC   Pacer  17   3350  4749
## 3  AMC  Spirit  22   2640  3799
```

```
test.csv <- read.csv("https://stats.idre.ucla.edu/wp-content/uploads/2016/02/test-1.csv", header=T)
head(test.csv, 3)
```

```
##   make   model mpg weight price
## 1  amc concord  22   2930  4099
## 2  amc   oacer  17   3350  4749
## 3  amc  spirit  22   2640  3799
```

```
test.csv <- read.delim("https://stats.idre.ucla.edu/wp-content/uploads/2016/02/test.txt")
head(test.csv, 3)
```

```
##   make.model.mpg.weight.price
## 1    AMC Concord 22 2930 4099
## 2    AMC Pacer   17 3350 4749
## 3    AMC Spirit  22 2640 3799
```

Fast Reading of Data From txt|csv Files into R: readr package

```
library("readr")
# Read tab separated values
read_tsv(file.choose())
# Read comma (",") separated values
read_csv(file.choose())
# Read semicolon (";") separated values
read_csv2(file.choose())
```

Reading data From Excel Files (xls|xlsx) into R

```
# Use readxl package to read xls|xlsx
library("readxl")
my_data <- read_excel(file, sheet = "------") # For sheet specify an index or name.

# Use xlsx package
library("xlsx")
my_data <- read.xlsx(file, sheetIndex, header = TRUE)
```

Exporting Data from R

- Base functions for writing data: `write.table()`, `write.csv()`, `write.csv2()`.
- Fast writing of data from R to txt|csv files involves using `readr` function: `write_tsv()`, `write_csv()`. From `xlsx` package we use the function `write.xlsx()` for Excel files.

```
data("iris")
write.csv(iris, file = "iris1.csv")
library(readr)
write_csv(iris, path = "iris2.csv")
```

Introduction to R Programming

- Click ⇒ Easy R Programming Basics.
- Click ⇒Basic R Programming.

Data Manipulation

Data Inspection

- `head()` for first few rows of a matrix or data frame.
- `tail()` for last few rows of a matrix or data frame.
- `dim()` for dimension of a matrix or data frame.
- `str()` for displaying the structure of an R object.
- `nrow()` for number of rows of a matrix or data frame.
- `ncol()` for number of columns of a matrix or data frame.
- `summary()` for numeric variables.
- `quantile()` for quartiles.
- `table()` for categorical variables.
- `sum(is.na())` for counting the number of `NAs` in the entire dataset.

If you need to change the data type for any column, use the following functions:

- `as.character()` converts to a text string.
- `as.numeric()` converts to a number.
- `as.factor()` converts to a categorical variable.
- `as.integer()` converts to an integer.

Inspect the credit dataset using str():

```
credit_data <- read.csv("credit.csv", header = TRUE,  stringsAsFactor = FALSE)
dim(credit_data)
```

```
## [1] 1000   17
```

```
str(credit_data)
```

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ checking_balance    : chr  "< 0 DM" "1 - 200 DM" "unknown" "< 0 DM" ...
##  $ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30 ...
##  $ credit_history      : chr  "critical" "good" "critical" "good" ...
##  $ purpose             : chr  "furniture/appliances" "furniture/appliances" "education" "furniture/ap
##  $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ savings_balance     : chr  "unknown" "< 100 DM" "< 100 DM" "< 100 DM" ...
##  $ employment_duration : chr  "> 7 years" "1 - 4 years" "4 - 7 years" "4 - 7 years" ...
##  $ percent_of_income   : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ years_at_residence  : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ other_credit        : chr  "none" "none" "none" "none" ...
##  $ housing             : chr  "own" "own" "own" "other" ...
##  $ existing_loans_count: int  2 1 1 1 2 1 1 1 1 2 ...
##  $ job                 : chr  "skilled" "skilled" "unskilled" "skilled" ...
##  $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ phone               : chr  "yes" "no" "no" "no" ...
##  $ default             : chr  "no" "yes" "no" "no" ...
```

`dplyr` Package

- We will use the `dplyr` package from the `tidyverse` packages to manipulate data.
- Here are some of the most useful functions in `dplyr`:
  - `select:` Choose which columns to include.
  - `filter:` Filter the data.
  - `arrange:` Sort the data, by size for continuous variables, by date, or alphabetically.
  - `group_by:` Group the data by a categorical variable.
  - `summarize:` Summarize, or aggregate (for each group if following group_by). Often used in conjunction with functions including: mean, median, max, min, sum, n etc.
  - `mutate:` Create new column(s) in the data, or change existing column(s).
  - `rename:` Rename column(s).
  - `bind_rows:` Merge two data frames into one, combining data from columns with the same name.
  - `glimpse:` Used to see the columns of the dataset and display some portion of the data with respect to each attribute that can fit on a single line.

