Good Coding Practice

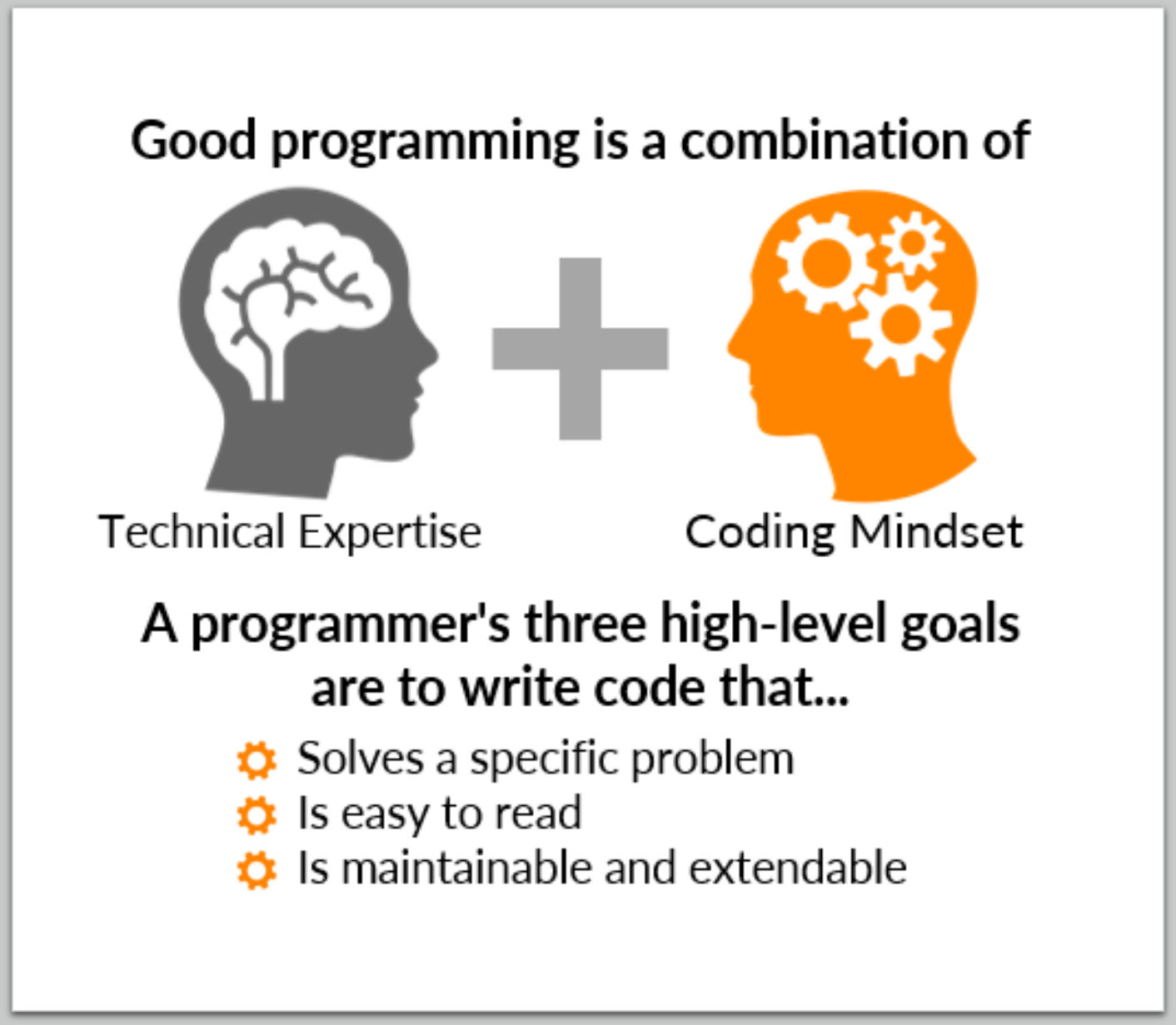
Introduction to Data Science for Health and Social Care

November 9th

Dr Holly Tibble

Fundamentals of good coding

- Commenting
- Folder structure
- File names
- Accessibility (Github)



