KEGG Pathway Drawing

Robert M Flight

The [final pathway](#final-pathway) that should be used is at the end of this document.

## Purpose

To generate interpretable graphs from KEGG Pathway maps.

## Data

Parsed KGML from KEGG for pathways for the MAPK, PI3K, RAS and EGFR pathways previously identified from pathway enrichment, as well as the differential kinases and their targets.

## Methods

knitr::opts\_chunk$set(warning = FALSE, message = FALSE, fig.width = 10, fig.height = 8, out.width= "70%")

library(tidygraph)  
library(dplyr)  
library(ggraph)  
library(org.Hs.eg.db)  
source(here::here("scripts", "functions.R"))  
theme\_set(cowplot::theme\_cowplot())

gene\_list\_files = dir(here::here("data", "inputs", "kea3"), pattern = "^z", full.names = TRUE)  
gene\_lists = purrr::map(gene\_list\_files, ~ scan(.x, character(), sep = "\n", quiet = TRUE, skip = 1))  
names(gene\_lists) = c("human\_ptk", "mouse\_ptk", "human\_stk", "mouse\_stk")  
  
human\_list = unique(unlist(gene\_lists[c("human\_ptk", "human\_stk")]))  
mouse\_list = unique(unlist(gene\_lists[c("mouse\_ptk", "mouse\_stk")]))  
  
specific\_list = scan(here::here("data", "inputs", "very\_specific\_genelist.txt"), what = character(), sep = "\n", quiet = TRUE)  
  
kea3\_files = dir(here::here("data", "inputs", "kea3"), full.names = TRUE, pattern = "gmt$")  
kea3\_files = grep("Cheng.KSIN|PTMsigDB|PhosD.All|STRING.bind", kea3\_files, value = TRUE)  
kea3\_data = purrr::map(kea3\_files, parse\_gmt)  
kea3\_df = purrr::map\_df(kea3\_data, function(in\_data){  
 purrr::imap\_dfr(in\_data, function(targets, kinase){  
 data.frame(from = kinase, to = targets)  
 })  
})  
  
kegg\_graphs = readRDS(here::here("data", "outputs", "rds\_files", "kegg\_pathway\_graphs.rds"))  
all\_entrez = keys(org.Hs.eg.db)  
org\_hs\_df = select(org.Hs.eg.db, keys = all\_entrez, columns = "SYMBOL")  
  
specific\_to\_collapsed = read.table(here::here("data", "inputs", "pathways", "full\_nodes\_collapsed.csv"), sep = "\t", header = TRUE)  
  
uka\_list = unique(c(human\_list, mouse\_list))

## Work With Full Graph

To start, we will work with the full graph and see what happens.

full\_symbol\_df = purrr::map\_df(kegg\_graphs, function(.x){  
 from\_translation = dplyr::left\_join(.x, org\_hs\_df, by = c("from" = "ENTREZID"))   
 to\_translation = dplyr::left\_join(.x, org\_hs\_df, by = c("to" = "ENTREZID"))   
 new\_graph = data.frame(from = from\_translation$SYMBOL, to = to\_translation$SYMBOL)  
 new\_graph  
}) %>%  
 unique()

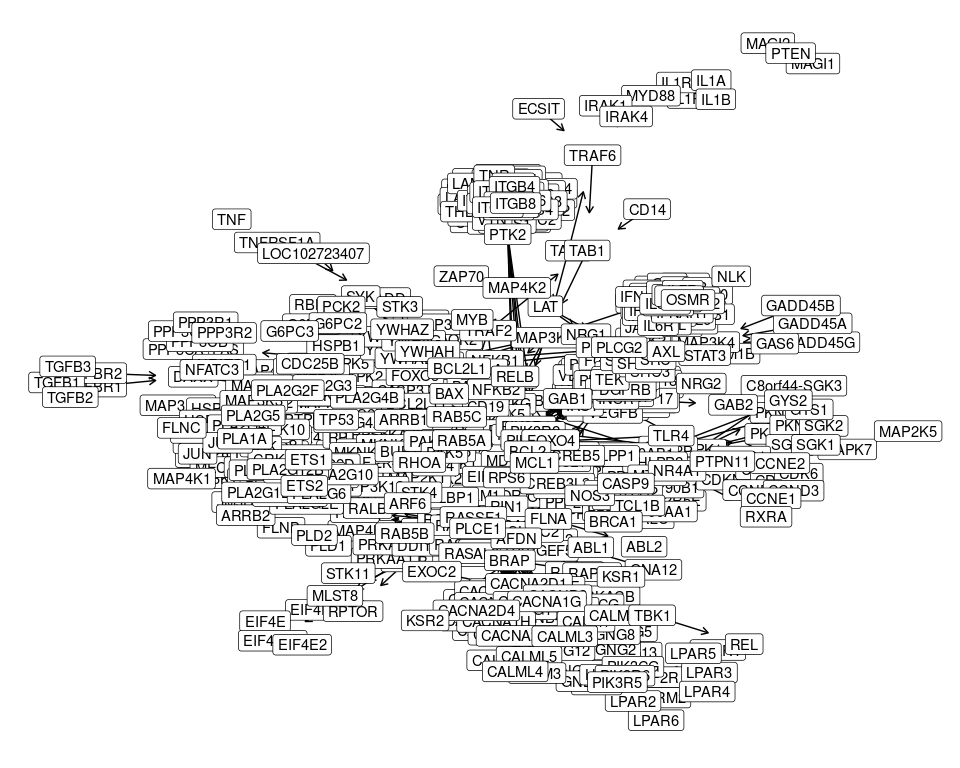
### Get From Significant and KEA3

Initially, we can grab the KEA3 nodes from both the kinases and their targets that match the peptide list from UKA.

kea3\_both = kea3\_df %>%  
 dplyr::filter((to %in% uka\_list) | (from %in% uka\_list)) %>%  
 unique() %>%  
 unlist() %>%  
 unique()

Now let’s filter to those in the KEGG list.

full\_symbol\_df\_kea3\_both = full\_symbol\_df %>%  
 dplyr::filter((to %in% kea3\_both) | (from %in% kea3\_both)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
ggraph(full\_symbol\_df\_kea3\_both, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name))

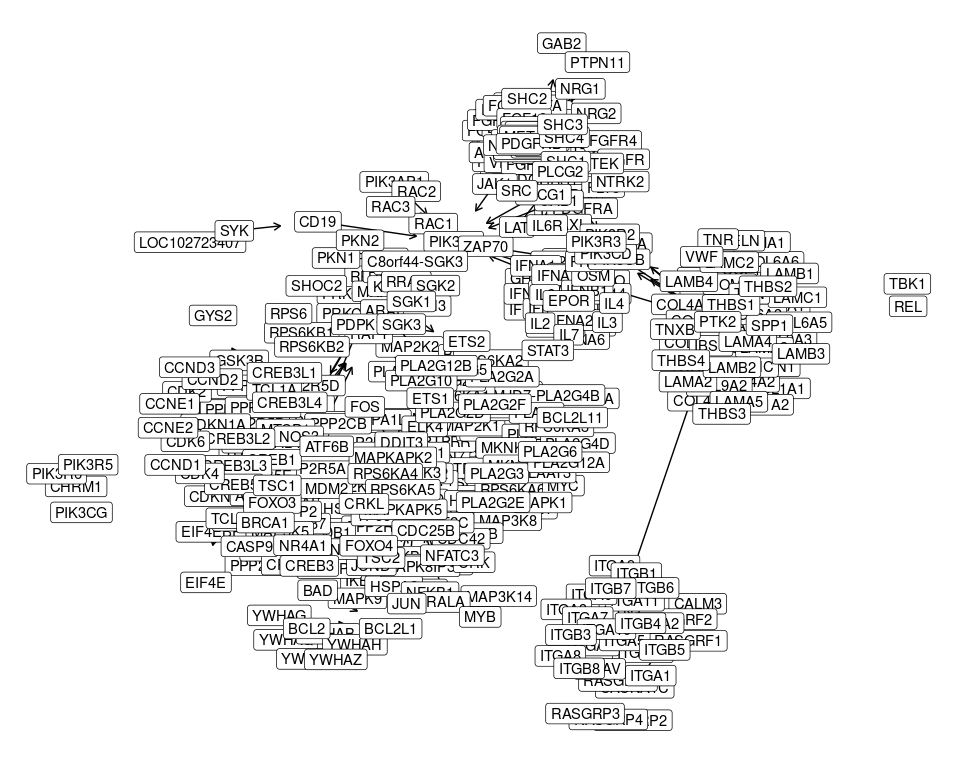


Aaannnddd, that’s just nasty. There is no way we are learning or demonstrating anything from that monstrosity.

### Significant Only

What if we use only the genes from UKA to filter the graph on?

full\_symbol\_df\_uka\_both = full\_symbol\_df %>%  
 dplyr::filter((to %in% uka\_list) | (from %in% uka\_list)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
ggraph(full\_symbol\_df\_uka\_both, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name))



So, this is better, but still not very interpretable.

## Collapsed Graph

**Part** of the problem seems to be genes that form large complexes. I actually spit out all the SYMBOLs in the graph in lexical order, and mapped them to **collapsed** SYMBOLs, where a \*\*"\*"\*\* indicates that it has been collapsed from multiple things.