These functions can be chained together using the operator `%>%` which makes the output of one line of code the input for the next.

Gapminder data: Excerpt of the Gapminder data on life expectancy, GDP per capita, and population by country

```
# install.packages("gapminder")
library(gapminder)
library(tidyverse)
gdf <- as.data.frame(gapminder)
str(gdf)
```

```
## 'data.frame':    1704 obs. of  6 variables:
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
##  $ pop      : int  8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 16317921 222
##  $ gdpPercap: num  779 821 853 836 740 ...
```

```
glimpse(gdf)
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ~
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
## $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12~
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
```

Filter()

```
filter_us <- filter(gapminder, country == "United States")
head(filter_us)
```

```
## # A tibble: 6 x 6
##   country       continent  year lifeExp       pop gdpPercap
##   <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
## 1 United States Americas   1952    68.4 157553000    13990.
## 2 United States Americas   1957    69.5 171984000    14847.
## 3 United States Americas   1962    70.2 186538000    16173.
## 4 United States Americas   1967    70.8 198712000    19530.
## 5 United States Americas   1972    71.3 209896000    21806.
## 6 United States Americas   1977    73.4 220239000    24073.
```

```
# filter(gapminder, lifeExp < 30)
# filter(gapminder, pop < 1000000)
# filter(gapminder, pop < 1000000, year == 2007)
# filter(gapminder, pop < 1000000 & year == 2007)
# filter(gapminder, country %in% c("United States", "Canada"), year > 2000)
```

Filter()

```
filter_us_can_1 <- filter(gapminder, country %in% c("United States", "Canada"), year > 2000)
head(filter_us_can_1)
```

```
## # A tibble: 4 x 6
##   country       continent  year lifeExp       pop gdpPercap
##   <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
## 1 Canada        Americas   2002    79.8  31902268    33329.
## 2 Canada        Americas   2007    80.7  33390141    36319.
## 3 United States Americas   2002    77.3 287675526    39097.
## 4 United States Americas   2007    78.2 301139947    42952.
```

```
filter_us_can_2 <- filter(gapminder, country == "United States" | country == "Canada", year > 2000)
head(filter_us_can_2)
```

```
## # A tibble: 4 x 6
##   country       continent  year lifeExp       pop gdpPercap
##   <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
## 1 Canada        Americas   2002    79.8  31902268    33329.
## 2 Canada        Americas   2007    80.7  33390141    36319.
## 3 United States Americas   2002    77.3 287675526    39097.
## 4 United States Americas   2007    78.2 301139947    42952.
```

Select()

```
tcont <- select(gapminder, -continent)
head(tcont)
```

```
## # A tibble: 6 x 5
##   country      year lifeExp      pop gdpPercap
##   <fct>       <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan  1952    28.8  8425333      779.
## 2 Afghanistan  1957    30.3  9240934      821.
## 3 Afghanistan  1962    32.0 10267083      853.
## 4 Afghanistan  1967    34.0 11537966      836.
## 5 Afghanistan  1972    36.1 13079460      740.
## 6 Afghanistan  1977    38.4 14880372      786.
```

```
tcont <- select(gapminder, lifeExp, pop, gdpPercap, continent)
head(tcont)
```

```
## # A tibble: 6 x 4
##   lifeExp      pop gdpPercap continent
##     <dbl>    <int>     <dbl> <fct>
## 1    28.8  8425333      779. Asia
## 2    30.3  9240934      821. Asia
## 3    32.0 10267083      853. Asia
## 4    34.0 11537966      836. Asia
## 5    36.1 13079460      740. Asia
## 6    38.4 14880372      786. Asia
```

Filter() and Select()

```
filter_select_1 <- filter(gapminder, year == 2007) %>% select(country, year, lifeExp)
head(filter_select_1)
```

```
## # A tibble: 6 x 3
##   country      year lifeExp
##   <fct>       <int>   <dbl>
## 1 Afghanistan  2007    43.8
## 2 Albania      2007    76.4
## 3 Algeria      2007    72.3
## 4 Angola       2007    42.7
## 5 Argentina    2007    75.3
## 6 Australia    2007    81.2
```

```
filter_select_2 <- filter(gapminder, country == "United States" | country == "Canada", year > 2000) %>%
  select(country, year, lifeExp)
head(filter_select_2)
```

```
## # A tibble: 4 x 3
##   country        year lifeExp
##   <fct>         <int>   <dbl>
## 1 Canada         2002    79.8
## 2 Canada         2007    80.7
## 3 United States  2002    77.3
## 4 United States  2007    78.2
```

