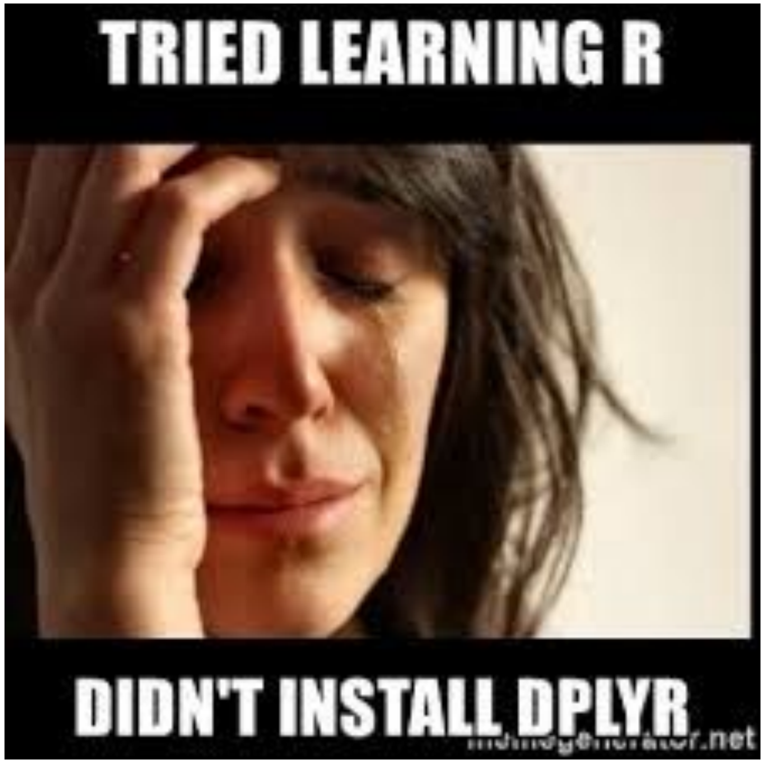
Intermediate R: Tidyverse

Lecture 6
BIOS 6660, Spring 2019
Instructor: Pam Russell



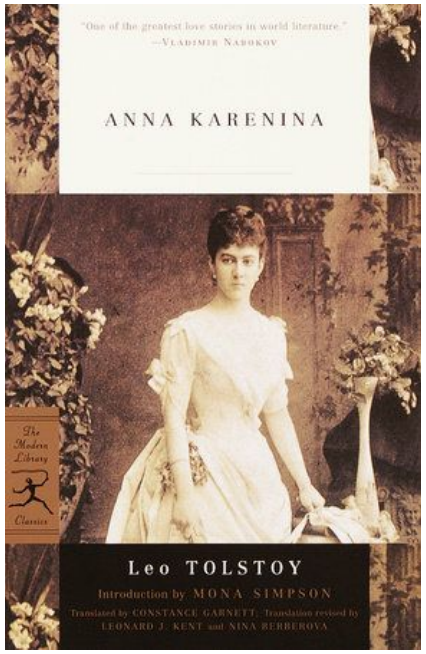
Tidy data

“Happy families are all alike; every unhappy family is unhappy in its own way.”

- Leo Tolstoy

“Tidy datasets are all alike but every messy dataset is messy in its own way.”

- Hadley Wickham

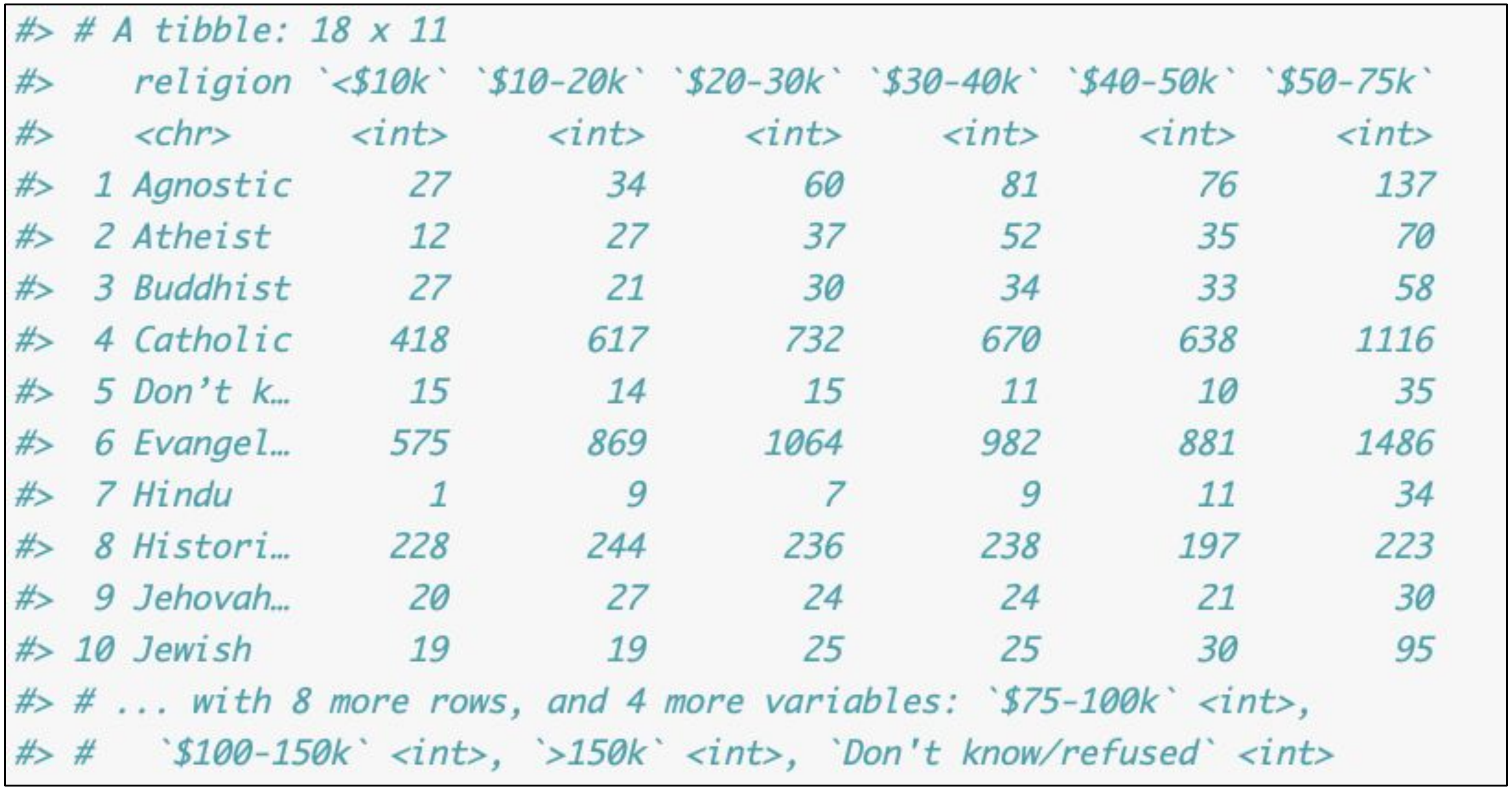

"One of the greatest love stories in world literature." —Vladimir Nabokov
ANNA KARENINA
Leo TOLSTOY
Introduction by MONA SIMPSON
Translated by CONSTANCE GARNETT; Translation revised by LEONARD J. KENT and NINA BERBEROVA

