Synaptome.db: A Bioconductor package for synaptic proteomics data

Bioinformatics Advances, YYYY, 0–0
doi: 10.1093/Bioinformatics Advances/xxxxx
Advance Access Publication Date: DD Month YYYY
Manuscript Category

Subject Section

Synaptome.db: A Bioconductor package for synaptic proteomics data.

Oksana Sorokina[1*], Anatoly Sorokin[2,3] and J Douglas Armstrong[1,4]

[1]School of informatics, University of Edinburgh, UK., [2]Department of Biochemistry and Systems Biology, Institute of Systems, Molecular and Integrative Biology, Faculty of Health and Life Sciences, University of Liverpool, Liverpool, UK, [3] [3]Biological Systems Unit, Okinawa Institute of Science and Technology, Okinawa, Japan, [4]Computational Biomedicine Institute (IAS-5 / INM-9), Forschungszentrum Jülich, Jülich, Germany.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Summary: The neuronal synapse is underpinned by a large and diverse proteome but the molecular evidence is spread across many primary datasets. These data were recently curated into a single dataset describing a landscape of ~ 8000 proteins found in studies of mammalian synapses. Here we describe programmatic access to the dataset via the R/Bioconductor package **Synaptome.db**, which enables convenient and in-depth data analysis from within the Bioconductor environment. Synaptome.db allows users to obtain the respective gene information, e.g. subcellular localization, brain region, gene ontology, disease association and construct custom protein-protein interaction network models for gene sets and entire subcellular compartments.

Availability and implementation: The package Synaptome.db is part of Bioconductor since release 3.14, https://bioconductor.org/packages/release/data/annotation/html/synaptome.db.html, it is open source and available under the Artistic license 2.0. The development version is maintained on GitHub (https://github.com/lptolik/synaptome.db). Full documentation including examples is provided in the form of vignettes on the package webpage.

Contact: oksana.sorokina@ed.ac.uk

Supplementary information: Supplementary data are available at Bioinformatics Advances online.

1 Introduction

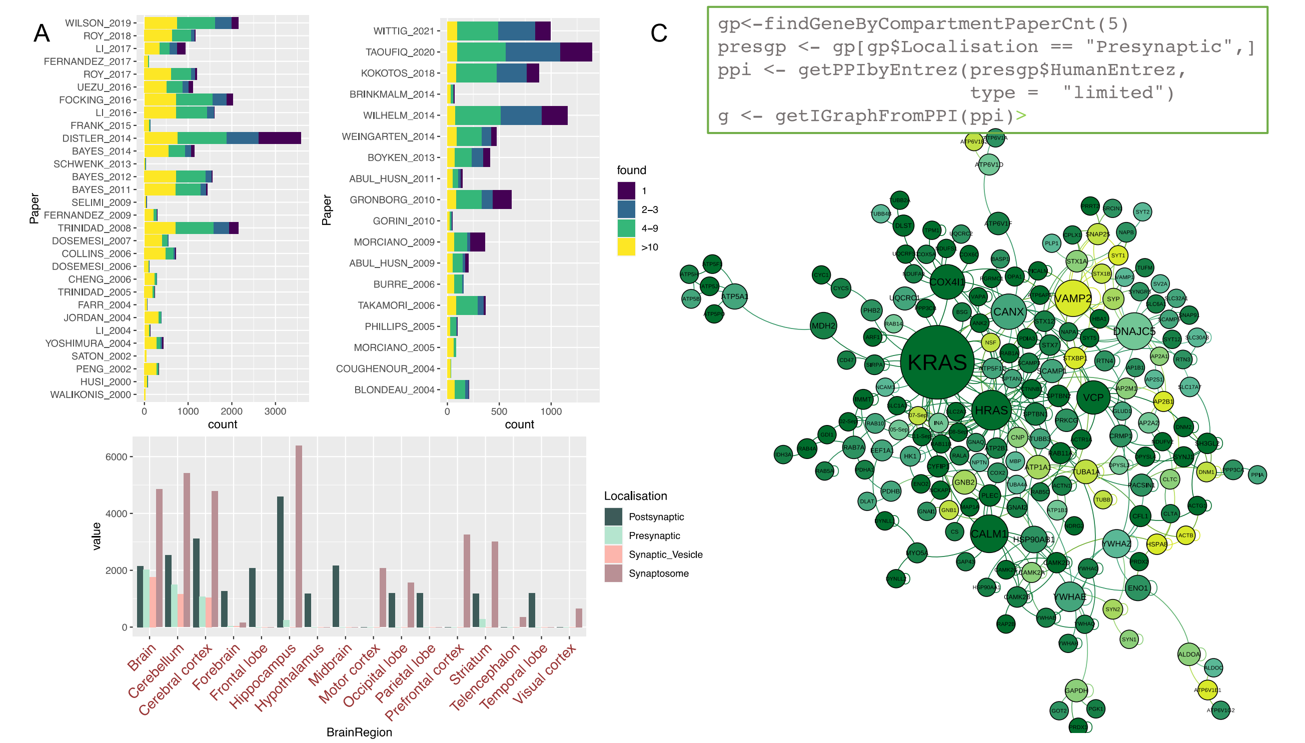
The proteomes of the presynaptic and postsynaptic compartments mediate information processing in the brain via complex and highly dynamic molecular networks. Sorokina et al., 2021 systematically curated 58 proteomic studies from 2000 to 2020, to produce a comprehensive dataset describing > 8000 proteins expressed at the mammalian synapse (*1*). The set includes 29 post synaptic proteome (PSP) studies (2000 to 2019) contributing to a total of 5560 mouse, human and rat unique gene identifiers; 18 presynaptic studies (2004 to 2020) resulting in 2772 unique gene IDs, and 11 studies for whole synaptosomes reporting 7198 unique gene IDs.

Each synaptic component was annotated with relevant metadata based on the respective study (author, year, method, subcellular compartment, brain region) and associated with function and disease information according to Gene Ontology and Human Disease Ontology. Figure 1, A shows studies aggregating pre- (right panel) and postsynaptic (left panel) compartments with numbers of identified proteins, while Figure 1, B shows the brain regions, annotated from the studies with respective numbers of proteins. It could be seen that coverage highly varies between regions, as the most of collected studies were performed on the whole brain, hippocampus, cerebellum and cerebral cortex.

Furthermore, the protein–protein interactions (PPI) were obtained for the pre- and post-synaptic proteomes based on combined human, mouse and rat data from BioGRID (*2*), Intact (*3*) and DIP (*4*). Interaction sources were filtered for methods that produce data on direct physical interactions with the highest confidence. The interaction data from each database was extracted in the PSI-MITAB format

To merge the datasets we standardised the IDs used, by mapping each onto Entrez gene IDs. To extract only direct interactions, the 'interaction type' column was then filtered for the PSI- MI terms“association” (MI:0914), “physical association” (MI:0915) and 'direct interaction' (MI:0407) and their 63 child-terms. Some of the source data used an obsolete interaction type MI:0218, “physical interaction” which could still be used, since it was updated to association and physical association, which we both include. PPIs based on the interaction types: “genetic interaction” (MI:0208) (including “suppression” (MI:0796) and “synthetic” (MI:0794)), “colocalization” (MI:0403), “genetic interference” (MI:0254) and “additive genetic interaction defined by inequality” (obsolete term, MI:0799) were excluded from the final set as these methods are designed to include both direct and indirect interactions.

