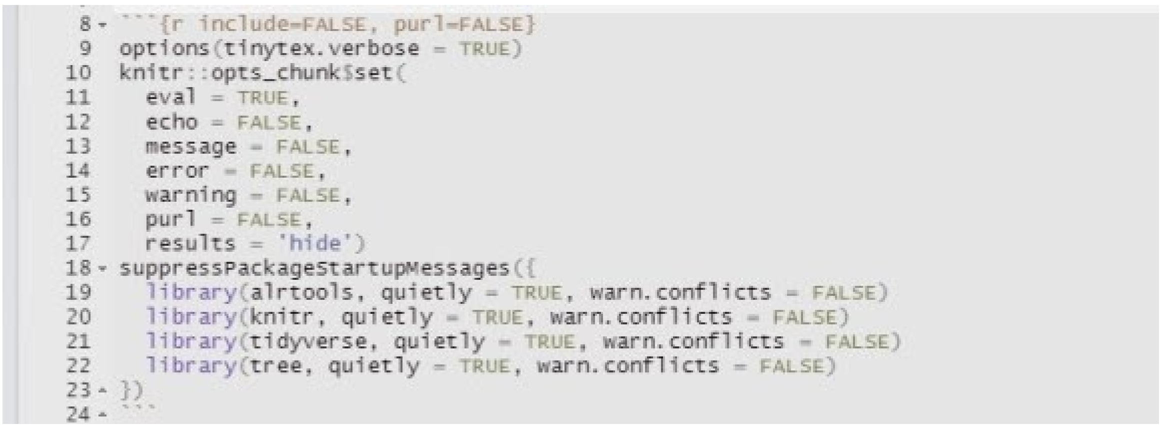
Intro

R markdown files allow you to show code and outputs in the order they were run.
However, in a class I'm taking currently,
our professor doesn't want to see our R code until the end of the report,
in an appendix.
So, she has said that our reports *should not* be compiled from R markdown files.
But, there is a way to create PDF reports from R markdown files where the code
echoing is suppressed and instead shown in an appendix! The Rmd file above is an example of that.

I'll show a bunch of example code chunks so you can see some different options.
The inline images below are not part of the rendered output –
they are screenshots from the Rmd file.

Setup Chunk



~~~
```{r include=FALSE, purl=FALSE}
options(tinytex.verbose = TRUE)
knitr::opts_chunk$set(
  eval = TRUE,
  echo = FALSE,
  message = FALSE,
  error = FALSE,
  warning = FALSE,
  purl = FALSE,
  results = 'hide')
suppressPackageStartupMessages({
  library(alrtools, quietly = TRUE, warn.conflicts = FALSE)
  library(knitr, quietly = TRUE, warn.conflicts = FALSE)
  library(tidyverse, quietly = TRUE, warn.conflicts = FALSE)
  library(tree, quietly = TRUE, warn.conflicts = FALSE)
})
```
~~~

Please notice above the `setup` chunk.
There are a couple of things I want to point out:

- The chunk options are very different from what you are used to
- Every package required anywhere in the report is loaded right up front
- The `setup` chunk *is not* included in the appendix! It is reserved solely for code that is required to facilitate document generation

So, why have I put `library` statements there?
You'll see that the `library` statements are wrapped in
`suppressPackageStartupMessages`
and that I've passed a few extra parameters that you may not have seen before.
This means that packages will not produce any pesky output in your report
when they are loaded. However, because we don't want to include the `setup` chunk
in the appendix, you will want to "re-load" every package
within code chunks that *will* end up in the appendix.

A Note About Default Chunk Options

You can ignore this section on the first read.
Just follow the conventions outlined below for the different examples.

Default option	Why?
`eval = TRUE`	All R code is executed by default
`echo = FALSE`	Do not show R code at the time it is run
`message = FALSE`	Do not show any messages
`error = FALSE`	Do not show any warnings
`warning = FALSE`	Do not show any errors
`purl = FALSE`	By default, code chunks *will not* appear in the appendix. You will have to explicitly mark the ones you want to include
`results = 'hide'`	You are probably used to code chunks outputing something to include in your report. If you want this, you'll have to explicitly override this option!

