

[Tutorial] SCN2A mutations in neurodevelopmental disorders

Contents

1. Introduction	1
2. Explore your data	2
3. Exercise	6

1. Introduction

```
library(dplyr)
```

```
##
##           : 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

1.1. Background

SCN2A is a voltage-gated sodium channel gene that encodes the neuronal sodium channel NaV1.2 and plays a critical role in action potential initiation during early neurodevelopment. The latest study demonstrated that it is loss of function mutations that in SCN2A that lead to autism spectrum disorders (ASD), in contrast to gain of function, which leads to infantile seizures (Ben-Shalom 2018). In this tutorial, we will handle genetic data for SCN2A mutations identified in latest genomic studies, and then explore the data format to describe genetic mutations using R basic functions. Our tutorial will utilize the summary data from Sanders et al. (2018).

1.2. Aims

What we will do with this dataset,

-Understand the dataset from a scientific journal -Apply some functions you have learnt from the Chapter 2 and 3

2. Explore your data

2.1. Unboxing your dataset

Here we obtain the list of mutations in the Supplementary Table 1 from Sanders et al. (2018). Using the rio package, reading the excel file from the file link into your workspace. If you don't have the rio package in your system, please install as following:

```
#install.packages('rio')
```

Now you can read the file from the website. This will create the d object.

```
d = rio::import('https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6015533/bin/NIHMS957592-supplement-1.xlsx')
```

Let's explore the object you just loaded. How would you check the class of the object d?

```
class(d)
```

```
## [1] "data.frame"
```

It shows that the d object is data.frame. Then, let's overview the data frame. We will use head function to print out first few lines.

```
head(d)
```

```
##      PatientID PatientSex PatientAgeAtAssessment Chr  Pos_hg19      Ref
## 1          .           M                7y    2 154370122 166384278
## 2          .           F                3y    2 163706754 166506754
## 3      Patient2         F               18m    2 163875903 166478766
## 4 Case1,280269         F                4y    2 165798270 166304847
## 5          .           M                3y    2 166019786 166249879
## 6          .           F               25y    2 166060054 166153823
##      Alt Type      Effect c.DNA p.Protein Inheritance
## 1 12Mb Duplication CNV DuplicationCNV .      . DeNovoMosaic
## 2  2.8Mb Deletion CNV DeletionCNV .      .      DeNovo
## 3 2.6Mb Duplication CNV DuplicationCNV .      .      Inherited
## 4 507kb Duplication CNV DuplicationCNV .      .      Inherited
## 5 230kb Deletion CNV DeletionCNV .      .      Unknown
## 6  94kb Deletion CNV DeletionCNV .      .      DeNovo
##      Source SourcePMID Ben-Shalom2017 Wolff2017 AnyRecurrence
## 1 Vecchi et al 2011 21893419 Y N 1
## 2 Chen et al 2010 20346423 Y N 1
## 3 Yoshitomi et al 2015 25843248 Y N 1
## 4 Thuresson et al 2016 27153334 Y N 1
## 5 Celle et al 2013 24080482 Y N 1
## 6 Bartnik et al 2011 20807223 Y N 1
## UniqueSample/Family TrueRecurrence Seizures SeizureOnsetDays SeizureType
## 1 Y 1 Y 90 Unknown
## 2 Y 1 N . None
## 3 Y 1 Y 3 Unknown
## 4 Y 1 Y 3 Unknown
## 5 Y 1 N . None
```

```
## 6          Y          1          Y          365          Unknown
##          ASD DD/ID DD/ID severity          OtherFeatures
## 1          N          Y          Mild          clumsiness
## 2          Y          Y          Severe          dysmorphia, immature myelination
## 3 Unknown          Y          Severe          cerebral atrophy
## 4          N          N          .          .
## 5          Y          Y          Severe          microcephaly
## 6          N          Y          Moderate hypotonia, bipolar disorder, behavioral problems
##          Classification
## 1 IEE_Mild/Ataxia
## 2          ASD/DD
## 3          IEE
## 4          BIS
## 5          ASD/DD
## 6          ASD/DD
```

When you execute code in a notebook chunk, an output will be visible immediately beneath the input. From this, you can see several rows and columns in the data frame. Let's look at the first column PatientID and check which class it is.