To maximise confidence in direct physical interactions we also excluded predicted interactions and interactions obtained by Co-IP experiments (spoke models), filtering out the PSI-MI terms like “Pull-down”, “Affinity technology”, etc.



```
gp<-findGeneByCompartmentPaperCnt(5)
presgp <- gp[gp$Localisation == "Presynaptic",]
ppi <- getPPIbyEntrez(presgp$HumanEntrez,
                      type =  "limited")
g <- getIGraphFromPPI(ppi)>
```

Figure 1. Summary statistics for data combined in Synaptome.data and an example of the functionality of Synaptome.db A) Studies used for aggregation of pre- (right) and postsynaptic (left) compartments with numbers of identified proteins; color code represents distribution of discovery rate for proteins found in each study, where yellow corresponds to the highly confident proteins, identified in at n>10 studies and dark blue corresponds to the proteins found in just a single study B) Disribution of numbers of proteins by brain region; color corresponds to distribution of subcellular localisation for the proteins in each of the region C) Code example for extraction of a presynaptic PPI network and the resulting graph visualised with Gephi (https://gephi.org/), node colors match those from Figure 1,A, node size correspond to node degree.

This resulted in two large-scale PPI networks (4817 nodes and 27,788 edges for PSP and 2221 nodes and 8678 edges for presynaptic proteome).

Combined these provide a unified and configurable resource for constructing customised networks for the synaptic proteome. The resulting network model is available in a SQLite implementation from Edinburgh DataShare https://doi.org/10.7488/ds/3771 and EBRAINS. Although highly extendible, the SQLite implementation requires specific database-related expertise restricting its use to specialist bioinformatics researchers; while gene information and network models stored in the database provide the much-in-demand resource for the broader community of the molecular neuroscientists. To make the database more widely accessible we developed the Bioconductor package **synaptome.db**, which enables direct access to the data (embedded into the satellite package **synaptome.data**) from witin the R environment.

We developed both packages as components of Bioconductor project (4), which is designed to facilitate rigorous and reproducible analysis of biological data by building customised pipelines and workflows (*5*). The incorporation into Bioconductor allows users to combine the synaptic PPI networks and protein annotation with external genomics (org.Hs.eg.db, org.Mm.eg.db, and org.Rn.eg.db packages (*6-8*), transcriptomics (via various ChipDB packages (*9*), mutations and polymorphism analysis (via PolyPhen.Hsapiens.dbSNP131 (*10*) to name just a few examples. Results of analysis could be presented in domain-specific manner by, for example ggbio (*11*) or KaryoploteR packages (*12*) (see the example below) . Synaptome.db can be also used to provide annotation for experimental datasets, or as a source for hypothesis generation and experimental design.

2 Implementation

To comply with the requirements of Bioconductor the database itself was wrapped into an AnnotationHub (*13*) package, synaptome.data, that fetches most recent version of the database from Edinburgh DataShare site and caches it for further use. The synaptome.db package provides a simple API for extracting the data from the database without understanding of the underlying database structure or using other database related skills. Users with SQL experience can still also query the database directly via synaptome.data package using the schema described in (*1*).

2.1 Synaptome.db functionality

The functions implemented in the current release were designed to support the most frequent user queries: When?, and by whom?, was my favorite gene (or list of genes) identified? Was my gene/list found pre- or post-synaptically? and how often? Was it found in a specific brain region? and which diseases it is associated with? Functions findGenesByEntrez and findGenesByName return the following identifiers for genes specified by Entrez ID or gene name, respectively: GeneID (internal database ID), MGI ID, Human Entrez ID, Mouse Entrez ID, Rat Entrez ID, Human gene name, Mouse gene name and Rat gene name. Here, Internal GeneID corresponds to our unique database ID, which helps to resolve ambiguity across the external IDS, for example where a mouse Entrez gene IDs matches the same Human one, etc. Internal GeneIDs can then be used to extract subcellular compartment `(getAllGenes4Compartment)` or brain region `(getAllGenes4BrainRegion)` protein composition, and for extracting PPIs for selected molecules `(getPPIbyIDs)`, , as shown in Figure 1. It is also possible to get Human disease information (HDO provided) for any subset of Human Entrez IDs (`getGeneDiseaseByEntres)`, internal Gene IDs and Human gene names. As it is based on manually curated data, synaptome.db provides a literature provenance trail `(getGeneInfoByIDs)` for each of its data points, including details such as Localisation (one of the following: presynaptic, postsynaptic, synaptosome), PaperPMID (PMID for the publications where the genes were reported), Paper (papers where specific genes were reported in a format FIRSTAUTHOR_YEAR), Year, SpeciesTaxID (species on which the original experiment was performed on), BrainRegion (Brain region where the specific genes were identified, according to the paper). Where a users wants to check whether query set of proteins have previously been identified as synaptic, we enabled a quick check by command `getGenes4Compartment` and `getGenes4BrainRegion,` were one needs to provide Compartment Id and Specie TaxID or/and BrainRegion ID, along with the list of internal Gene Ids for the proteins obtained from experiment.

Given that the diversity across synaptic proteomics datasets (e.g. low overlap between some synaptosome datasets) could easily be due to

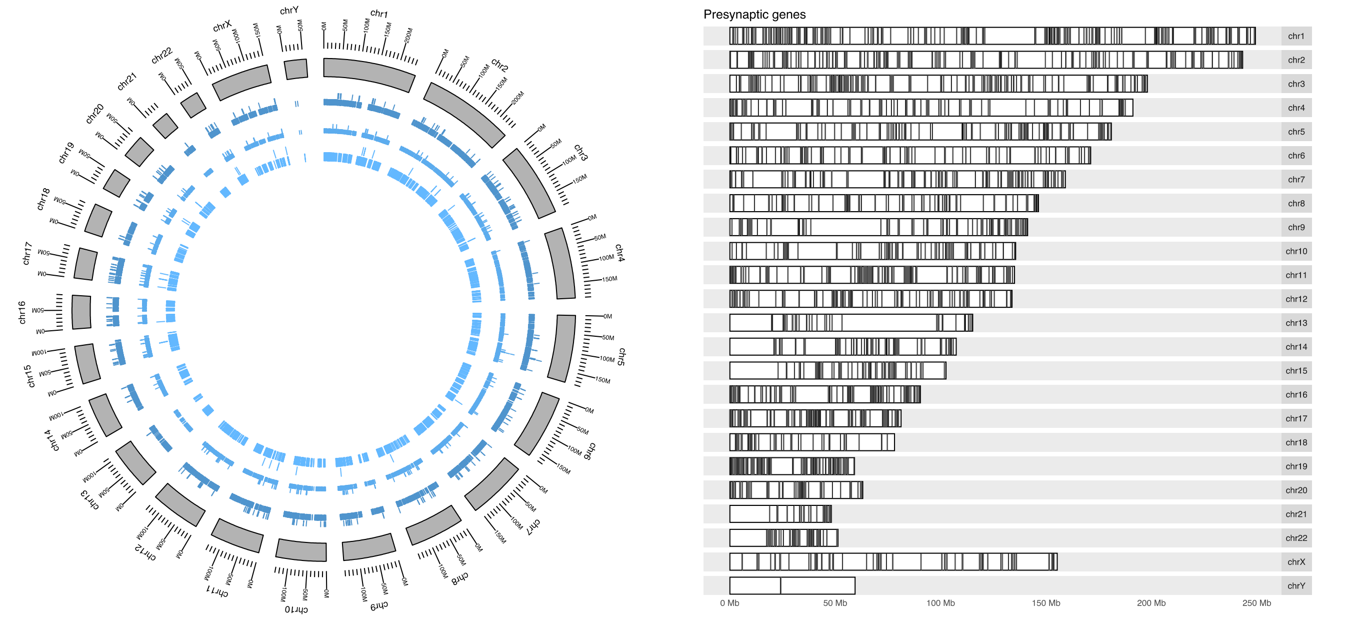


Figure 2. Distribution of synaptic genes over the Human chromosomes. A. Circus diagram showing the distribution of pre, post and synaptosome genes on each chromosome. B The localisation of presynaptic genes on the Human chromosomes.