Tidy data

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

Example: not tidy

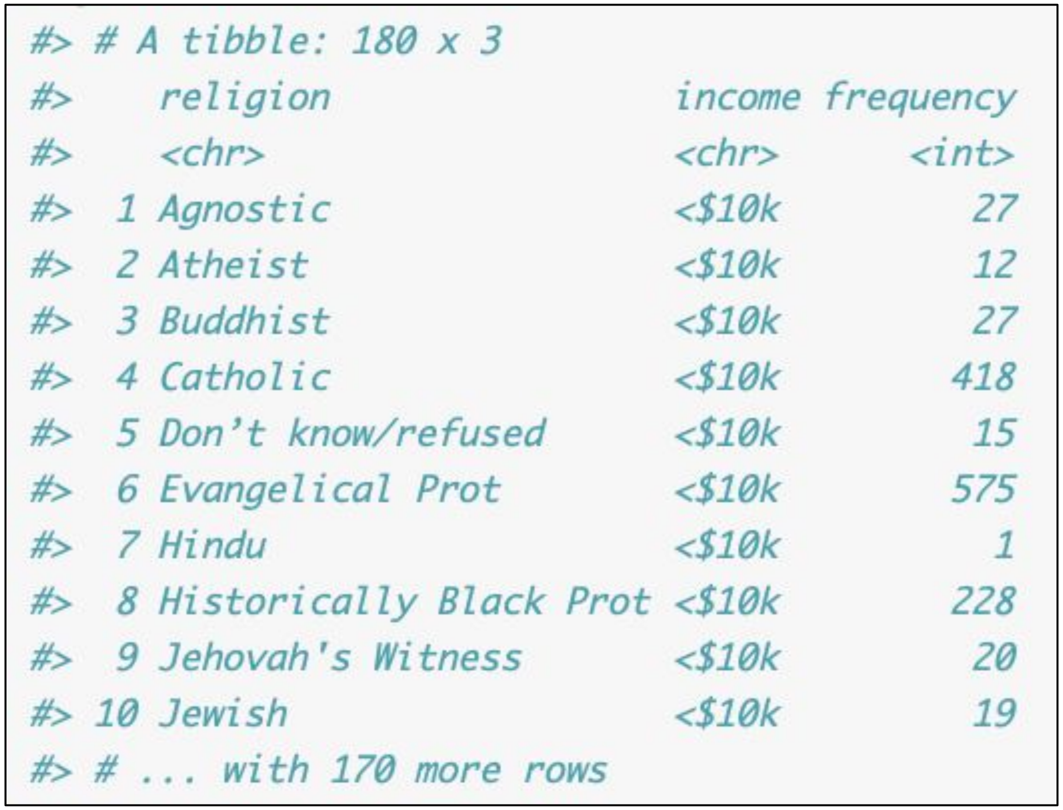


```
#> # A tibble: 18 x 11
#>    religion `<$10k` `$10-20k` `$20-30k` `$30-40k` `$40-50k` `$50-75k`
#>    <chr>      <int>     <int>     <int>     <int>     <int>     <int>
#>  1 Agnostic      27        34        60        81        76       137
#>  2 Atheist       12        27        37        52        35        70
#>  3 Buddhist      27        21        30        34        33        58
#>  4 Catholic     418       617       732       670       638      1116
#>  5 Don't k…      15        14        15        11        10        35
#>  6 Evangel…     575       869      1064       982       881      1486
#>  7 Hindu          1         9         7         9        11        34
#>  8 Histori…     228       244       236       238       197       223
#>  9 Jehovah…      20        27        24        24        21        30
#> 10 Jewish        19        19        25        25        30        95
#> # ... with 8 more rows, and 4 more variables: `$75-100k` <int>,
#> #   `$100-150k` <int>, `>150k` <int>, `Don't know/refused` <int>
```

Column headers are values, not variable names

Actually has three variables: religion, income, frequency

Tidy version



```
#> # A tibble: 180 x 3
#>    religion                income frequency
#>    <chr>                   <chr>      <int>
#>  1 Agnostic                <$10k         27
#>  2 Atheist                 <$10k         12
#>  3 Buddhist                <$10k         27
#>  4 Catholic                <$10k        418
#>  5 Don’t know/refused      <$10k         15
#>  6 Evangelical Prot        <$10k        575
#>  7 Hindu                   <$10k          1
#>  8 Historically Black Prot <$10k        228
#>  9 Jehovah's Witness       <$10k         20
#> 10 Jewish                  <$10k         19
#> # ... with 170 more rows
```

Tidyverse

A coherent system of packages for data manipulation, exploration and visualization that share a common design philosophy.

Data manipulation

Visualization

Tidying messy datasets

Examples of messy datasets and how to tidy them, including the one on the previous slides:
https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

Tidy data paper:
https://www.jstatsoft.org/article/view/v059i10

Pipe operator: %>%
Organize your steps as a story

```
# Equivalent to f(x)
x %>% f()


# Equivalent to f(x, y)
# x is passed as first argument to f
x %>% f(y)


# Equivalent to h(g(f(x, y)), z, w)
x %>% f(y)
  %>% g()
  %>% h(z, w)
```

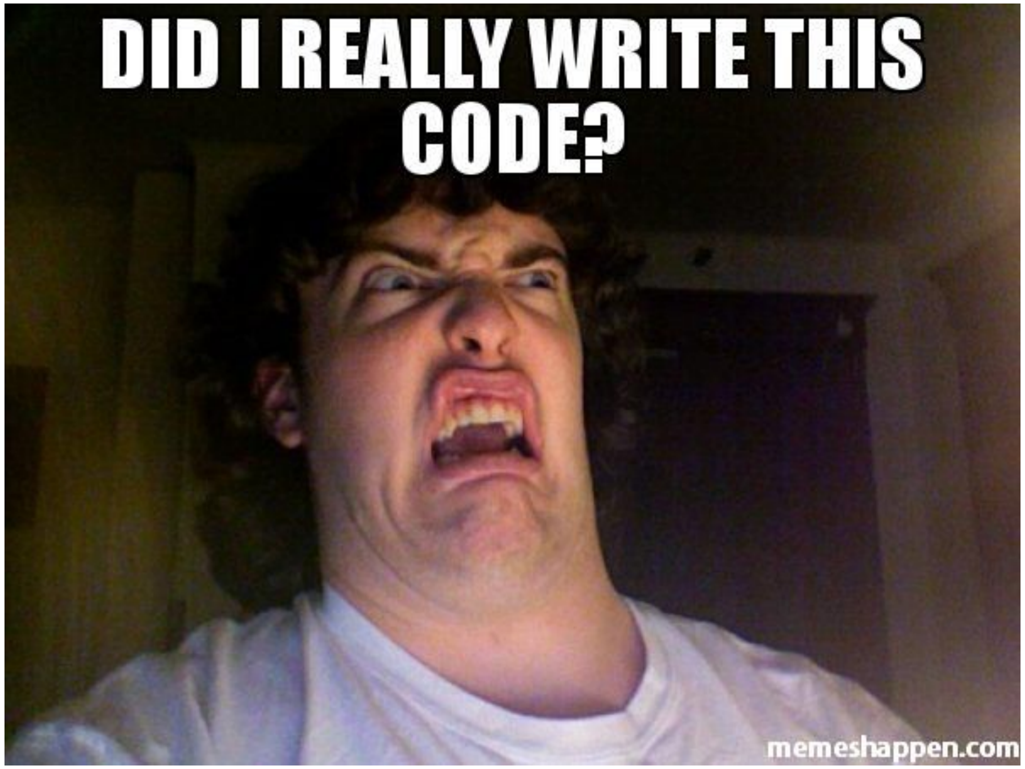
dplyr and ggplot2

Slides:

`L6_dplyr_ggplot2.html`

Code organization

Lecture 6
BIOS 6660, Spring 2019
Instructor: Pam Russell



http://memeshappen.com/meme/oh-no-meme/did-i-really-write-this-code-35113/68

Code organization

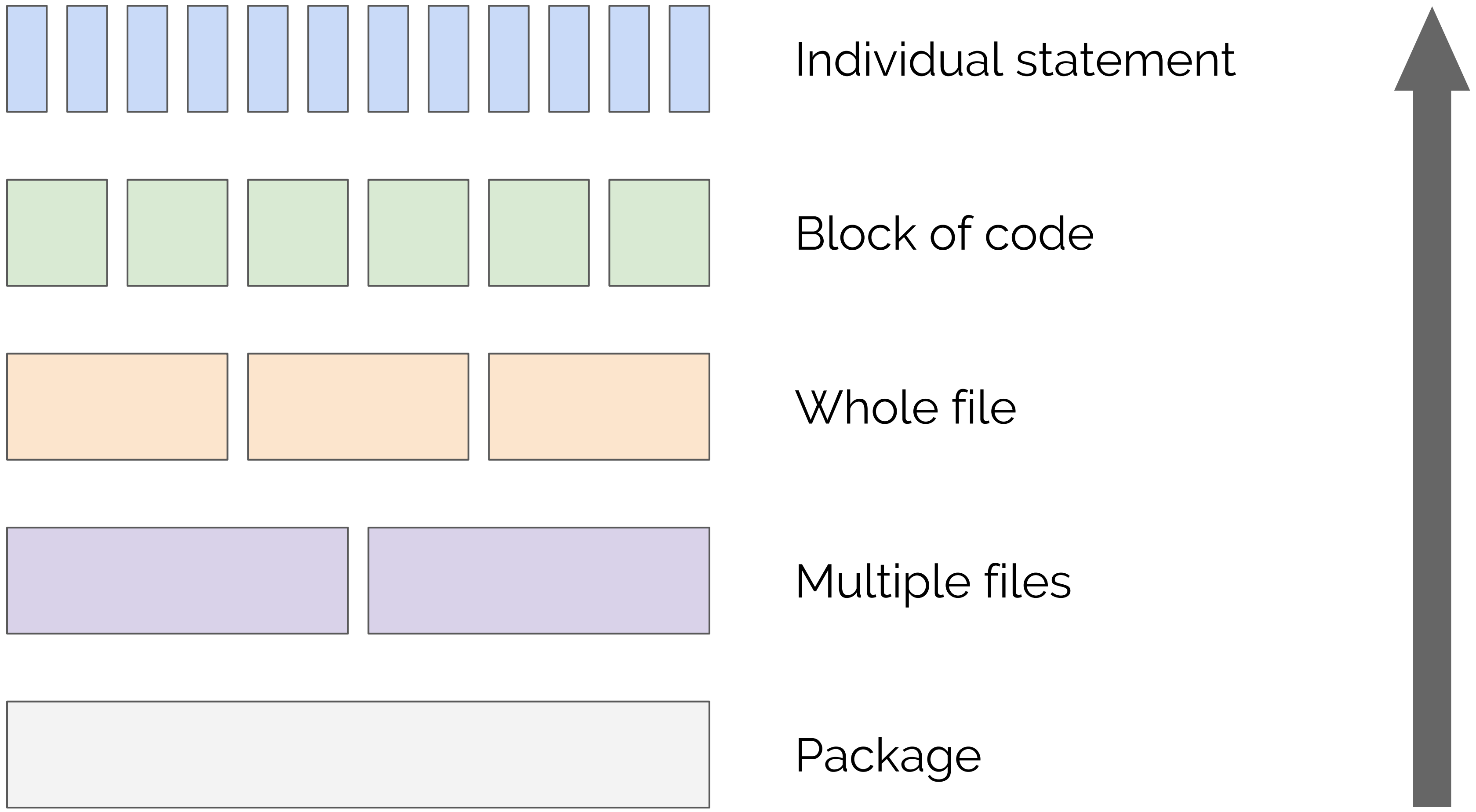
Used in Homework 4 out next week

Why care?

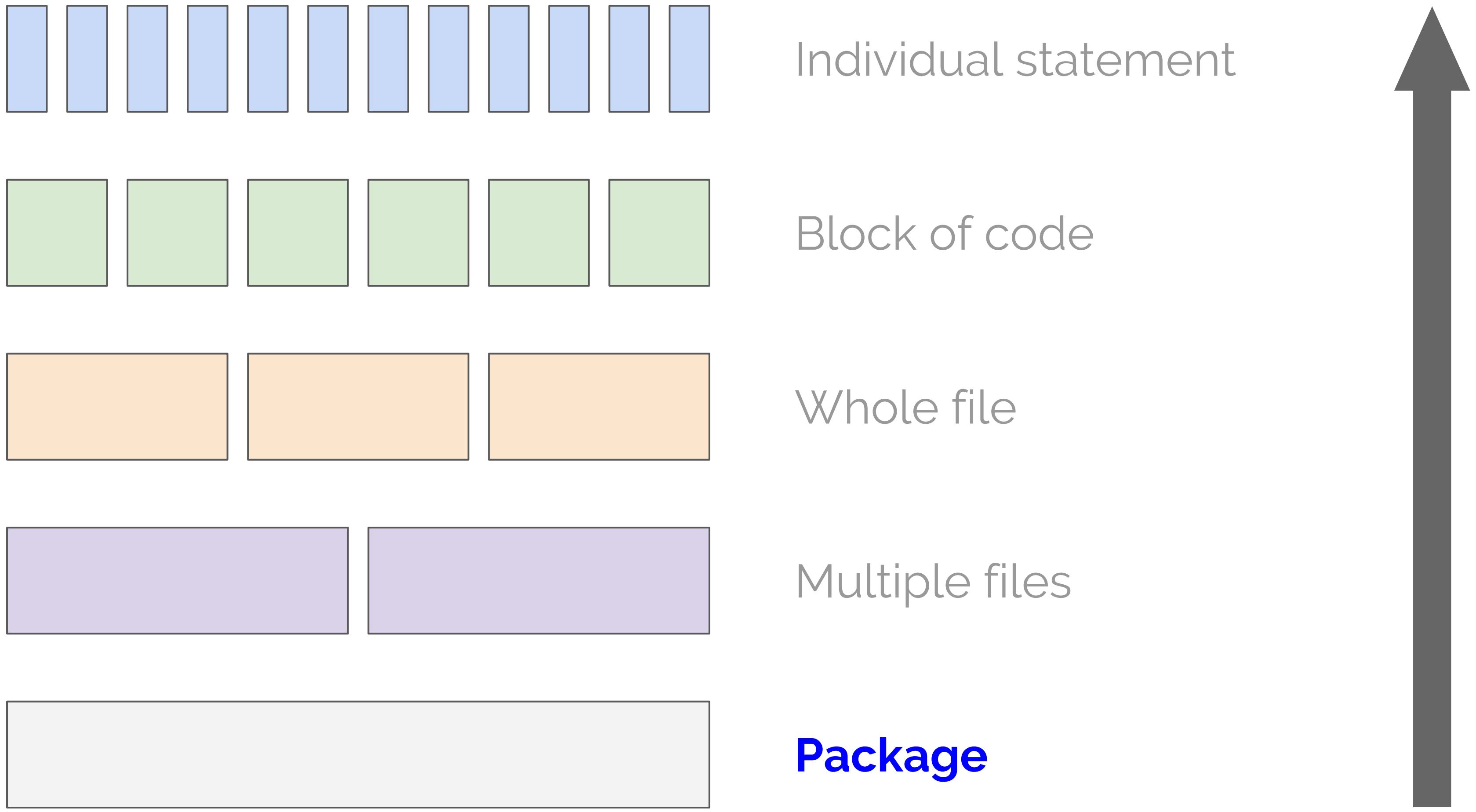
Easier to:

- Understand
- Maintain
- Change

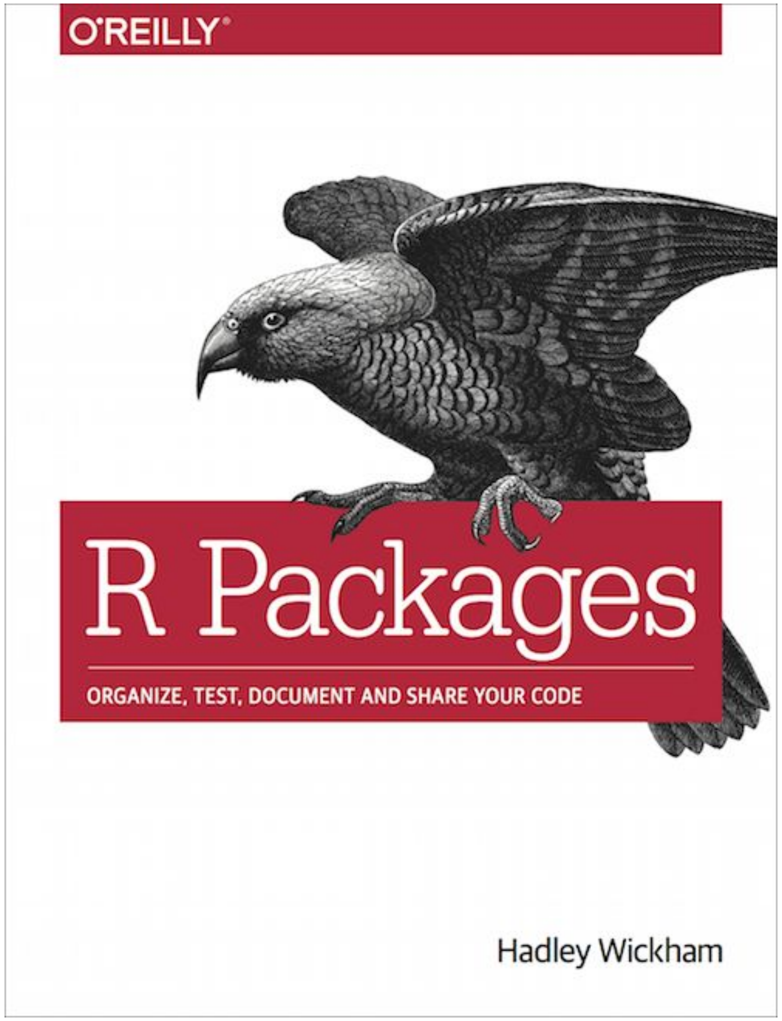
Organization at different scales



Organization at different scales

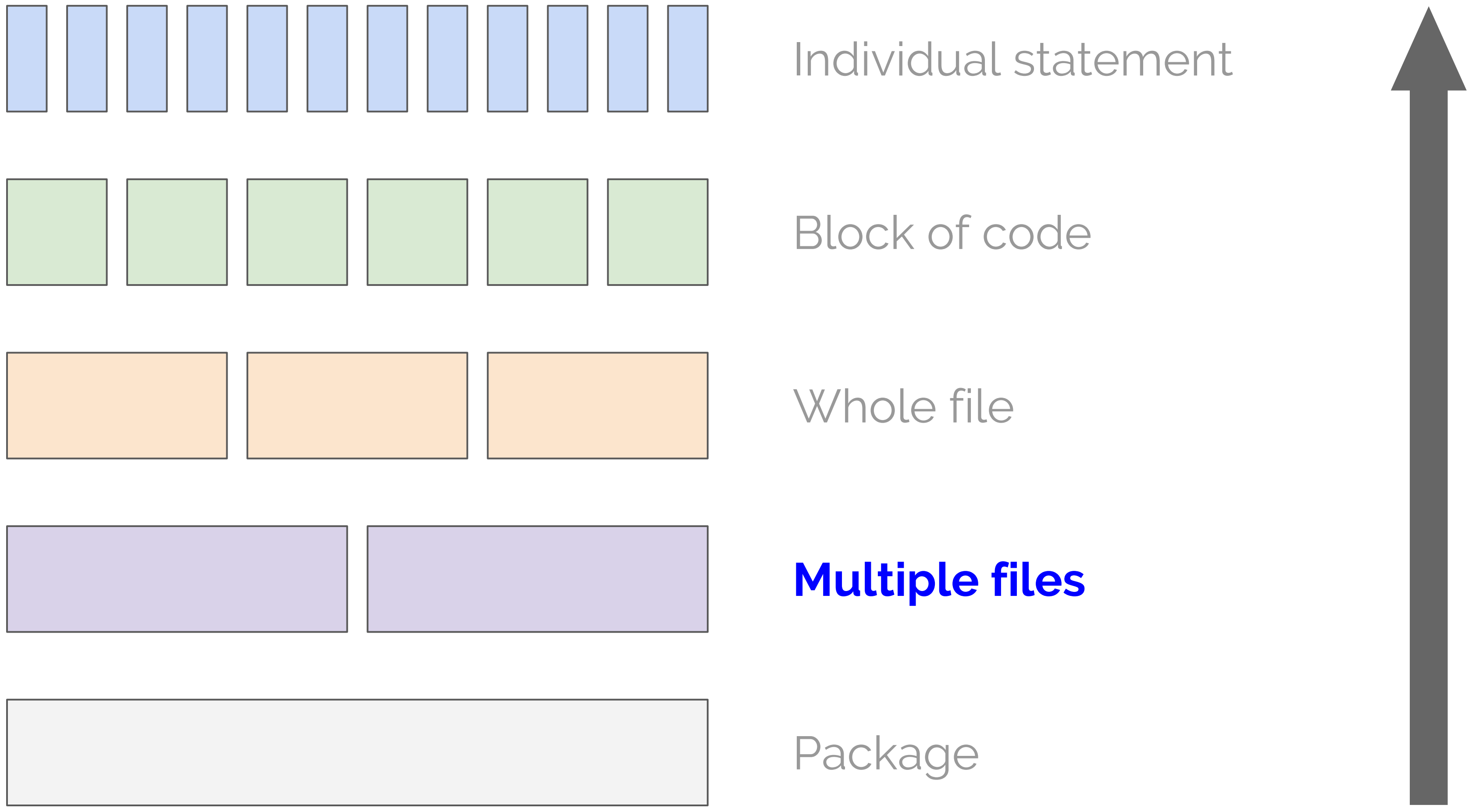


The most important unit of organization of R code... that we won't cover

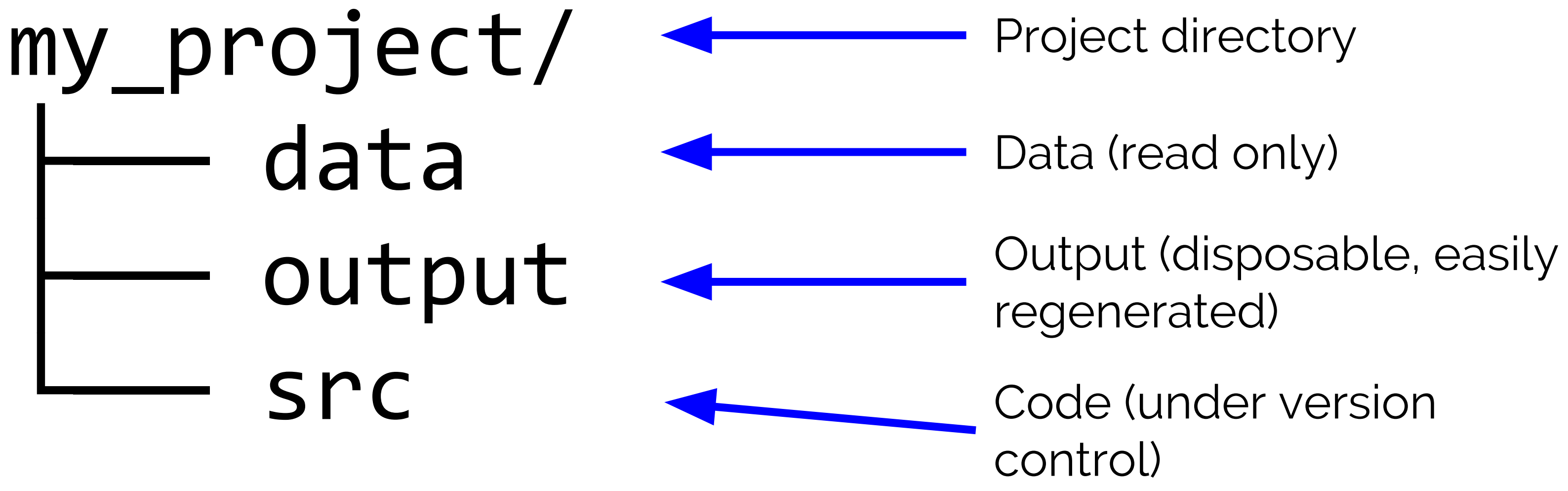


http://r-pkgs.had.co.nz/

Organization at different scales



Organizing an analysis project: directory structure


my_project/
data
output
src
Project directory
Data (read only)
Output (disposable, easily regenerated)
Code (under version control)

