Widespread rewiring of genetic networks upon cancer signaling pathway activation

Maximilian Billmann, Varun Chaudhary, Mostafa ElMagraphy, Bernd Fischer and Michael Boutros

September 26, 2017

Contents

Prepare data and workspace

Load required packages.

```r
library(abind)
library(gplots)
library(RColorBrewer)
library(LSD)
library(limma)
library(boot)
library(plotrix)
library(igraph)
library(vioplot)
library(calibrate)
library(geneplotter)
library(Hmisc)
library(WntSGI)
```

Load data

Load **raw data** and **annotation**

```r
data("RawDataWntSGI", package="WntSGI")
```

Define **content annotation** for the raw data file. Each gene (template = `t`, query = `q`) is targeted by two independent RNAi designs (`dsRNA 1 or 2`). Wnt signaling pathway actvivity was measured using a dTCF-luc reporter, which expresses the firefly luciferase (Fluc). This signal was normalized for the viability effect, measured using a Rp128-rluc reporter, which expresses the renilla luciferase (Rluc).

```
dsDesign = c("dsRNA 1","dsRNA 2")
designCombi = c("t1q1","t1q2","t2q1","t2q2")
geneticBackground = c("uninduced","ligand-induced","Apc LOF-induced")
measurement = c("Rluc", "Fluc","Fluc/Rluc","normRluc", "normFluc","normFluc/Rluc")
```

The **data array structure** is the following:

- dim1: 384 wells
- dim2: 72 query genes times template design x query design combinations (= 288)
- dim3: 3 Wnt pathway states (baseline, ligand-induced, Apc LoF-induced)
- dim4: Wnt pathway activity and cell viability measurements, and their ratio (= 3)

Flagging of plate reader failures

Define an array that contains **raw data** and **processed raw data**.

```
proRawData = abind(RawDataWntSGI$RawData, RawDataWntSGI$RawData, along=4)
dimnames(proRawData) = list(paste(RawDataWntSGI$templateAnno$Dest_well384[1:384],
                                  RawDataWntSGI$templateAnno$Content[1:384]),
                            RawDataWntSGI$queryNames,geneticBackground,measurement)
```

The instrument **failed to read** reporter levels on plates and locations as listed here:

- t2q1 171 r, wells M to P 1:24 *in Ra_files*
- t2q2 171 f, wells O 1:4 and P 1:24 *in Fa_files*

Those values were set to NA in the **normalized data dimension**.

```
tdata = proRawData[,,,c("normRluc","normFluc")]
dim(tdata) = c(384,length(RawDataWntSGI$plateNames))
tdata[c(289:384),which(RawDataWntSGI$plateNames == "t2q1_171_r.TXT")] <- NA
tdata[c(337:340,361:384),which(RawDataWntSGI$plateNames == "t2q2_171_f.TXT")] <- NA
dim(tdata) = c(384,
               length(RawDataWntSGI$queryNames),
               length(geneticBackground),
               length(c("normRluc","normFluc")))
proRawData[,,,c("normRluc","normFluc")] = tdata
```

Normalization

Normalize all plates to the **median** of their **biological controls** (384-well plate columns 12 and 13). Those biological controls are spotted in 32 wells of each plate and involve non-targeting dsRNAs, viability controls and reagents targeting known Wnt pathway components. Their location is annotated in RawDataWntSGI$templateAnno$Content as ctrl.

```
biolCtrl = which(RawDataWntSGI$templateAnno$Content[1:384] == "ctrl")
for (i in 1:dim(proRawData)[2]) {
  for(n in 1:dim(proRawData)[3]) {
    for(m in 4:5) {
      proRawData[,i,n,m] = proRawData[,i,n,m] / median(proRawData[biolCtrl,i,n,m]
                                                       , na.rm=TRUE)
    }
  }
}
proRawData[,,,"normFluc/Rluc"] = proRawData[,,,"normFluc"] / proRawData[,,,"normRluc"]
```

Order data

Reorder template genes: 1st and 2nd dsRNA designs targeting template genes had been placed on different 384-well plates and on those plates they had a different order. Bring **measurements** for **both designs** to **same order**.

```
dim(proRawData) = c(384,
                    length(RawDataWntSGI$queryNames)/2,
                    length(dsDesign),
                    length(geneticBackground),
                    length(measurement))
proRawData[,,1,,] = proRawData[order(RawDataWntSGI$templateAnno$Symbol[1:384]),,1,,]
proRawData[,,2,,] = proRawData[order(RawDataWntSGI$templateAnno$Symbol[385:768]),,2,,]
dim(proRawData) = c(384,
                    length(RawDataWntSGI$queryNames),
                    length(geneticBackground),
                    length(measurement))
```

Bring **annotation** to the measurement order.

```
tAnno = RawDataWntSGI$templateAnno[order(RawDataWntSGI$templateAnno$Symbol[1:384]),]
dimnames(proRawData) = list(tAnno$Symbol, RawDataWntSGI$queryNames
                            , geneticBackground, measurement)
```

Re-define **ctrl well location** based on new annotation order.

```
ctrlMainEff = which(tAnno$Content == "ctrlMainEff")
biolCtrl = which(tAnno$Content == "ctrl")
```

Reorder query genes: Plates must be re-ordered for overview of the **screening batches**. Each query (`q`) gene was measured with all template (`t`) genes in the following order in which `1` marked the first and `2` the second dsRNA design:

- t1q1__xxx
- t1q2__xxx
- t2q1__xxx
- t2q2__xxx

The `.txt` files were imported in an alpha-numerical order: all query plates for `t1q1_xxx` came first, then `t1q2_xxx`, `t2q1_xxx` and `t2q2`. Change order of plates to have all 4 combinations for query one, then for the second until the 72nd query gene.

The screening order is defined in **orderQ**.

```
orderQ=(1:length(RawDataWntSGI$queryNames))
iterLoop = length(RawDataWntSGI$queryAnno$Symbol)
for(i in 1:iterLoop) {
  orderQ[c((1:4)+(4*(i-1)))] = c(i,iterLoop+i,iterLoop*2+i,iterLoop*3+i)
}
```

Generate array that contains all **processed data**. Write data into array for all normalization steps.

```
proMeasurement = c("normRluc", "normFluc","normFluc/normRluc")
pD = array(1, dim=c(dim(proRawData)[1],
                        dim(proRawData)[2],
                        length(geneticBackground),
                        length(proMeasurement)))
dimnames(pD) = list(tAnno$Symbol,
                    RawDataWntSGI$queryNames[orderQ],
                    geneticBackground,
                    proMeasurement)
```

Transform data

log2-transform ctrl median-normalized Rluc and Fluc values to get more normally distributed data. Build their ratio to normalize the **Wnt-specific** signal for **viability effects** cased by a given dsRNA (*note: at log2 scale* / *becomes* -).

```
pD[,,,"normRluc"] = log2(proRawData[,orderQ,,"normRluc"])
pD[,,,"normFluc"] = log2(proRawData[,orderQ,,"normFluc"])
pD[,,,"normFluc/normRluc"] = pD[,,,"normFluc"] - pD[,,,"normRluc"]
processedData = pD
```

Quality control

Exclude potentially contaminated wells

Rational: the measurements in each channel and for the first and second template RNAi design have been done in two genetic backgrounds: the first and the second RNAi design against each of the query genes. The measurements can be compared between those two backgrounds with the expectation that they are similar. If Rp128-Rluc (viability) values between those backgrounds deviate strongly, the lower value indicates a failure of the experiment, due to e.g. a contamination.

