Eco 5316 Time Series Econometrics

Pipes

Introduction

- pipe operator %>% comes from `magrittr` package, see cran.r-project.org/web/packages/magrittr/vignettes/magrittr.html

- https://en.wikipedia.org/wiki/The_Treachery_of_Images
- they help write code that is easier to read and understand
- in particular, pipes are useful for clearly expressing a sequence of operations
- see http://r4ds.had.co.nz/pipes.html for more details

Introduction

- a sequence of operations can be carried out in several ways
- you can
  1. save each intermediate step as a new object
  2. overwrite the original object over and over
  3. compose functions
  4. use the pipe

Example

- how to get a Ph.D. in 5 easy steps
- consider functions
  - `take_courses()`
  - `pass_comps()`
  - `write_papers()`
  - `defend()`
  - `celebrate()`

Save Intermediate Steps

- how to get a Ph.D.

```
you1 <- take_courses(you, n = 20)
you2 <- pass_comps(you1, max.attempts = 2)
you3 <- write_papers(you2, n = 3)
you4 <- defend(you3, committee = c("the good","the bad","the ugly"))
you5 <- celebrate(you4, how.long = "until 3am")
```

- downside: forces you to name intermediate elements
- good idea if there are natural names and we want to inspect and further use the intermediate elements
- often there aren't natural names and code becomes cluttered with unintuitive names

Overwrite the Original

- how to get a Ph.D.

```
you <- take_courses(you, n = 20)
you <- pass_comps(you, max.attempts = 2)
you <- write_papers(you, n = 3)
you <- defend(you, committee = c("the good","the bad","the ugly"))
you <- celebrate(you, how.long = "until 3am")
```

- downsides: if you make a mistake you'll need to re-run the sequence from the beginning, results for intermediate steps are not available

Function Composition

- how to get a Ph.D.

```
celebrate(
    defend(
        write_papers(
            pass_comps(
                take_courses(you, n = 20),
                max.attempts = 2),
            n = 3),
        committee = c("the good","the bad","the ugly")),
    how.long = "until 3am")
```

- big disadvantage: hard to see whats going on since
  1. you have to read from inside-out,
  2. function arguments are spread far apart from the function itself

Use the Pipe

- how to get a Ph.D.

```
you %>%
    take_courses(n = 20) %>%
    pass_comps(max.attempts = 2) %>%
    write_papers(n = 3) %>%
    defend(committee = c("the good","the bad","the ugly")) %>%
    celebrate(how.long = "until 3am")
```

- big advantage: easy to follow and understand

When Not to Use the Pipe

- pipes are powerful but don't solve every problem
- they are great for rewriting a reasonably short linear sequence of operations
- Hadley Wickham's recommendation: you should reconsider using pipes if
  - your pipes are longer than (say) ten steps - create some intermediate objects with meaningful names (this will make debugging easier and also make it easier to understand code)
  - if there isn't one primary object being transformed and instead you have multiple inputs or outputs
  - if you have a complex dependency structure - pipes are fundamentally linear, expressing complex relationships with them will yield confusing code

Other Tools From magrittr

- %$% is useful for functions that don't take a data frame but rather individual vectors as arguments

```
library(magrittr)
mtcars %$%
    cor(disp, mpg)
```

```
## [1] -0.8475514
```

- %<>% operator which allows you to replace code like

```
mtcars <- mtcars %>%
    transform(cyl = cyl * 2)
```

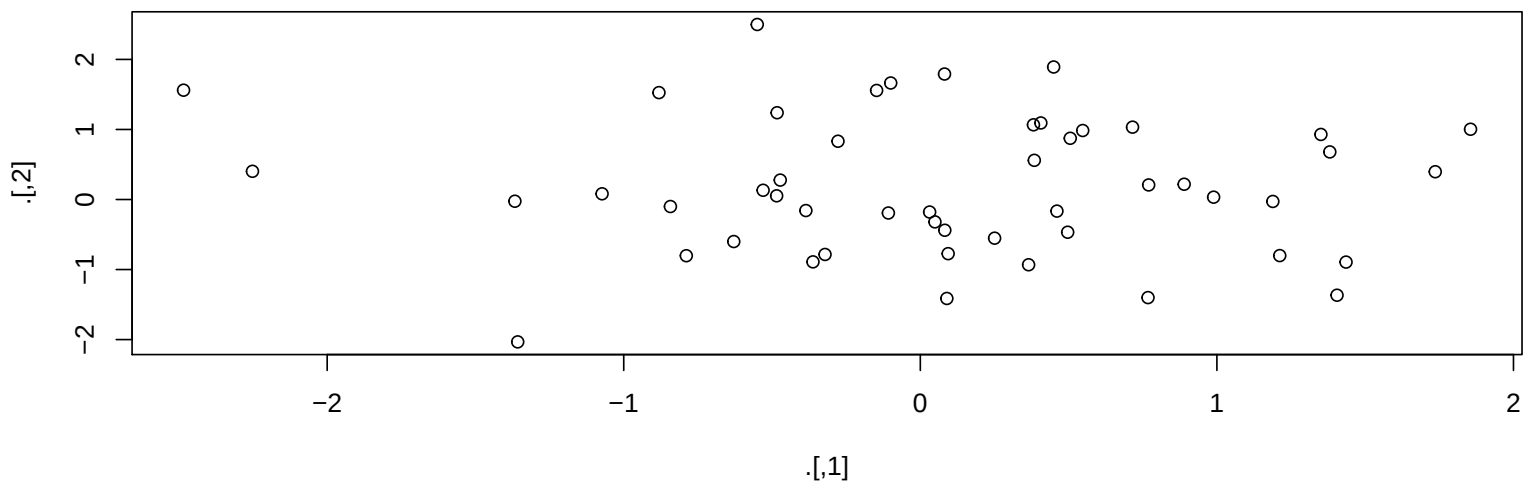
with

```
mtcars %<>% transform(cyl = cyl * 2)
```