Mutate()

```
mut_1 <- mutate(gapminder, popMil = round(pop / 1000000, 1))
head(mut_1, 3)
```

```
## # A tibble: 3 x 7
##   country     continent  year lifeExp      pop gdpPercap popMil
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>  <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.    8.4
## 2 Afghanistan Asia       1957    30.3  9240934      821.    9.2
## 3 Afghanistan Asia       1962    32.0 10267083      853.   10.3
```

Group Data

```
dat <- select(gapminder, country, year, pop) %>%
  group_by(country) %>%
  mutate(pop_lag = lag(pop), pop_chg = pop - pop_lag,
         pop_pctchg = round(pop_chg/pop_lag * 100, 1)) %>%
  filter(pop_chg < 0)
head(dat, 3)
```

```
## # A tibble: 3 x 6
## # Groups:   country [2]
##   country                  year      pop  pop_lag  pop_chg pop_pctchg
##   <fct>                   <int>    <int>    <int>    <int>      <dbl>
## 1 Afghanistan              1982 12881816 14880372 -1998556      -13.4
## 2 Bosnia and Herzegovina   1992  4256013  4338977   -82964       -1.9
## 3 Bosnia and Herzegovina   1997  3607000  4256013  -649013      -15.2
```

Summarize()

```
group_by(gapminder, continent, country) %>%
summarise(avg_le_cc = mean(lifeExp))
```

```
## # A tibble: 142 x 3
## # Groups:   continent [5]
##    continent country                  avg_le_cc
##    <fct>     <fct>                        <dbl>
##  1 Africa    Algeria                       59.0
##  2 Africa    Angola                        37.9
##  3 Africa    Benin                         48.8
##  4 Africa    Botswana                      54.6
##  5 Africa    Burkina Faso                  44.7
##  6 Africa    Burundi                       44.8
##  7 Africa    Cameroon                      48.1
##  8 Africa    Central African Republic      43.9
##  9 Africa    Chad                          46.8
## 10 Africa    Comoros                       52.4
## # ... with 132 more rows
```

```
group_by(gapminder, continent) %>%
summarise(avg_le_cc = mean(lifeExp))
```

```
## # A tibble: 5 x 2
##   continent avg_le_cc
##   <fct>         <dbl>
## 1 Africa         48.9
```

Summarize()

```
group_by(gapminder, continent, year) %>%
summarise_at(c("lifeExp", "pop"), funs(min, median, max, mean, sd, IQR))
```

```
## # A tibble: 60 x 14
## # Groups:   continent [5]
##    continent  year lifeExp_min pop_min lifeExp_median pop_median lifeExp_max
##    <fct>     <int>       <dbl>   <int>          <dbl>      <dbl>       <dbl>
##  1 Africa     1952        30     60011           38.8   2668124.        52.7
##  2 Africa     1957        31.6   61325           40.6   2885790.        58.1
##  3 Africa     1962        32.8   65345           42.6   3145210         60.2
##  4 Africa     1967        34.1   70787           44.7   3473692.        61.6
##  5 Africa     1972        35.4   76595           47.0   3945594.        64.3
##  6 Africa     1977        36.8   86796           49.3   4522666         67.1
##  7 Africa     1982        38.4   98593           50.8   5668228.        69.9
##  8 Africa     1987        39.9  110812           51.6   6635612.        71.9
##  9 Africa     1992        23.6  125911           52.4   7140388.        73.6
## 10 Africa     1997        36.1  145608           52.8   7805422.        74.8
## # ... with 50 more rows, and 7 more variables: pop_max <int>,
## #   lifeExp_mean <dbl>, pop_mean <dbl>, lifeExp_sd <dbl>, pop_sd <dbl>,
## #   lifeExp_IQR <dbl>, pop_IQR <dbl>
```

Summarize()

```
filter(gapminder, year == 2007) %>%
group_by(continent) %>%
summarise(year = mean(year), ncountries = n(),
avg_country_le = mean(lifeExp), sd_country_le = sd(lifeExp))
```

```
## # A tibble: 5 x 5
##   continent  year ncountries avg_country_le sd_country_le
##   <fct>     <dbl>      <int>          <dbl>         <dbl>
## 1 Africa     2007         52           54.8         9.63
## 2 Americas   2007         25           73.6         4.44
## 3 Asia       2007         33           70.7         7.96
## 4 Europe     2007         30           77.6         2.98
## 5 Oceania    2007          2           80.7         0.729
```

```
filter(gapminder, year == 1952) %>%
group_by(continent) %>%
summarise(year = mean(year), ncountries = n(),
avg_country_le = mean(lifeExp), sd_country_le = sd(lifeExp))
```

```
## # A tibble: 5 x 5
##   continent  year ncountries avg_country_le sd_country_le
##   <fct>     <dbl>      <int>          <dbl>         <dbl>
## 1 Africa     1952         52           39.1         5.15
## 2 Americas   1952         25           53.3         9.33
## 3 Asia       1952         33           46.3         9.29
## 4 Europe     1952         30           64.4         6.36
## 5 Oceania    1952          2           69.3         0.191
```