Current stucture of *processedData* is:

- `tAnno$Symbol`
- `RawDataWntSGI$queryNames[orderQ]`, *this is crucial, because all plates with the same query are next to each other!*
- pathway state
- `proMeasurement`

Generate dimensions: wells, queries, t x q combinations, uninduced/induced/ApcKD, measurements.

```
dim(pD) = c(384,
            length(designCombi),
            length(RawDataWntSGI$queryAnno$Symbol),
            length(geneticBackground),
            length(proMeasurement))
pD = aperm(pD, c(1,3,2,4:5))
dimnames(pD) = list(tAnno$Symbol,
                    RawDataWntSGI$queryAnno$Symbol,
                    designCombi,
                    geneticBackground,
                    proMeasurement)
```

Swap t x q combinations with the screening conditions **uninduced/induced/ApcKD**.

```
pD = aperm(pD, c(1,2,4,3,5))
```

Test all template reagents (design #1 and #2 independently) in their two measurements (with query design #1 vs. #2).

Swap t1q2 with **t2q1** and distinguish all txq**1** from txq**2** data in the second dimension of the array.

```
pD = pD[,,,c(1,3,2,4),]
dim(pD) = c(384*length(RawDataWntSGI$queryAnno$Symbol)*length(geneticBackground)
            *(length(designCombi)/length(dsDesign)),
            length(dsDesign), length(proMeasurement))
```

Define potentially **contaminated wells** (very low normRluc values).

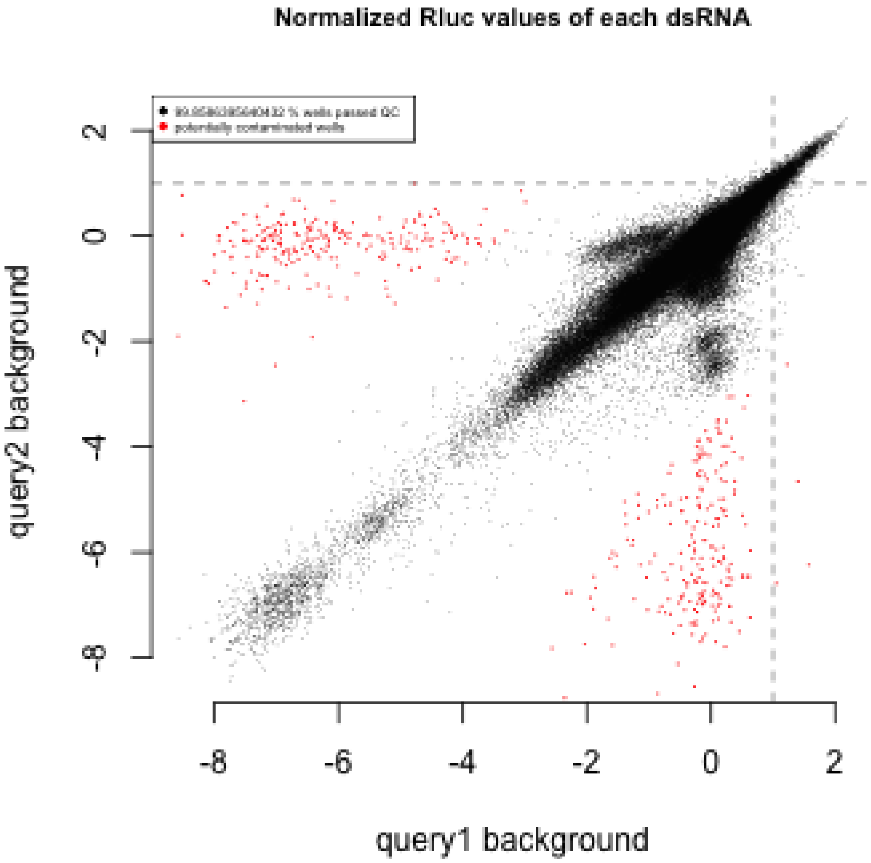
Comment: Rluc values, which are high (>=1.5) in design 1 and low (<=0.75) in design 2, were measured with slmb des#2 as query (all 3 conditions).

```
rlDim = which(proMeasurement == "normRluc")
exclWells1 = which(pD[,2,rlDim] - pD[,1,rlDim] >= log2(10))
exclWells2 = which(pD[,2,rlDim] - pD[,1,rlDim] <= log2(1/10))
passedWells = paste(((dim(pD)[1]*2) - length(c(exclWells1,exclWells2)))
                    / (dim(pD)[1]*2) *100, "%", sep=" ")
```

The entire screen (over all three conditions) contained 469 potentially contaminated wells.

Visualize potentially contaminated wells.

```
contaminationShow(d = pD, e1 = exclWells1, e2 = exclWells2)
```



Replace potentially **contaminated** well values by **'NA'**.

```
pD[exclWells1,1,] <- NA
pD[exclWells2,2,] <- NA
```

Reconstitute data array structure.

```
dim(pD) = c(384,
            length(RawDataWntSGI$queryAnno$Symbol),
            length(geneticBackground), length(designCombi), length(proMeasurement))
pD = pD[,,,c(1,3,2,4),]
```

Correlation of independent RNAi reagents

Test the **correlation** between the two independent dsRNA designs using their **ratio (normFluc/normRluc)** data across all pathway states. Each pair of dsRNAs against a given gene has **432 data** points, which are viability-normalized Wnt activity values in `72 query genes * 3 pathway states = 216 unique genetic backgrounds` (times two dsRNA designs for each query).

Note: in the 3rd dimension all t1 can be compared to all t2

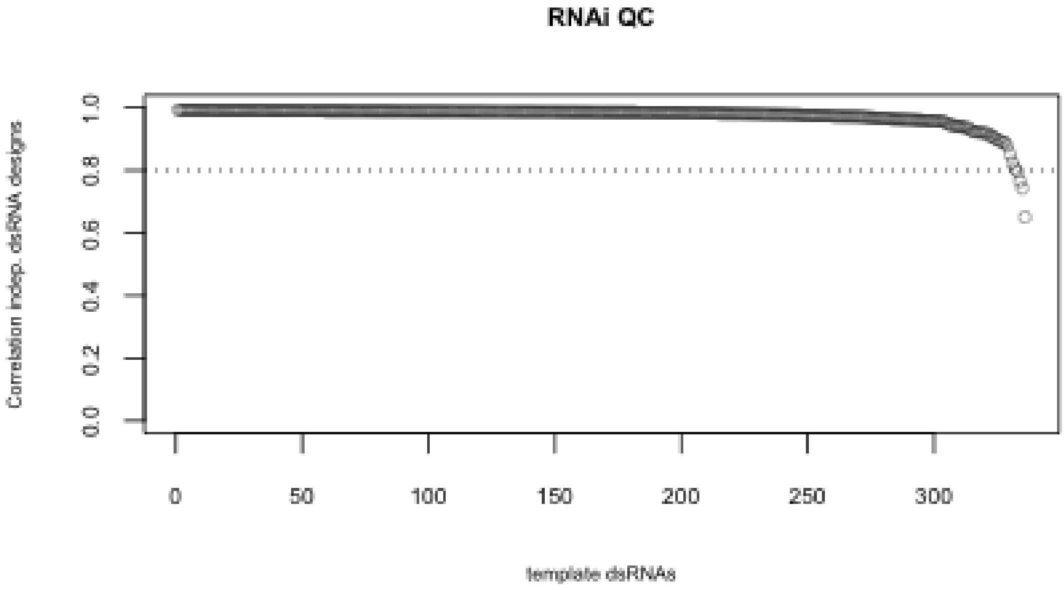
```
dim(pD) = c(384,
            length(RawDataWntSGI$queryAnno$Symbol)*length(geneticBackground)*
                (length(designCombi)/length(dsDesign)),
            length(dsDesign),length(proMeasurement))
ratioMeasures = pD[,,,3]
```