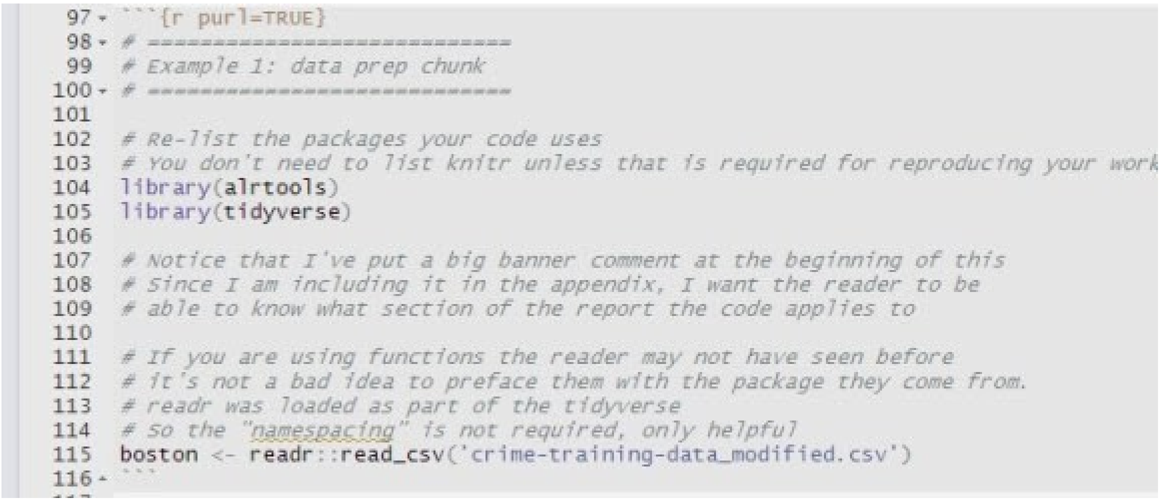
Examples of Different Configurations

Example 1: Data Prep Chunk

You'll use this kind of code chunk when you are prepping data
for use in other chunks, but there won't be any output to the
report. You want the code in the appendix so the reader can reproduce your work,
but there isn't any output yet.

Chunk options:

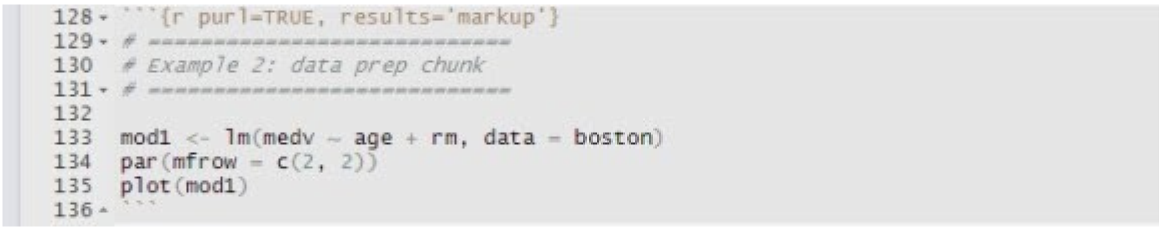
- Default options apply
- `purl=TRUE` means "include in appendix"