Data Visualization

- ggplot2 is a plotting system developed by Hadley Wickham in 2005.
- It implements the graphics scheme described in the book *The Grammar of Graphics* by Leland Wilkinson.
- It has two primary functions for creating graphics, qplot() and ggplot().
- It uses a standardized system of syntax that makes it flexible to understand.
- It takes care of other features of graphics such as colors, scales, axis, legend position etc.

data, aesthetics and geometric shapes

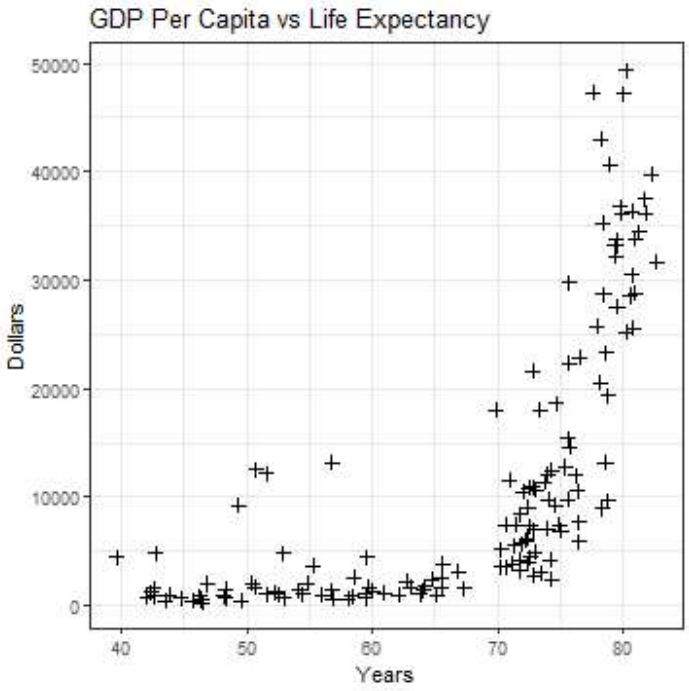
```
ggplot(data, aes(x=, y=, color=, shape=, size=)) +
geom_point(), or geom_histogram(), or geom_boxplot(), etc.
```

- The data must be a data frame.
- The aes() function maps columns of the data frame to aesthetic properties of geometric shapes to be plotted. Components are added with +.

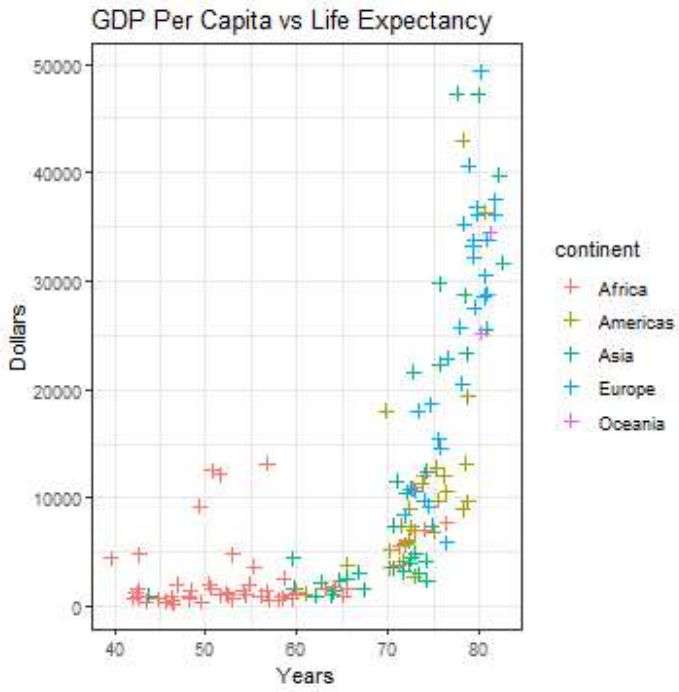
- We will use the `ggplot2` package from the `tidyverse` packages to visualize data.

