SheafCanon_OF

Olivia Freides

4/06/2022

Modeling Robinson's SheafCanon Sheaf:

For specifics and citations, reference https://arxiv.org/abs/1603.01446

Robinson, Michael. "Sheaves Are the Canonical Data Structure for Sensor Integration." Information Fusion, vol. 36, Elsevier B.V, 2017, pp. 208–24, https://doi.org/10.1016/j.inffus.2016.12.002.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
Table1 <- read.csv("Assignment2.csv") # Copied Table 1, page 218.
Table1
```

```
##         Sensor Key entity     Case1     Case2     Case3  Units
## 1  Flight plan  U1      x    70.662    70.663    70.612      W
## 2  Flight plan  U1      y    42.829    42.752    42.834      N
## 3  Flight plan  U1      z 11178.000 11299.000 11237.000      m
## 4          ATC  U2      x    70.587    70.657    70.617      W
## 5          ATC  U2      y    42.741    42.773    42.834      N
## 6          ATC  U2      z 11346.000 11346.000 11236.000      m
## 7          ATC  U2    v_x  -495.000  -495.000  -419.000 km/h W
## 8          ATC  U2    v_y   164.000   164.000   310.000 km/h N
## 9        RDF 1  U3 Theta1    77.100    77.200    77.200 true N
## 10       RDF 1  U3      t     0.943     0.930     0.985      h
## 11       RDF 2  U4 Theta2    61.300    63.200    63.300 true N
## 12       RDF 2  U4      t     0.890     0.974     1.050      h
## 13         Sat  U5      s        NA        NA        NA
## 14         Sat  U5    s_x    64.599    64.630    62.742      W
## 15         Sat  U5    s_y    44.243    44.287    44.550      N
```

```
## 16       Field    X       x     70.649     70.668     70.626      W
## 17       Field    X       y     42.753     42.809     42.814      N
## 18       Field    X       z 11220.000 11431.000 11239.000      m
## 19       Field    X     v_x   -495.000   -495.000   -419.000 km/h W
## 20       Field    X     v_y    164.000    164.000    311.000 km/h N
## 21       Field    X       t      0.928      1.050      1.020      h
```

```r
#Should these be global variables?
r_1x <- -73.662574
r_1y <- 42.733838
r_2x <- -77.0897
r_2y <- 38.935
```

What should be done with variables outside of the assignment table that we need? Constraints for functions not in any assignment table…

Restriction Functions:

Page 214: s__x, s__y are coordinates of an object detected in the satellite image, r__1x, r__1y are coordinates of the first RDF sensor and r__2x, r__2y are coordinates of the second RDF sensor.

$$A(x, y, z, v_x, v_y, t) = \left(tan^{-1} \frac{x + v_x t - r_{1x}}{y + v_y t - r_{1y}}, t \right)$$

```r
A <- function(stalk) {
  r_1x <- -73.662574
  r_1y <- 42.733838

  stalk %>%
    mutate(Theta1=atan2(x + v_x*t - r_1x, y + v_y*t - r_1y)) %>%
    select(Theta1, t)

}
```

$$B(x, y, z, v_x, v_y, t) = \left(tan^{-1} \frac{x + v_x t - r_{2x}}{y + v_y t - r_{2y}}, t \right)$$

```r
B <- function(stalk) {
  r_2x <- -77.0897
  r_2y <- 38.935

    stalk %>%
      mutate(Theta2=atan2(x + v_x*t - r_2x, y + v_y*t - r_2y)) %>%
      select(Theta2, t)
}
```

$$C(s_x, s_y) = tan^{-1} \frac{s_x - r_{1x}}{s_y - r_{1y}}$$

```
C <- function(stalk) {
  r_1x <- -73.662574
  r_1y <- 42.733838

  stalk %>%
    mutate(Theta1=atan2(s_x - r_1x, s_y - r_1y)) %>%
    select(c(Theta1))
}
```

$$D(s_x, s_y) = tan^{-1}\frac{s_x - r_{2x}}{s_y - r_{2y}}$$