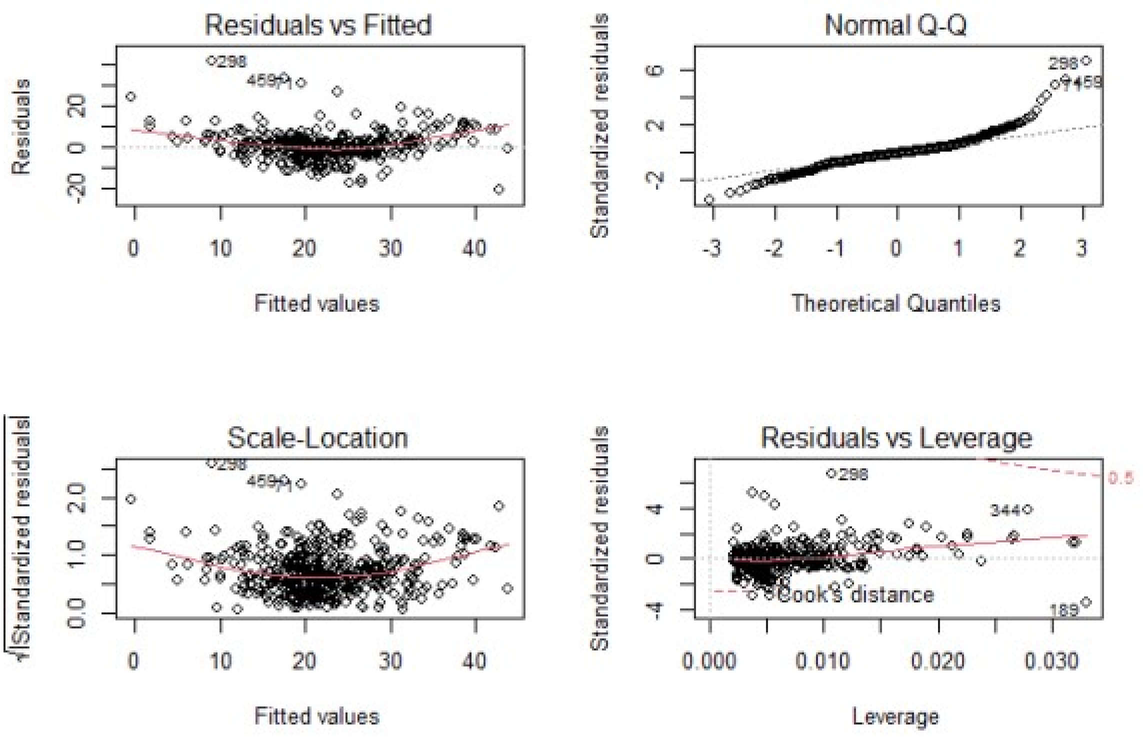


~~~
```{r purl=TRUE}
# =============================
# Example 1: data prep chunk
# =============================

# Re-list the packages your code uses
# You don't need to list knitr unless that is required for reproducing your work
library(alrtools)
library(tidyverse)

# Notice that I've put a big banner comment at the beginning of this
# Since I am including it in the appendix, I want the reader to be
# able to know what section of the report the code applies to

# If you are using functions the reader may not have seen before
# it's not a bad idea to preface them with the package they come from.
# readr was loaded as part of the tidyverse
# so the "namespacing" is not required, only helpful
boston <- readr::read_csv('crime-training-data_modified.csv')
```
~~~

Example 2: Content Chunk

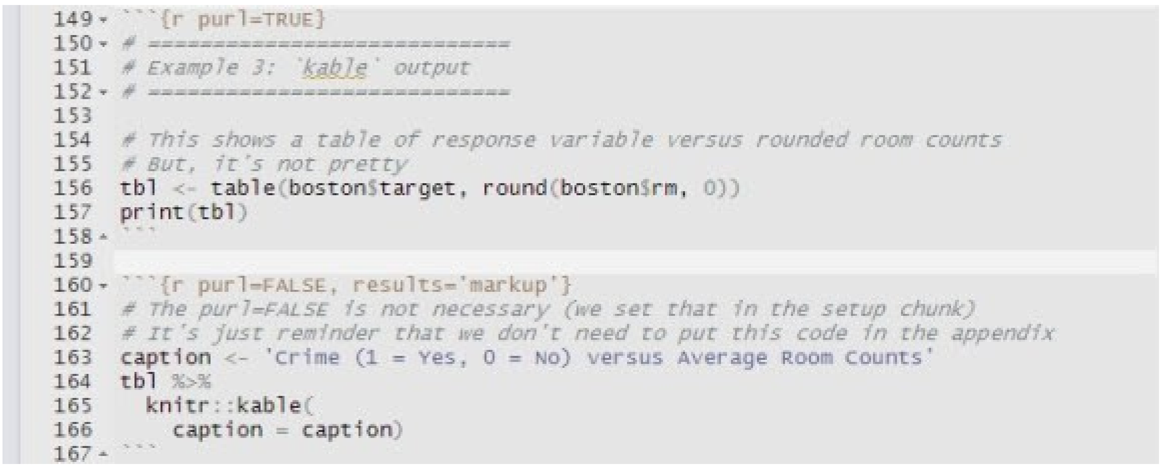
The option `results='markup'` is what you are used to working with in Rmd files.
There are other values you can set `results` to, but you probably won't use them very often.
(Except for `asis`, and you will see an example of that below when we bootstrap in the appendix.)


~~~
```{r purl=TRUE, results='markup'}
# ============================
# Example 2: data prep chunk
# ============================