Scatter Plot

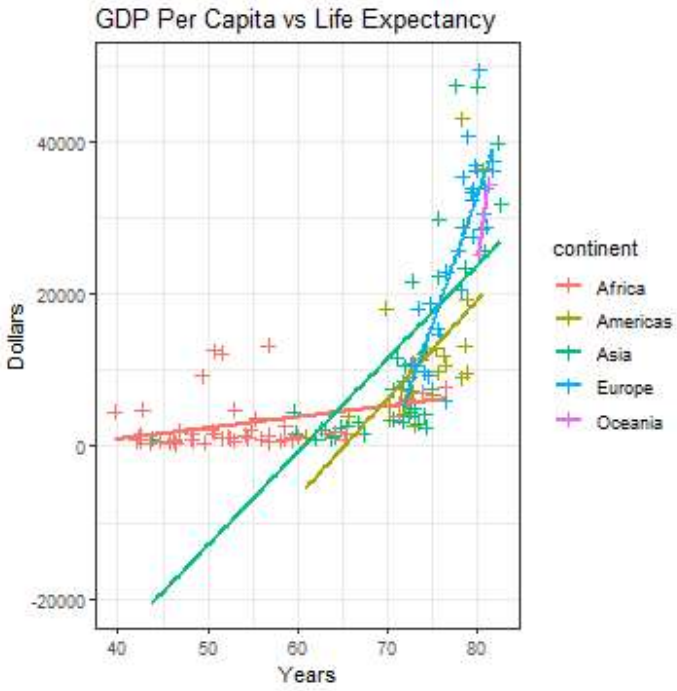
```
library(tidyverse)
filter(gapminder, year == 2007) %>%
ggplot(aes(x=lifeExp, y=gdpPercap)) +
  geom_point(shape = 3, size = 2) +
  labs(title="GDP Per Capita vs Life Expectancy", x="Years", y="Dollars")
```



```
filter(gapminder, year == 2007) %>%
ggplot(aes(x=lifeExp, y=gdpPercap, color = continent)) +
  geom_point(shape = 3, size = 2) +
  labs(title="GDP Per Capita vs Life Expectancy", x="Years", y="Dollars")
```



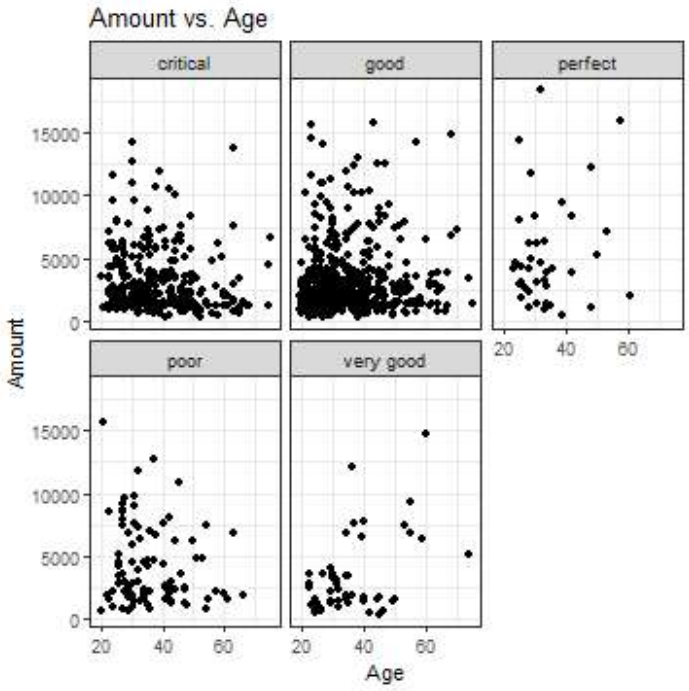
```
filter(gapminder, year == 2007) %>%
ggplot(aes(x=lifeExp, y=gdpPercap, color = continent)) +
  geom_point(shape = 3, size = 2) +
  geom_smooth(method= "lm", se = FALSE) +
  labs(title="GDP Per Capita vs Life Expectancy", x="Years", y="Dollars")
```



Scatter Plot with facet_wrap = Credit History

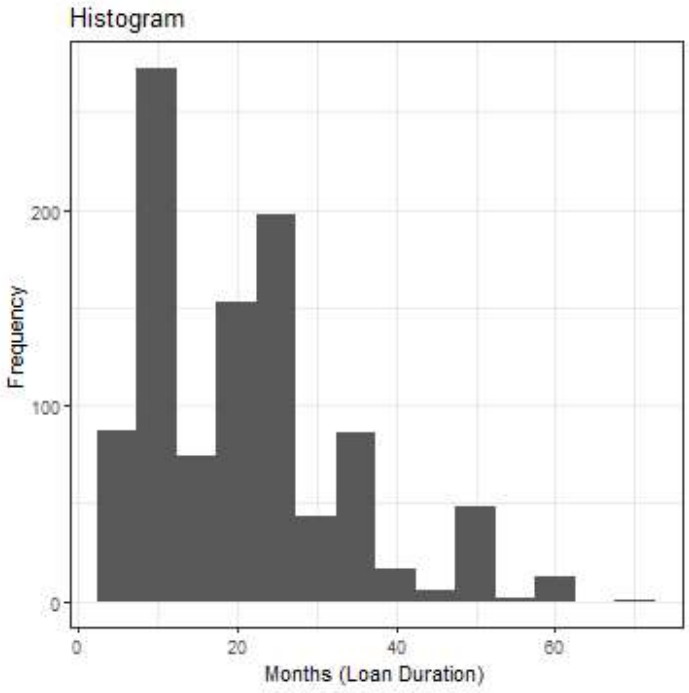
- Use of facet_wrap

```
ggplot(credit_data, aes(x=age, y=amount)) +
  geom_point() +
  facet_wrap(~ credit_history) +
  labs(title="Amount vs. Age", x="Age", y="Amount")
```