Quality Assurance of Code for Analysis and Research

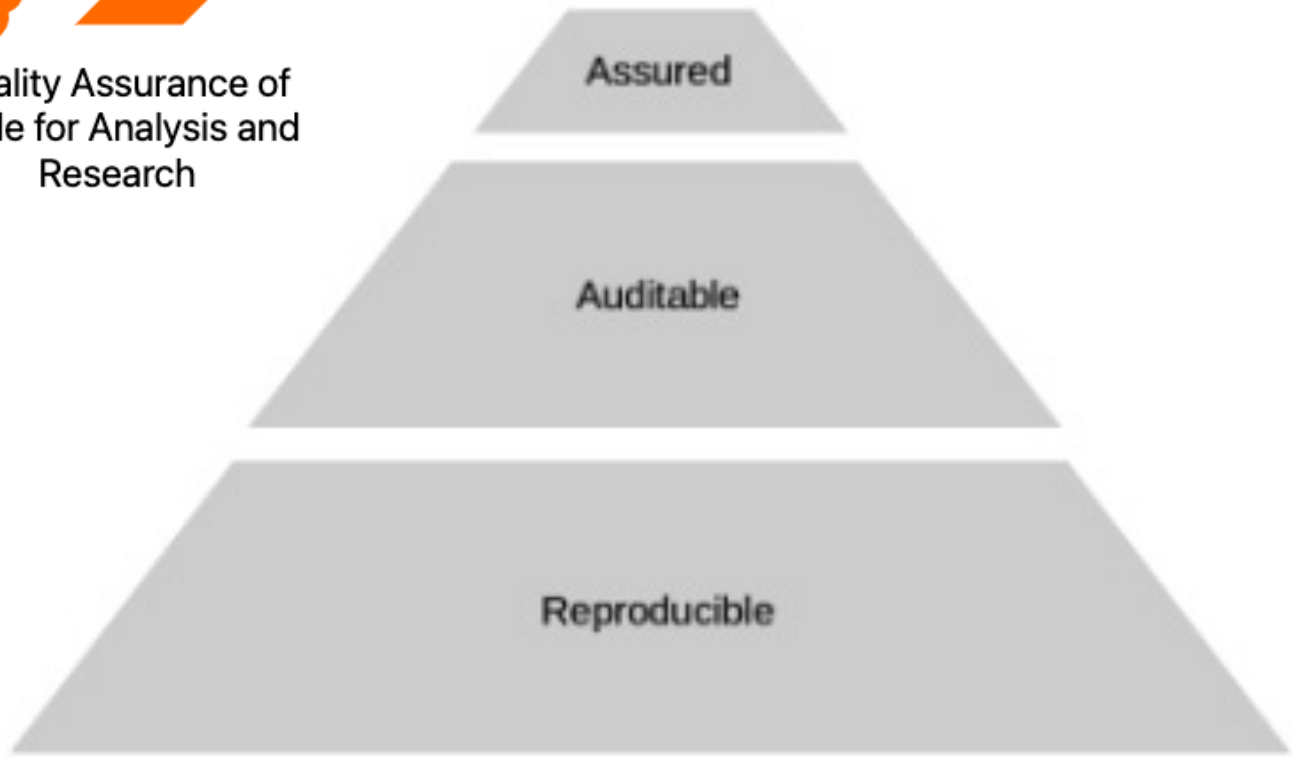
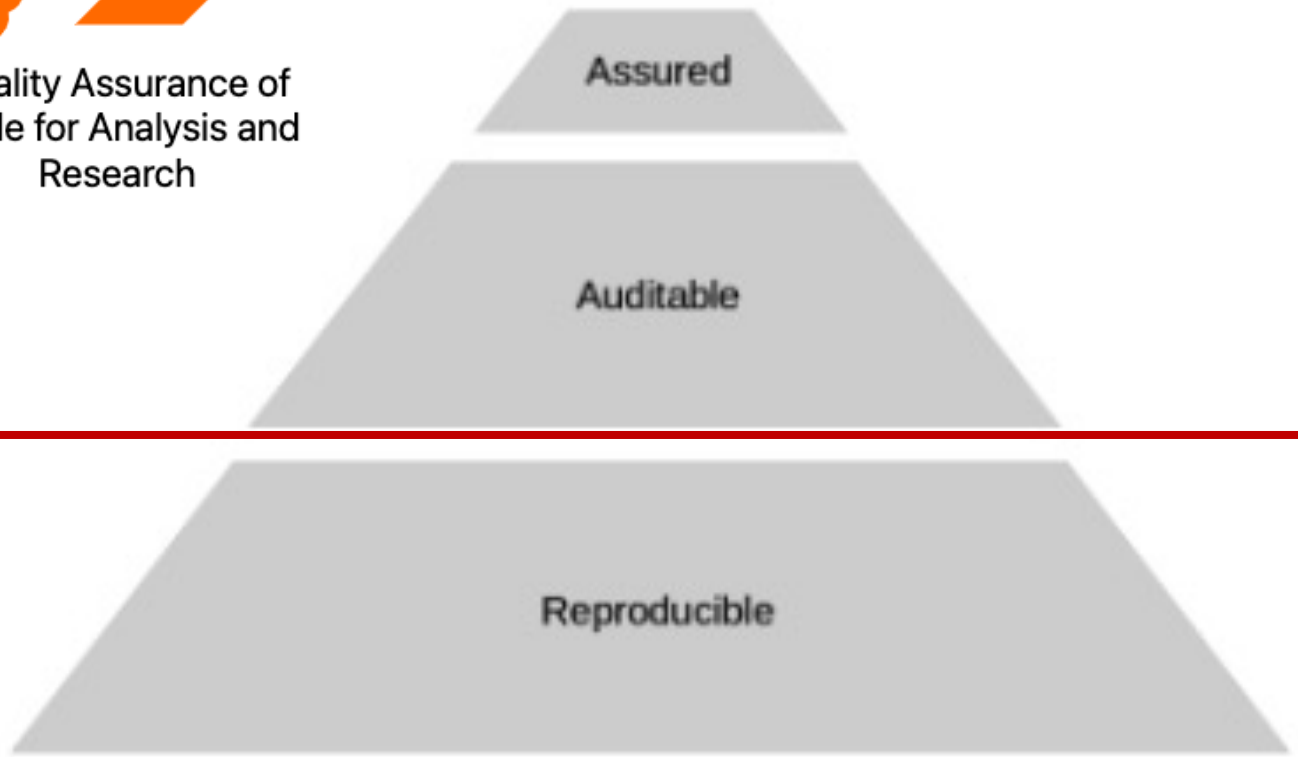