Create an Rstudio project associated with the src directory

Rstudio projects let you maintain different contexts for different projects (working directory, workspace, history, files)

Advantages:

- No need to set working directory
- Remembers which files you had open
- Optionally save workspace (environment)

In Rstudio:
`File -> New Project…`
Creates an .Rproj file that you open each time you work on the project

Organizing the src directory: options

```
my_project/
├── data
├── output
└── src
    ├── helper_functions.R
    └── workflow_script.R
```

All functions in one R file

One R or Rmd file loads the file of functions and runs the workflow

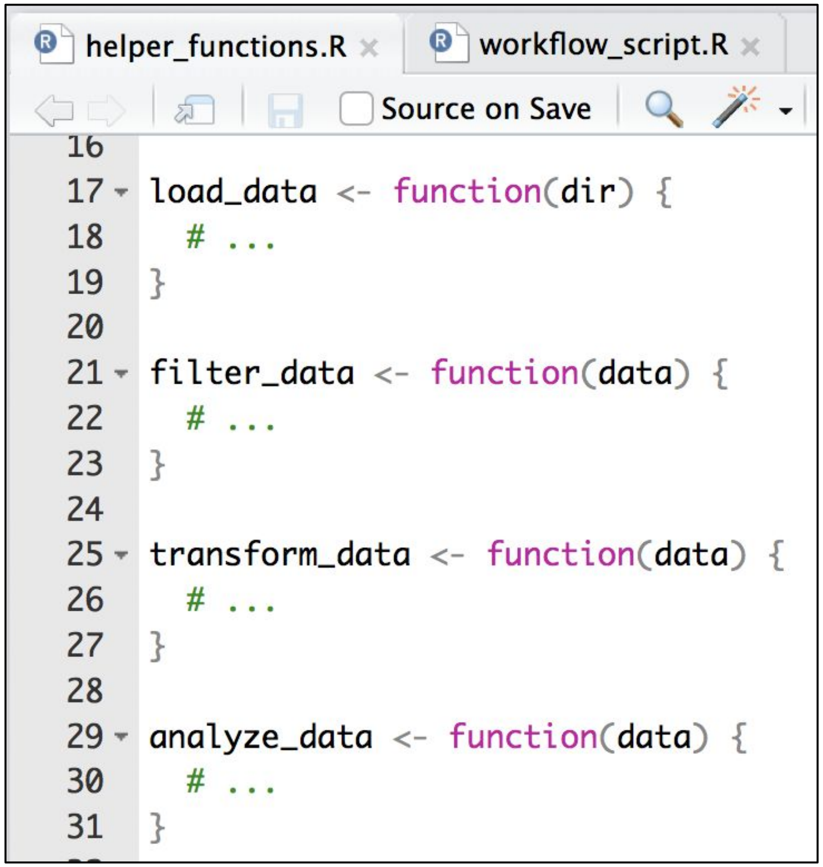
---

```
my_project/
├── data
├── output
└── src
    ├── func_analysis.R
    ├── func_data_processing.R
    ├── func_qc.R
    └── workflow.Rmd
```

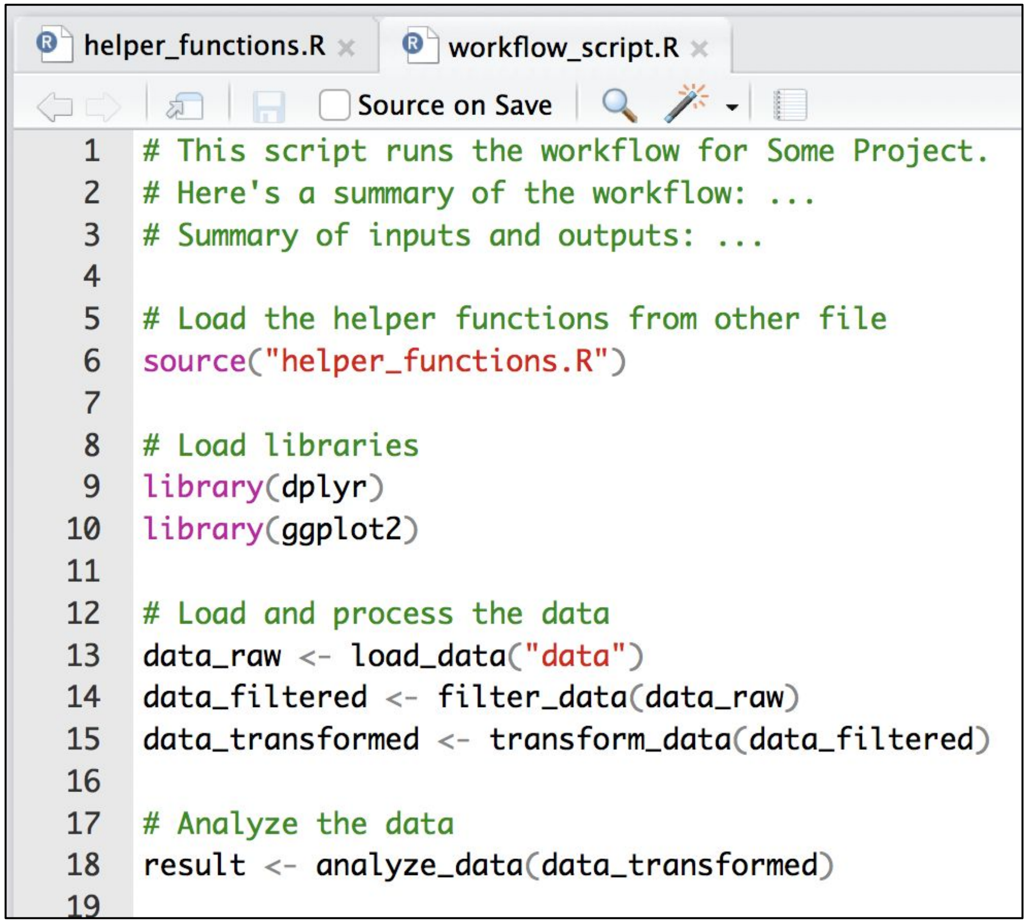
If you have lots of functions, organize them into multiple R files