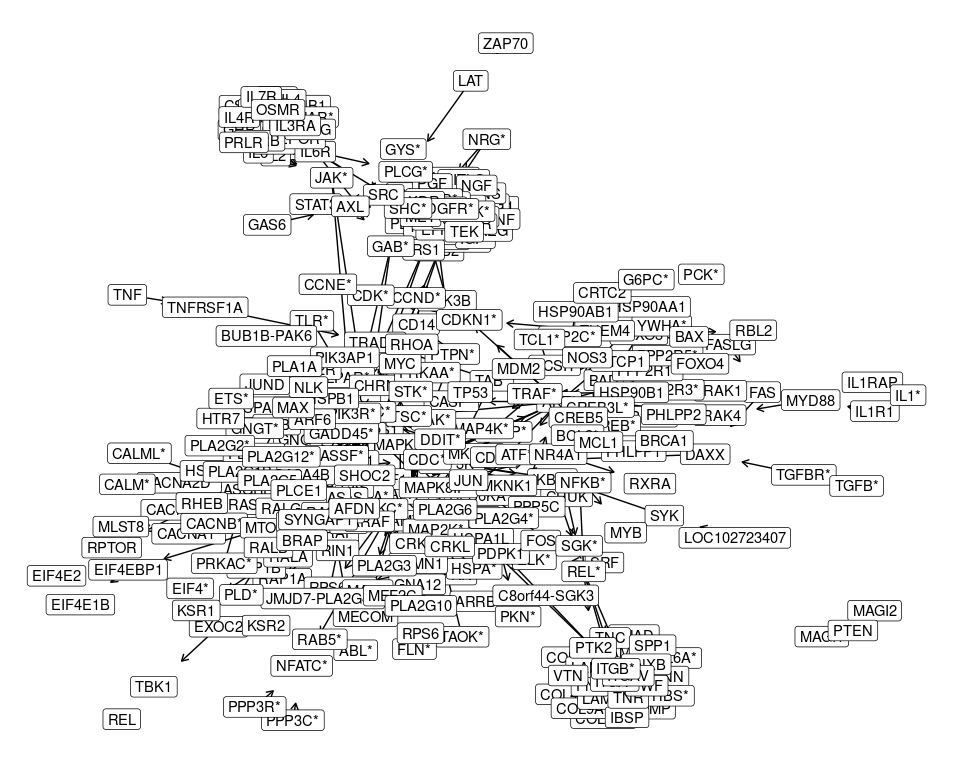
collapsed\_symbol\_df = purrr::map\_df(kegg\_graphs, function(.x){  
 from\_translation = dplyr::left\_join(.x, org\_hs\_df, by = c("from" = "ENTREZID")) %>%  
 dplyr::left\_join(., specific\_to\_collapsed, by = c("SYMBOL" = "name"))  
 to\_translation = dplyr::left\_join(.x, org\_hs\_df, by = c("to" = "ENTREZID")) %>%  
 dplyr::left\_join(., specific\_to\_collapsed, by = c("SYMBOL" = "name"))  
  
 new\_graph = data.frame(from = from\_translation$collapsed, to = to\_translation$collapsed)  
 new\_graph  
}) %>%  
 unique()  
  
nrow(collapsed\_symbol\_df) / nrow(full\_symbol\_df)

## [1] 0.2617419

Cool, we have 1/4 the edges we started with overall. Ideally, this should help things.

### KEA3 & Significant

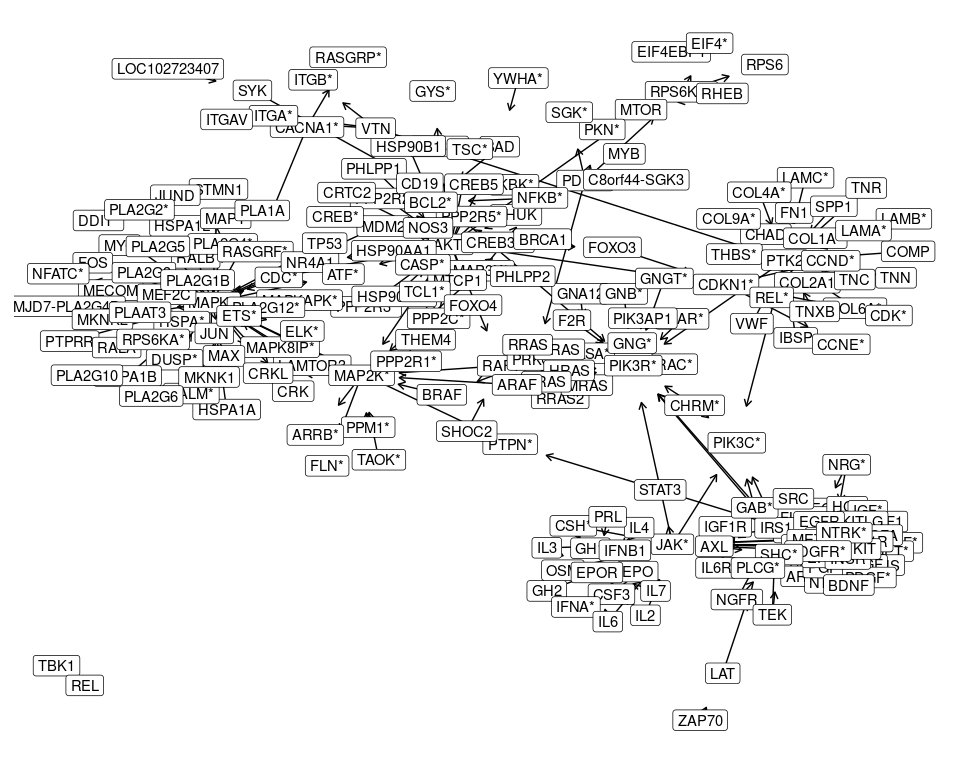
kea3\_2\_collapsed = dplyr::inner\_join(data.frame(name = kea3\_both),  
 specific\_to\_collapsed, by = "name") %>%  
 dplyr::pull(collapsed) %>%  
 unique()  
collapsed\_symbol\_df\_kea3\_both = collapsed\_symbol\_df %>%  
 dplyr::filter((to %in% kea3\_2\_collapsed) | (from %in% kea3\_2\_collapsed)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
ggraph(collapsed\_symbol\_df\_kea3\_both, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name))



Still **really** hard to interpret.

### Significant Only

uka\_2\_collapsed = dplyr::inner\_join(data.frame(name = uka\_list),  
 specific\_to\_collapsed, by = "name") %>%  
 dplyr::pull(collapsed) %>%  
 unique()  
collapsed\_symbol\_df\_uka = collapsed\_symbol\_df %>%  
 dplyr::filter((to %in% uka\_2\_collapsed) | (from %in% uka\_2\_collapsed)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
ggraph(collapsed\_symbol\_df\_uka, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name))



This still looks really painful to make sense of.

## Individual Pathways

One thing we’ve not talked about, is what the individual pathways look like.