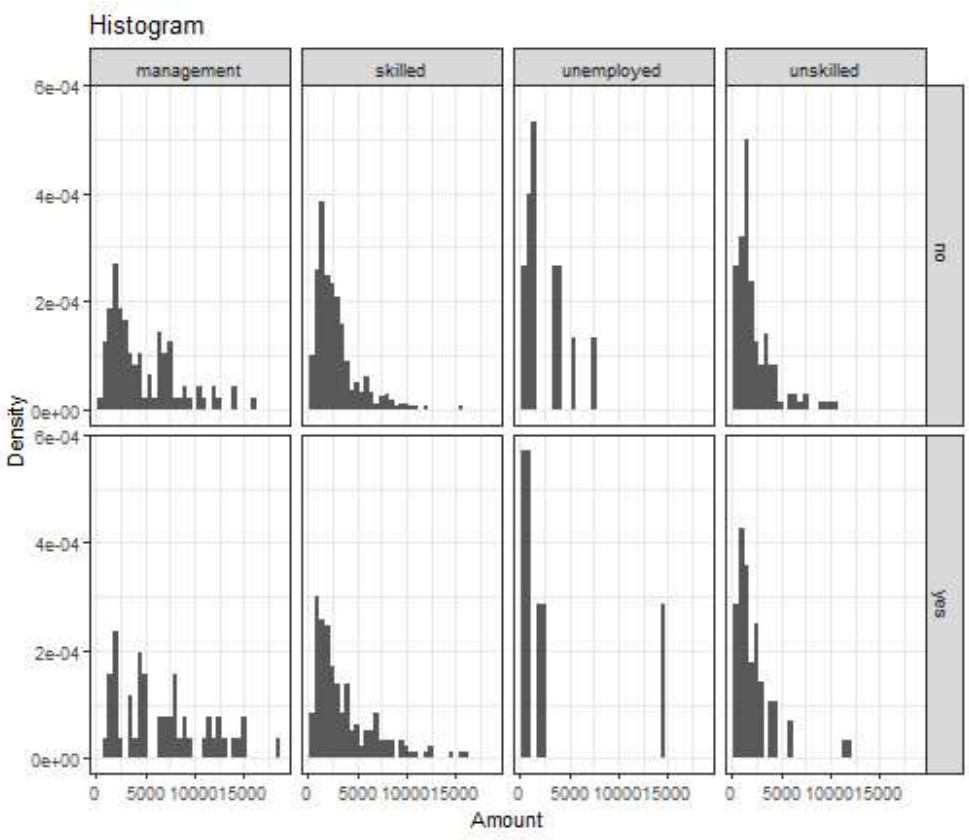
Histogram

```
ggplot(credit_data, aes(x = months_loan_duration)) +
  geom_histogram(aes(y = ..count..), binwidth = 5) +
labs(title="Histogram", x="Months (Loan Duration)", y="Frequency")
```