Estaimte the **agreement** between dsRNA designs by computing the `Pearson correlation coefficient`.

```
dsRNAcor = rep(NA, times = dim(pD)[1])

for(i in 1:dim(ratioMeasures)[1]) {
  dsRNAcor[i] = cor(ratioMeasures[i,,1], ratioMeasures[i,,2]
                      , use="complete.obs", method="pearson")
}

dsRNAcor = dsRNAcor[-c(biolCtrl,ctrlMainEff)]
names(dsRNAcor) = tAnno$Symbol[-c(biolCtrl,ctrlMainEff)]
```

Plot the **correlations coefficients** in **descending order**.

```
plot(sort(dsRNAcor, decreasing=TRUE), ylim=c(0,1), lwd=0.4, cex=0.8,
     ylab="Correlation indep. dsRNA designs", xlab="template dsRNAs",
     main="RNAi QC", cex.axis=0.7, cex.lab=0.6, cex.main=0.8)
abline(h=0.8, lty=3)
```



Print gene names with **lowly correlating** dsRNA designs (smaller than 0.8).

```
print(sort(dsRNAcor[which(dsRNAcor < 0.8)]))
```

```
##      slmb l(2)NC136  CkIalpha   Hsc70-4
## 0.6506253 0.7427107 0.7650024 0.7967743
```

Conclusion: since **only well-known regulators** (*CkIalpha* and *slmb*) with differences in amplitude of the single knockdown effects of the two dsRNA designs are below a `PCC` of `0.8`, all 336 template gene dsRNA reagents were kept.

Genetic interaction estimation

Reorder data for genetic interaction calling

Reorder data array dimensions: wells, query genes, t x q combinations, baseline/Wnt-active/Apc-loss, F/R measurements.

```
dim(pD) = c(384,
            length(RawDataWntSGI$queryAnno$Symbol),
            length(geneticBackground),
            length(designCombi),
            length(proMeasurement))
processedData = aperm(pD, c(1,2,4,3,5))
dimnames(processedData) <- list(tAnno$Symbol,
                                RawDataWntSGI$queryAnno$Symbol,
                                designCombi,
                                geneticBackground,
                                proMeasurement)
```

Arrange normalized **combinatorial** data array (*use the same dimension of the GI array*).

```
comData = processedData[-biolCtrl,,,,"normFluc/normRluc"]
dimnames(comData) = list(tAnno$Symbol[-biolCtrl],RawDataWntSGI$queryAnno$Symbol
                         ,designCombi,geneticBackground)
```

Arrange data array to have the:

- **template** dsRNAs (two per gene, plus non-targeting controls) in dimension 1 (704 levels)
- **query** dsRNAs (two per gene) in dimension 2 (144 levels)
- **pathway states** in dimension 3 (3 levels)

```
tmpd = comData
dim(tmpd) = c(352,72,2,2,3)
tmpd = aperm(tmpd, c(1,4,2:3,5))
dim(tmpd) = c(704,144,3)
dimnames(tmpd) = list(c(paste(tAnno$Symbol[-biolCtrl],"design1")
                        ,paste(tAnno$Symbol[-biolCtrl],"design2")),
                      c(paste(RawDataWntSGI$queryAnno$Symbol,"design1")
                        ,paste(RawDataWntSGI$queryAnno$Symbol,"design2"))
                      , geneticBackground)
```

Estimate interaction values by medpolishing

The genetic interaction scores (GI scores) are estimated using the `medpolish()` function, which subtracts the main effects (ME) from all columns and rows of a two-dimensional data array in an iterative fashion. The ME represent the single depletion phenotypes of the query and template genes, respectively. The residual value after medpolishing is the genetic interaction term. For each gene pair, four such terms are estimated, considering two independent RNAi designs against each gene. Medpolishing is applied for the two independent dsRNA designs against each of the 336 tested genes, as well as additional `2 * 16` GFP dsRNA controls. These non-targeting controls help controlling a possible bias. Medpolishing for each of those 704 reagents is done over 144 measurements (72 query genes * 2 dsRNA designs).

Define empty array for the template dsRNA main effect (ME) **tME** the query dsRNA ME **qME**, and separate the **four GI values** for each gene pair.

```
templateME = array(1,dim=c(dim(tmpd)[1],length(geneticBackground)))
queryME = array(1,dim=c(dim(tmpd)[2],length(geneticBackground)))
sepGI = array(1,dim=c(dim(tmpd)[1]*dim(tmpd)[2],length(geneticBackground)))

ctrlMainEff = which(tAnno$Content[-biolCtrl] == "ctrlMainEff")

for(j in 1:length(geneticBackground)) {
  mp = medpolish(tmpd[,,j],na.rm=TRUE)

  tME1 = mp$row[1:352]
  tME2 = mp$row[353:704]

  qME1 = mp$col[1:72]
  qME2 = mp$col[73:144]

  templateME[,j] = mp$row
  queryME[,j] = mp$col
  sepGI[,j] = mp$residuals
}
```

```
## 1: 28229.53
## 2: 27436.13
## Final: 27433.71
## 1: 28136.98
## 2: 26778.14
## Final: 26774.89
## 1: 25349.2
## 2: 24940.91
## Final: 24939.38
```

Re-arrange array.

```
dim(sepGI) = c(704,144,3)
dimnames(sepGI) = list(c(paste(tAnno$Symbol[-biolCtrl],"design1")
                         ,paste(tAnno$Symbol[-biolCtrl],"design2"))
                      ,c(paste(RawDataWntSGI$queryAnno$Symbol,"design1"),
                         paste(RawDataWntSGI$queryAnno$Symbol,"design2"))
                      , geneticBackground)
dimnames(templateME) = list(c(paste(tAnno$Symbol[-biolCtrl],"design1"),
                              paste(tAnno$Symbol[-biolCtrl],"design2"))
                            , geneticBackground)
dimnames(queryME) = list(c(paste(RawDataWntSGI$queryAnno$Symbol,"design1"),
                           paste(RawDataWntSGI$queryAnno$Symbol,"design2"))
                         , geneticBackground)
```

Get **query main effect** per gene for illustration of GI profiles. The main effect is estimated as the mean of the main effect of both dsRNAs targeting a given query gene.

```
singleKDq = (queryME[1:72,] + queryME[73:144,]) / 2
dimnames(singleKDq) = list(RawDataWntSGI$queryAnno$Symbol, geneticBackground)
```

Arrange the **genetic interaction** data, which was generated by medpolishing, in the **sepGI** array to have the:

- **template** genes in dimension 1 (352 levels)
- **query** genes in dimension 2 (72 levels)