mod1 <- lm(medv ~ age + rm, data = boston)
par(mfrow = c(2, 2))
plot(mod1)
```
~~~



Example 3: `kable` Output

Let’s say you want to put some table output in your report.
But, you want the reader, when they run your code, to be able to
get readable output.
(Nicely formatted stuff will have a lot of extra tags around it and isn’t always the easiest to read.)


~~~
```{r purl=TRUE}
# ============================
# Example 3: `kable` output
# ============================

# This shows a table of response variable versus rounded room counts
# But, it's not pretty
tbl <- table(boston$target, round(boston$rm, 0))
print(tbl)
~~~

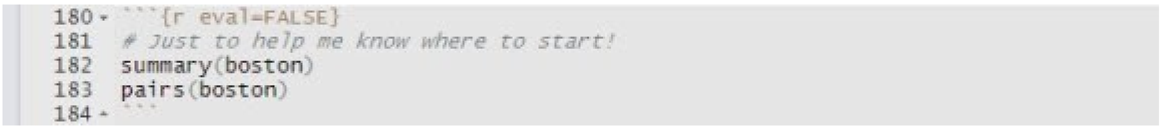
```{r purl=FALSE, results='markup'}
# The purl=FALSE is not necessary (we set that in the setup chunk)
# It's just reminder that we don't need to put this code in the appendix
caption <- 'Crime (1 = Yes, 0 = No) versus Average Room Counts'
tbl %>%
  knitr::kable(
    caption = caption)
```
```

Table: Crime (1 = Yes, 0 = No) versus Average Room Counts

4 5 6 7 8 9
```

|   | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|
| 0 | 0 | 4 | 148 | 73 | 12 | 0 |
| 1 | 4 | 33 | 136 | 42 | 11 | 3 |

## Example 4: Experiments

You're going to try lots of stuff when you are writing your report.
But, why should you have to delete the code just because it ended up
not being needed?

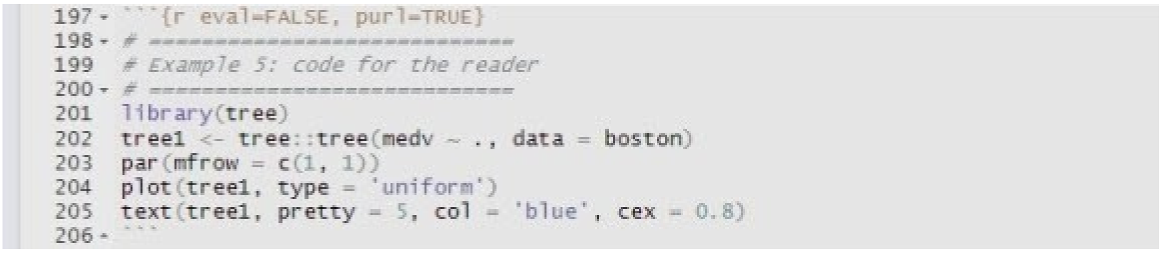
Remember `purl=FALSE` and `results='hide'` are set by default.