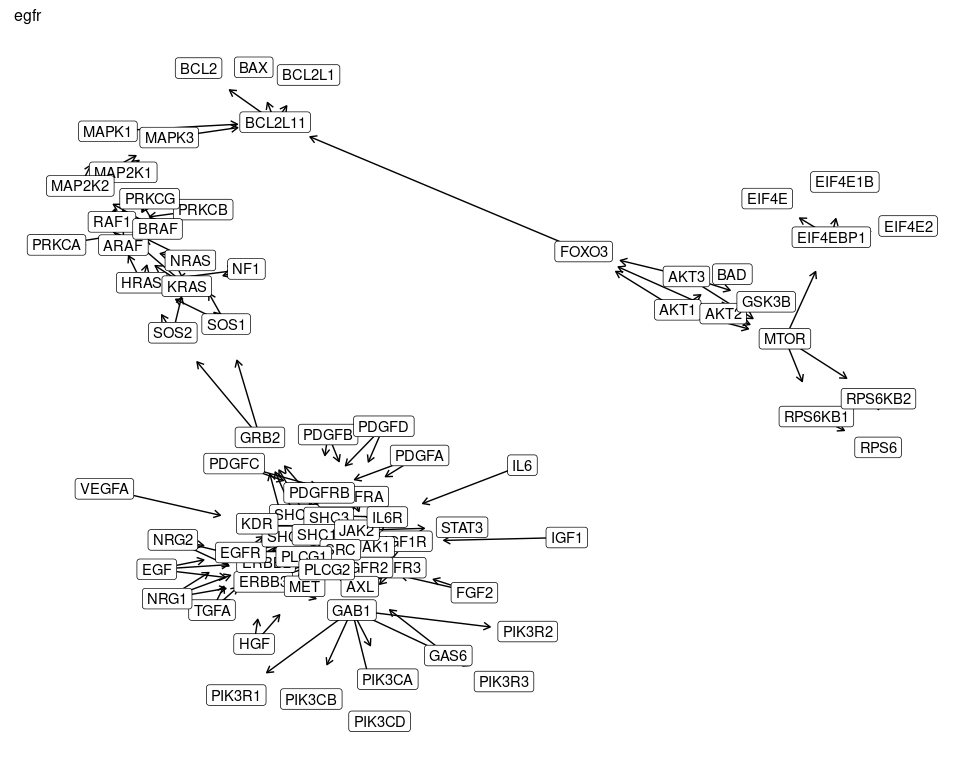
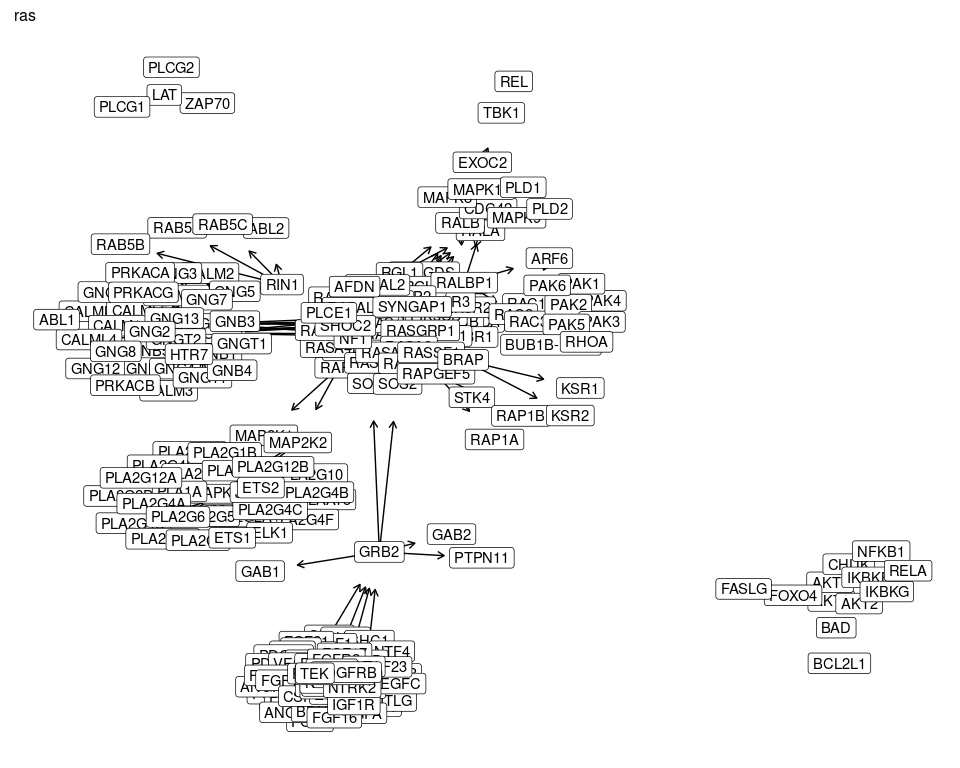
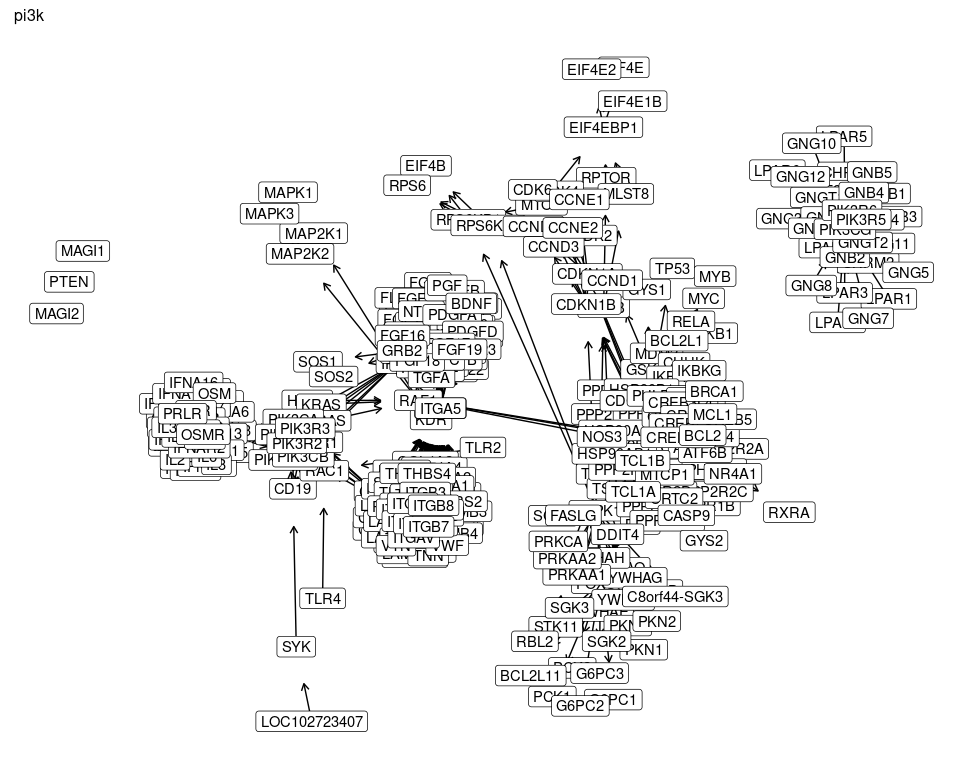
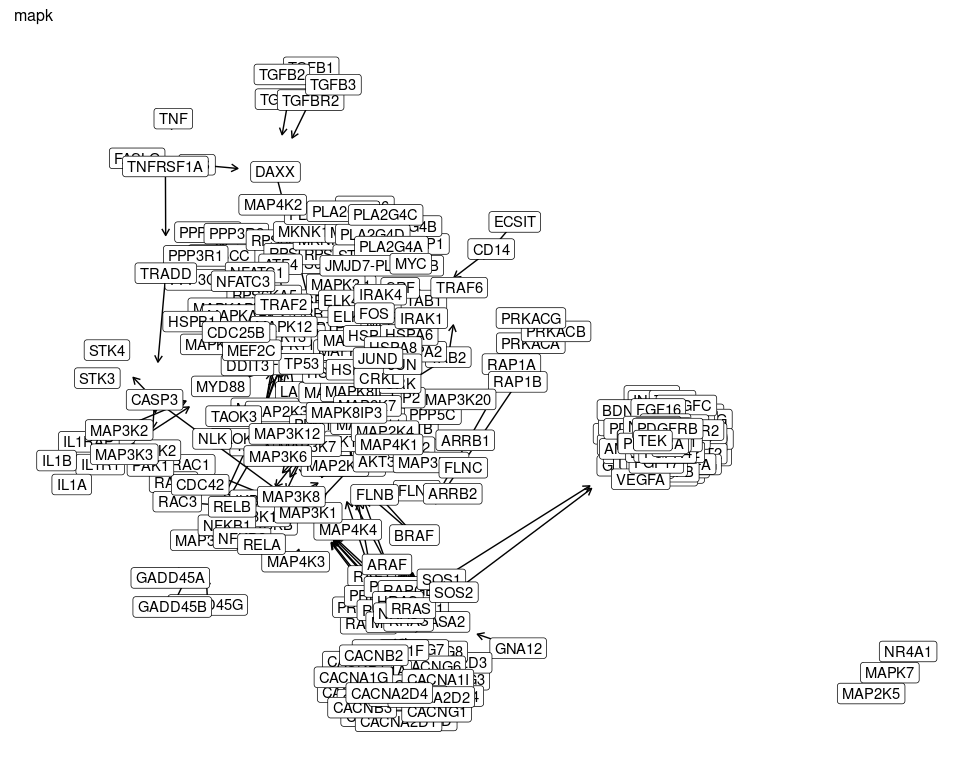
separate\_symbol\_df = purrr::map(kegg\_graphs, function(.x){  
 from\_translation = dplyr::left\_join(.x, org\_hs\_df, by = c("from" = "ENTREZID"))   
 to\_translation = dplyr::left\_join(.x, org\_hs\_df, by = c("to" = "ENTREZID"))   
 new\_graph = data.frame(from = from\_translation$SYMBOL, to = to\_translation$SYMBOL)  
 unique(new\_graph)  
})  
  
collapsed\_separate\_df = purrr::map(separate\_symbol\_df, function(.x){  
 from\_translation = dplyr::left\_join(.x, specific\_to\_collapsed, by = c("from" = "name"))  
 to\_translation = dplyr::left\_join(.x, specific\_to\_collapsed, by = c("to" = "name"))  
 new\_graph = data.frame(from = from\_translation$collapsed,  
 to = to\_translation$collapsed) %>%  
 unique()  
 new\_graph  
})  
  
# how big are these things?  
purrr::map(separate\_symbol\_df, nrow)

## $mapk  
## [1] 1969  
##   
## $pi3k  
## [1] 3092  
##   
## $ras  
## [1] 1571  
##   
## $egfr  
## [1] 230

Wow! Some of these are rather large, 2K and 3K for MAPK and PI3K, respectively. So maybe there is something to be gained by plotting them individually.