R or Rmd file loads functions and runs workflow

Functions in one file, workflow in another

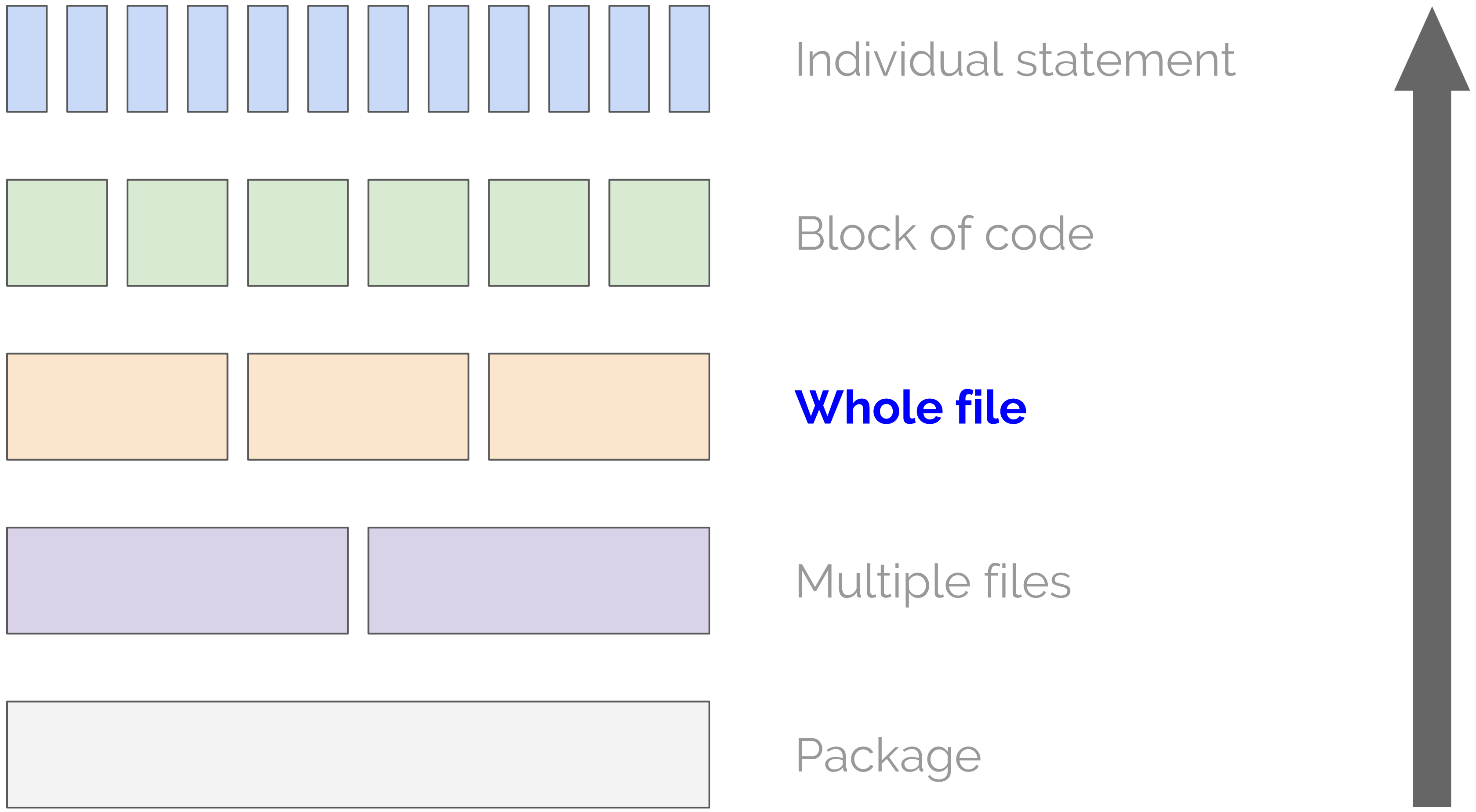


```r

load_data <- function(dir) {
  # ...
}

filter_data <- function(data) {
  # ...
}

transform_data <- function(data) {
  # ...
}

analyze_data <- function(data) {
  # ...
}
```

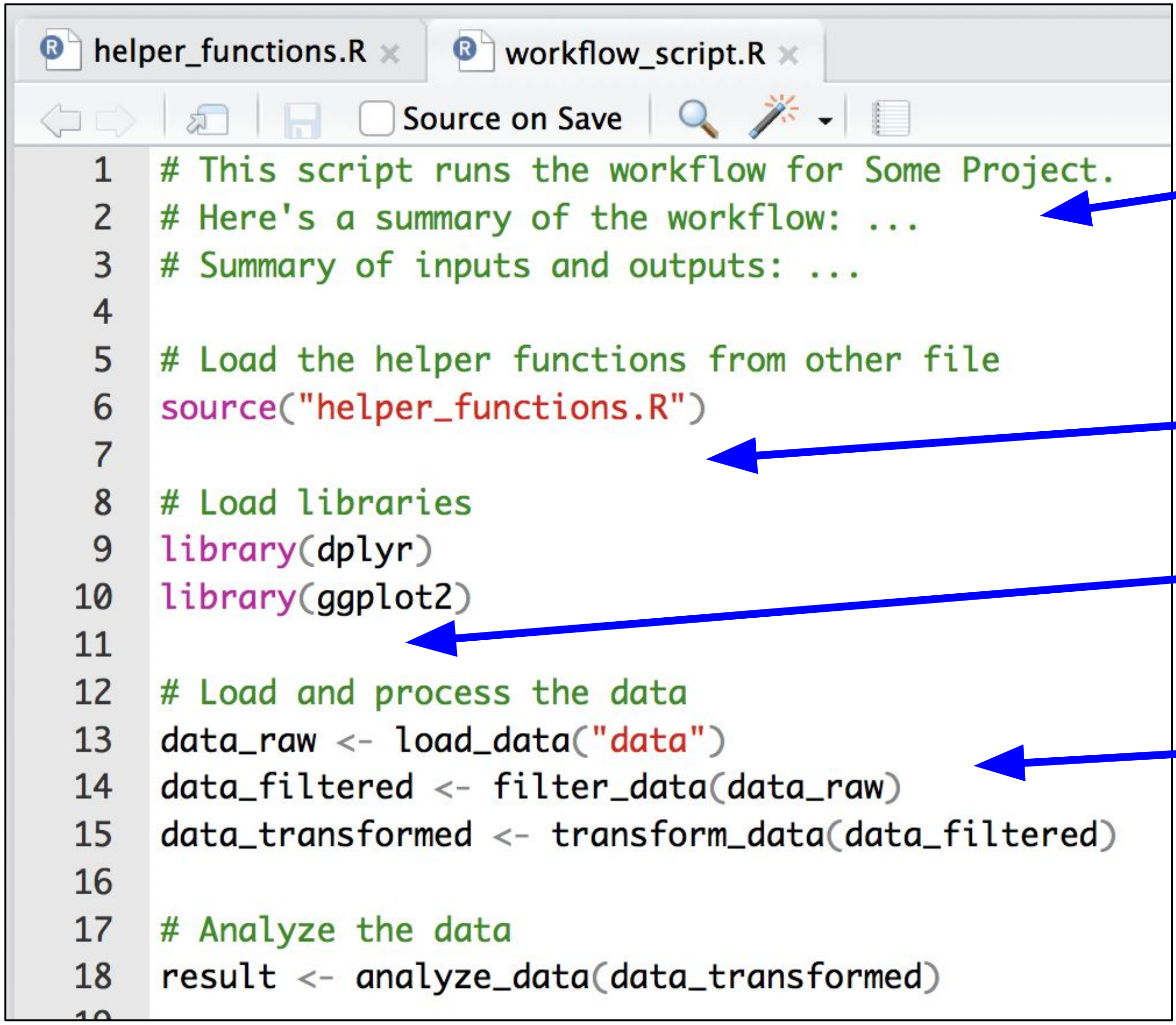


```r
# This script runs the workflow for Some Project.
# Here's a summary of the workflow: ...
# Summary of inputs and outputs: ...

# Load the helper functions from other file
source("helper_functions.R")

# Load libraries
library(dplyr)
library(ggplot2)

# Load and process the data
data_raw <- load_data("data")
data_filtered <- filter_data(data_raw)
data_transformed <- transform_data(data_filtered)

# Analyze the data
result <- analyze_data(data_transformed)

```

Organization at different scales



Script order



```r
# This script runs the workflow for Some Project.
# Here's a summary of the workflow: ...
# Summary of inputs and outputs: ...

# Load the helper functions from other file
source("helper_functions.R")

# Load libraries
library(dplyr)
library(ggplot2)

# Load and process the data
data_raw <- load_data("data")
data_filtered <- filter_data(data_raw)
data_transformed <- transform_data(data_filtered)

# Analyze the data
result <- analyze_data(data_transformed)
```

File description

`source` and `library` statements

Function definitions (none here)