```
head(d$PatientID)
```

```
## [1] "."          "."          "Patient2"    "Case1,280269" "."
## [6] "."
```

Cool. Now you can see the TrueRecurrence column. What is the class of the column TrueRecurrence?

```
class(d$TrueRecurrence)
```

```
## [1] "numeric"
```

To check the class of columns, you don't need to type an individual column. We can overview the summary of the dataset using summary function. Which column has the character class?

```
summary(d)
```

```
## PatientID          PatientSex      PatientAgeAtAssessment      Chr
## Length:293      Length:293      Length:293      Min. :2
## Class :character Class :character Class :character 1st Qu.:2
## Mode :character Mode :character Mode :character Median :2
##                                     Mean :2
##                                     3rd Qu.:2
##                                     Max. :2
## Pos_hg19          Ref          Alt          Type
## Length:293      Length:293      Length:293      Length:293
## Class :character Class :character Class :character Class :character
## Mode :character Mode :character Mode :character Mode :character
##
##
## Effect          c.DNA          p.Protein          Inheritance
```

```
## Length:293      Length:293      Length:293      Length:293
## Class :character Class :character Class :character Class :character
## Mode :character Mode :character Mode :character Mode :character
##
##
##
## Source          SourcePMID      Ben-Shalom2017    Wolff2017
## Length:293      Length:293      Length:293      Length:293
## Class :character Class :character Class :character Class :character
## Mode :character Mode :character Mode :character Mode :character
##
##
##
## AnyRecurrence   UniqueSample/Family TrueRecurrence     Seizures
## Min. : 1.000     Length:293      Min. :1.000     Length:293
## 1st Qu.: 1.000    Class :character 1st Qu.:1.000    Class :character
## Median : 1.000    Mode :character  Median :1.000    Mode :character
## Mean : 2.119
## 3rd Qu.: 2.000
## Max. :10.000
## SeizureOnsetDays SeizureType      ASD              DD/ID
## Length:293      Length:293      Length:293      Length:293
## Class :character Class :character Class :character Class :character
## Mode :character Mode :character Mode :character Mode :character
##
##
##
## DD/ID severity   OtherFeatures      Classification
## Length:293      Length:293      Length:293
## Class :character Class :character Class :character
## Mode :character Mode :character Mode :character
##
##
##
```

2.2. Difference between data frame and matrix

Here we will convert the data frame into a matrix, and compare which part will be different in this. To convert a data frame into a matrix, you can use the command called `as.matrix`.

```
m = as.matrix(d)
```

Let's overview the matrix object. Can you tell difference with data frame?

```
head(m[, 'TrueRecurrence'])
```

```
## 1 2 3 4 5 6
## "1" "1" "1" "1" "1" "1"
```

2.3. Subset and Sort

Some patients who have the SCN2A mutation (hereafter called “SCN2A patient”) often have seizures. So we want to know when the seizure occurs in development. Let's check the class first.

```
class(d$SeizureOnsetDays)
```

```
## [1] "character"
```

Why this column contains character? Let's head the first few lines.

```
head(d$SeizureOnsetDays)
```

```
## [1] "90"  "."   "3"   "3"   "."   "365"
```

It seems that some rows contain samples who do not have seizure or unknown information. It's represented by ".", and also recorded in another column called Seizures.

```
head(d$Seizures)
```

```
## [1] "Y" "N" "Y" "Y" "N" "Y"
```

So we want to subset rows where the seizure phenotype is available.

```
d1 = d[d$Seizures == 'Y',]
```

Let's see how many samples have seizure phenotypes Then, you can ask when is the earliest days for the representation of seizure phenotype? How can we check this? The first, as seen previously the SeizureOnsetDays column is character so we cannot apply functions for numeric.

```
head(d1$SeizureOnsetDays)
```

```
## [1] "90"  "3"   "3"   "365" "30"  "1825"
```

So we have to convert this into numeric first.