differences in biochemical enrichment protocols and mass-spec setups, it is likely that only a subset of proteins in each dataset described here are truly synaptic. Figure 1, A demonstrates the distribution of proteins with different discovery rates over pre- and post- synaptic studies. It could be seen that most stable (yellow) population makes more or less regular proportion, while the number of proteins discovered only in single studies (dark blue) varies between the datasets. To tackle this issue we enabled a few functions that use "count" (discovery rate, or number of protein identification in different studies) to enable custim filters for the proteins that were identified more frequently than others, thus, may correspond to more probable synaptic residents. One of them, `findGeneByPaperCnt`, selects the proteins from the total list off ~ 8000, which were found more than defined "count" of studies, e. g. one can select the genes that were identified in more than 5 studies in all compartments. Another, `findGeneByCompartmentPaperCnt`, allows similar filtering for specific compartment.

The use of this command in illustrated in Figure 1C, were we selected the most confident protein set (for example, "count" = 5, proteins identified in at least 5 presynaptic studies). In addition, the command `findGeneByPapers` enables extraction of protein lists from specific studies, which can listed with the command `getPapers`.

Finally, the package supports extraction of PPIs for the gene list or entire compartment/brain region and their export in a form of a network graph or a table (example code and network presented in Figure 1, C) . Custom protein-protein interactions based on bespoke subsets of molecules can be extracted in two general ways: "induced" and "limited." In the first case, the command will return all possible interactors for the genes within the whole interactome. In the second case it will return only interactions between the genes of interest. PPIs could be obtained by submitting list of EntrezIDs or gene names, or Internal IDs – in all cases the interactions will be returned as a list of interacting pairs of Internal GeneIDs.

To summarize, the package allows users to do the following:

- Finding a variety of Gene ID information for specific gene/lists(s)
- Finding molecular composition for specific compartments or brain regions
- Finding the most confident set of proteins for the total synaptosome or specific compartments
- Extracting the protein lists from specific papers
- Finding disease associations for selected genes
- Comparing user defined protein lists against specific compartments and/or brain regions
- Finding PPIs for selected genes/compartments/brain regions.
- Constructing custom PPI graphs and network models

(See Supplementary materials for package vignette and manual with detailed functionality)

3 Example

The following brief example demonstrates how the SynaptomeDB can be used in combination with other Bioconductor packages (Figure 2).

We extracted a complete list of human gene IDs for each of the presynaptic compartment, the postsynaptic compartment and the entire synaptosome. For each of these gene sets we mapped genes onto the Human kariotype to get a distribution map of the respective gene positions across all human chromosomes using the ggbio package (*11*).

We could then select genes that are annotated to any specific disorder, e.g. Alzheimer disease (AD). Supplementary Figure 1 shows the distribution AD related synaptic genes across human chromosomes. The colour code corresponds to each gene's subcellular localization. R code for the example is available from Supplementary materials.

4 Conclusions

We developed the Bioconductor packages synaptome.data and synaptome.db to provide a simple and intuitive access to the data in SynaptomeDB. These packages can easily be incorporated into custom bioinformatics data pipelines along with other annotations, experimental data and statistical methods exploiting the features of Bioconductor and

R for further analysis. We aim to update the package twice a year to incorporate newly available datsets and are open to suggestions.

Acknowledgement

We thank the EBRAINs facility of the Human Brain Project for hosting a public version of the database at https://doi.org/10.25493/VA01-BRD and Edinburgh DataShare for hosting the raw datasets.

Funding

This research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement Nos. 945539 (Human Brain Project SGA3).

Conflict of Interest: none declared

References

1. O. Sorokina *et al.*, A unified resource and configurable model of the synapse proteome and its role in disease. *Scientific Reports* **11**, 9967 (2021).

2. R. Oughtred *et al.*, The BioGRID interaction database: 2019 update. *Nucleic Acids Res* **47**, D529-d541 (2019).

3. S. Kerrien *et al.*, The IntAct molecular interaction database in 2012. *Nucleic Acids Res* **40**, D841-846 (2012).

4. L. Salwinski *et al.*, The Database of Interacting Proteins: 2004 update. *Nucleic Acids Res* **32**, D449-451 (2004).

5. W. Huber *et al.*, Orchestrating high-throughput genomic analysis with Bioconductor. *Nat Methods* **12**, 115-121 (2015).

6. https://bioconductor.org/packages/release/data/annotation/html/org.Hs.eg.db.html.

7. https://bioconductor.org/packages/release/data/annotation/html/org.Mm.eg.db.html.

8. https://bioconductor.org/packages/release/data/annotation/html/org.Rn.eg.db.html.

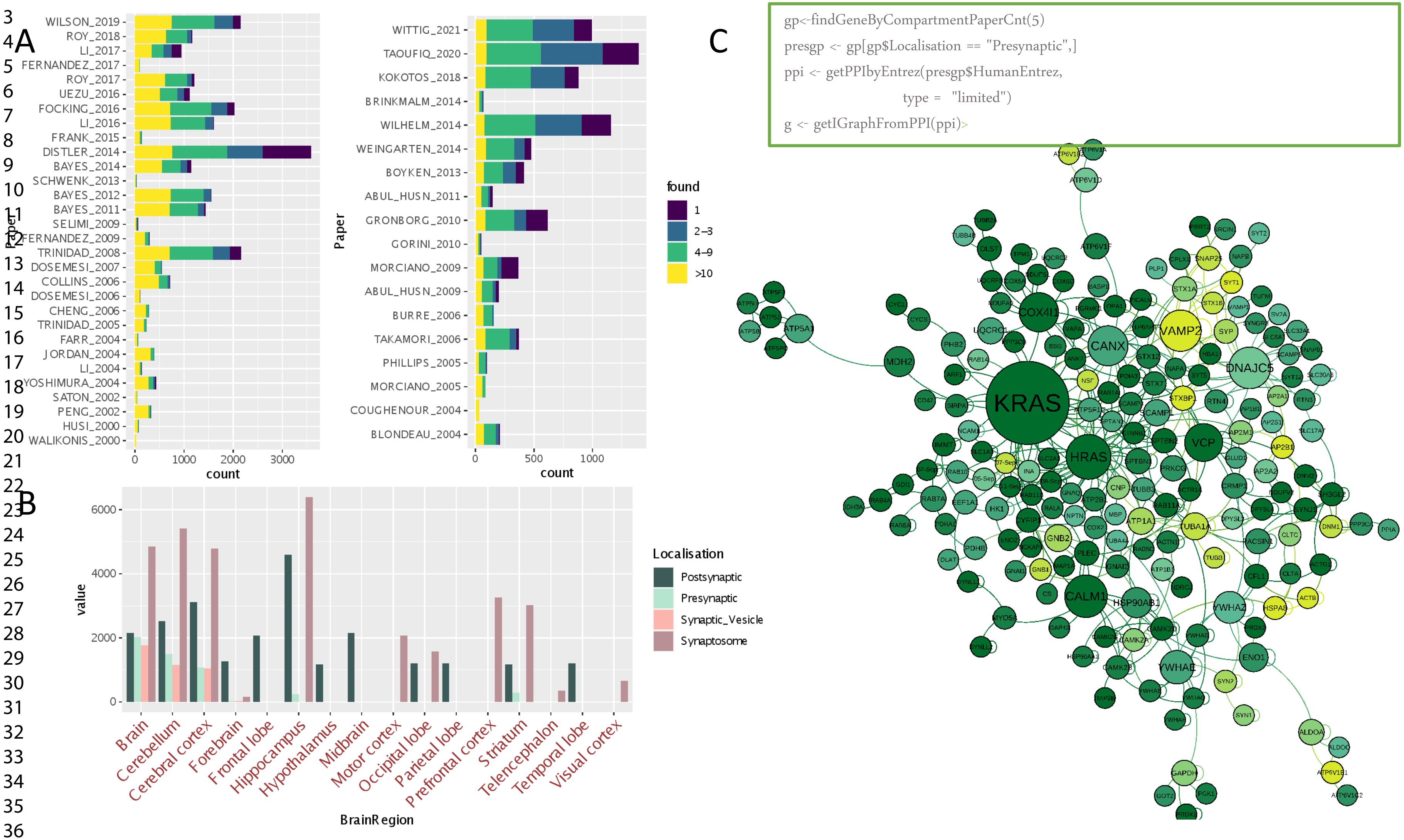
9. https://bioconductor.org/packages/release/BiocViews.html#___ChipDb.

