To produce PDF file, you need TeX files.

- Easy way: Install the tinytex package: `install.packages("tinytex")`. Then run `tinytex::install_tinytex()`.
- If you want full version of TeX: For Mac install MacTeX. For Windows install TeX Live.

**Basic console output**

To insert an R code chunk, you can type it manually or just press Chunks - Insert chunks or use the shortcut key.

You can quickly insert chunks into your R Markdown file with the keyboard shortcut Cm + Option + I (Windows Ctr + Alt + I). This will produce the following code chunk:

**Prerequisite**

After cleaning the memory, we will first call the library `tidyverse`. Tidyverse loads a set of commonly used packages for data science, including `ggplot2` and `dplyr`.

```r
rm(list = ls()) # Clear memory

# Load packages

#install.packages("magrittr")
#install.packages("lattice")
#install.packages("stargazer")
#install.packages("pander")
#install.packages("kableExtra")
#install.packages("tidyverse")

library(magrittr)
library(lattice)
library(stargazer)
library(pander)
library(kableExtra)
library(tidyverse)
```
R Code chunk features

Create Markdown code from R

Frequently used chunk options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>include</td>
<td>If FALSE, knitr will run the chunk but not include the chunk in the final document</td>
</tr>
<tr>
<td>echo</td>
<td>If FALSE, knitr will not display the code in the code chunk above it’s results in the final document.</td>
</tr>
<tr>
<td>error</td>
<td>If FALSE, knitr will not display any error messages generated by the code.</td>
</tr>
<tr>
<td>message</td>
<td>If FALSE, knitr will not display any messages generated by the code.</td>
</tr>
<tr>
<td>warning</td>
<td>If FALSE, knitr will not display any warning messages generated by the code.</td>
</tr>
</tbody>
</table>

Recommendation for Homework

<table>
<thead>
<tr>
<th>Option</th>
<th>HW setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>include</td>
<td>TRUE</td>
</tr>
<tr>
<td>echo</td>
<td>TRUE</td>
</tr>
<tr>
<td>error</td>
<td>FALSE</td>
</tr>
<tr>
<td>message</td>
<td>FALSE</td>
</tr>
<tr>
<td>warning</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Data Frames Practice 1

1. Load econ.csv in R

```r
df <- read.csv("data/econ.csv")
```

2. What is the data structure? What does that tell us about type?

```r
# Check structure
str(df) # character, integer, numeric, and integer.
```

## `data.frame`: 557 obs. of 4 variables:
## $ country : chr "Afghanistan" "Afghanistan" "Afghanistan" "Albania" ...
## $ GwN : int 700 700 700 339 615 615 615 615 540 ...
## $ gdpPerCap: num 863 819 601 2962 1824 ...

3. Check the names and summary statistics of the data. Fix any names that are less than good.
# Check and fix names

names(df)

## [1] "country" "GWn" "year" "gdpPercap"

# Removing an uninteresting variable:

df <- df %>% select(-GWn)

# Two ways to change the names of a df:

# Using dplyr:

df <-
    df %>% rename(
        COUNTRY = country,
        gdp_PC = gdpPercap
    )

# Using base R

names(df) <- c("COUNTRY","Year","GDP_PC_PPP")

# Summary Statistics

summary(df)

##
##   COUNTRY Year GDP_PC_PPP
##  Length:557 Min. :1900 Min. : 228
##  Class :character 1st Qu.:1949 1st Qu.: 1357
##  Mode :character Median :1970 Median : 3386
##  Mean :1966 Mean : 6297
##  3rd Qu.:1989 3rd Qu.: 8657
##  Max. :2010 Max. :39157

4. Load pop.csv in R

```
df2 <- read.csv("data/pop.csv")
```

5. Check and remove observations with missing values

# I like to use the package questionr, but you will first have to install it

#questionr::freq.na(df2)

# Remove NAs

df2 <- na.omit(df2)
6. Merging two datasets

```r
# Makes sure that the reference columns have the same names:

names(df)
## [1] "COUNTRY" "Year" "GDP_PC_PPP"

names(df2)
## [1] "country" "GWn" "year" "pop" "region"

df2 %<>% # Note the assignment pipe, from magrittr
  rename(
    COUNTRY = country,
    Year = year
  )

df_m <- merge(df,df2,by=c("COUNTRY","Year"))

# check out ?merge

# Moreover, cbind() and rbind() may be useful in another applications
```

7. Calculate the average GDP per capita for Brazil for the observed period. Repeat the calculation for all countries.

```r
# Base R

mean(df_m[df_m$COUNTRY == "Brazil", "GDP_PC_PPP"])
## [1] 4184.175

# Tidy way

df_m %>%
  filter(COUNTRY == "Brazil") %>%
  summarize(mean(GDP_PC_PPP))

## mean(GDP_PC_PPP)
## 1 4184.175

# Average gdp.per.cap for all countries

df_m %>%
  group_by(COUNTRY) %>%
  summarize(mean(GDP_PC_PPP))%>%
  head() %>%   # Let's only show the first six observations
  kbl()
```
<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>mean(GDP_PC_PPP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afghanistan</td>
<td>760.6971</td>
</tr>
<tr>
<td>Albania</td>
<td>2962.2794</td>
</tr>
<tr>
<td>Algeria</td>
<td>2649.4482</td>
</tr>
<tr>
<td>Angola</td>
<td>825.4988</td>
</tr>
<tr>
<td>Argentina</td>
<td>4798.4565</td>
</tr>
<tr>
<td>Australia</td>
<td>7914.3872</td>
</tr>
</tbody>
</table>

Sometimes we are interested in reporting specific numbers from a dataset. To minimize human mistakes, we can embed R code into an R Markdown document directly with `r`.