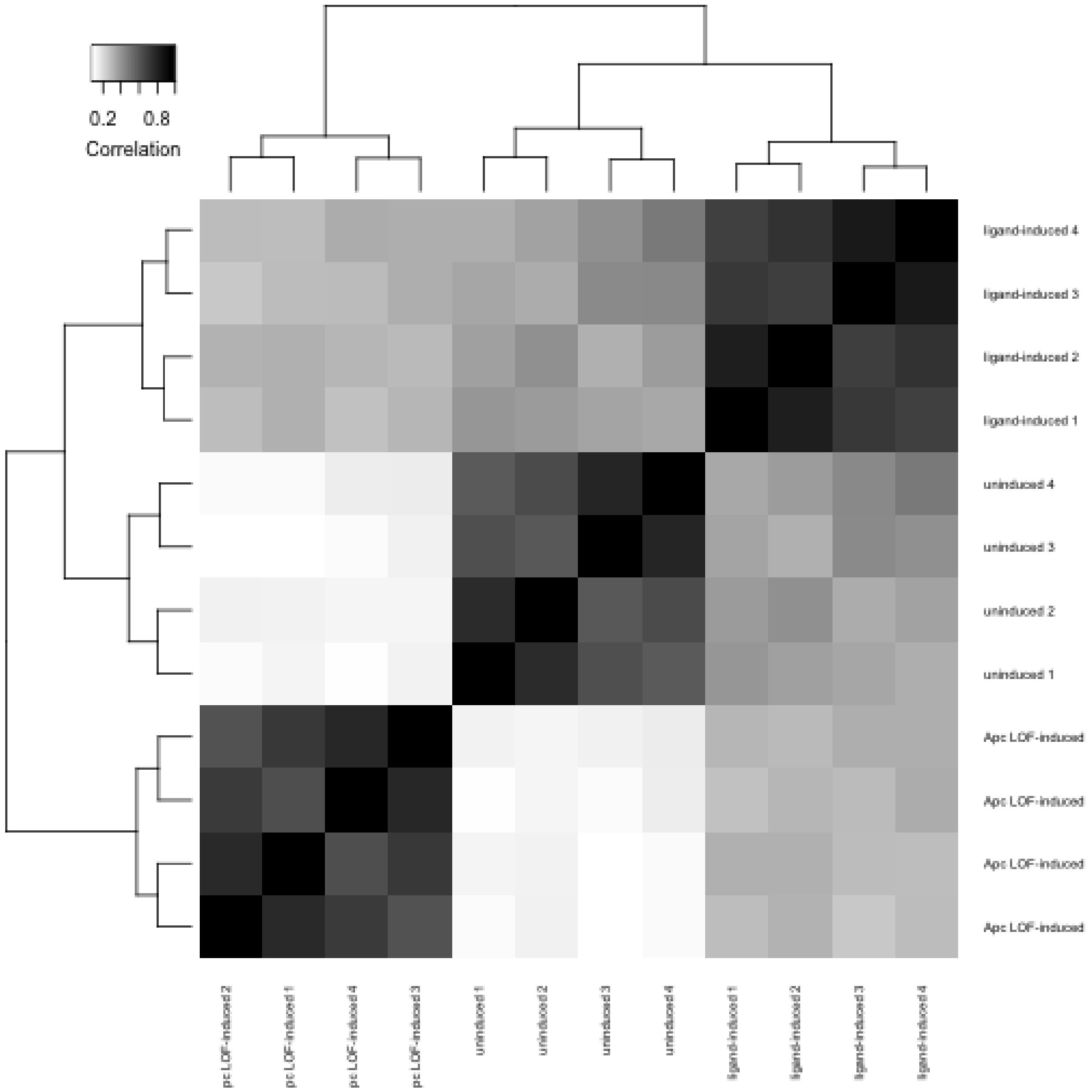
- **template X query** combination in dimension 3 (4 levels)
- **pathway states** in dimension 4 (3 levels)

```
dim(sepGI) = c(352,2,72,2,3)
sepGI = aperm(sepGI, c(1,3:4,2,5))
dim(sepGI) = c(352,72,4,3)
dimnames(sepGI) = list(tAnno$Symbol[-biolCtrl],RawDataWntSGI$queryAnno$Symbol
                       ,designCombi,geneticBackground)
```

Compare **GIs** across **all query genes** in the measured **txqx combinations** over **all measured conditions**.

Q: do scores measured with different RNAi (dsRNA) designs in the same condition are more similar to each other than measurements with the same RNAi designs in a different condition?

```
X = sepGI[-ctrlMainEff,,,]
dim(X) = c(336*72,4*3)
colnames(X) <- paste(rep(geneticBackground, each=4), 1:4)
x <- cor(X, use='complete.obs')
```



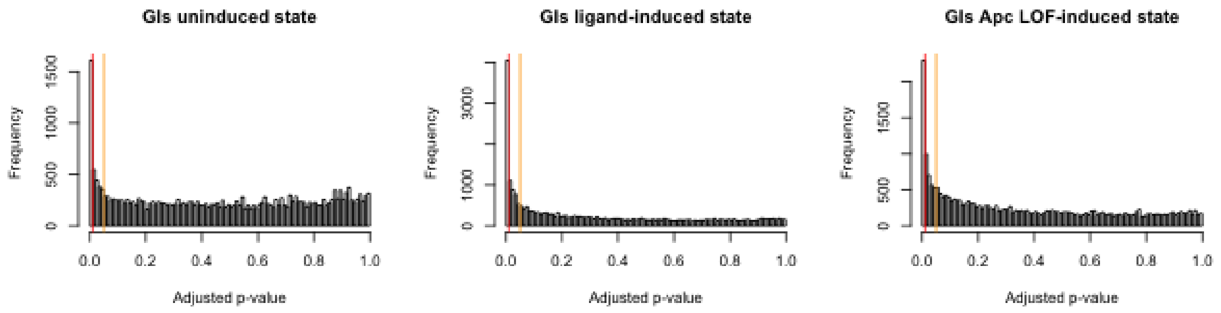
CONCLUSION: GIs are highly **reproducible** over the 4 **txqx combinations**. At the same time, the 4 replicates in a given state correlate very lowly with replicates measured in another state.

Significance of GIs

The moderated *t*-test as described in *Smyth 2004* is used to calculate p-values for the GI scores. The four independent measurements (txqx) are considered biologically independent replicates for a given gene pair. Adjust *p*-values for multiple testing using the method by Benjamini & Hochberg.

```
X = sepGI
selfGI = match(colnames(X), row.names(X))
padj = array(FALSE, dim = dim(X)[c(1,2,4)], dimnames = dimnames(X)[c(1,2,4)])

par.adj <- par(mfrow = c(1,length(geneticBackground)))
```

```
for (i in 1:length(geneticBackground)) {
  Y = X[,,,i]
  dim(Y) = c(prod(dim(Y)[1:2]),dim(Y)[3])
  fit = lmFit(Y)
  efit = eBayes(fit)
  padj[,,i] = p.adjust(efit$p.value, method="BH")
  hist(padj[,,i], breaks=100, main=paste("GIs",geneticBackground[i], "state")
       , xlab="Adjusted p-value", cex=1)
  abline(v=c(0.01,0.05), col=c("red","orange"))
}
par(par.adj)
```



Write out significant GIs for `FDR = 0.05` and `FDR = 0.01`. The genetic interaction terms derived from the four co-RNAi measurements per gene pair are used.

```
giFdr001 = padj <= 0.01
giFdr005 = padj <= 0.05
for (i in 1:length(geneticBackground)) {
  for(j in 1:72) {
    giFdr001[selfGI[j],j,i] = FALSE
    giFdr005[selfGI[j],j,i] = FALSE
  }
  print(geneticBackground[i])
  print(paste("Nr. GIs at FDR = 1%:",sum(giFdr001[-ctrlMainEff,,i])))
  print(paste("Nr. GIs at FDR = 5%:",sum(giFdr005[-ctrlMainEff,,i])))
}
```

```
## [1] "uninduced"
## [1] "Nr. GIs at FDR = 1%: 1588"
## [1] "Nr. GIs at FDR = 5%: 3269"
## [1] "ligand-induced"
## [1] "Nr. GIs at FDR = 1%: 3976"
## [1] "Nr. GIs at FDR = 5%: 7183"
## [1] "Apc LOF-induced"
## [1] "Nr. GIs at FDR = 1%: 2016"
## [1] "Nr. GIs at FDR = 5%: 4556"
```

Summarize interaction terms acorss dsRNA designs

The `median` was applied, because it was more robust against outliers. The median absolute deviation `mad` was used along with `median`.