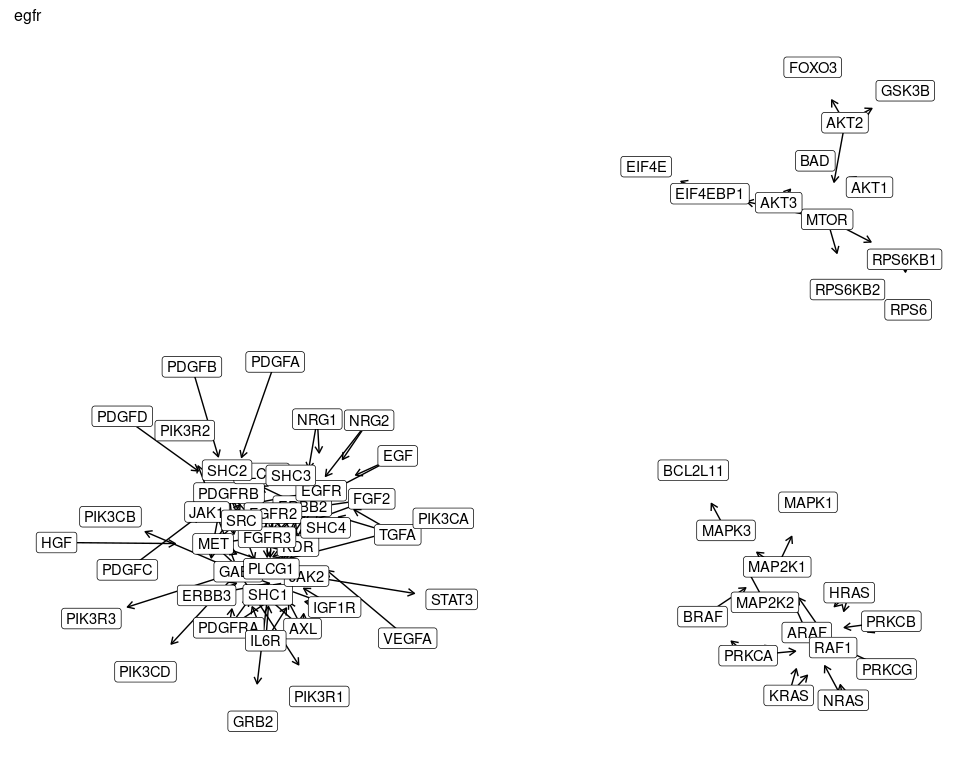
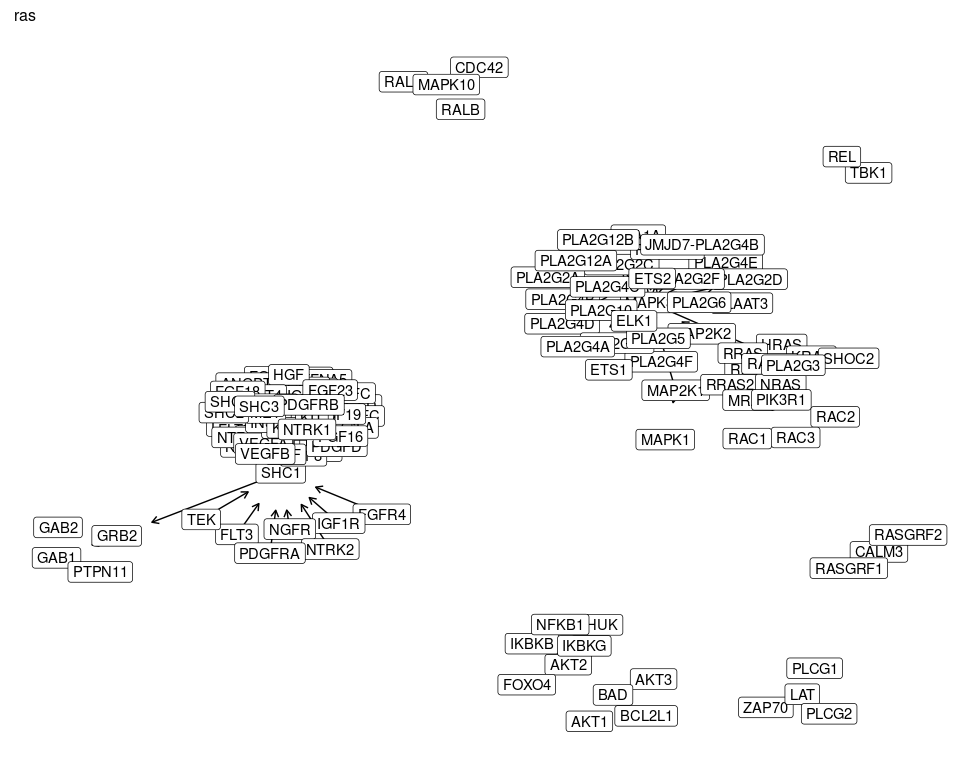
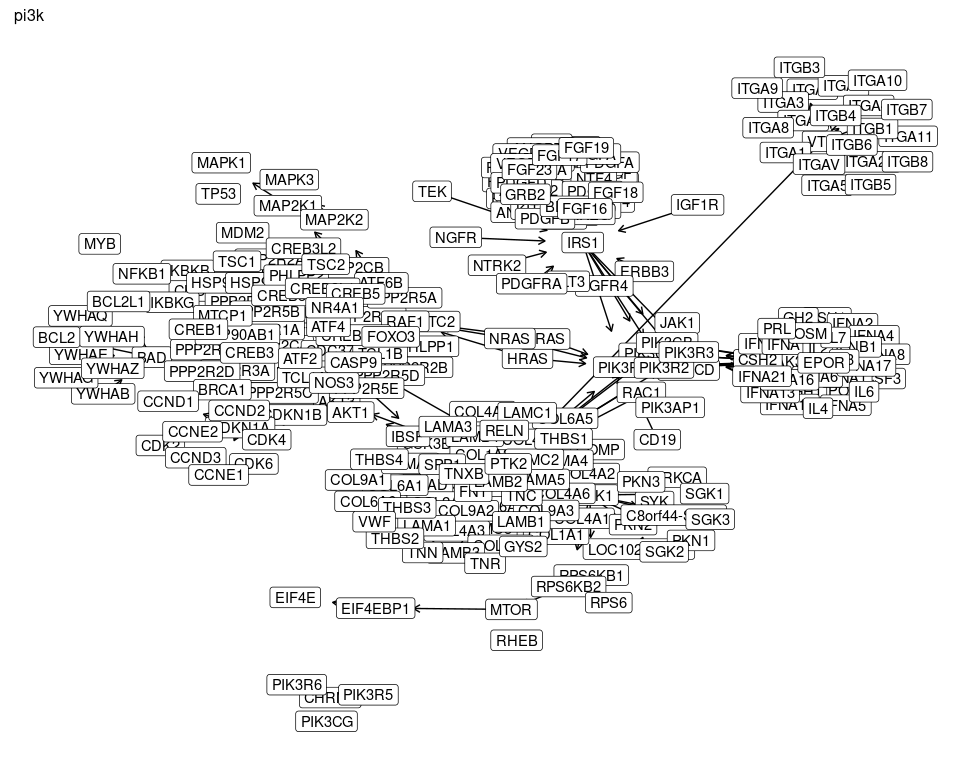
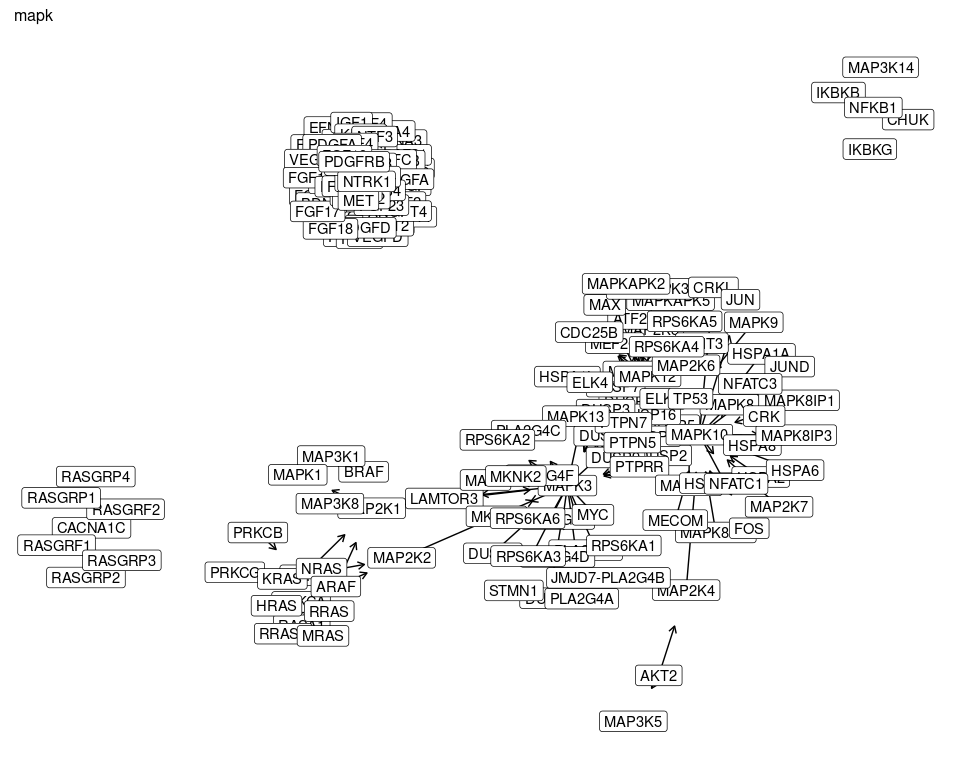
### Full KEA3 & UKA

purrr::iwalk(separate\_symbol\_df, function(graph, id){  
 graph\_uka = graph %>%  
 dplyr::filter((to %in% kea3\_both) | (from %in% kea3\_both)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
print(ggraph(graph\_uka, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name)) +  
 labs(subtitle = id))  
})