```
d1$SeizureOnsetDays2 <- as.numeric(d1$SeizureOnsetDays)
```

```
## Warning:      NA
```

Hmm. There's an warning for NA introduction. This is because some rows do not have character that we can properly convert from character to numeric. So possible solutions are either you can bear with this in your downstream analyses or 2) convert character into an appropriate form of numeric conversion. Then, the question is how can we find the rows with NA? We will ask whether the rows contains NA or not using is.na function. This will return boolean as to NA presence.

```
head(is.na(d1$SeizureOnsetDays2))
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

See we can find some rows with NA. One of them is the 13th row. Let's see how it looks like.

```
d1$SeizureOnsetDays[13]
```

```
## [1] "<365"
```

Here you have < (angle bracket) in the character so it won't properly converted to numeric information. Did you find more of these cases?

```
d1[is.na(d1$SeizureOnsetDays2),]$SeizureOnsetDays
```

```
## [1] "<365" "<365" "<365" "<28" "<30" "<365" "<365" "<365" "<365" "<365"
```

Our NAs all contains <, which prevent converting a character into a numeric. We would fix for downstream analyses. For example, we can convert <365 into 365. One function we can try is gsub. This replace your string into a format that you may not get NA. For example,

```
# gsub('pattern in your character', 'new character you want to replace', vectors for your character)
d1$SeizureOnsetDays3 <- gsub('<', '', d1$SeizureOnsetDays)
head(d1$SeizureOnsetDays3)
```

```
## [1] "90" "3" "3" "365" "30" "1825"
```

Let's convert them into numerics.

```
d1$SeizureOnsetDays3 <- as.numeric(d1$SeizureOnsetDays3)
```

Did you get warning for this? Now we can ask our initial question. When is the earliest day for having seizure?

```
min(d1$SeizureOnsetDays3)
```

```
## [1] 0
```

3. Exercise

The dataset contains more details for genetic mutations in SCN2A patients. From this information, what can we analyze further?

Here I list up few questions you can examine further.

- Finding the position of the genetic mutations within SCN2A. Which information you would use? If you are not familiar with positional information on genetic variants (or mutations), please find the Figure 1 or the slides for Mutation (BSMS205 Session 3-1).
- Counting the recurrent mutations at the same protein position (in other words, the same mutations seen across different patients), and examine whether the patients have similar phenotype.
- Finding the position where different consequences mutations occur. Please note that “consequences” are loss-of-function (Nonsense, Frameshift) or missense.
- Sketch a plot to visualize your analysis.

Figure 1. Standard mutation nomenclature (Ogino et al. 2007). According to this guideline, genetic mutations can be represented for coding DNA reference sequences (c. prefix) and protein-level amino acid sequences (p. prefix).

3.1. For-loops and Vectorization

Here we examine more details of genetic mutations as to their functional consequence and position of SCN2A mutations. In the dataset includes, there are two columns called c.DNA and p.Protein, containing the cDNA or protein position for the genetic mutations. During these exercises, we will look at the concept of for-loop and vectorization, which you learn from the Chapter 3.4. Let's look at the column p.Protein. It contains protein positions from each patient. What would you check at the first place?

-Task 1 -Task 2 -Task 3 -.... -Task N

Let's write down your code to explore this column.

```
head(d$p.Protein)
```

```
## [1] "." "." "." "." "." "."
```

If you need more information on SCN2A, please visit the Uniprot description for SCN2A. The Uniprot database contains description for protein domains.

Then, I would remove the characters from the string so we can have only numerics for positions. Here I use gsub function to extract numbers from string. Let's remove non-numeric characters from the string.

```
gsub('[^0-9]', '', 'p.R102X')
```

```
## [1] "102"
```

In the dataset, we have many rows for protein positions. One way we might try is to set up for-loop to process each row.

```
a <- vector(length = 293)
for (i in 1:293) {
  a[i] <- gsub('[^0-9]', '', d[i, 'p.Protein'])
}
```