Executed statements if applicable

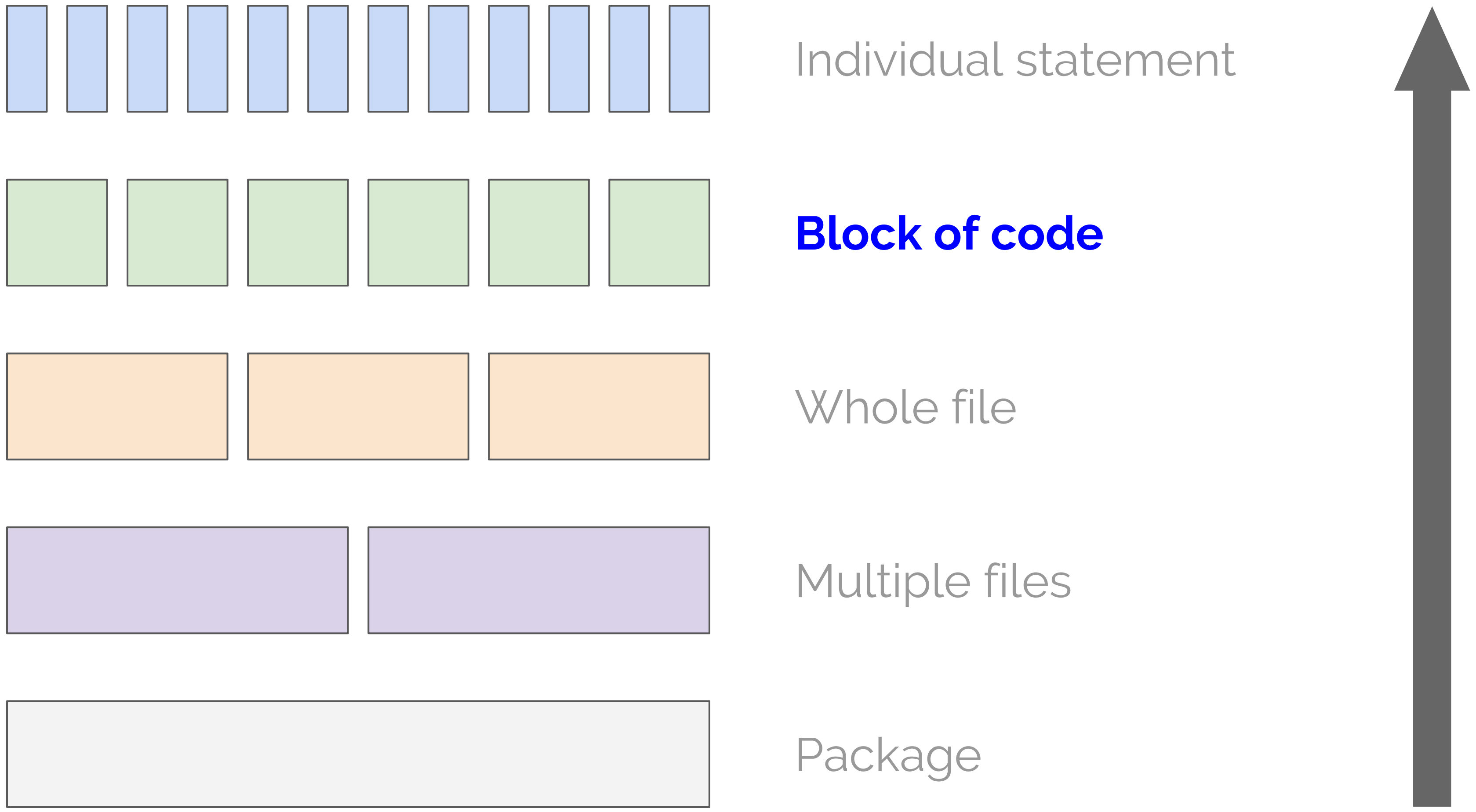
Based on Google's R Style Guide (https://google.github.io/styleguide/Rguide.xml)

Code grouping

Group tasks into separate blocks of code with explanatory comments

```
# This script runs the workflow for Some Project.
# Here's a summary of the workflow: ...
# Summary of inputs and outputs: ...

# Load the helper functions from other file
source("helper_functions.R")

# Load libraries
library(dplyr)
library(ggplot2)

# Load and process the data
data_raw <- load_data("data")
data_filtered <- filter_data(data_raw)
data_transformed <- transform_data(data_filtered)

# Analyze the data
result <- analyze_data(data_transformed)
```

Organization at different scales



Functions: organization and code reuse

Organize
Each function is one unit of action

Encapsulate
Hide details when they're not relevant

DRY
Don't repeat yourself

Function documentation

A multi-line comment inside the function body

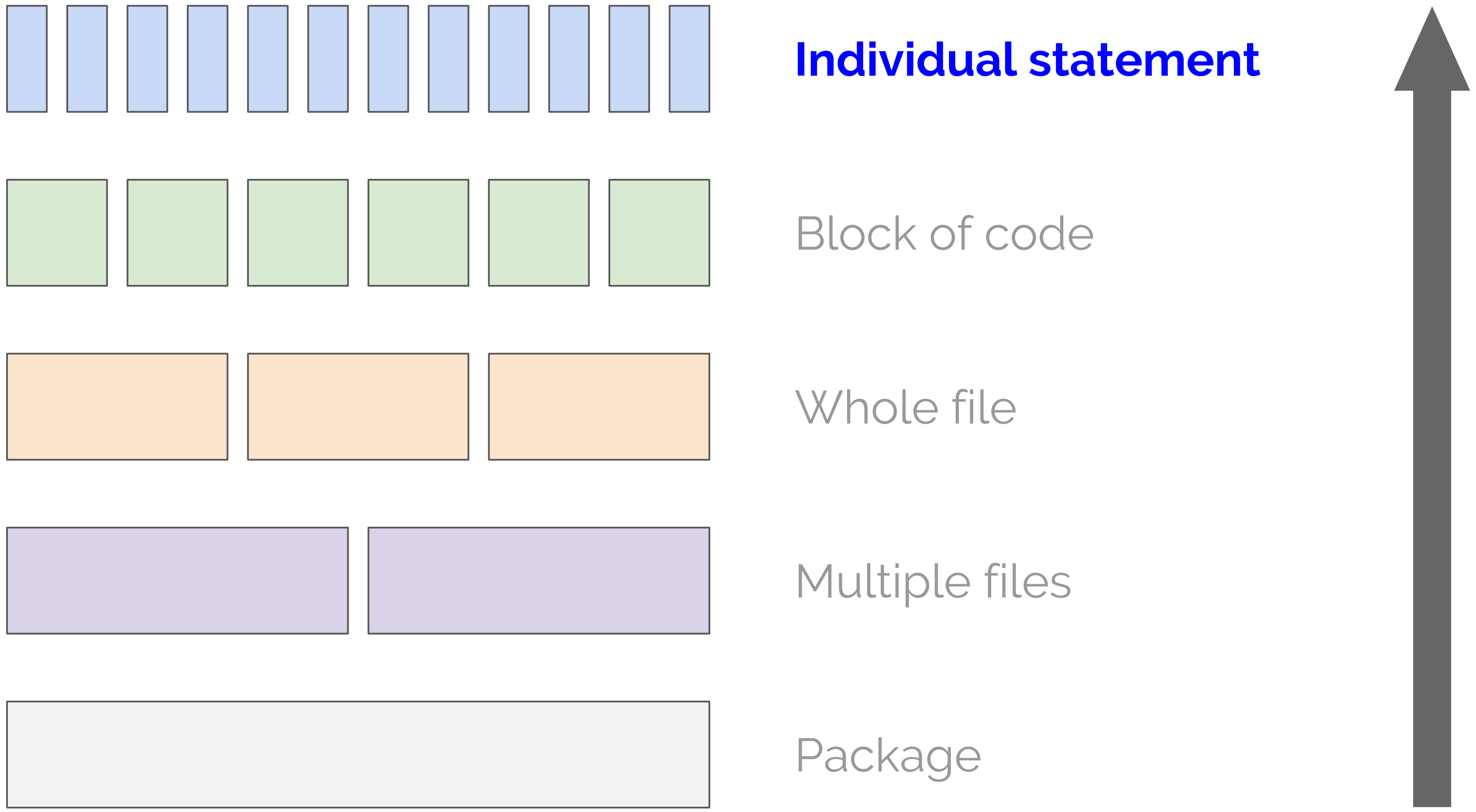
What the function does (verb phrase)