10. https://bioconductor.org/packages/release/data/annotation/html/PolyPhen.Hsapiens.dbSNP131.html.

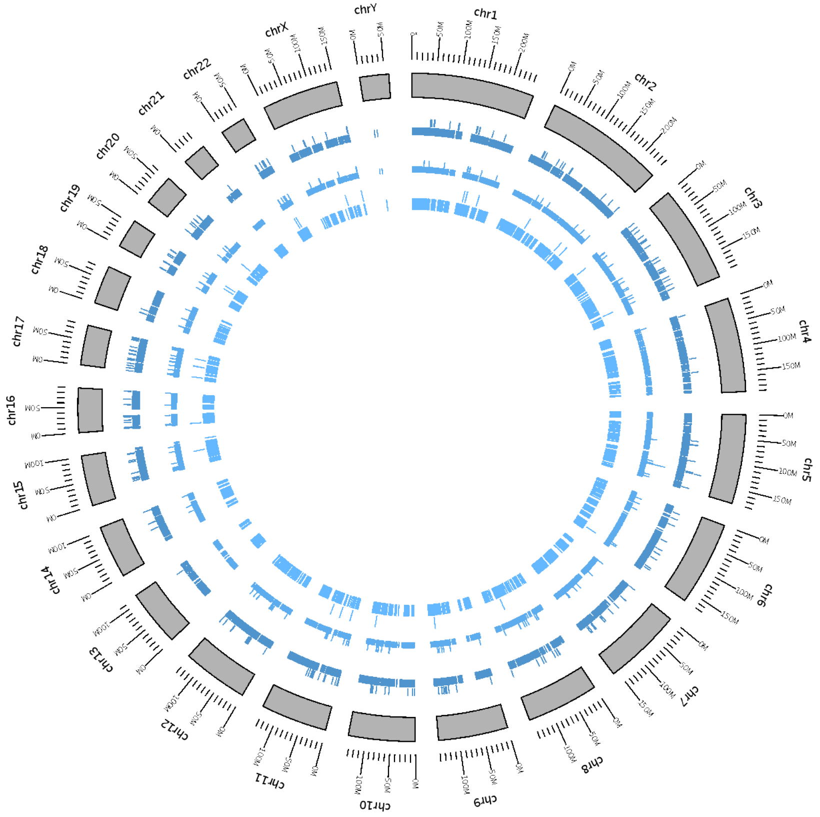
11. T. Yin, D. Cook, M. Lawrence, ggbio: an R package for extending the grammar of graphics for genomic data. *Genome Biol* **13**, R77 (2012).

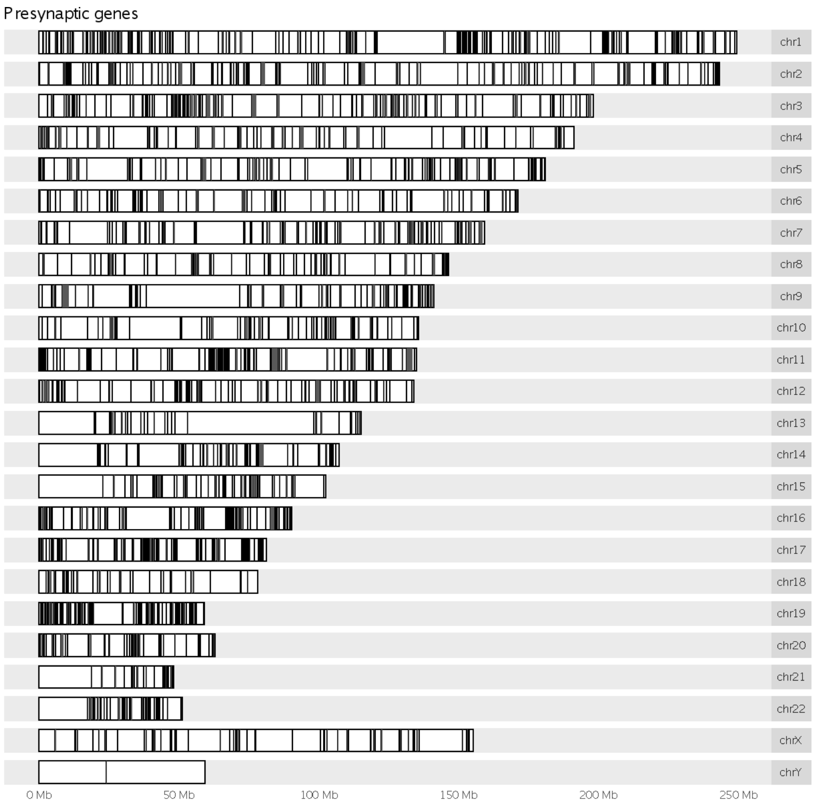
12. B. Gel, E. Serra, karyoploteR: an R/Bioconductor package to plot customizable genomes displaying arbitrary data. *Bioinformatics* **33**, 3088-3090 (2017).