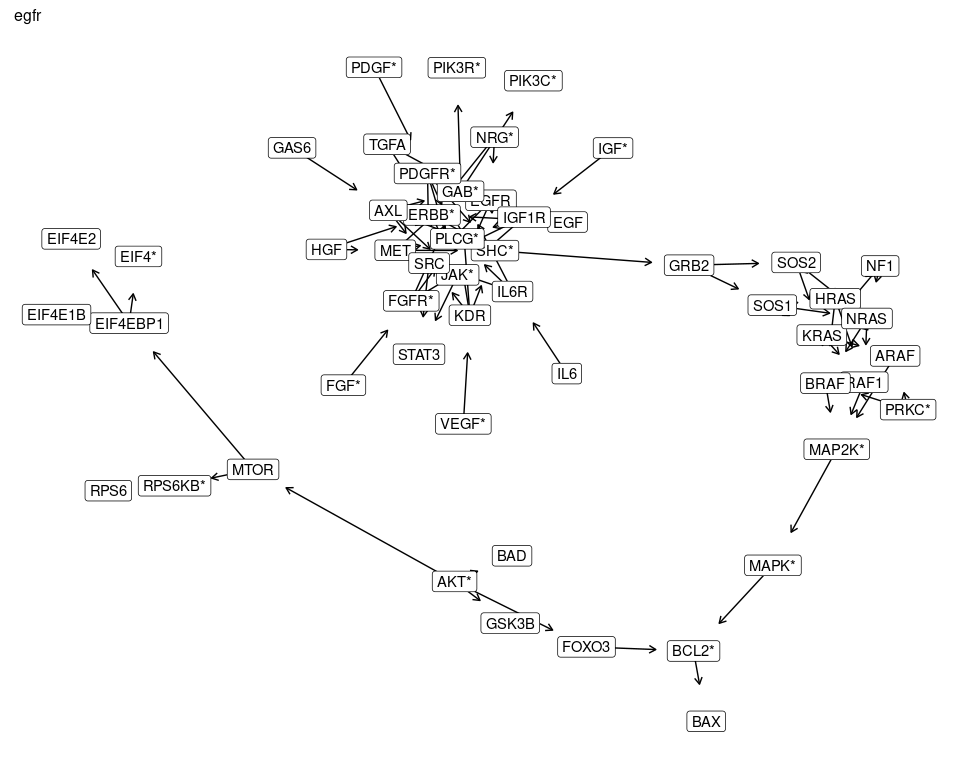
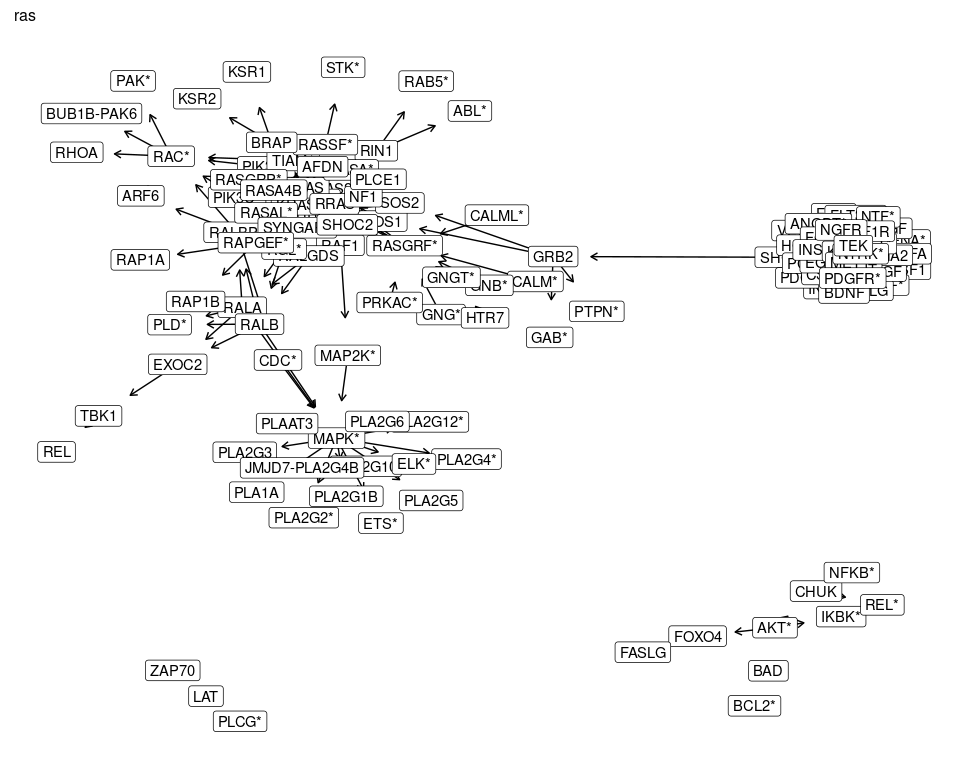
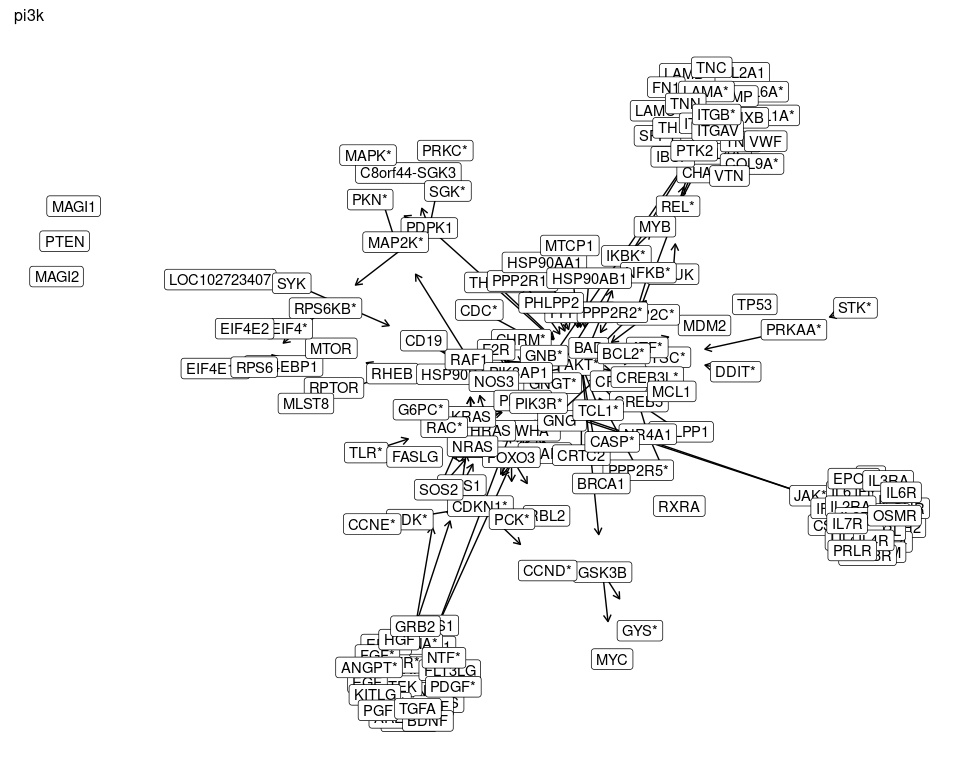
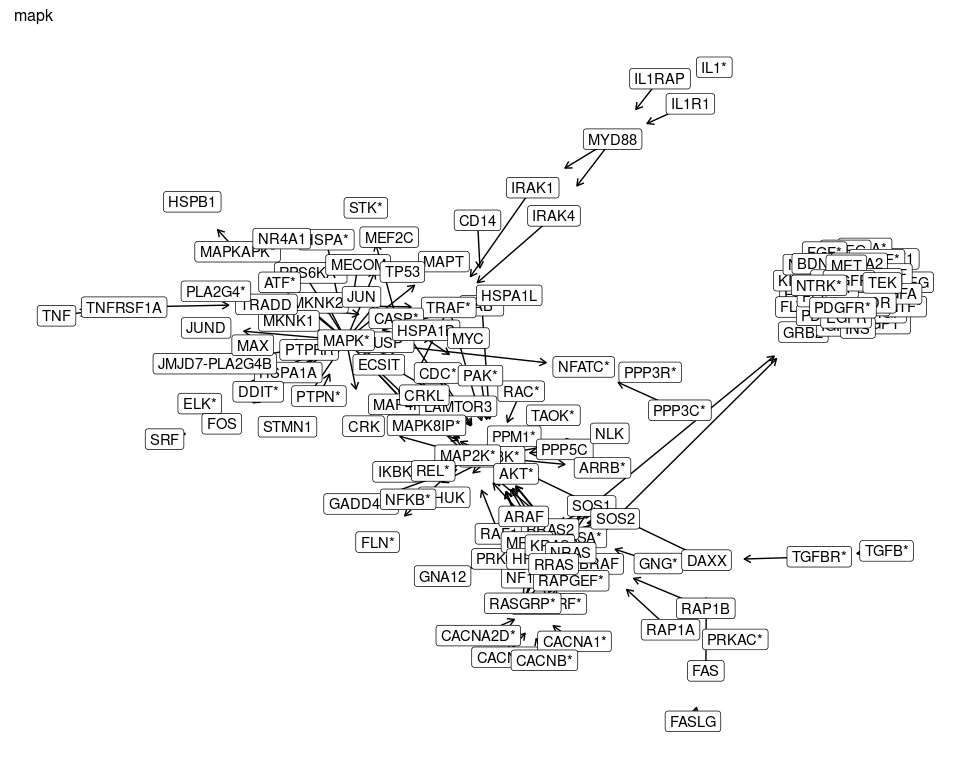
### Full UKA Only

purrr::iwalk(separate\_symbol\_df, function(graph, id){  
 graph\_uka = graph %>%  
 dplyr::filter((to %in% uka\_list) | (from %in% uka\_list)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
print(ggraph(graph\_uka, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name)) +  
 labs(subtitle = id))  
})