~~~
```{r eval=FALSE}
# Just to help me know where to start!
summary(boston)
pairs(boston)
```
~~~

Example 5: Code for the Reader

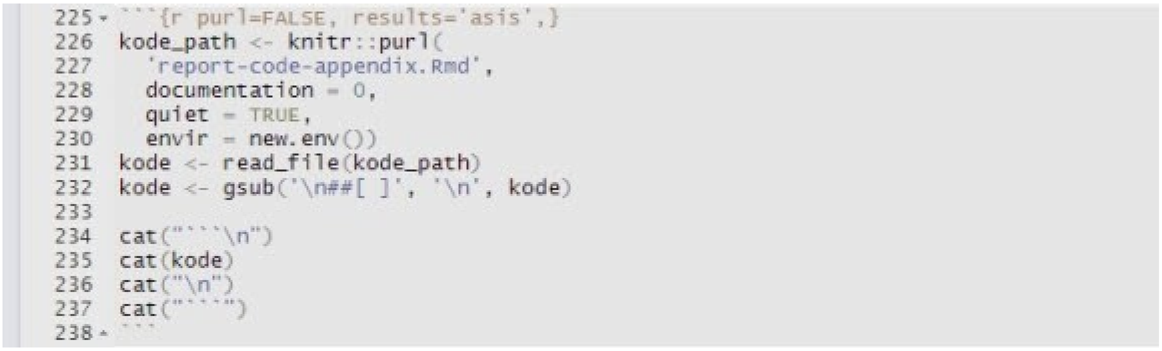
The following chunk won't do anything for your report or analysis, but will show up in the appendix.
This might be used for something that you experimented with and talked about,
but doesn't have any content for your report.
The reader might want to see what you tried if you've mentioned it in your write-up.



~~~
```{r eval=FALSE, purl=TRUE}
# ============================
# Example 5: code for the reader
# ============================
library(tree)
tree1 <- tree::tree(medv ~ ., data = boston)
par(mfrow = c(1, 1))
plot(tree1, type = 'uniform')
text(tree1, pretty = 5, col = 'blue', cex = 0.8)
```
~~~

Appendix 1: R Code for Analysis

And, here is the appendix.
I haven't figured out how to get the file name of the
Rmd file knitr is compiling, so that is hardcoded.
(It's the name of this Rmd file!)



~~~
```{r purl=FALSE, results='asis',}
kode_path <- knitr::purl(
  'report-code-appendix.Rmd',
  documentation = 0,
  quiet = TRUE,
  envir = new.env())
kode <- read_file(kode_path)
kode <- gsub('\n##[ ]', '\n', kode)

cat("```\n")
cat(kode)
cat("\n")
cat("```")
```
~~~

```
# ============================
# Example 1: data prep chunk
# ============================
```

```
# Re-list the packages your code uses
# You don't need to list knitr unless that is required for reproducing
your work
library(alrtools)
library(tidyverse)

# Notice that I've put a big banner comment at the beginning of this
# Since I am including it in the appendix, I want the reader to be
# able to know what section of the report the code applies to

# If you are using functions the reader may not have seen before
# it's not a bad idea to preface them with the package they come from.
# readr was loaded as part of the tidyverse
# So the "namespacing" is not required, only helpful
boston <- readr::read_csv('crime-training-data_modified.csv')

# ============================
# Example 2: data prep chunk
# ============================
mod1 <- lm(medv ~ age + rm, data = boston)
par(mfrow = c(2, 2))
plot(mod1)

# ============================
# Example 3: `kable` output
# ============================
# This shows a table of response variable versus rounded room counts
# But, it's not pretty
tbl <- table(boston$target, round(boston$rm, 0))
print(tbl)

# ============================
# Example 5: code for the reader
# ============================
library(tree)
tree1 <- tree::tree(medv ~ ., data = boston)
par(mfrow = c(1, 1))
plot(tree1, type = 'uniform')
text(tree1, pretty = 5, col = 'blue', cex = 0.8)…
```