13. M. M, S. L, AnnotationHub: Client to access AnnotationHub resources. R package version 3.0.2., (2021).



```
gp<-findGeneByCompartmentPaperCnt(5)
presgp <- gp[gp$Localisation == "Presynaptic",]
ppi <- getPPIbyEntrez(presgp$HumanEntrez,
                     type =  "limited")
g <- getIGraphFromPPI(ppi)>
```


chr1
chr2
chr3
chr4
chr5
chr6
chr7
chr8
chr9
chr10
chr11
chr12
chr13
chr14
chr15
chr16
chr17
chr18
chr19
chr20
chr21
chr22
chrX
chrY
0M
50M
100M
150M
200M


Presynaptic genes
chr1
chr2
chr3
chr4
chr5
chr6
chr7
chr8
chr9
chr10
chr11
chr12
chr13
chr14
chr15
chr16
chr17
chr18
chr19
chr20
chr21
chr22
chrX
chrY
0 Mb
50 Mb
100 Mb
150 Mb
200 Mb
250 Mb

Untitled

Oksana Sorokina

2022-10-13

0.1 R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
gp<-findGeneByCompartmentPaperCnt(1)
papers <- getPapers()
```

1 Presynaptic

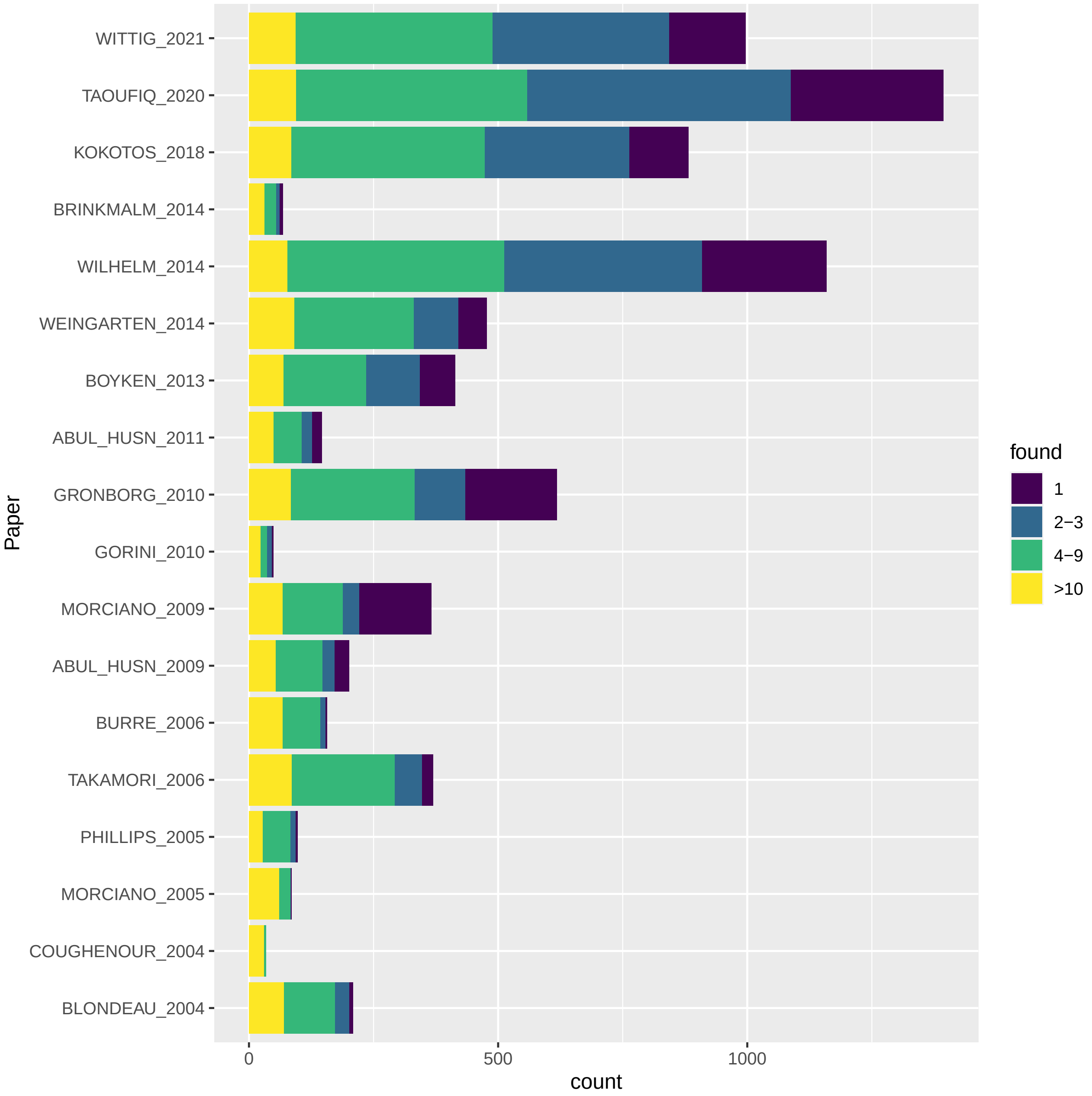
```
# presynaptic stats
presgp <- gp[gp$Localisation == "Presynaptic",]
svgp <- gp[gp$Localisation == "Synaptic_Vesicle",]
syngp <- gp[gp$Localisation == "Synaptosome",]
presg <- getGeneInfoByIDs(presgp$GeneID)
#mpres <- merge(presgp, presg, by = "GeneID")
mpres <- merge(presgp, presg, by = c("GeneID","Localisation"))
#mmpres <- mpres[, c(1,3,6, 10, 17, 18, 19)]
mmpres <- mpres[, c('GeneID','HumanEntrez.x','HumanName.x','Npmid','PaperPMID','Paper','Year')]
head(mmpres)
```

```
##   GeneID HumanEntrez.x HumanName.x Npmid PaperPMID           Paper Year
## 1      1          1742        DLG4     4  24534009 WEINGARTEN_2014 2014
## 2      1          1742        DLG4     4  30301801    KOKOTOS_2018 2018
## 3      1          1742        DLG4     4  24876496    WILHELM_2014 2014
## 4      1          1742        DLG4     4  23622064     BOYKEN_2013 2013
## 5     10         10458      BAIAP2     4  24534009 WEINGARTEN_2014 2014
## 6     10         10458      BAIAP2     4  24876496    WILHELM_2014 2014
```

```
prespap <- papers[papers$Localisation == "Presynaptic",]
mmmpres <- mmpres[mmpres$PaperPMID %in% prespap$PaperPMID,]
table(mmmpres$Npmid)
```

```
## 
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16 
## 1416 1172  923  828  667  518  476  398  318  255  261  148  137  131  141  136
```

```
mmmpres$found <- 0
for(i in 1:dim(mmmpres)[1]) {
  if (mmmpres$Npmid[i] == 1) {
    mmmpres$found[i] <- '1'
    } else if (mmmpres$Npmid[i] > 1 & mmmpres$Npmid[i] < 4) {
        mmmpres$found[i] <- '2-3'
    } else if (mmmpres$Npmid[i] >= 4 & mmmpres$Npmid[i] < 10) {
        mmmpres$found[i] <- '4-9'
    } else if (mmmpres$Npmid[i] >= 10) {
        mmmpres$found[i] <- '>10'
    }
}

mmmpres$found<- factor(mmmpres$found,levels = c('1','2-3','4-9','>10'),ordered=TRUE)
tp<-unique(mmmpres$Paper)
mmmpres$Paper<- factor(mmmpres$Paper,
                       levels =tp[order(as.numeric(sub('^[^0-9]+_([0-9]+)',
                                                       '\\1',tp)))],
                       ordered=TRUE)