Fig. 1 Founding principles of good analysis

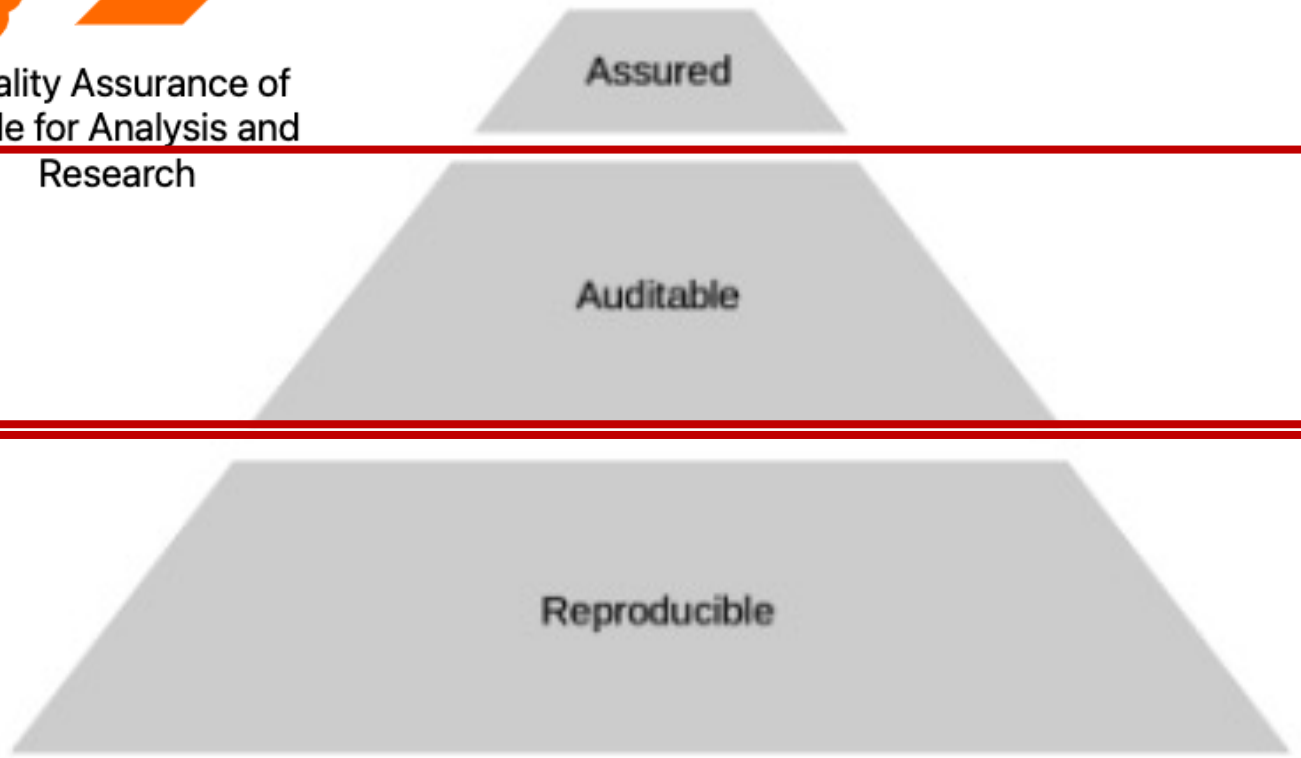
Quality Assurance of Code for Analysis and Research



- Publicly available code and data
- Runs without error
- No random element without seed
- Portable (machines and versions)