```
D <- function(stalk){
  r_2x <- -77.0897
  r_2y <- 38.935

  stalk %>%
    mutate(Theta2=atan2(s_x - r_2x, s_y - r_2y)) %>%
    select(c(Theta2))
}
```

$$E(x, y, z, v_x, v_y, t) = (x + v_x t, y + v_y t)$$

s = expected location, where coordinates = y + displacement and x+ displacement from the equation

```
E <- function(stalk) {
    stalk %>%
      mutate(s_x = c(x + v_x*t), s_y = c(y + v_y*t)) %>%
      select(c(s_x, s_y))
}
```

Check Example 15:

```
pr1xpr2 <- function(stalk){
  stalk %>%
      select(c(x, y, z, v_x, v_y))
}
```

pr1 for u2 -> u1

```
U2_pr1 <- function(stalk){
  stalk %>%
  select(c(x, y, z))
}
```

pr1 for u3 -> v1

```
U3_pr1 <- function(stalk){
  stalk %>%
      select(c(Theta1))
}
```

pr2 for u3 -> v3

```
U3_pr2 <- function(stalk){
  stalk %>%
      select(c(t))
}
```

pr1 for u4 -> v2

```
U4_pr1 <- function(stalk){
  stalk %>%
      select(c(Theta2))
}
```

pr1 for u4 -> v3

```
U4_pr2 <- function(stalk){
  stalk %>%
      select(c(t))
}
```

ID function return itself, refer to image for components.

```
ID_X <- function(stalk){
  stalk %>%
    select(c(x, y, z, v_x, v_y, s_x, s_y, t, Theta1, Theta2))
}
```

```
ID_U1 <- function(stalk){
  stalk %>%
    select(c(x, y, z))
}
```

```
ID_U2 <- function(stalk){
  stalk %>%
    select(c(x, y, z, v_x, v_y))
}
```

```
ID_U3 <- function(stalk){
  stalk %>%
    select(c(Theta1, t))
}
```

```
ID_U4 <- function(stalk){
  stalk %>%
    select(c(Theta2, t))
}
```

```
ID_U5 <- function(stalk){
  stalk %>%
    select(c(s_x, s_y, Theta1, Theta2))
}
```

```
ID_V1 <- function(stalk){
  stalk %>%
      select(c(Theta1))
}
```

```
ID_V2 <- function(stalk){
  stalk %>%
      select(c(Theta2))
}
```

```
ID_V3 <- function(stalk){
  stalk %>%
      select(c(t))
}
```

Table representation of Figure 6 (b), page 214:

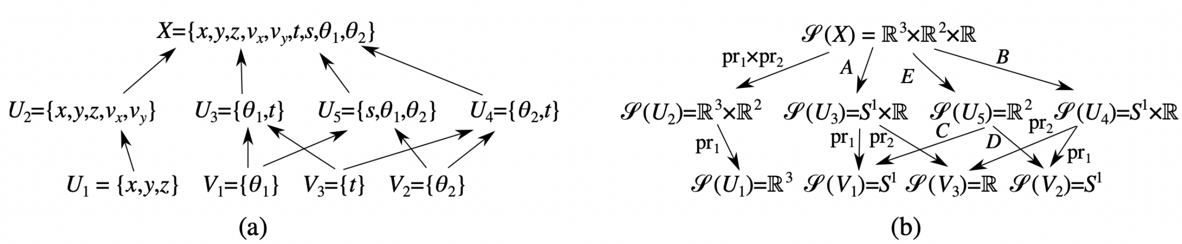

Figure 1: Figure 6

```
SixB <- tibble(SSource = c("X", "X", "X", "X", "U2", "U3", "U3", "U5", "U5", "U4",
                           "U4", "X", "U1", "U2", "U3", "U4", "U5", "V1", "V2", "V3"),
               SDest =  c("U2", "U3", "U5", "U4", "U1", "V1", "V3", "V1", "V2", "V3",
                          "V2", "X", "U1", "U2", "U3", "U4", "U5", "V1", "V2","V3"),
               DMap = c(pr1xpr2, A, E, B, U2_pr1, U3_pr1, U3_pr2, C, D, U4_pr2,
                        U4_pr1, ID_X, ID_U1, ID_U2, ID_U3, ID_U4, ID_U5, ID_V2,
                        ID_V2, ID_V3))