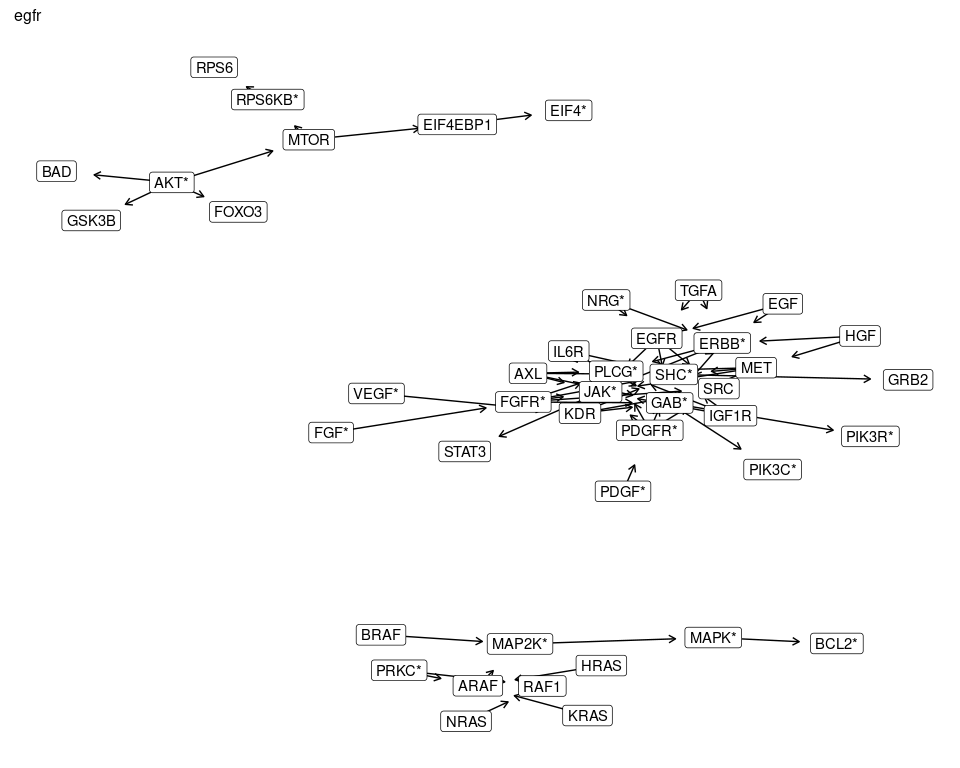
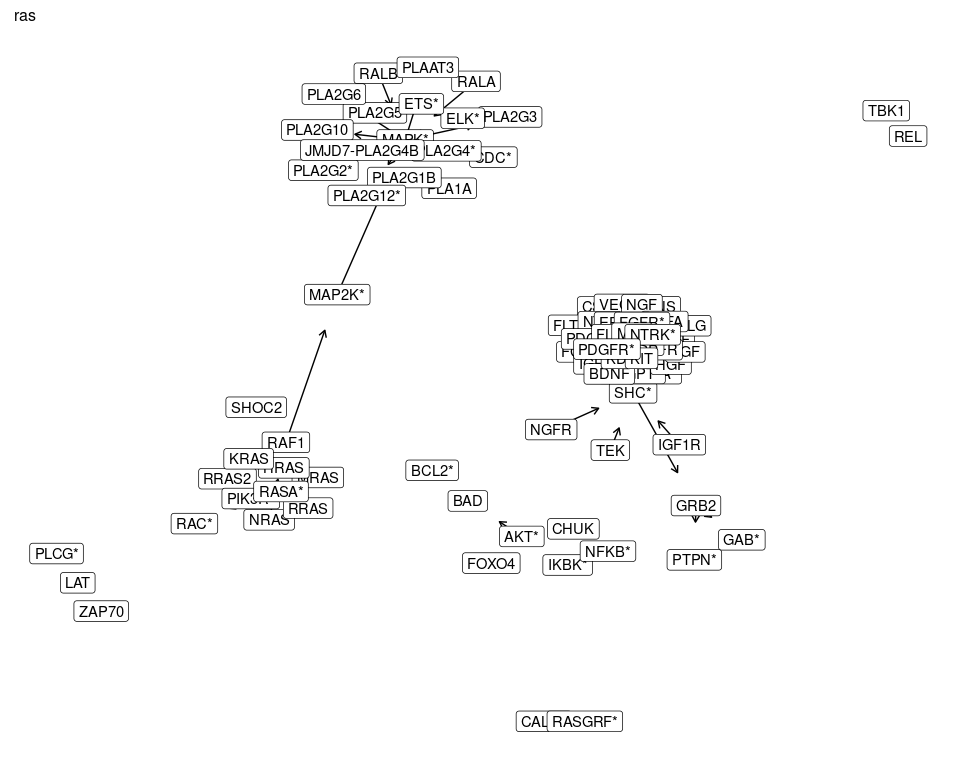
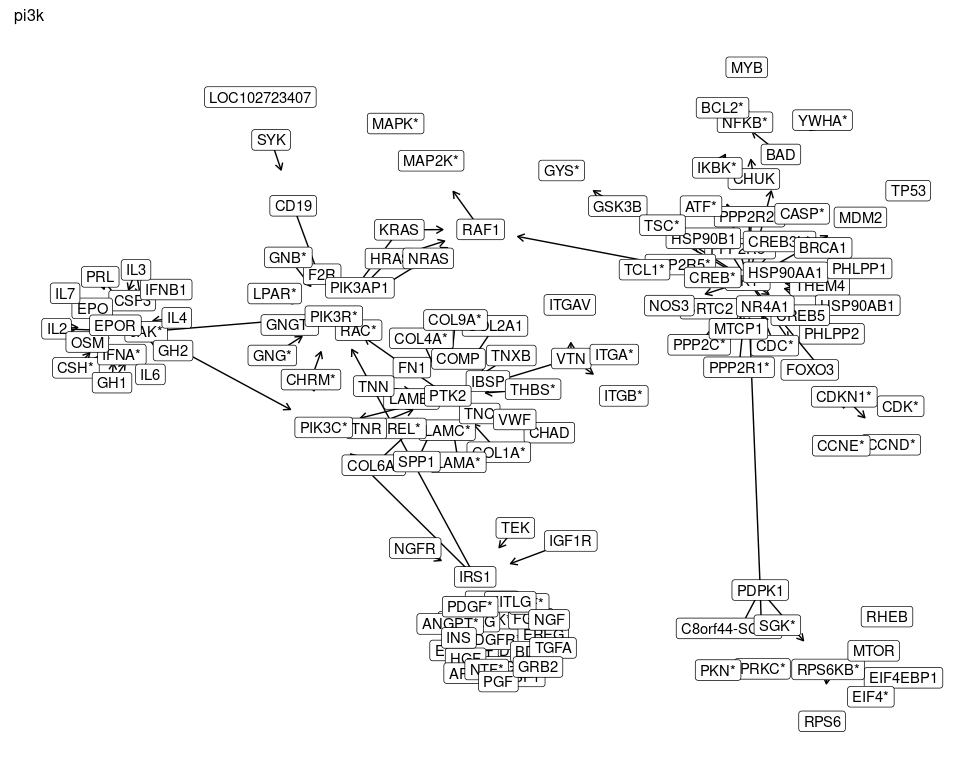
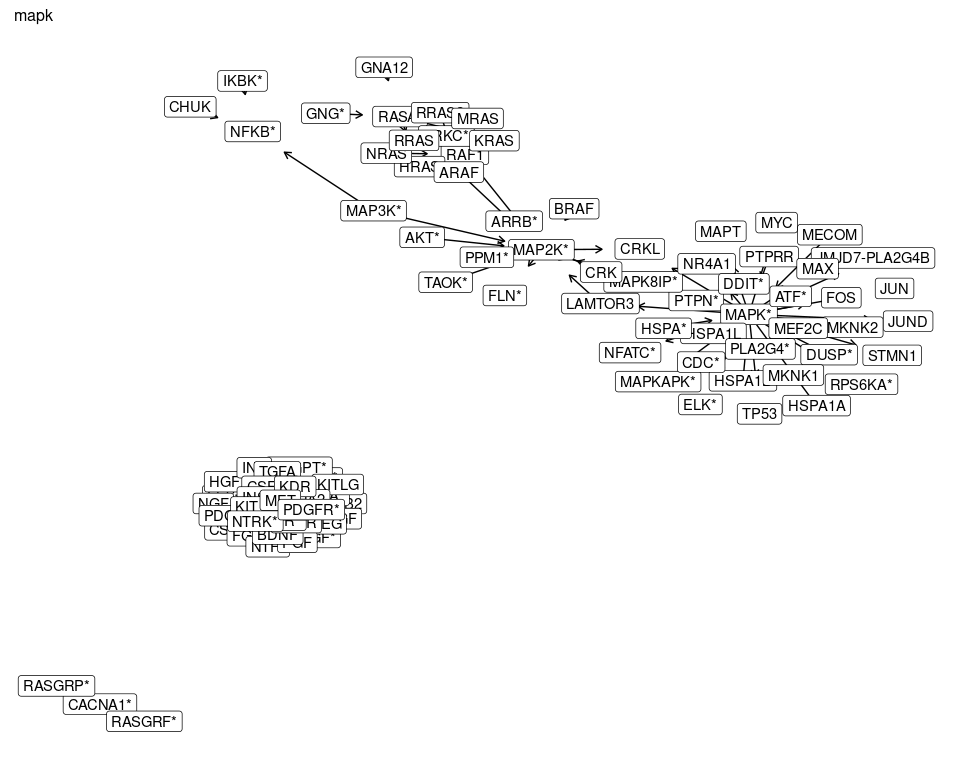
### Collapsed KEA3 & UKA

purrr::iwalk(collapsed\_separate\_df, function(graph, id){  
 graph\_kea3 = graph %>%  
 dplyr::filter((to %in% kea3\_2\_collapsed) | (from %in% kea3\_2\_collapsed)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
print(ggraph(graph\_kea3, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name)) +  
 labs(subtitle = id))  
})



### Collapsed UKA Only

purrr::iwalk(collapsed\_separate\_df, function(graph, id){  
 graph\_uka = graph %>%  
 dplyr::filter((to %in% uka\_2\_collapsed) | (from %in% uka\_2\_collapsed)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
print(ggraph(graph\_uka, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name)) +  
 labs(subtitle = id))  
})



## Common Nodes

Instead of combining the **full** graphs, what if we combined them where at least one member of an edge had to be in more than **three** of the graphs?

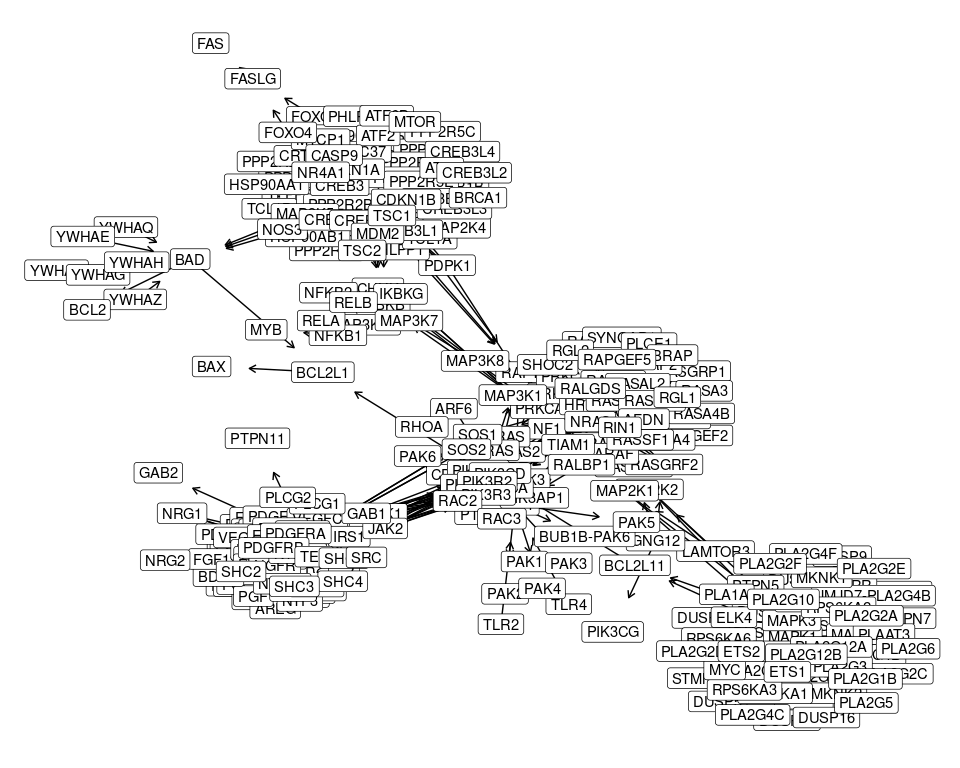
all\_nodes = data.frame(nodes = unique(unlist(full\_symbol\_df)), n\_path = 0)  
  
for (ipath in separate\_symbol\_df) {  
 ipath\_nodes = unique(unlist(ipath))  
 all\_nodes[all\_nodes$nodes %in% ipath\_nodes, "n\_path"] = all\_nodes[all\_nodes$nodes %in% ipath\_nodes, "n\_path"] + 1  
}  
  
multi\_nodes = all\_nodes %>%  
 dplyr::filter(n\_path > 2)  
nrow(multi\_nodes)