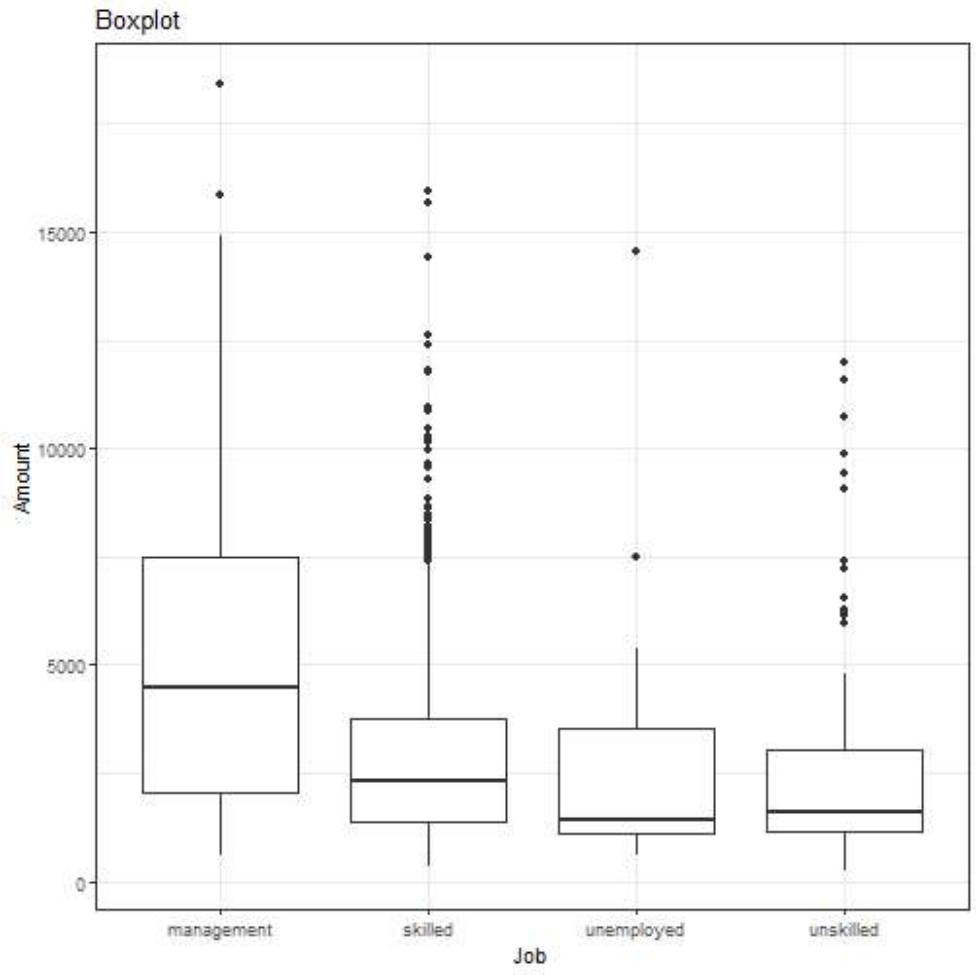
facet_grid histograms

```
ggplot(credit_data, aes(x=amount, y = stat(density))) +
  geom_histogram(aes(y = ..density..), binwidth = 500) +
  facet_grid(default ~ job) +
  labs(title="Histogram", x="Amount", y="Density")
```



Boxplot

```
ggplot(credit_data, aes(x=job, y=amount)) +
geom_boxplot() +
labs(title="Boxplot", x="Job", y="Amount")
```

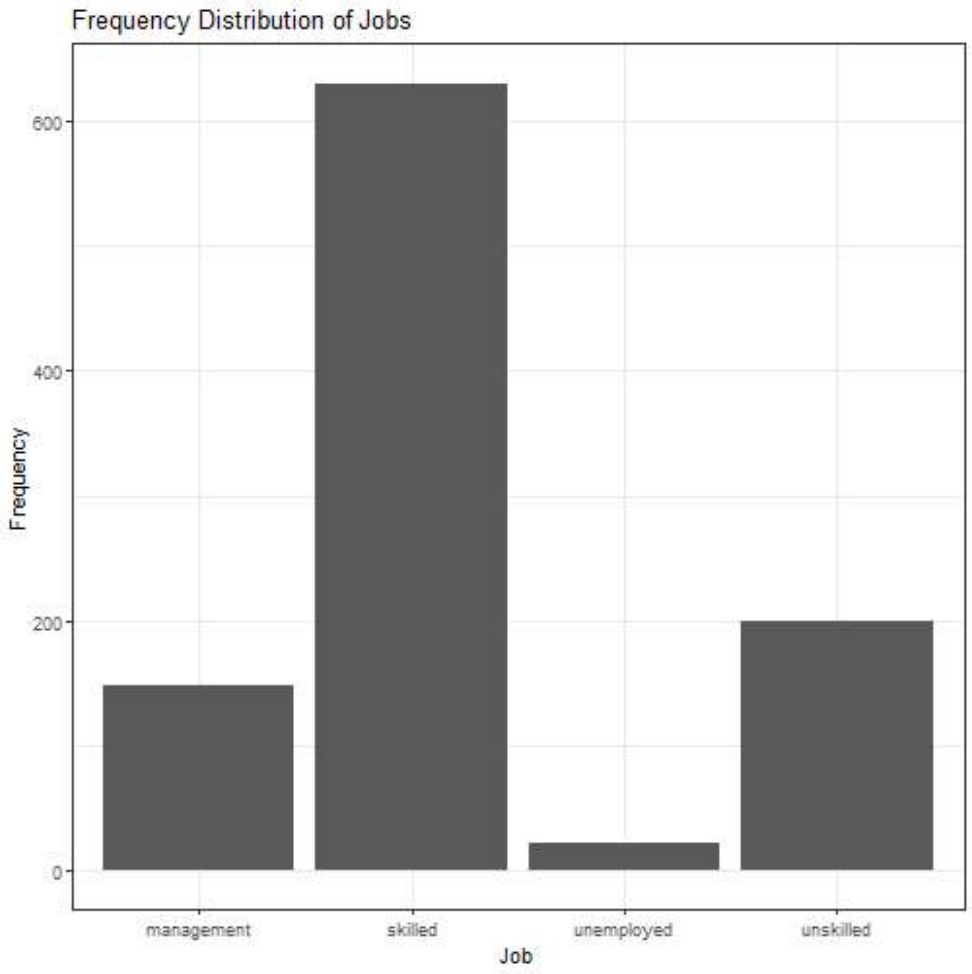


Boxplot

```
ggplot(credit_data, aes(x=checking_balance, y=amount)) +
geom_boxplot() +
facet_grid(job ~ .) +
labs(title="Boxplot", x="Checking Balance", y="Amount")
```