Do you think this is an effective approach? As we have done in your assignment, for-loop is not a good choice to process vectors because R can do vectorization for this process with a shorter and clearer code. So this mean you can apply gsub on vector and return your output to another column (could be new assign).

```
i <- 1:293
b <- gsub('[^0-9]', '', d[i, 'p.Protein'])
b
```

```
## [1] "" "" "" "" "" "" "" "" "12" "28"
## [11] "36" "36" "36" "76" "82" "102" "102" "" "" "132"
## [21] "136" "136" "" "" "169" "169" "172" "188" "188" "202"
## [31] "" "" "207" "208" "211" "211" "212" "213" "213" "218"
## [41] "220" "223" "227" "" "236" "237" "237" "240" "252" "258"
## [51] "261" "261" "263" "263" "263" "263" "263" "263" "263" "263"
## [61] "263" "266" "281" "328" "328" "343" "365" "379" "379" "389"
## [71] "423" "423" "424" "430" "430" "430" "439" "439" "440" "478"
## [81] "502" "564" "565" "583" "607" "609" "612" "614" "649" "674"
## [91] "686" "700" "" "717" "733" "772" "784" "" "812" "828"
## [101] "849" "850" "853" "853" "853" "853" "853" "853" "853" "853"
```

```
## [111] "853" "853" "856" "863" "873" "876" "879" "880" "881" "882"
## [121] "882" "882" "885" "887" "891" "892" "895" "899" "899" "905"
## [131] "905" "908" "928" "928" "930" "937" "937" "937" "959" "965"
## [141] "976" "978" "978" "983" "987" "999" "999" "999" "999" "999"
## [151] "999" "999" "1001" "1003" "1013" "1021" "1021" "1050" "1114" "1115"
## [161] "1125" "1128" "1155" "1170" "1211" "1211" "1211" "1223" "1235" "1260"
## [171] "1260" "1281" "1282" "1282" "1310" "1312" "1319" "1319" "1319" "1319"
## [181] "1319" "1319" "1319" "1319" "1319" "1321" "1323" "310" "310" "1326"
## [191] "1326" "1330" "1336" "1336" "1338" "1341" "1342" "1342" "1342" "1342"
## [201] "1386" "1387" "1398" "1408" "310" "1420" "1422" "1435" "1435" "1435"
## [211] "" "" "1473" "1479" "1479" "1500" "1501" "1515" "" "1521"
## [221] "1522" "1522" "1528" "1531" "1531" "1536" "1537" "1545" "1548" "1563"
## [231] "1576" "1589" "1593" "1593" "1594" "1596" "1597" "1598" "1598" ""
## [241] "1622" "1623" "1626" "1626" "1627" "1629" "1629" "1629" "1632" "1634"
## [251] "1634" "1635" "1640" "1641" "1650" "1651" "1652" "1656" "1660" "1665"
## [261] "1711" "1716" "1731" "1744" "1744" "1758" "1773" "1773" "1773" "1773"
## [271] "1778" "1780" "1811" "1829" "1853" "1875" "1880" "1882" "1882" "1882"
## [281] "1882" "1882" "1882" "1882" "1882" "1882" "1882" "1902" "1902" "1904"
## [291] "1918" "1933" "1974"
```

```
d2 <- d %>%
  mutate(proteinpositions = b)
```

3.2. Counting the recurrent mutations

Recurrent mutations are the ones that the same genetic mutations occur in multiple individuals. Recurrent mutations can be common 1) when the mutation does not affect on natural selection, 2) when the mutation is beneficial, 3) in the hotspot for a disease or strongly associated with trait. However, given we are dealing with the genetic mutations from rare disorders, the mutations in the dataset are supposed to be uniquely present in general population. Otherwise, the recurrent mutations can indicate strong association with the phenotype.