## [1] 103

multi\_symbol\_df = full\_symbol\_df %>%  
 dplyr::filter((from %in% multi\_nodes$nodes) | (to %in% multi\_nodes$nodes))

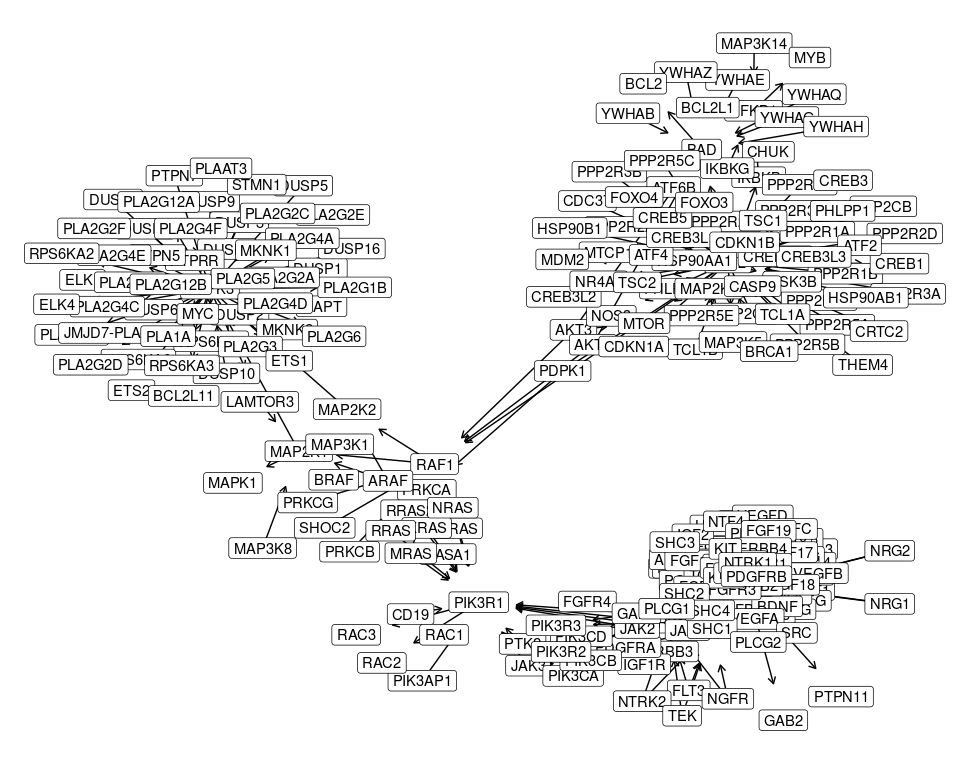
### KEA3 & UKA

multi\_kea3 = multi\_symbol\_df %>%  
 dplyr::filter((from %in% kea3\_both) | (to %in% kea3\_both)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
ggraph(multi\_kea3, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name))



### UKA Only

multi\_uka = multi\_symbol\_df %>%  
 dplyr::filter((from %in% uka\_list) | (to %in% uka\_list)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
ggraph(multi\_uka, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name))



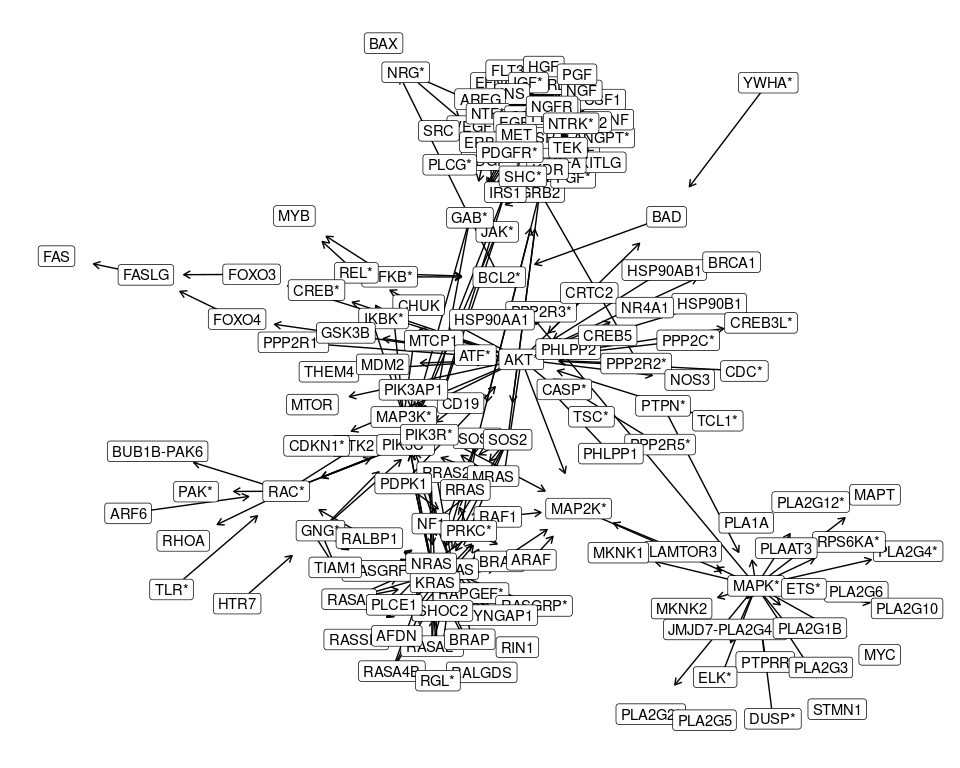
## Common Nodes Collapsed

And now we try collapsing them again, because the graphs are still really, really big and hard to interpret.

multi\_collapsed\_df = data.frame(from =   
 dplyr::left\_join(multi\_symbol\_df, specific\_to\_collapsed, by = c("from" = "name")) %>% dplyr::pull(collapsed),  
 to = dplyr::left\_join(multi\_symbol\_df, specific\_to\_collapsed, by = c("to" = "name")) %>% dplyr::pull(collapsed)) %>%  
 unique()

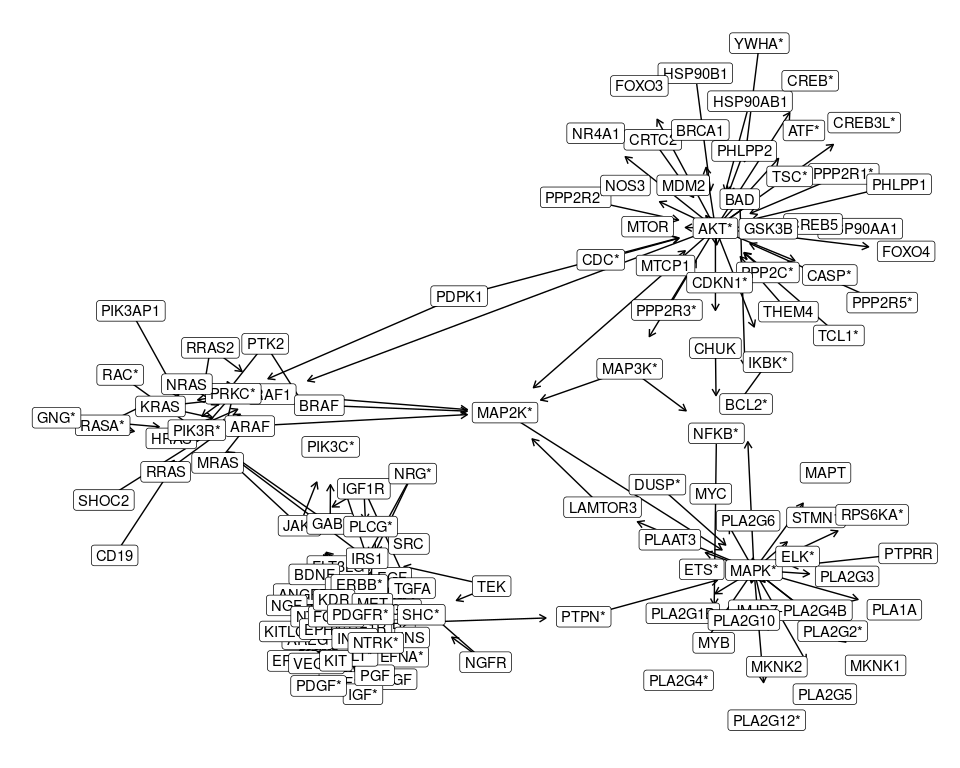
### KEA3 & UKA

multi\_collapsed\_kea3 = multi\_collapsed\_df %>%  
 dplyr::filter((from %in% kea3\_2\_collapsed) | (to %in% kea3\_2\_collapsed)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
ggraph(multi\_collapsed\_kea3, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name))



### UKA

multi\_collapsed\_uka = multi\_collapsed\_df %>%  
 dplyr::filter((from %in% uka\_2\_collapsed) | (to %in% uka\_2\_collapsed)) %>%  
 unique() %>%  
 tidygraph::as\_tbl\_graph()  
set.seed(1234)  
ggraph(multi\_collapsed\_uka, "graphopt") +  
geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name))



OK, now this finally looks **almost** interpretable. If we can break off that group on the left and plot it separately, we could probably make even more sense of it.