Take **median of 4 t x q** measurements for each **combinatorial KD**.

```
wntSignal = apply(comData, c(1:2,4), median, na.rm=TRUE)
dimnames(wntSignal) = list(tAnno$Symbol[-biolCtrl], RawDataWntSGI$queryAnno$Symbol
                           , geneticBackground)
```

Take **mad of 4 t x q** measurements for each **combinatorial KD**.

```
sdWntSignal = apply(comData, c(1:2,4), mad, na.rm=TRUE)
dimnames(sdWntSignal) = list(tAnno$Symbol[-biolCtrl], RawDataWntSGI$queryAnno$Symbol
                             , geneticBackground)
```

Take **mean of 2 dsRNA designs** (each estimated by mp) for each **single KD value / tME**.

```
singleKD = templateME
dim(singleKD) = c(352,2,3)
singleKD = apply(singleKD, c(1,3), mean, na.rm=TRUE)
dimnames(singleKD) = list(tAnno$Symbol[-biolCtrl], geneticBackground)
```

Estimate **standard error** for the **single KD** effects. The single knockdown effects for each of the two dsRNA designs against a gene were estimated via the main effect over 144 genetic backgrounds (72 queries with 2 dsRNA designs each). The mean of those two values represented the single knockdown effect of a gene. The standard error was estimated by performing **bootstrapping** of the median of all 288 combinatorial effects *minus* the respective query main effects for a given query gene.

```
sdSingleKD = singleKD
x = comData
dx = queryME

bsf <- function(data, ind) {median(data[ind], na.rm=TRUE)}

for(i in 1:dim(x)[1]) {
  for(j in 1:dim(x)[4]) {
    x[i,,1,j] = x[i,,1,j] - dx[1:72,j]
    x[i,,3,j] = x[i,,3,j] - dx[1:72,j]
    x[i,,2,j] = x[i,,2,j] - dx[73:144,j]
    x[i,,4,j] = x[i,,4,j] - dx[73:144,j]
    bsv <- boot(data = x[i,,,j], R=100, statistic = bsf)
    sdSingleKD[i,j] <- sd(bsv$t)
  }
}
```

Take **median of 4 t x q** measurements for each separate **GI**.

```
geneticInteractionsPlus = apply(sepGI, c(1:2,4), median, na.rm=TRUE)
dimnames(geneticInteractionsPlus) = list(tAnno$Symbol[-biolCtrl],
                                         RawDataWntSGI$queryAnno$Symbol,
                                         geneticBackground)
geneticInteractions = geneticInteractionsPlus[-ctrlMainEff,,]
```

Estimate **mad** for each **GI** (across the four independent measurements).

```
giSD = apply(sepGI, c(1:2,4), mad, na.rm=TRUE)
dimnames(giSD) = list(tAnno$Symbol[-biolCtrl], RawDataWntSGI$queryAnno$Symbol
                      , geneticBackground)
giSD = giSD[-ctrlMainEff,,]
```

Re-screened query gene comparison

Five of 72 query genes have been re-screened under Wnt-active (high Wg) conditions in an inpendent batch. To facilitate maximal comparability, genetic interactions were scored separately on the same data set as shown here but with those five re-screened query genes instead of the original ones. The scores were saved and are loaded to visualize the comparison of genetic interaction reproducibility.

```
data("replicateGI", package="WntSGI")
```

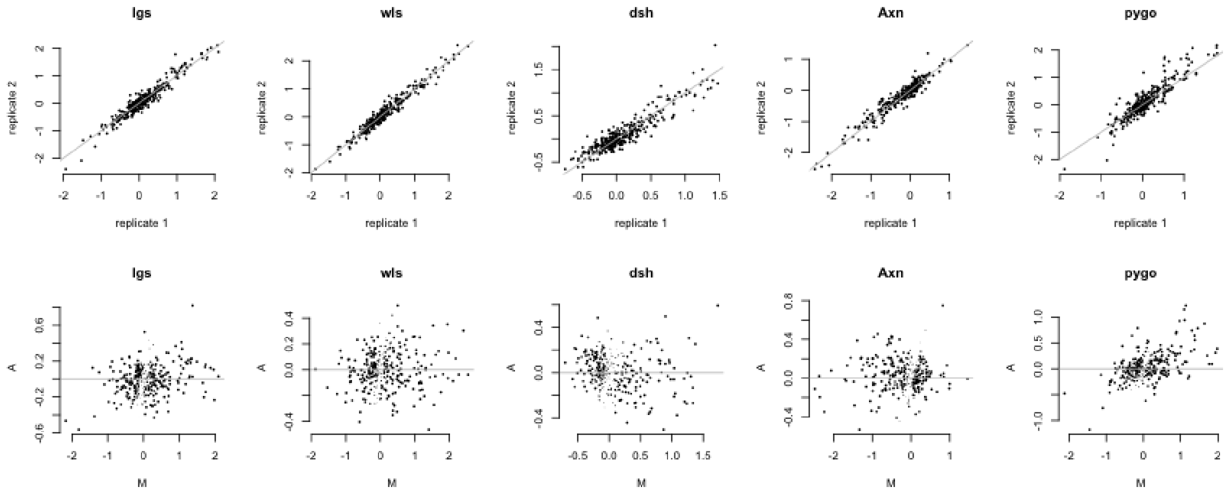
Compute the correlation between the 336 genetic interaction scores of each of the five query genes with the matched replicate scores.

```
repGenes <- colnames(replicateGI$giNew)
x <- array(NA, dim = c(length(repGenes), 2), dimnames = list(repGenes, c("cor","p-val")))

for(i in 1:length(repGenes)) {
  xtemp <- cor.test(geneticInteractions[,repGenes[i],2], replicateGI$giNew[,repGenes[i]])
  x[i,"cor"] <- xtemp$estimate
  x[i,"p-val"] <- xtemp$p.value
}

x
```

```
##            cor         p-val
## lgs  0.9624930 2.703711e-191
## wls  0.9769747 3.718821e-226
## dsh  0.9249331 2.347923e-142
## Axn  0.9599120 1.459602e-186
## pygo 0.9073408 9.629315e-128
```

Illustrate the 336 genetic interaction scores of a given query gene in the **initial data** set and compare to the **re-screened** query. Use **M-A-plot** to visualize the deviation of the Pi-score and how this deviation is reflected in the FDR.