Bar chart

```
ggplot(credit_data, aes(job)) +
geom_bar() +
labs(title="Frequency Distribution of Jobs", x="Job", y="Frequency")
```



Explorating Data

Summary of Functions for Exploring Data in R

Exploratory Data Analysis (EDA)

- EDA is an approach used to summarise and visualise the main characteristics of a dataset.
- It focuses on:
  - Exploring data by understanding its structure and variables.
  - Developing an intuition about the dataset.
  - Considering how the dataset came into existence.
  - Deciding how to investigate by providing a formal statistical method.
  - Extending better insights about the dataset.
  - Formulating a hypothesis that leads to new data collection.
  - Handling any missing values.
  - Investigating with more formal statistical methods.
- Some of the graphical techniques used by EDA are:
  - Box plot
  - Histogram
  - Scatter plot
  - Bar chart
  - Density plots
  - Pareto chart

Functions for Obtaining Data Summary in R

Function Name	*Description*
`summary(x)`	Returns the min, max, median, and mean
`min(x)`	Returns the minimum value
`max(x)`	Returns the maximum value
`range(x)`	Returns the range of the given input
`mean(x)`	Returns the mean value
`median(x)`	Returns the median value
`mad(x)`	Returns the median absolute deviation value
`IQR(x)`	Returns the interquartile range
`quantile(x)`	Returns quartiles
`summary(x)`	Summarises the data frame
`apply(x,1,mean)`	Calculates the row mean value
`apply(x,2,mean)`	Calculates the column mean which is similar to the function of mean(x)

Function for Finding Missing Entities in a Dataset in R

Function Name	*Description*

`sum(is.na(mydata))`
Example:
`> sum(is.na(c))`
`[1] 1`
	Number of missing data in dataset

`rowSums(is.na(data))`
Example:
`> rowSums(is.na(c));`
`[1] 0 0 1`
The third row has one missing value.
	Number of missing data per variable

`rowMeans(is.na(data))*length(data)`
Example:
`> rowMeans(is.na(c))*length(c)`
`[1] 0 0 1`
	Number of missing data per row

Generating Reports with R Markdown

- R Markdown is used to generating high-quality reports combining both text and code that can be shared with an audience.
- It can also be used for executing and saving code.
- These links: link 1, link 2 contain R Markdown cheat sheet (Make sure you know how to use R Markdown) and link 3.
- There is an R Notebook which is also an R Markdown document. Take a look at this link for information on R Notebook.
- Commonly Used Symbols in R Markdown.

R Markdown Basic Requirements

- Install the following R packages:
  - `install.packages("rmarkdown", dependencies=TRUE)`
  - `install.packages("tidyverse", dependencies=TRUE)`
  - `install.packages("tinytex", dependencies=TRUE)`
- Run the following in RStudio to install TeX on your computer (this may take some time):
  - `tinytex::install_tinytex()`
- restart RStudio once you are done.

Working with R Markdown

- The following steps are used to start with R Markdown:
  1. Open RStudio IDE.
  2. Install the respective packages needed.
  3. Create a new R Markdown document as follows: select the Menu option by navigating to `File | New File | R Markdown`.
- The following screenshot shows the lookup of the required Markdown page:

Stay up to date

- RStudio Community - https://community.rstudio.com/
- Developer Blog - https://rstudio.link/developer_blog
- R Views Blog - https://rstudio.link/rviews_blog
- TensorFlow Blog - https://rstudio.link/tensorflow_blog
- Twitter - https://rstudio.link/twitter
- GitHub - https://rstudio.link/github
- LinkedIn - https://rstudio.link/linkedin
- YouTube - https://rstudio.link/youtube
- Facebook - https://rstudio.link/facebook
- Cookbook for R - http://www.cookbook-r.com/
- Quick R - https://www.statmethods.net/management/subset.html
- ggplot2: Elegant Graphics for Data Analysis - https://ggplot2-book.org/collective-geoms.html
- RStudio Cheatsheets - https://rstudio.com/resources/cheatsheets/
- Graphs - https://socviz.co/groupfacettx.html