ummpres<-unique(mmmpres[,c('GeneID','Paper','found')])
ggplot(ummpres) + geom_bar(aes(y = Paper, fill = found))
```



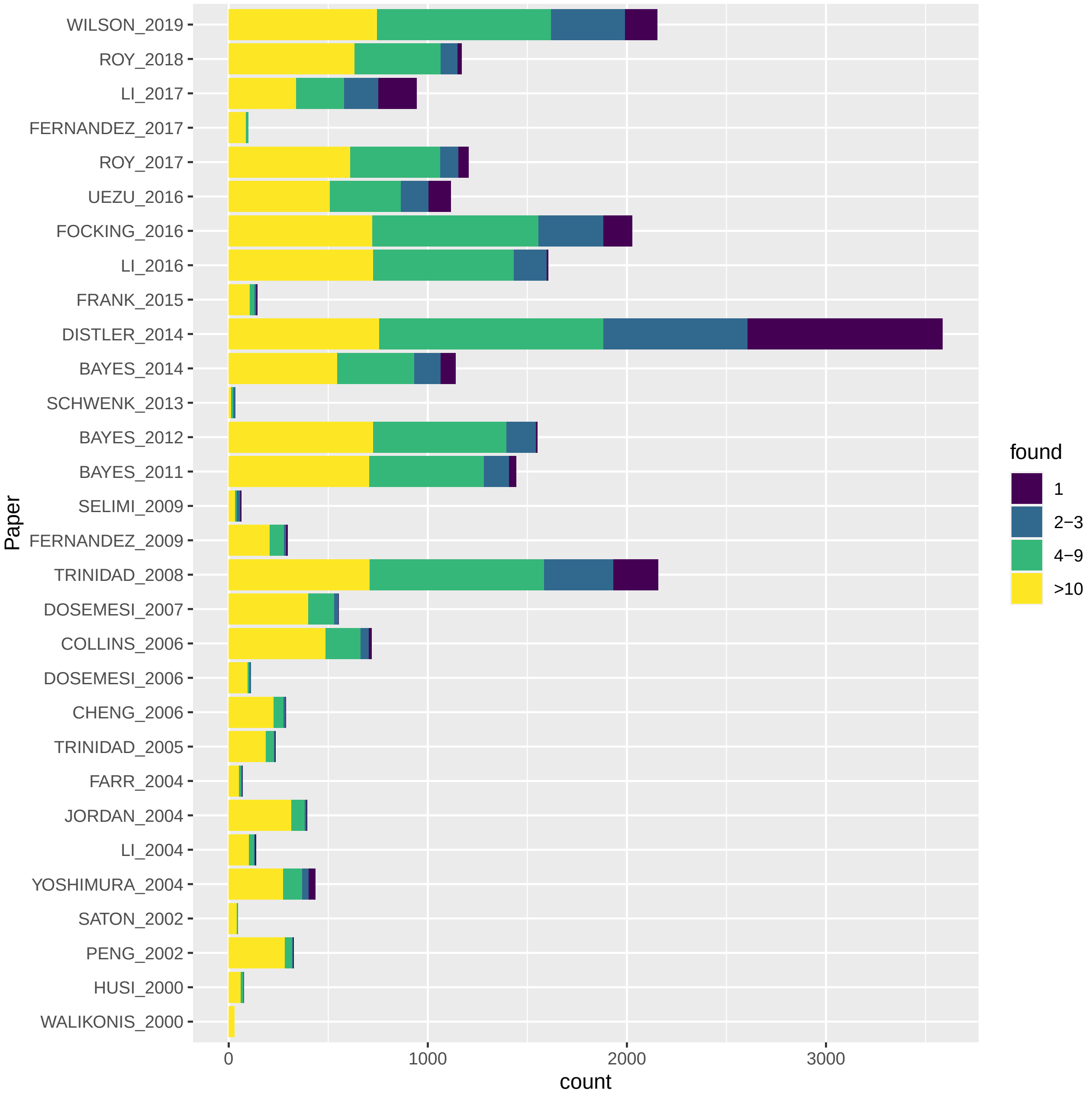
2 Postsynaptic

```
#postsynaptic stats

pstgp <- gp[gp$Localisation == "Postsynaptic",]
postg <- getGeneInfoByIDs(pstgp$GeneID)
#mpost <- merge(pstgp, postg, by = "GeneID")
mpost <- merge(pstgp, postg, by = c("GeneID","Localisation"))
#mmpost <- mpost[, c(1,3,6, 10, 17, 18, 19)]
mmpost <- mpost[, c('GeneID','HumanEntrez.x','HumanName.x','Npmid','PaperPMID','Paper','Year')]
postspap <- papers[papers$Localisation == "Postsynaptic",]
mmmpost <- mmpost[mmpost$PaperPMID %in% postspap$PaperPMID,]
```

```
table(mmmpost$Npmid)
```

```
##
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2820 2235 2120 2415 2090 2248 2118 2205 2114 2397 2026 1776 1917 1415 1507 1030
##   17   18   19   20   21   22   23   24   25   26   28   29
##  971  880  705  382  485  396  219  265   70  176   38   39
```

```
mmmpost$found <- 0
for(i in 1:dim(mmmpost)[1]) {
    if (mmmpost$Npmid[i] == 1) {
        mmmpost$found[i] <- '1'
    } else if (mmmpost$Npmid[i] > 1 & mmmpost$Npmid[i] < 4) {
        mmmpost$found[i] <- '2-3'
    } else if (mmmpost$Npmid[i] >= 4 & mmmpost$Npmid[i] < 10) {
        mmmpost$found[i] <- '4-9'
    } else if (mmmpost$Npmid[i] >= 10) {
        mmmpost$found[i] <- '>10'
    }
}

mmmpost$found<- factor(mmmpost$found,levels = c('1','2-3','4-9','>10'),ordered=TRUE)
tp<-unique(mmmpost$Paper)
mmmpost$Paper<- factor(mmmpost$Paper,
                       levels =tp[order(as.numeric(sub('^[^0-9]+_([0-9]+)',
                                                       '\\1',tp)))],
                       ordered=TRUE)