For example, the final data set `df_m` has 508 observations and the average GDP per capita in Brazil is 4184.2.

8. Let’s create some data

```r
rm(list = ls())  # Clear memory
set.seed(1234)   # For replication
x <- 1:10
y <- round(rnorm(10, mean=x, sd=1), digits=2)
df <- data.frame(x, y)
```

9. Basic markdown functionality

For those not familiar with standard Markdown, the following may be useful. See the source code for how to produce such points. However, RStudio does include a Markdown quick reference button that adequately covers this material.

**Dot Points**

Simple dot points:

- Point 1
- Point 2
- Point 3

and numeric dot points:

1. Number 1
2. Number 2
3. Number 3

and nested dot points:

- A
  - A.1
  - A.2
- B
  - B.1
  - B.2
10. Equations

Equations are included by using LaTeX notation and including them either between single dollar signs (inline equations) or double dollar signs (displayed equations). If you hang around the Q&A site CrossValidated you’ll be familiar with this idea.

There are inline equations such as \( y_i = \alpha + \beta x_i + e_i \).

And displayed formulas:

\[
\frac{1}{1+\exp(-x)}
\]

\[
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]

\[
X = (x + a)(x - b)
\]

\[
= x(x - b) + a(x - b)
\]

\[
= x^2 + x(a - b) - ab
\]

More info: LaTeX wiki

11. Tables

Tables can be included using the following notation

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Male</td>
<td>Blue</td>
</tr>
<tr>
<td>2</td>
<td>Female</td>
<td>Pink</td>
</tr>
</tbody>
</table>

Or you want to show nice regression tables

```r
Mod1 <- y ~ x
Res1 <-
  lm(formula = Mod1,
      data = df)
Mod2 <- y ~ x^2
Res2 <-
  lm(formula = Mod2,
      data = df)

stargazer(Res1, Res2)
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Jan 07, 2022 - 1:36:02 AM

#For html
#stargazer(Res1, Res2, type = "html")

More info: Cheat Sheet

If you want to create a fancy table from data.frame, you can use “pander” or “kable”
Table 4:

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.965***</td>
<td>0.965***</td>
</tr>
<tr>
<td></td>
<td>(0.116)</td>
<td>(0.116)</td>
</tr>
<tr>
<td>Constant</td>
<td>-0.192</td>
<td>-0.192</td>
</tr>
<tr>
<td></td>
<td>(0.718)</td>
<td>(0.718)</td>
</tr>
<tr>
<td>Observations</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>R²</td>
<td>0.897</td>
<td>0.897</td>
</tr>
<tr>
<td>Adjusted R²</td>
<td>0.884</td>
<td>0.884</td>
</tr>
<tr>
<td>Residual Std. Error (df = 8)</td>
<td>1.051</td>
<td>1.051</td>
</tr>
<tr>
<td>F Statistic (df = 1; 8)</td>
<td>69.541***</td>
<td>69.541***</td>
</tr>
</tbody>
</table>

Note: *p<0.1; **p<0.05; ***p<0.01

```r
Table <-
df %>%
  mutate(z = if_else(y>5, 1, 0)) %>%
t()

Table
```
```
#> x 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00
#> y -0.21 2.28 4.08 1.65 5.43 6.51 6.43 7.45 8.44 9.11
#> z  0.00 0.00 0.00 0.00 1.00 1.00 1.00 1.00 1.00 1.00

With pandar

```
Table %>%
pander(caption ="Fancy Table")
```

Table 5: Fancy Table

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>-0.21</td>
<td>2.28</td>
<td>4.08</td>
<td>1.65</td>
<td>5.43</td>
<td>6.51</td>
<td>6.43</td>
<td>7.45</td>
<td>8.44</td>
<td>9.11</td>
</tr>
<tr>
<td>z</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

With kable

```
Table %>%
kableExtra::kable()
```
```
x  | 1.00 | 2.00 | 3.00 | 4.00 | 5.00 | 6.00 | 7.00 | 8.00 | 9.00 | 10.00 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>-0.21</td>
<td>2.28</td>
<td>4.08</td>
<td>1.65</td>
<td>5.43</td>
<td>6.51</td>
<td>6.43</td>
<td>7.45</td>
<td>8.44</td>
<td>9.11</td>
</tr>
<tr>
<td>z</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>
12. Plots

You can also show plots

```r
df %>%
ggplot(aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x) +
  labs(title = "Sample Plot",
       y = "Happiness",
       x = "Exam score") +
  theme_bw()
```

![Sample Plot](images/terry-schiavo-misleading-graph.jpg)

13. Images

Images can be called using `include_graphics`.

```r
knitr::include_graphics("images/terry-schiavo-misleading-graph.jpg")
```
Source: Statistics How To “Misleading Graphs: Real Life Examples”