To assess the recurrent mutations, the first thing we can try is to examine whether the same mutations occur in multiple individuals. Since the dataset contains individual patients for each row, we can simply check the frequency using:

```
c <- as.data.frame(table(d$p.Protein))
```

or we can check the number of unique variants in the dataset by:

```
d2 %>%
  group_by(p.Protein) %>%
  count() %>%
  filter(n == '1')
```

```
## # A tibble: 163 x 2
## # Groups:   p.Protein [163]
##   p.Protein     n
##   <chr>       <int>
## 1 p.A1310S     1
## 2 p.A1500T     1
## 3 p.A1652P     1
```



```
## 4 p.A1773T      1
## 5 p.A202V       1
## 6 p.A240S       1
## 7 p.A263T       1
## 8 p.A733T       1
## 9 p.A812V       1
## 10 p.A880S      1
## # ... with 153 more rows
```

```
d2 %>%
  group_by(p.Protein) %>%
  count() %>%
  filter(n != '1')
```

```
## # A tibble: 40 x 2
## # Groups:   p.Protein [40]
##   p.Protein      n
##   <chr>        <int>
## 1 .              8
## 2 p.A1773V       3
## 3 p.A263V        8
## 4 p.A439fs       2
## 5 p.E1211K       3
## 6 p.E999K        6
## 7 p.F328V        2
## 8 p.F928C        2
## 9 p.F978L        2
## 10 p.G1522A      2
## # ... with 30 more rows
```

How many unique variants can you find? and which variants are occurred in multiple times?

Then, you can use other columns to check frequency for different groups. Which columns you would use for more grouping?

If you find that, please check the recurrent mutations for each group.

```
d2 %>%
  group_by(c.DNA) %>%
  count() %>%
  filter(n == '1')
```

```
## # A tibble: 176 x 2
## # Groups:   c.DNA [176]
##   c.DNA              n
##   <chr>            <int>
## 1 c.1028A>G         1
## 2 c.1094C>T         1
## 3 c.1165C>T         1
## 4 c.1270G>T         1
## 5 c.1288G>C         1
## 6 c.1289A>C         1
## 7 c.1289A>G         1
## 8 c.1316CTGAACAGAAGGAAGCTGAATTCAGCAGATGC>C 1
```

```
## 9 c.1317_1348del 1
## 10 c.1317T>TC 1
## # ... with 166 more rows
```

```
d2 %>%
  group_by(c.DNA) %>%
  count() %>%
  filter(n != '1')
```

```
## # A tibble: 40 x 2
## # Groups:   c.DNA [40]
##   c.DNA      n
##   <chr>    <int>
## 1 .         8
## 2 ???       2
## 3 c.106A>G   3
## 4 c.1136G>A  2
## 5 c.1267G>C  2
## 6 c.2558G>A  9
## 7 c.2645G>A  2
## 8 c.2695G>A  2
## 9 c.2715G>C  2
## 10 c.2783T>G  2
## # ... with 30 more rows
```

```
d2 %>%
  group_by(Pos_hg19) %>%
  count() %>%
  filter(n == '1')
```

```
## # A tibble: 169 x 2
## # Groups:   Pos_hg19 [169]
##   Pos_hg19      n
##   <chr>    <int>
## 1 .         1
## 2 154370122  1
## 3 163706754  1
## 4 163875903  1
## 5 165798270  1
## 6 166019786  1
## 7 166060054  1
## 8 166084399  1
## 9 166152059  1
## 10 166152367  1
## # ... with 159 more rows
```

```
d2 %>%
  group_by(Pos_hg19) %>%
  count() %>%
  filter(n != '1')
```

```
## # A tibble: 44 x 2
```

```
## # Groups:   Pos_hg19 [44]
##   Pos_hg19      n
##   <chr>      <int>
## 1 166152439      3
## 2 166153563      2
## 3 166153647      2
## 4 166164379      2
## 5 166164448      2
## 6 166165261      2
## 7 166165305      2
## 8 166165701      2
## 9 166165707      2
## 10 166166845      2
## # ... with 34 more rows
```

3.3. What is the proportion of diagnosis for SCN2A patient?

SCN2A mutation can have multiple different consequences for disease phenotypes. It can cause ASD but also other neurodevelopmental conditions. In total cases, how many phenotypes occur in SCN2A patients. Then, calculate the proportion of the phenotypes among total cases.

```
d %>%
  group_by(Classification) %>%
  count() %>% ungroup() %>%
  mutate(proportion = n/sum(n))
```

```
## # A tibble: 7 x 3
##   Classification      n proportion
##   <chr>      <int>      <dbl>
## 1 ASD/DD           92      0.314
## 2 BIS              36      0.123
## 3 IEE             111      0.379
## 4 IEE_Mild/Ataxia    7      0.0239
## 5 Other              3      0.0102
## 6 Schizophrenia      5      0.0171
## 7 Unclear           39      0.133
```