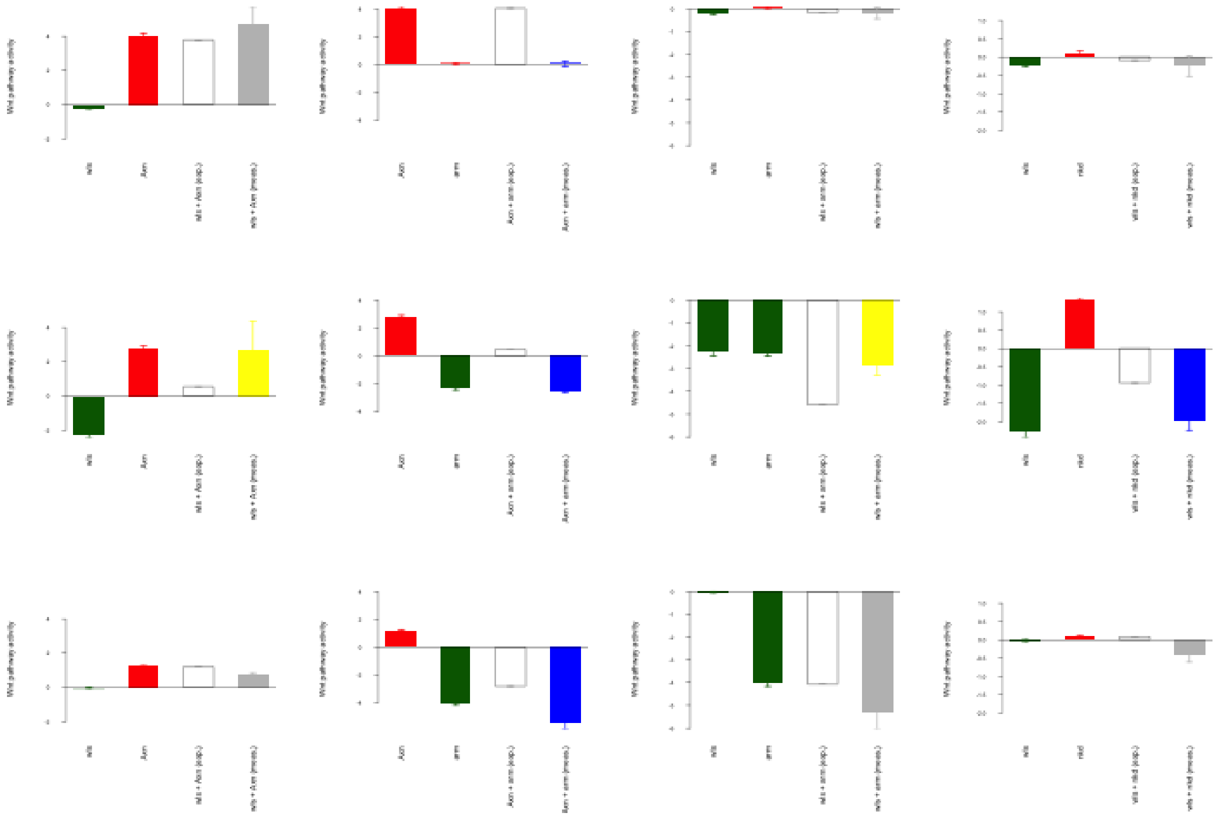
```
replicatePlot(x = geneticInteractions, y = replicateGI$giNew, q = padj[-ctrlMainEff,,])
```



Visualize the multiplicative model to estimate genetic interactions

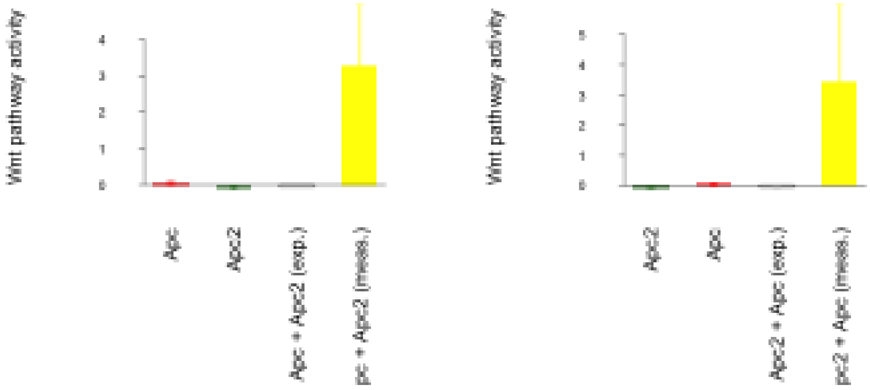
Plot **single KD phenotypes**, estimated and measured **combinatorial effects** for **selected gene pairs** for example gene pairs: *wls* & *Axn*, *Axn* & *arm*, *wls* & *arm*, *evi/wls* & *nkd*

```
giBarplot(goi = list(template=c("wls","Axn","wls","wls")
                     , query=c("Axn","arm","arm","nkd")),
          pState=1:3, singleKD, wntSignal, sdSingleKD, sdWntSignal
          , InteractionFDR=giFdr001)
```



Plot **single KD phenotypes**, estimated and measured **combinatorial effects** for: *Apc*, *Apc2*. This is done in the **baseline** state, because the phenotypic penetrance of the destruction complex is strongest here.

```
giBarplot(goi = list(template=c("Apc","Apc2"), query = c("Apc2","Apc")), pState=1,
          singleKD, wntSignal, sdSingleKD, sdWntSignal, InteractionFDR = giFdr005)
```

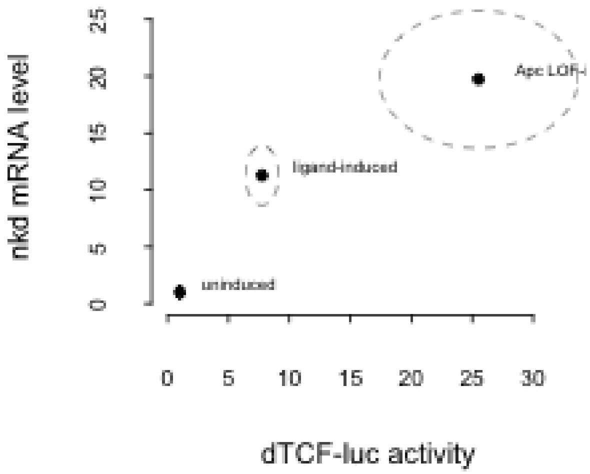


State-dependency of genetic interactions

Pathway induction level in the three states

Wnt pathway induction levels were quantified using a dTCF-luc reporter. This signal was validated in the three states by qPCR of mRNA level of the Wnt pathway target gene *nkd*.

```r
mNames = c("dTCF-luc activity","nkd mRNA level")
ctrlPlateEff = which(RawDataWntSGI$templateAnno$Symbol[1:384] == "GFP"
                     & RawDataWntSGI$templateAnno$Content[1:384] == "ctrl")

sEffects = array(NA, dim = c(3,2,2), dimnames = list(geneticBackground, mNames
                                                     , c("mean","sd")))

sEffects[,1,1] = apply(RawDataWntSGI$RawData[ctrlPlateEff,,,3],3, mean)
sEffects[,1,2] = apply(RawDataWntSGI$RawData[ctrlPlateEff,,,3],3, sd)
sEffects[,2,1] = c(1.11,12.51,21.89) #from qPCR data
sEffects[,2,2] = c(0.6,2.92,6.72) #from qPCR data
for(i in 1:2) {
  sEffects[,i,] = sEffects[,i,] / sEffects[1,i,1] # normation to: low Wg = 1
}

plot(sEffects[,1,1], sEffects[,2,1], xlim=c(0,33), ylim=c(0,26), pch=19, bty="n",
     xlab=mNames[1], ylab=mNames[2], tck=-0.02, cex.axis=0.7, cex.lab=0.9, cex=0.7)
draw.ellipse(sEffects[,1,1], sEffects[,2,1], sEffects[,1,2], sEffects[,2,2]
             , lty=2, lwd=0.5)
textxy(sEffects[,1,1], sEffects[,2,1], labs=geneticBackground)
```