#ID maps w functions and SSource+SDest =.
```

Note: exec takes the function in .x and runs with input .y

```r
Table1 %>%
  select(entity, Case1, Key) %>%
  pivot_wider(names_from = entity, values_from = Case1) %>%
  right_join(SixB, by = c(Key = "SSource")) %>%
  nest(stalkinput = 2:12) %>%
  mutate(stalkoutput = map2(.x= DMap, .y = stalkinput, .f = exec)) -> FinSheaf

FinSheaf
```

```
## # A tibble: 20 x 5
##    Key   SDest DMap   stalkinput        stalkoutput
##    <chr> <chr> <list> <list>            <list>
##  1 U1    U1    <fn>   <tibble [1 x 11]> <tibble [1 x 3]>
##  2 U2    U1    <fn>   <tibble [1 x 11]> <tibble [1 x 3]>
##  3 U2    U2    <fn>   <tibble [1 x 11]> <tibble [1 x 5]>
##  4 U3    V1    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
##  5 U3    V3    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
##  6 U3    U3    <fn>   <tibble [1 x 11]> <tibble [1 x 2]>
##  7 U4    V3    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
##  8 U4    V2    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
##  9 U4    U4    <fn>   <tibble [1 x 11]> <tibble [1 x 2]>
## 10 U5    V1    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
## 11 U5    V2    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
## 12 U5    U5    <fn>   <tibble [1 x 11]> <tibble [1 x 4]>
## 13 X     U2    <fn>   <tibble [1 x 11]> <tibble [1 x 5]>
## 14 X     U3    <fn>   <tibble [1 x 11]> <tibble [1 x 2]>
## 15 X     U5    <fn>   <tibble [1 x 11]> <tibble [1 x 2]>
## 16 X     U4    <fn>   <tibble [1 x 11]> <tibble [1 x 2]>
## 17 X     X     <fn>   <tibble [1 x 11]> <tibble [1 x 10]>
## 18 V1    V1    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
## 19 V2    V2    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
## 20 V3    V3    <fn>   <tibble [1 x 11]> <tibble [1 x 1]>
```

Consistency Radius: radius=ish sd/var of stalkoutputs/ diameter of stalkoutputs. Coord. comp Unnest. pivot wider, aggregate along all of the columns. UNNESTWIDER Put the STD together, remember units are diff chi square, norm. var. Ideally have user supply aggregation function.

process below should be a specific function, so to optimize consistency radius. best consistency radius function like lm() taking consistency radius function.

```r
FinSheaf %>%
  group_by(SDest) %>%
  unnest(stalkoutput)
```

```
## # A tibble: 20 x 14
## # Groups:   SDest [9]
##    Key   SDest DMap   stalkinput     x     y     z   v_x   v_y Theta1      t
##    <chr> <chr> <list> <list>     <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
##  1 U1    U1    <fn>   <tibble>    70.7  42.8 11178    NA    NA  NA     NA
##  2 U2    U1    <fn>   <tibble>    70.6  42.7 11346    NA    NA  NA     NA
##  3 U2    U2    <fn>   <tibble>    70.6  42.7 11346  -495   164  NA     NA
##  4 U3    V1    <fn>   <tibble>    NA    NA      NA    NA    NA  77.1   NA
##  5 U3    V3    <fn>   <tibble>    NA    NA      NA    NA    NA  NA      0.943
```