Then, you might be intrigued to whether females and males have different occurrence in each disorder. Let's check it.

```
d %>%
  group_by(PatientSex, Classification) %>%
  filter(PatientSex %in% c("F", "M")) %>%
  summarise(n=n(), .groups = "drop_last") %>%
  mutate(freq = n / sum(n))
```

```
## # A tibble: 13 x 4
## # Groups:   PatientSex [2]
##   PatientSex Classification      n   freq
##   <chr>      <chr>      <int> <dbl>
## 1 F          ASD/DD           26 0.228
## 2 F          BIS              11 0.0965
```

```
## 3 F      IEE      57 0.5
## 4 F      IEE_Mild/Ataxia  3 0.0263
## 5 F      Other      2 0.0175
## 6 F      Schizophrenia  1 0.00877
## 7 F      Unclear     14 0.123
## 8 M      ASD/DD      44 0.370
## 9 M      BIS        12 0.101
## 10 M     IEE        43 0.361
## 11 M     IEE_Mild/Ataxia  4 0.0336
## 12 M     Other      1 0.00840
## 13 M     Unclear     15 0.126
```

Another question you can ask is whether different mutation consequences occur in each phenotype. Let's find out how many mutation consequences are observed in each phenotype.

```
d %>%
  group_by(Classification, Effect) %>%
  count()
```

```
## # A tibble: 25 x 3
## # Groups:   Classification, Effect [25]
##   Classification Effect      n
##   <chr>          <chr>    <int>
## 1 ASD/DD        DeletionCNV      3
## 2 ASD/DD        DuplicationCNV    1
## 3 ASD/DD        Frameshift     17
## 4 ASD/DD        Missense       40
## 5 ASD/DD        Nonsense       16
## 6 ASD/DD        PopulationVariantInExAC  3
## 7 ASD/DD        SpliceSite     12
## 8 BIS           DuplicationCNV    1
## 9 BIS           Frameshift       1
## 10 BIS          Missense       34
## # ... with 15 more rows
```

3.4. Find the position where different consequences of mutations occur

If you checked the recurrent mutations, you might want to find a locus where two or more variants occur. Such loci might indicate functionally important position of the gene and you might find some insight as to a cause of disease.

```
# 1 locus, different 'variants'
i <- 1:293
p <- gsub('[^0-9]', '', d[i, 'c.DNA'])

d3 <- d2 %>%
  mutate(c.DNApositions = p)
c.p <- d3 %>%
  group_by(c.DNApositions, Alt) %>%
  count() %>% ungroup() %>%
  group_by(c.DNApositions) %>%
  count() %>%
```

```
filter(n>1)
c.p
```

```
## # A tibble: 10 x 2
## # Groups:   c.DNApositions [10]
##   c.DNApositions      n
##   <chr>           <int>
## 1 ""              10
## 2 "1289"           2
## 3 "2567"           2
## 4 "3778"           2
## 5 "3956"           3
## 6 "4793"           2
## 7 "4886"           2
## 8 "4901"           2
## 9 "5645"           3
## 10 "6051"          2
```

```
#6051 = 605+1
```

```
# 1 locus, different 'Effects'
p.p <- d2 %>%
  group_by(proteinpositions, Effect) %>%
  count() %>% ungroup() %>%
  group_by(proteinpositions) %>%
  count() %>%
  filter(n>1)
p.p
```

```
## # A tibble: 3 x 2
## # Groups:   proteinpositions [3]
##   proteinpositions      n
##   <chr>           <int>
## 1 ""              3
## 2 "1626"           2
## 3 "169"            2
```

3.5. Sketch a plot to visualize your analysis

When you examine the dataset, you would draw something to show your output. Though we haven't learnt how to plot data yet, we can have a quick sketch for the dataset. There's no restriction on your suggestion. Please submit your hand-drawing for the plot you would like to show from this dataset.