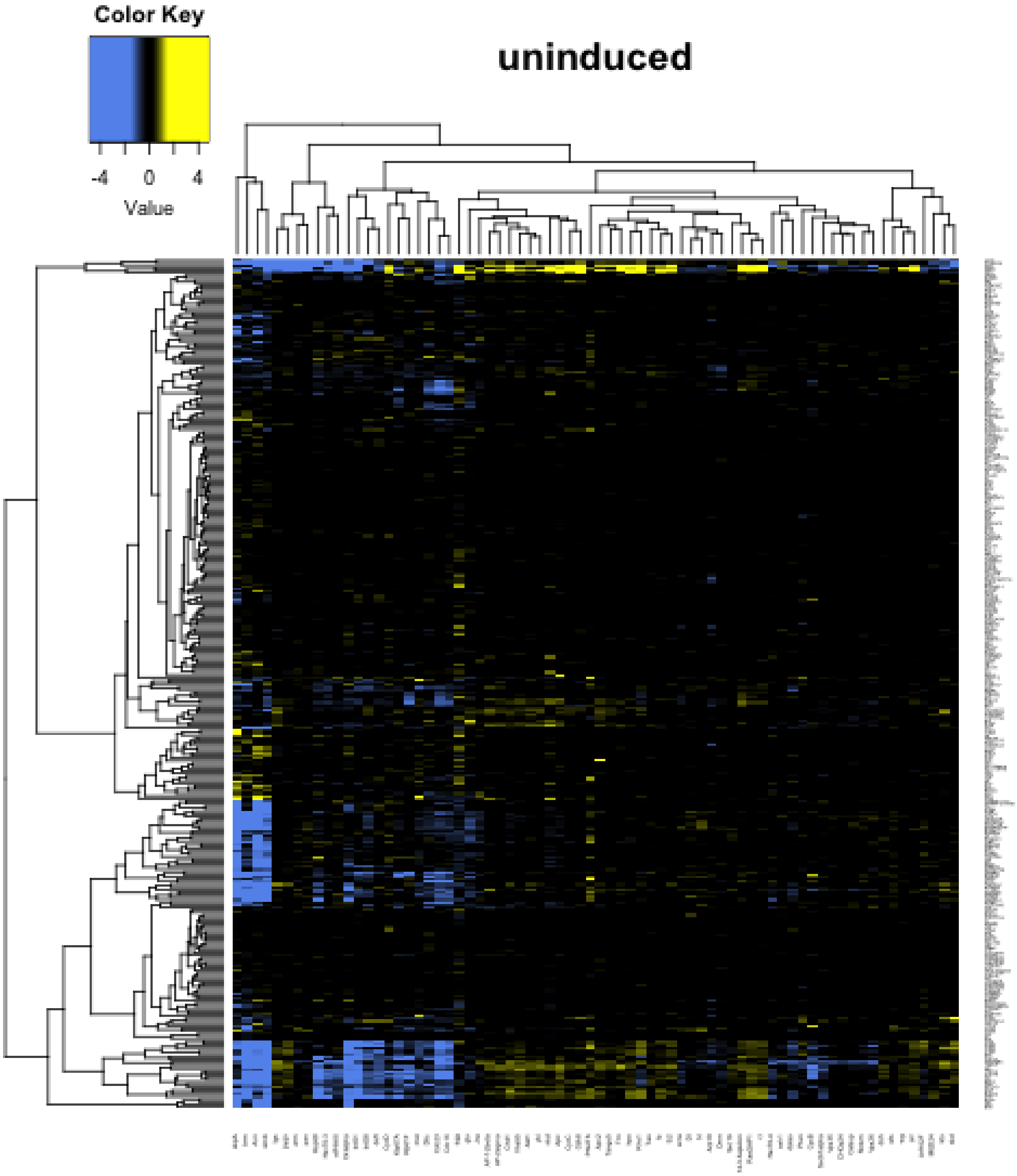


Distribution of the genetic interactions

Plot genetic interactions for each state as a **heatmap**. To keep the same order for the template genes across the three pathway states, the order (for individual states) was determined by state-spanning hierarchical clustering (query genes are ordered based on the profiles from the baseline state).

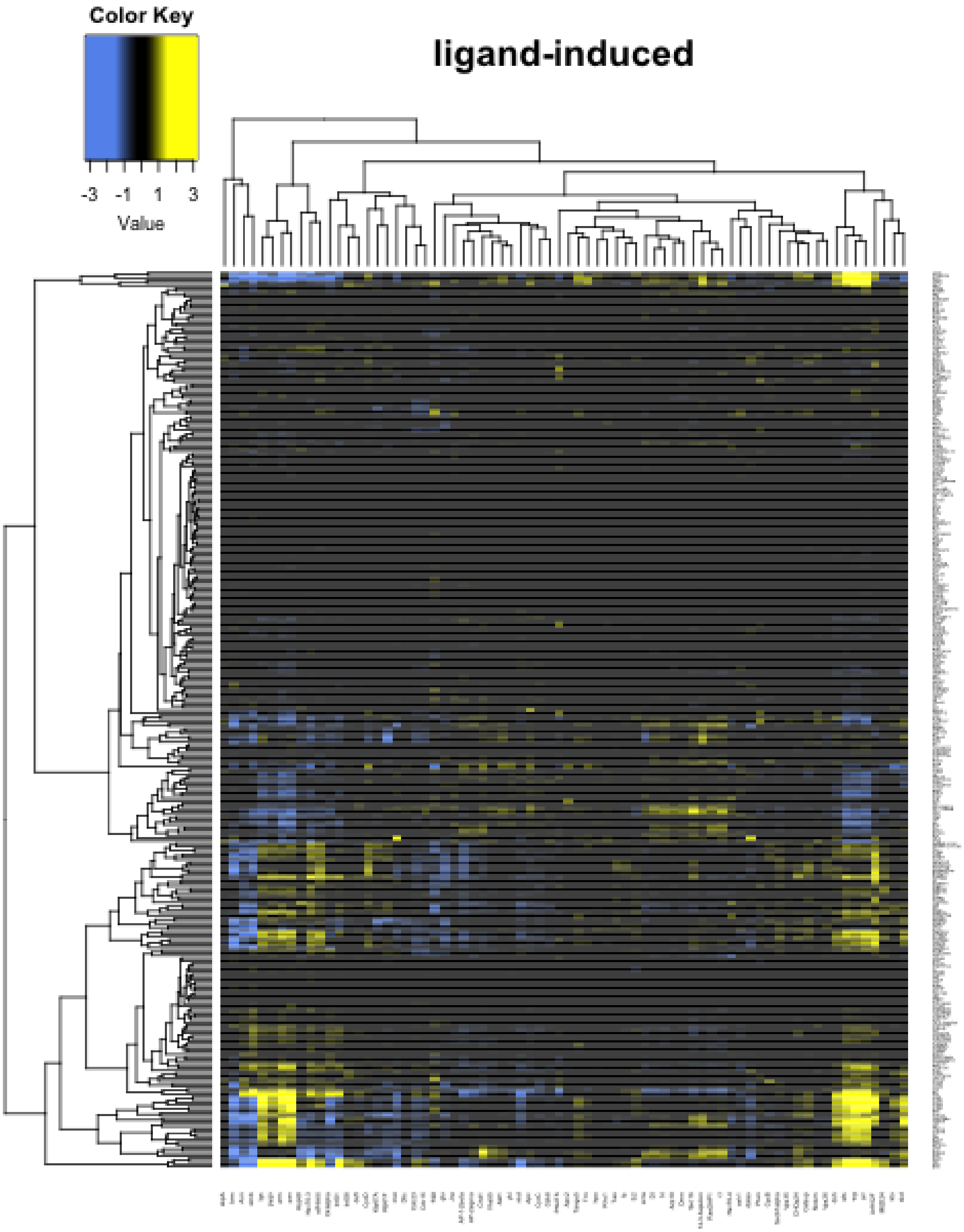
Genetic interactions in the uninduced state.

```
heatmap_func(X = geneticInteractions, ps = geneticBackground[1])
```