Other Tools From magrittr

- %T>% works like %>% except that it returns the left-hand side instead of the right-hand side

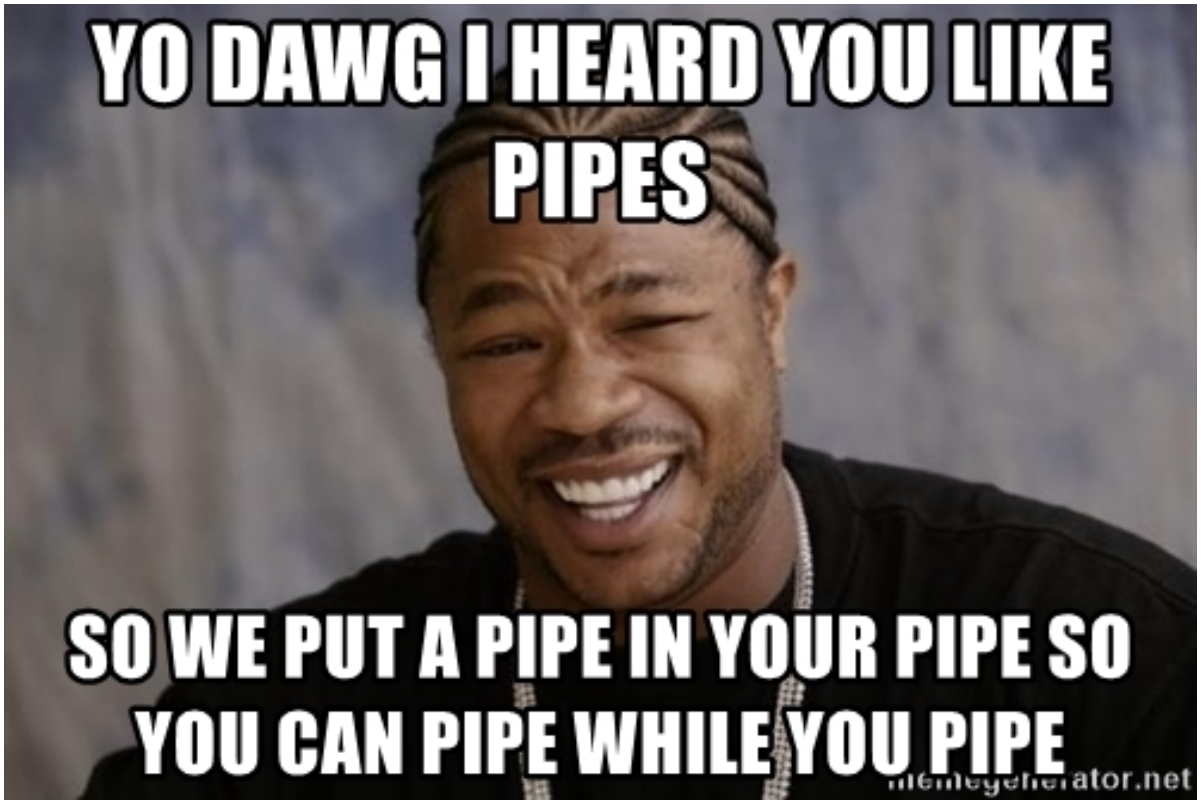
```
rnorm(100) %>%
    matrix(ncol = 2) %T>%
    plot() %>%
    str()
```




```
##  num [1:50, 1:2] -0.629 1.8553 -0.5301 0.4496 0.0816 ...
```

Nested Pipes

- note: you can nest pipes



Nested Pipes

```
# estimate rolling ARIMA model, create 1 period ahead rolling forecasts
results <-
    data.tbl %>%
    as_tbl_time(index = date) %>%
    mutate(arma.model = roll_Arima(PAYEMS)) %>%
    filter(!is.na(arma.model)) %>%
    mutate(arma.coefs = map(arma.model, tidy, conf.int = TRUE),
           arma.f = map(arma.model, (. %>% forecast(1) %>% sw_sweep())))
```