Fig. 1 Founding principles of good analysis

Quality Assurance of Code for Analysis and Research

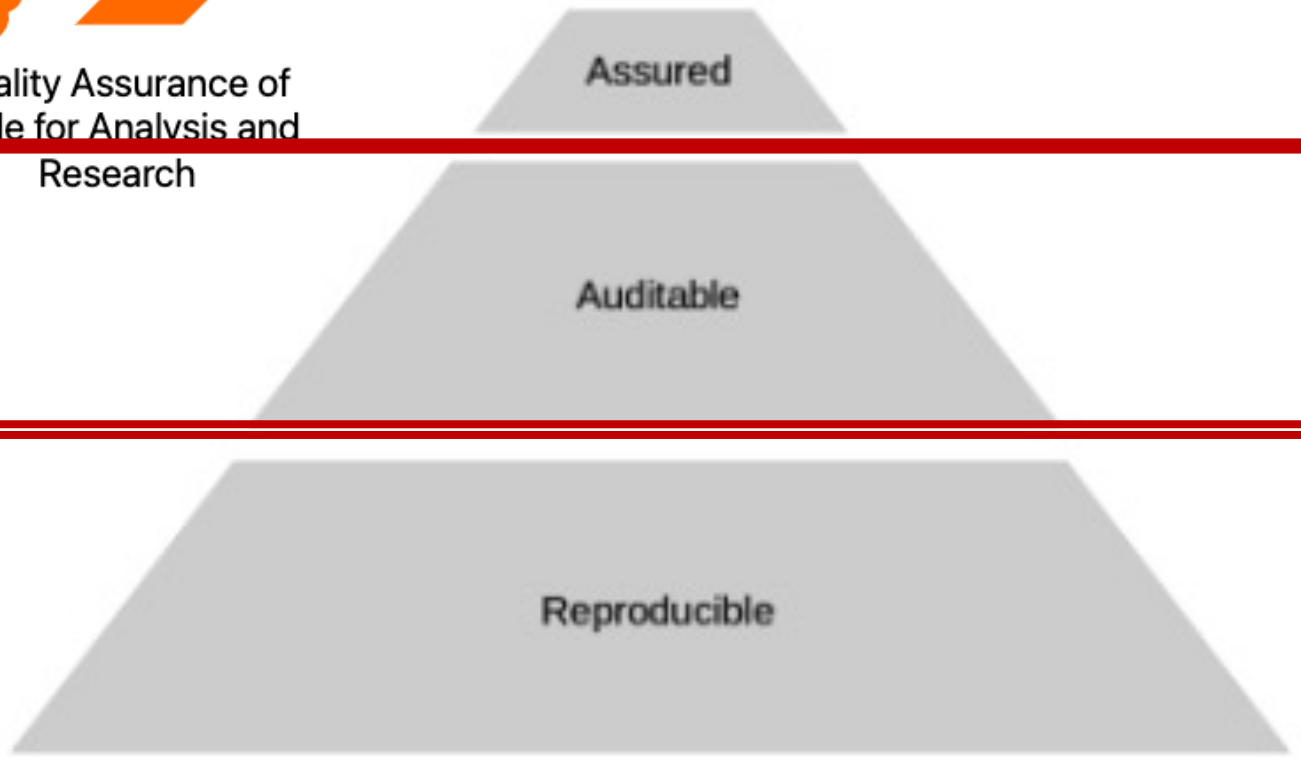


- Owner named, with contact details
- Evidence-based (cited) decisions

- Publicly available code and data
- Runs without error
- No random element without seed
- Portable (machines and versions)

Fig. 1 Founding principles of good analysis

Quality Assurance of Code for Analysis and Research



- Quality Assured

- Owner named, with contact details
- Evidence-based (cited) decisions

- Publicly available code and data
- Runs without error
- No random element without seed
- Portable (machines and versions)

Fig. 1 Founding principles of good analysis

Code Review Best Practices

Method 1

Total rewrite of code from initial protocol

Paired review of differences

Method 1

Total rewrite of code from initial protocol

Paired review of differences

Method 2

antagonist review – try to create data that will break the code

(important for code that will be used routinely)

A software tester walks into a bar.

Runs into a bar.

Crawls into a bar.

Dances into a bar.

Flies into a bar.

Jumps into a bar.

And orders:

a beer.

2 beers.

0 beers.

99999999 beers.

a lizard in a beer glass.

-1 beer.

“qwertyuiop” beers.

Testing complete.

A real customer walks into the bar and asks where the bathroom is.

The bar goes up in flames.

Method 1

Total rewrite of code from initial protocol

Paired review of differences

Method 2

antagonist review – try to create data that will break the code

(important for code that will be used routinely)

Method 3

review and comment on code

Questions to ask when peer-reviewing code

- What does this code do?
- Does the code function as I expect it to?
- Does this code fulfil regulatory requirements?
- How can this code be made more:
  - understandable
  - reliable
  - future-proof
  - efficient
  - reusable

Questions to ask when peer-reviewing code

- What does this code do?
- Does the code function as I expect it to?
- Does this code fulfil regulatory requirements?
- How can this code be made more:
  - understandable
  - reliable
  - future-proof
  - efficient
  - reusable

Set a time or line-of-code limit per session, to prevent reviewing with tired eyes

Code Indenting

```
Group_A<- data %>% filter(group == ”A”) %>%
select(name, Height, Weight, Age)
```

Code Indenting

```
Group_A<- data %>% filter(group == ”A”) %>%
select(name, Height, Weight, Age)
```

```
Group_A<- data %>%
        filter(group == ”A”) %>%
        select(name,
            Height,
            Weight,
            Age)
```