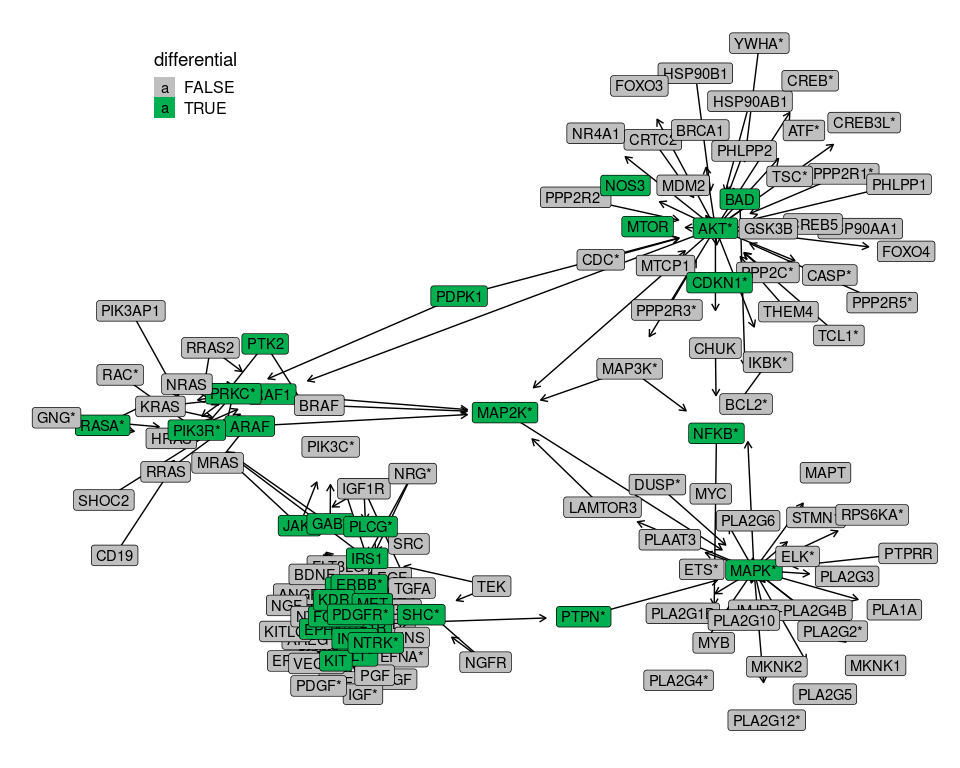
Let’s see where the UKA peptides are within this network.

## Final Pathway

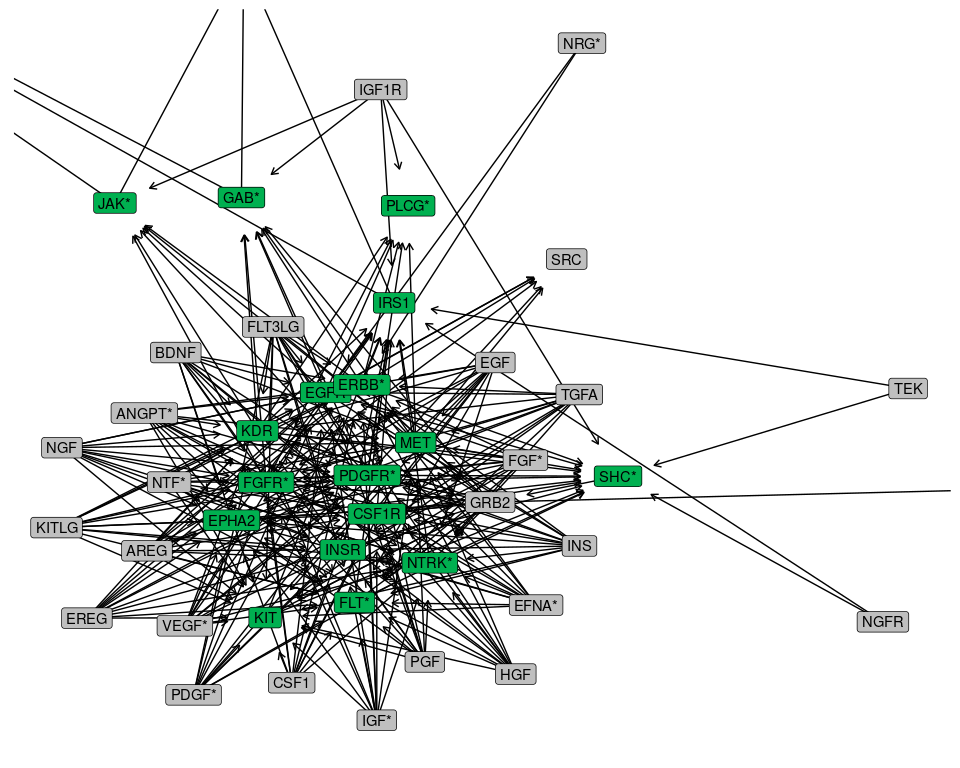
diff\_colors = c("FALSE" = "#BFBFBF", "TRUE" = "#00B050")  
multi\_collapsed\_uka = multi\_collapsed\_uka %>%  
 activate(nodes) %>%  
 dplyr::mutate(differential = name %in% uka\_2\_collapsed)

set.seed(1234)  
mcu\_layout = create\_layout(multi\_collapsed\_uka, "graphopt")  
  
walk\_membership = igraph::cluster\_walktrap(multi\_collapsed\_uka)  
walk\_communities = igraph::membership(walk\_membership)  
split\_comms = split(names(walk\_communities), walk\_communities)  
names(split\_comms) = NULL  
  
which\_comm = split\_comms[[which(purrr::map\_lgl(split\_comms, ~ "ERBB\*" %in% .x))]]  
  
erbb\_layout = mcu\_layout %>%  
 dplyr::filter(name %in% which\_comm)  
erbb\_xlim = range(erbb\_layout$x)  
erbb\_ylim = range(erbb\_layout$y)

full\_graph = ggraph(mcu\_layout) +  
 geom\_edge\_link(arrow = arrow(length = unit(2, 'mm')),  
 end\_cap = circle(10, "mm")) +  
 geom\_node\_label(aes(label = name, fill = differential)) +  
 scale\_fill\_manual(values = diff\_colors) +  
 theme(legend.position = c(0.15, 0.9))  
full\_graph



erbb\_graph = full\_graph +  
 coord\_cartesian(xlim = erbb\_xlim, ylim = erbb\_ylim, expand = TRUE, clip = "on") +  
 theme(legend.position = "none")  
erbb\_graph



knitr::kable(purrr::map\_df(split\_comms, function(in\_comm){  
 purrr::imap\_dfc(collapsed\_separate\_df, function(in\_path, path\_id){  
 all\_path = unique(unlist(in\_path))  
 tmp = data.frame(perc = sum(in\_comm %in% all\_path) / length(in\_comm))  
 names(tmp) = path\_id  
 tmp  
 })  
}), digits = 2)

|  |  |  |  |
| --- | --- | --- | --- |
| mapk | pi3k | ras | egfr |
| 0.23 | 0.95 | 0.21 | 0.15 |
| 0.68 | 0.64 | 0.64 | 0.45 |
| 0.83 | 0.88 | 0.83 | 0.50 |
| 0.57 | 0.09 | 0.65 | 0.04 |

I don’t think we have exact pathways, unfortunately.

Cairo::CairoPNG(  
 filename = here::here("data", "outputs", "figures", "full\_kegg\_pathway.png"),  
 width = 10, height = 10, res = 300, bg = "white", units = "in"  
)  
full\_graph  
dev.off()

## png   
## 2

Cairo::CairoPNG(  
 filename = here::here("data", "outputs", "figures", "erbb\_kegg\_pathway.png"),  
 width = 10, height = 10, res = 300, bg = "white", units = "in"  
)  
erbb\_graph  
dev.off()

## png   
## 2

#plot\_list = list(collapsed\_uka = full\_graph,  
# sub\_uka = erbb\_graph)  
#export\_plots(plot\_list, "kegg\_pathway\_plot.pptx")

Figure X. Combined KEGG pathways from MAPK, PI3K, RAS and EGFR. Gene nodes were kept if they formed edges with genes in at least three of the four pathways and genes that were noted as differential peptides from human and mouse. Genes from complexes or where multiple genes have very similar symbols were collapsed to single entries, noted with an asterisk (\*). Genes from the differential peptides list are blue.

## Methods

Downloaded KEGG KGML files for each pathway (MAPK, PI3K, RAS and EGFR) using KEGGREST (Tenenbaum and Maintainer 2021) and extracted all gene product - gene product edges from the pathway using KEGGgraph (Zhang and Wiemann 2009; Zhang 2021a, 2021b). Converted Entrez identifiers for each node to Symbols using org.Hs.eg.db. To deal with the many subunits for some protein complexes, collapsed or group identifier was created by hand where possible to reduce the number of edges and nodes. A count of how many of the pathways a gene product was present in was generated, and a subset of “core” gene products that are present in at least three of the four pathways kept. Those gene products that form an edge with the core ones were also kept as an initial network, discarding all others. Finally, from that network, the differential gene products and their partners were extracted and plotted.

## References

Tenenbaum, Dan, and Bioconductor Package Maintainer. 2021. *KEGGREST: Client-Side REST Access to the Kyoto Encyclopedia of Genes and Genomes (KEGG)* (version 1.34.0).

Zhang, Jitao David. 2021a. *KEGGgraph* (version 1.54.0).

———. 2021b. “KEGGgraph: Application Examples.” Manual.

Zhang, Jitao David, and Stefan Wiemann. 2009. “KEGGgraph: A Graph Approach to KEGG PATHWAY in r and Bioconductor.” *Bioinformatics (Oxford, England)*, no. 11: 1470–71.