ummpos<-unique(mmmpost[,c('GeneID','Paper','found')])
ggplot(ummpos) + geom_bar(aes(y = Paper, fill = found))
```



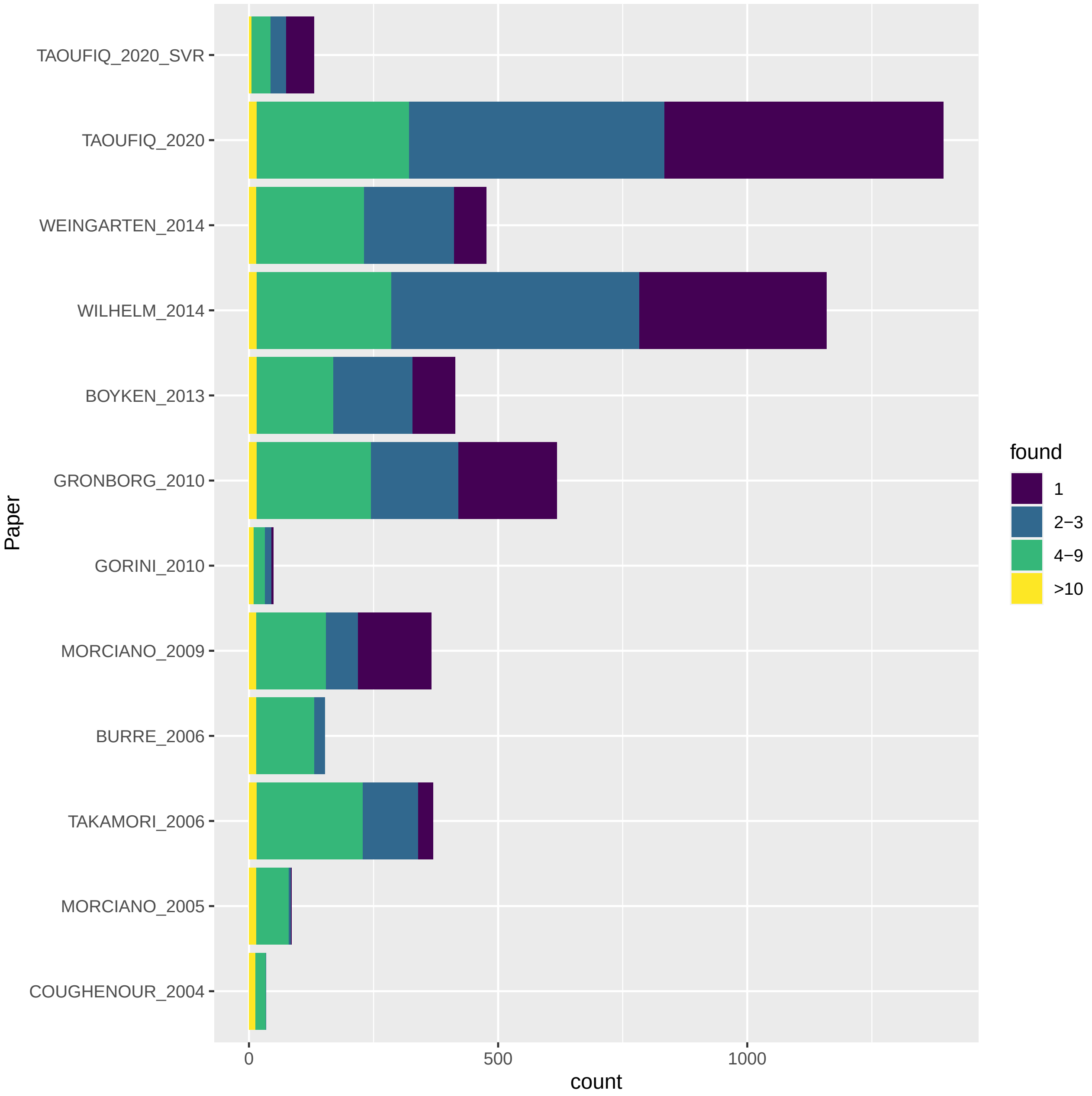
3 Synaptic Vesicle

```
#postsynaptic stats

svgp <- gp[gp$Localisation == "Synaptic_Vesicle",]
svg <- getGeneInfoByIDs(svgp$GeneID)
#mpost <- merge(pstgp, postg, by = "GeneID")
mpost <- merge(svgp, svg, by = c("GeneID","Localisation"))
mpost$Paper<-paste0(mpost$Paper,ifelse('FULL'==mpost$Dataset,'','_SVR'))
#mmpost <- mpost[, c(1,3,6, 10, 17, 18, 19)]
mmpost <- mpost[, c('GeneID','HumanEntrez.x','HumanName.x','Npmid','PaperPMID','Paper','Year')]
postspap <- papers[papers$Localisation == "Synaptic_Vesicle",]
```

```
mmmpost <- mmpost[mmpost$PaperPMID %in% postspap$PaperPMID,]

table(mmmpost$Npmid)
```

```
##
##    1    2    3    4    5    6    7    8    9   10   11
## 1527  974  795  540  379  309  208  193  166  124   34
```

```
mmmpost$found <- 0
for(i in 1:dim(mmmpost)[1]) {
    if (mmmpost$Npmid[i] == 1) {
        mmmpost$found[i] <- '1'
    } else if (mmmpost$Npmid[i] > 1 & mmmpost$Npmid[i] < 4) {
        mmmpost$found[i] <- '2-3'
    } else if (mmmpost$Npmid[i] >= 4 & mmmpost$Npmid[i] < 10) {
        mmmpost$found[i] <- '4-9'
    } else if (mmmpost$Npmid[i] >= 10) {
        mmmpost$found[i] <- '>10'
    }
}

mmmpost$found<- factor(mmmpost$found,levels = c('1','2-3','4-9','>10'),
                       ordered=TRUE)
tp<-unique(mmmpost$Paper)
mmmpost$Paper<- factor(mmmpost$Paper,
                       levels =tp[order(as.numeric(sub('^[^0-9]+_([0-9]+)_?.*',
                                                       '\\1',tp)))],
                       ordered=TRUE)

ummpos<-unique(mmmpost[,c('GeneID','Paper','found')])
ggplot(ummpos) + geom_bar(aes(y = Paper, fill = found))
```

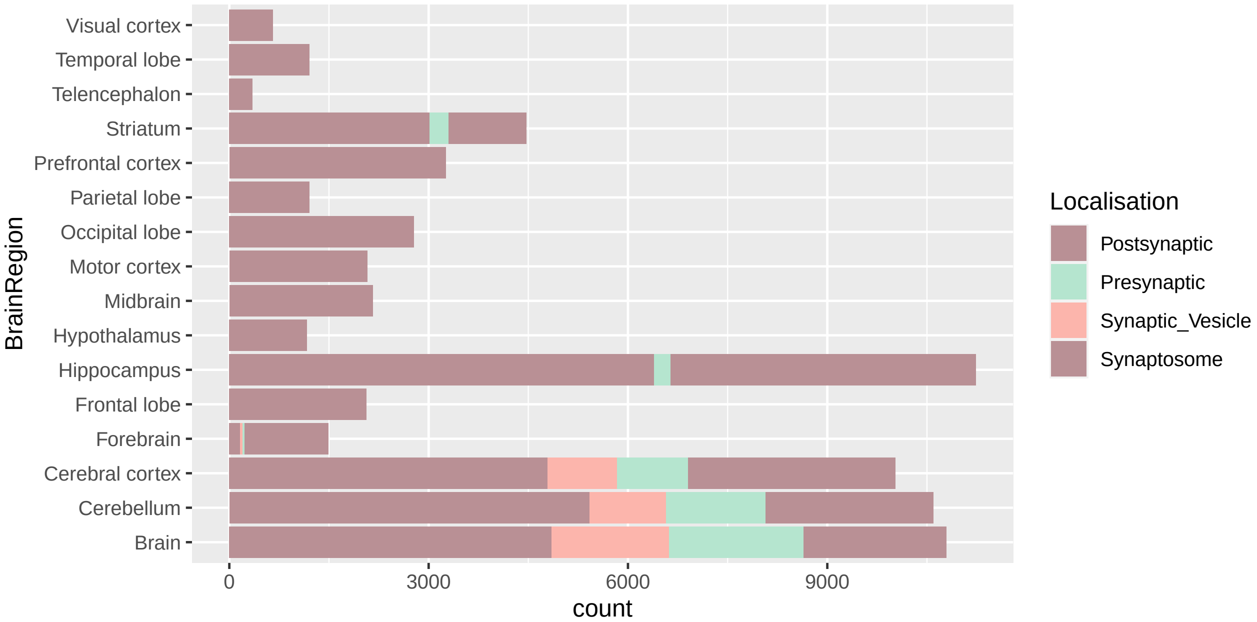


4 Brain region

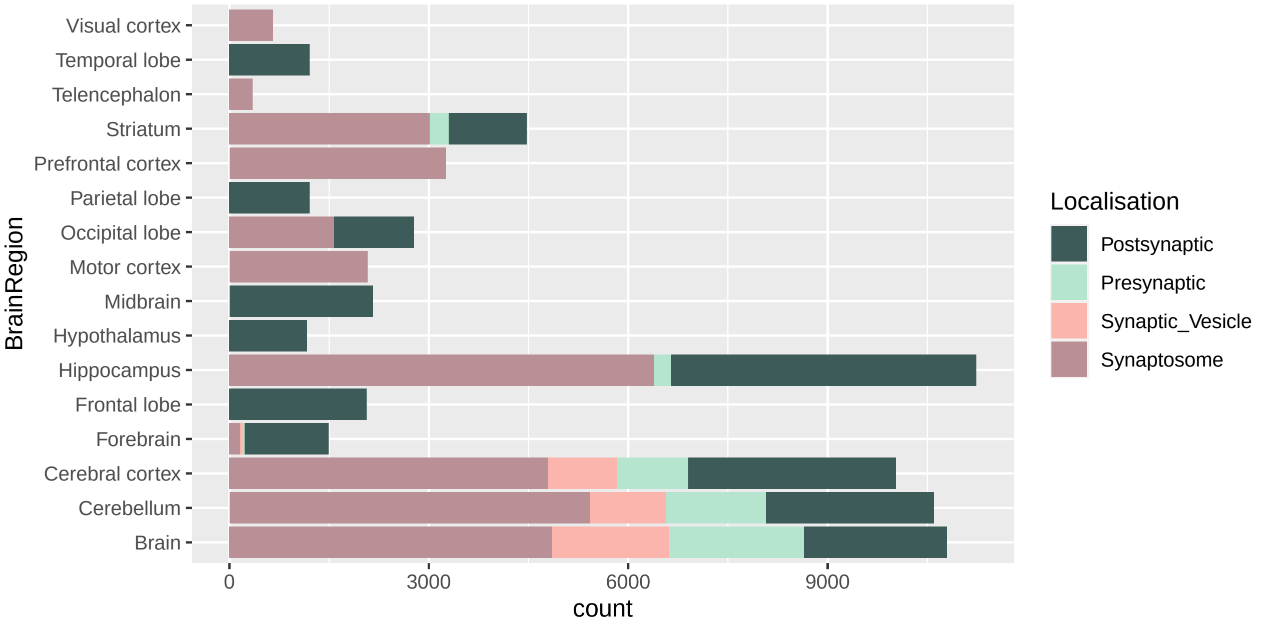
```
#region stats
totg <- getGeneInfoByIDs(gp$GeneID)
#mtot <- merge(gp, totg, by = "GeneID")
mtot <- merge(gp, totg, by = c("GeneID","Localisation"))
#mmptot <- mtot[, c(1,3,6, 9, 10, 18, 21)]
mmptot <- mtot[, c('GeneID','HumanEntrez.x','HumanName.x','Localisation','Npmid','Paper','BrainRegion')]
head(mmptot)
```