Overwriting vs. generating R objects

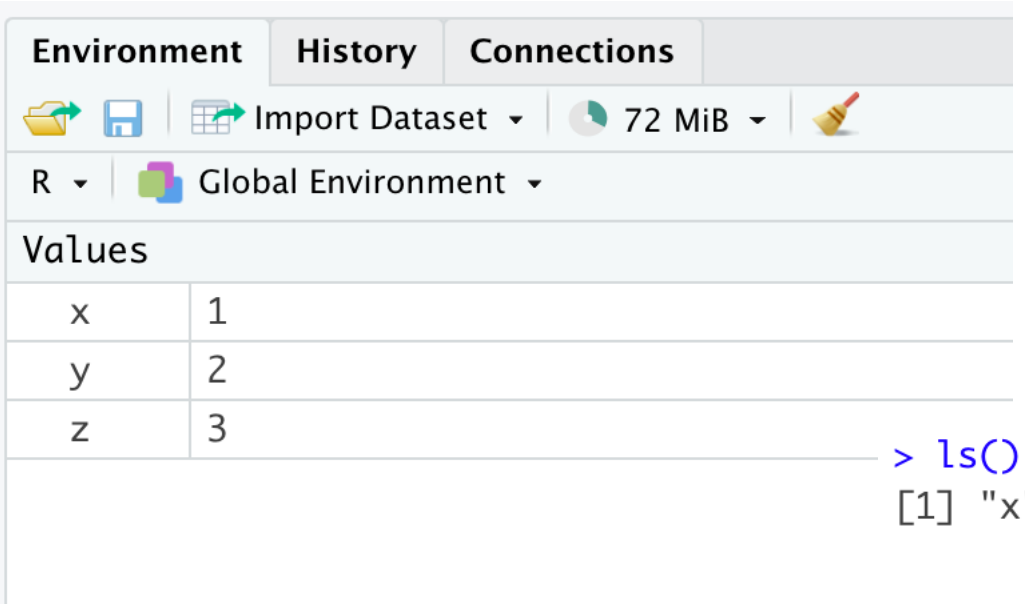
If I have made a mistake I have to rerun 540 lines to get back to before this line

Some of my actual R code!

```
540 ### how long has it been since your last prescription of each stage?
541 prescribing_asthma<-prescribing_asthma %>%
542   group_by(ID) %>%
543   arrange(ID,PrescDate) %>%
544   mutate(last_ICS=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class=="ICS",
545                                                                          PrescDate,NA)),na.rm=F)),9999),
546          last_combo=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class=="ICS+LABA",
547                                                                            PrescDate,NA)),na.rm=F)),9999),
548          last_ICS_low=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class %in% c("ICS","ICS+LABA") &
549                                                                               dose_category=="low",
550                                                                              PrescDate,NA)),na.rm=F)),9999),
551          last_ICS_medium=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class %in% c("ICS","ICS+LABA") &
552                                                                               dose_category=="medium",
553                                                                              PrescDate,NA)),na.rm=F)),9999),
554          last_ICS_high=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class %in% c("ICS","ICS+LABA") &
555                                                                               dose_category=="high",
556                                                                              PrescDate,NA)),na.rm=F)),9999),
557          last_ICS_dose=ifelse(last_ICS_low<last_ICS_high & last_ICS_low<last_ICS_medium,"low",
558                               ifelse(last_ICS_medium<last_ICS_high,"medium","high")),
559          last_LTRA=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class=="LTRA",
560                                                                           PrescDate,NA)),na.rm=F)),9999),
561          last_LABA=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class=="LABA",
562                                                                            PrescDate,NA)),na.rm=F)),9999),
563          last_LAMA=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class=="LAMA",
564                                                                           PrescDate,NA)),na.rm=F)),9999),
565          last_Theo=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class=="Theophylline",
566                                                                            PrescDate,NA)),na.rm=F)),9999),
567          last_ICS_SOL=replace_na(as.numeric(PrescDate-na.locf(as.Date(ifelse(drug_class=="ICS_SOL",
568                                                                              PrescDate,NA)),na.rm=F)),9999))
569
570
```

Clearing objects from your workspace

```
> x<-1
> y<-2
> z<-3
```




```
> ls()
[1] "x" "y" "z"
```

rm(x,y)

rm(list=setdiff(ls(),c(“x”,”y”)))

Practicing identifying and fixing coding errors

Short code
10 errors
- Identify
- Explain
- Fix

Survey!