```
## 6 U3    U3    <fn>   <tibble>    NA    NA      NA    NA    NA  77.1   0.943
##  7 U4    V3    <fn>   <tibble>    NA    NA      NA    NA    NA  NA     0.89
##  8 U4    V2    <fn>   <tibble>    NA    NA      NA    NA    NA  NA    NA
##  9 U4    U4    <fn>   <tibble>    NA    NA      NA    NA    NA  NA     0.89
## 10 U5    V1    <fn>   <tibble>    NA    NA      NA    NA    NA   1.56 NA
## 11 U5    V2    <fn>   <tibble>    NA    NA      NA    NA    NA  NA    NA
## 12 U5    U5    <fn>   <tibble>    NA    NA      NA    NA    NA  NA    NA
## 13 X     U2    <fn>   <tibble>    70.6  42.8 11220  -495   164  NA    NA
## 14 X     U3    <fn>   <tibble>    NA    NA      NA    NA    NA  -1.12  0.928
## 15 X     U5    <fn>   <tibble>    NA    NA      NA    NA    NA  NA    NA
## 16 X     U4    <fn>   <tibble>    NA    NA      NA    NA    NA  NA     0.928
## 17 X     X     <fn>   <tibble>    70.6  42.8 11220  -495   164  NA     0.928
## 18 V1    V1    <fn>   <tibble>    NA    NA      NA    NA    NA  NA    NA
## 19 V2    V2    <fn>   <tibble>    NA    NA      NA    NA    NA  NA    NA
## 20 V3    V3    <fn>   <tibble>    NA    NA      NA    NA    NA  NA    NA
## # ... with 3 more variables: Theta2 <dbl>, s_x <dbl>, s_y <dbl>
```

```
# have pre-consistency radii , un-group and aggregate all rads to
# get consistency radius.
# ends with: for each stalk you have a radius, then aggregate them, max, sum of squares.
```

Questions:

Consistency radius vs consistency filtration? From PNLL+Robinson paper

Joslyn, Cliff A., et al. “A Sheaf Theoretical Approach to Uncertainty Quantification of Heterogeneous Geolocation Information.” Sensors (Basel, Switzerland), vol. 20, no. 12, MDPI, 2020, p. 3418–, https://doi.org/10.3390/s20123418.

The consistency radius (more fully developed by Robinson [2] in later work) provides a native global measure of the uncertainty among the sensors present in any reading. Beyond that, the consistency filtration provides a detailed breakdown of the contributions of particular sensors and sensor combinations to that overall uncertainty.

$$\epsilon^* = max\ d_u((S(U \subset$$

Small means min. disagreement among observations, large some sensors disagree.

Per Pysheaf: Computes the error between the data assignments and extended assignments on the sheaf :param numpyNormType: Optional, how the errors across the sheaf are combined. default is take maximum error. :param cellStartIndices: Optional, which cells may start the data flowed into this cell :returns: Error between the data assignments and extended assignments on the sheaf

norm(differences in assignment?)

Relevant Testing:

```
# Keep in mind when pivoting for consistency rad.

Table1 %>%
  mutate(label = str_c(Sensor, ".", entity))%>%
  select(label, Case1, Key)%>%
  pivot_wider(names_from = label, values_from = Case1)
```

```
## # A tibble: 6 x 22
##   Key   `Flight plan.x` `Flight plan.y` `Flight plan.z` ATC.x ATC.y ATC.z
##   <chr>           <dbl>           <dbl>           <dbl> <dbl> <dbl> <dbl>
## 1 U1               70.7            42.8           11178  NA    NA      NA
## 2 U2               NA              NA                NA  70.6  42.7 11346
## 3 U3               NA              NA                NA  NA    NA      NA
## 4 U4               NA              NA                NA  NA    NA      NA
## 5 U5               NA              NA                NA  NA    NA      NA
## 6 X                NA              NA                NA  NA    NA      NA
## # ... with 15 more variables: ATC.v_x <dbl>, ATC.v_y <dbl>,
## #   `RDF 1.Theta1` <dbl>, `RDF 1.t` <dbl>, `RDF 2.Theta2` <dbl>,
## #   `RDF 2.t` <dbl>, Sat.s <dbl>, Sat.s_x <dbl>, Sat.s_y <dbl>, Field.x <dbl>,
## #   Field.y <dbl>, Field.z <dbl>, Field.v_x <dbl>, Field.v_y <dbl>,
## #   Field.t <dbl>
```