```
##   GeneID HumanEntrez.x HumanName.x Localisation Npmid           Paper
## 1      1          1742        DLG4 Postsynaptic    29 WALIKONIS_2000
## 2       1          1742       DLG4 Postsynaptic    29       HUSI_2000
## 3       1          1742       DLG4 Postsynaptic    29      SATON_2002
## 4       1          1742       DLG4 Postsynaptic    29         LI_2004
## 5       1          1742       DLG4 Postsynaptic    29 YOSHIMURA_2004
## 6       1          1742       DLG4 Postsynaptic    29       PENG_2002
##   BrainRegion
## 1   Forebrain
## 2   Forebrain
## 3   Forebrain
## 4   Forebrain
## 5   Forebrain
## 6   Forebrain
```

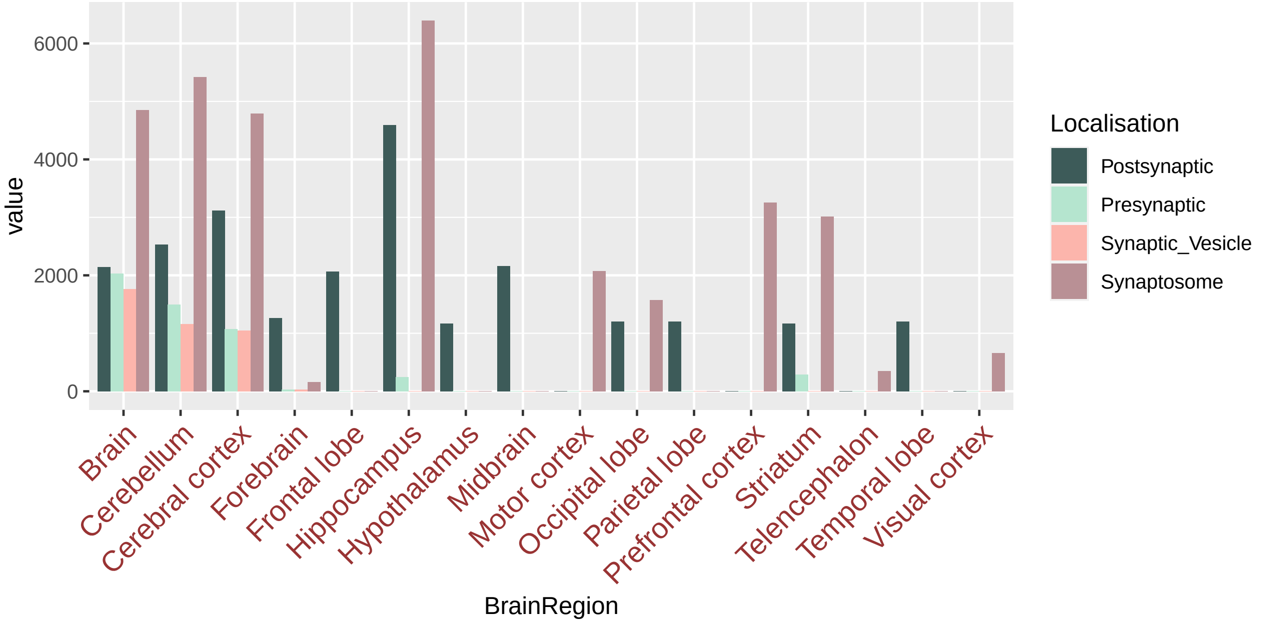
```
#untot<-unique(mmptot[,c('GeneID','Paper','BrainRegion','Localisation.x')])
untot<-unique(mmptot[,c('GeneID','BrainRegion','Localisation')])
#names(untot)
#names(untot)[4] <- "Localisation"
ggplot(untot) + geom_bar(aes(y = BrainRegion, fill = Localisation)) + scale_fill_manual(values = c("#B99095","#B5E5
```



```
ggplot(untot) + geom_bar(aes(y = BrainRegion, fill = Localisation)) + scale_fill_manual(values = c("#3D5B59","#B5E5
```



```
table(untot$Localisation,untot$BrainRegion)-> m
as.data.frame(m)->udf
names(udf)<-c('Localisation','BrainRegion','value')
ggplot(udf, aes(fill=Localisation, y=value, x=BrainRegion)) +
geom_bar(position="dodge", stat="identity")+ scale_fill_manual(values = c("#3D5B59","#B5E5CF","#FCB5AC", "#
```


value
Brain
Cerebellum
Cerebral cortex
Forebrain
Frontal lobe
Hippocampus
Hypothalamus
Midbrain
Motor cortex
Occipital lobe
Parietal lobe
Prefrontal cortex
Striatum
Telencephalon
Temporal lobe
Visual cortex
BrainRegion
Localisation
Postsynaptic
Presynaptic
Synaptic_Vesicle
Synaptosome