Arguments

Return value

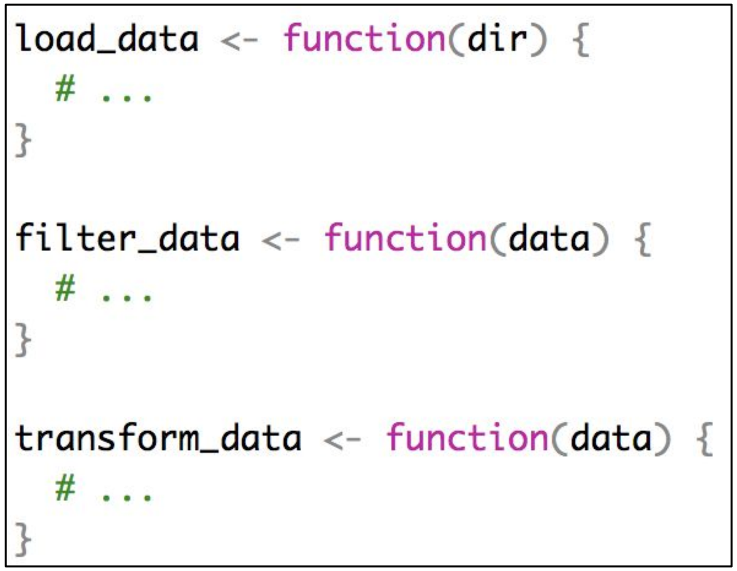
```r
load_data <- function(dir, another_arg) {
  # Loads the data in a directory.
  #
  # Args:
  #   dir: Directory path
  #   another_arg: Another argument
  #
  # Returns:
  #   A data frame containing the data

  do_something()
  return_something
}
```

Organization at different scales

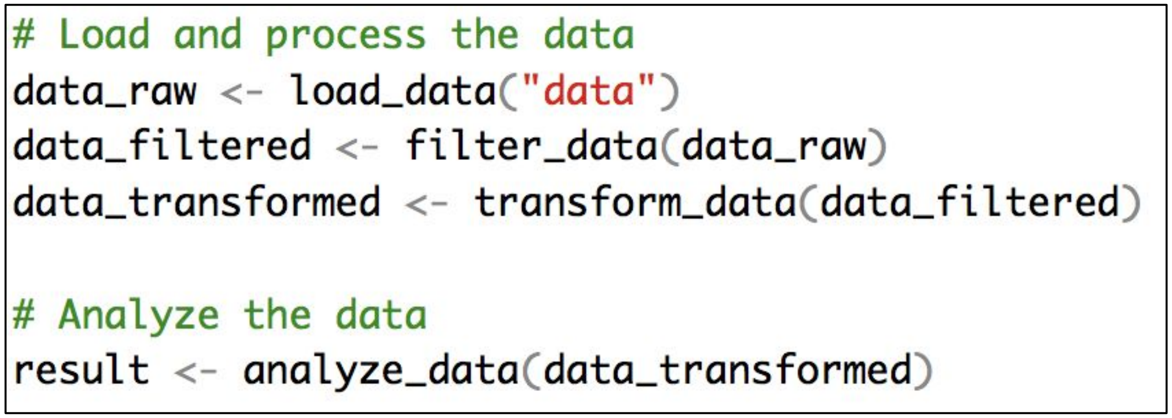


Object naming

- Concise and meaningful
- Lowercase
- Separate words with underscores
- Functions are verbs
- Variables are nouns



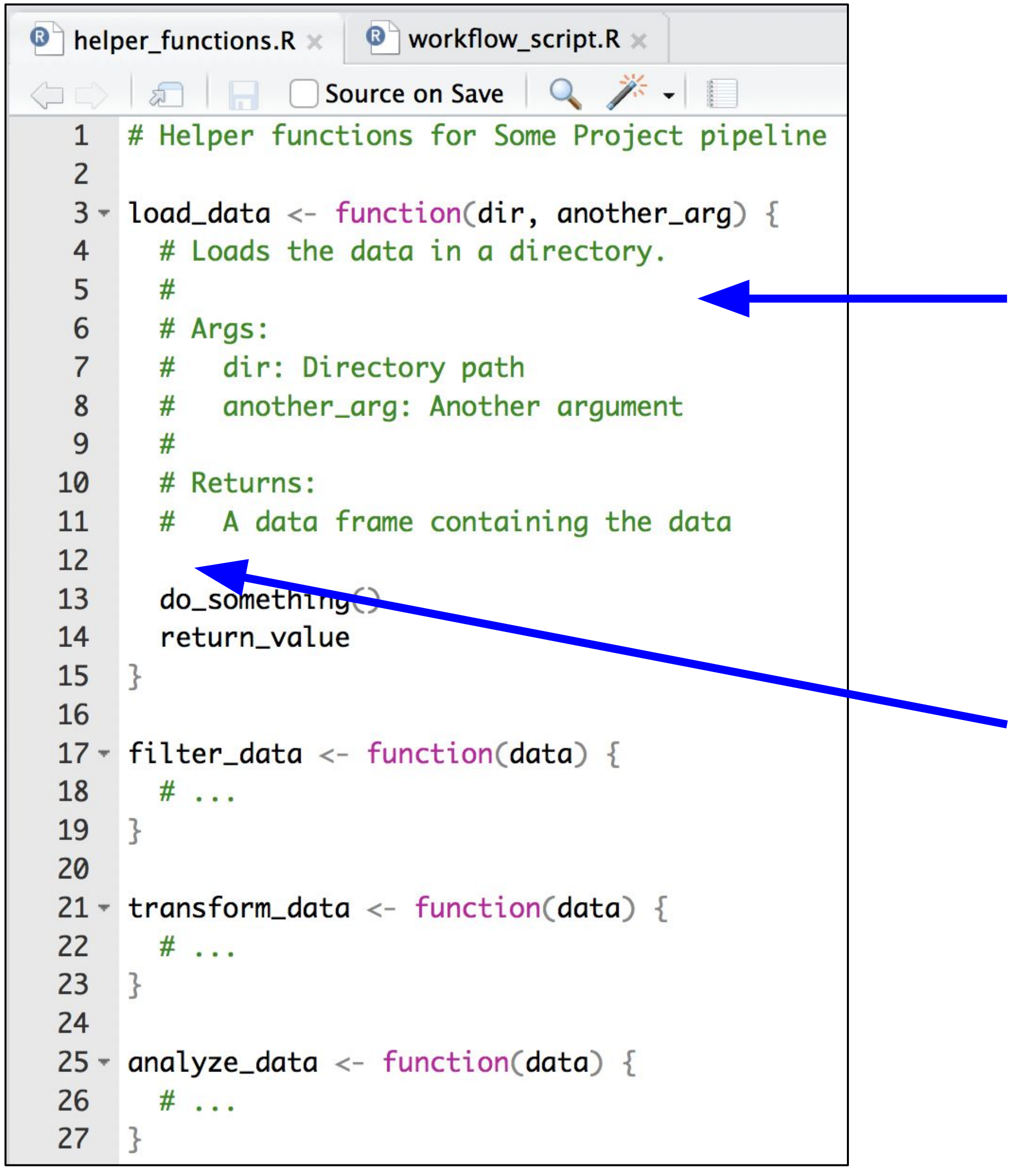
```
load_data <- function(dir) {
  # ...
}

filter_data <- function(data) {
  # ...
}

transform_data <- function(data) {
  # ...
}
```



```
# Load and process the data
data_raw <- load_data("data")
data_filtered <- filter_data(data_raw)
data_transformed <- transform_data(data_filtered)

# Analyze the data
result <- analyze_data(data_transformed)
```

http://adv-r.had.co.nz/Style.html

Line style for readability



```
# Helper functions for Some Project pipeline

load_data <- function(dir, another_arg) {
  # Loads the data in a directory.
  #
  # Args:
  #   dir: Directory path
  #   another_arg: Another argument
  #
  # Returns:
  #   A data frame containing the data

  do_something()
  return_value
}

filter_data <- function(data) {
  # ...
}

transform_data <- function(data) {
  # ...
}

analyze_data <- function(data) {
  # ...
}
```

Lines no longer than 80 characters

Indentation is important for readability. Rstudio automatically indents for you.

To refresh automatic indentation in Rstudio: Code -> Reindent Lines

Next time

Code quality