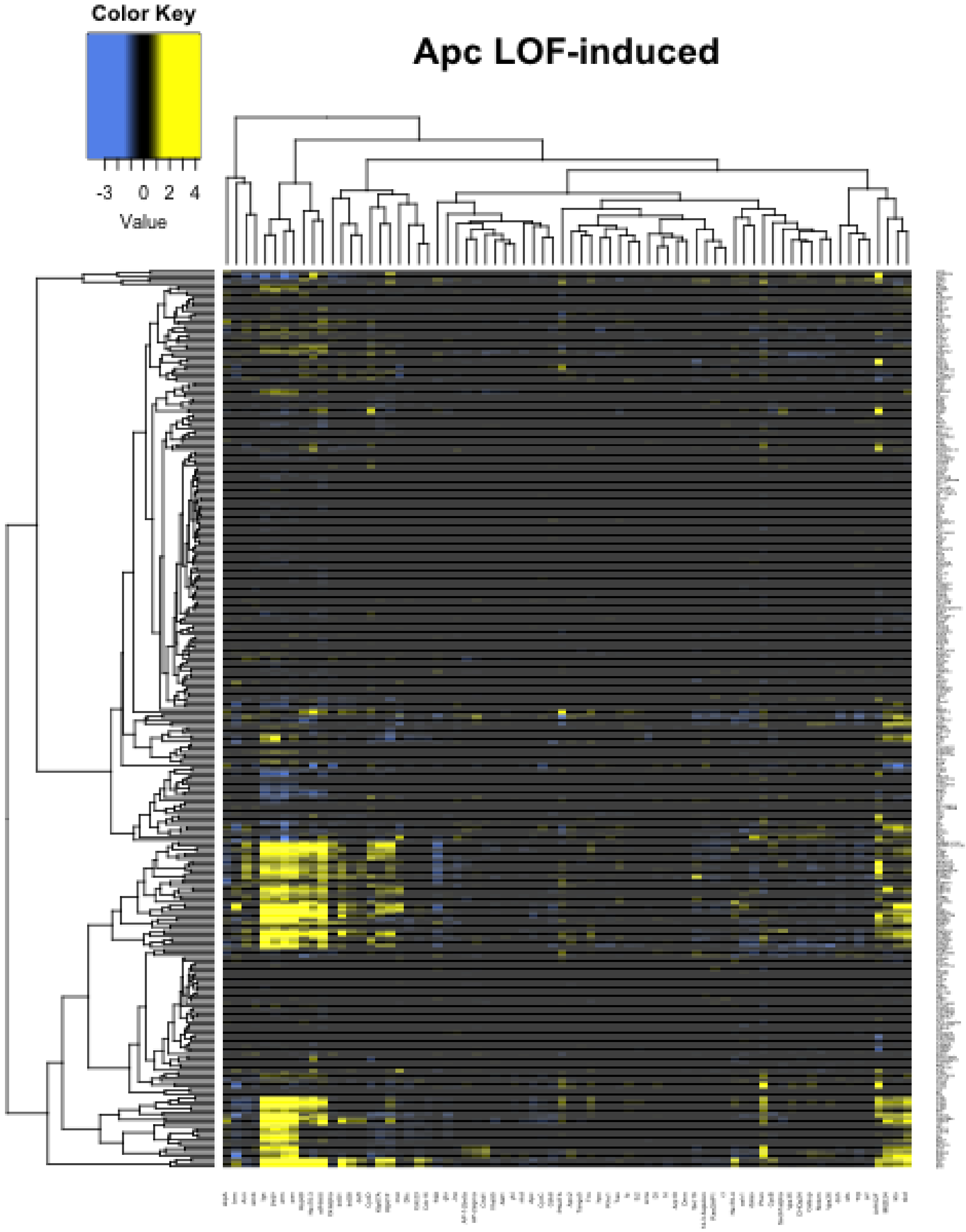
Genetic interactions in the ligand-induced state.

```
heatmap_func(X = geneticInteractions, ps = geneticBackground[2])
```



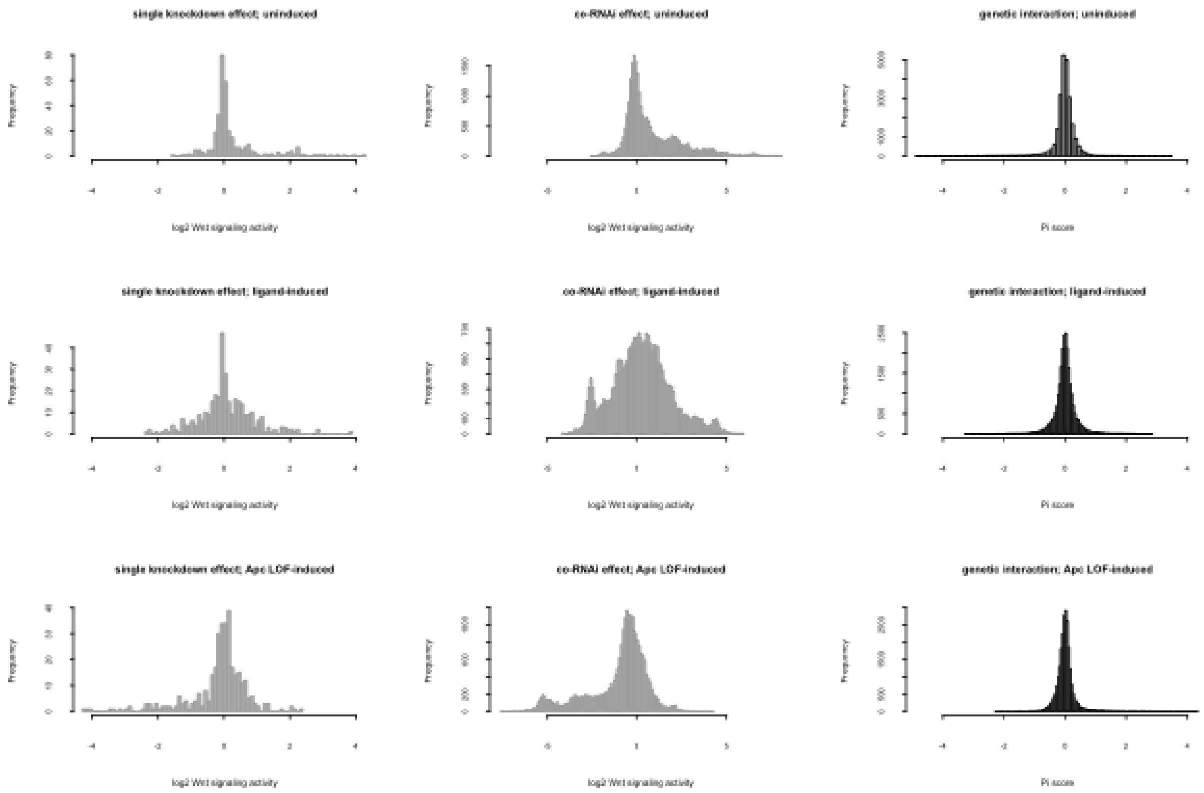
Genetic interactions in the Apc LOF-induced state.

```
heatmap_func(X = geneticInteractions, ps = geneticBackground[3])
```



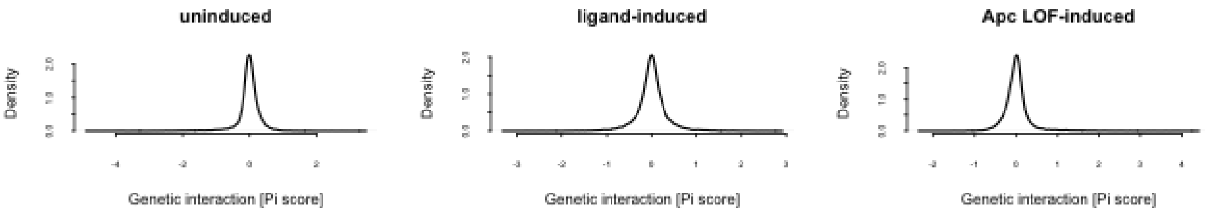
Plot **distribution** (histogram) of **single (template) gene effects**, **combinatorial depletion effects** and **genetic interactions (GI)** in each pathway state.

```
giHist(singleKD, wntSignal, geneticInteractions, geneticBackground, ctrlMainEff)
```



Plot **density of GIs** in each state.

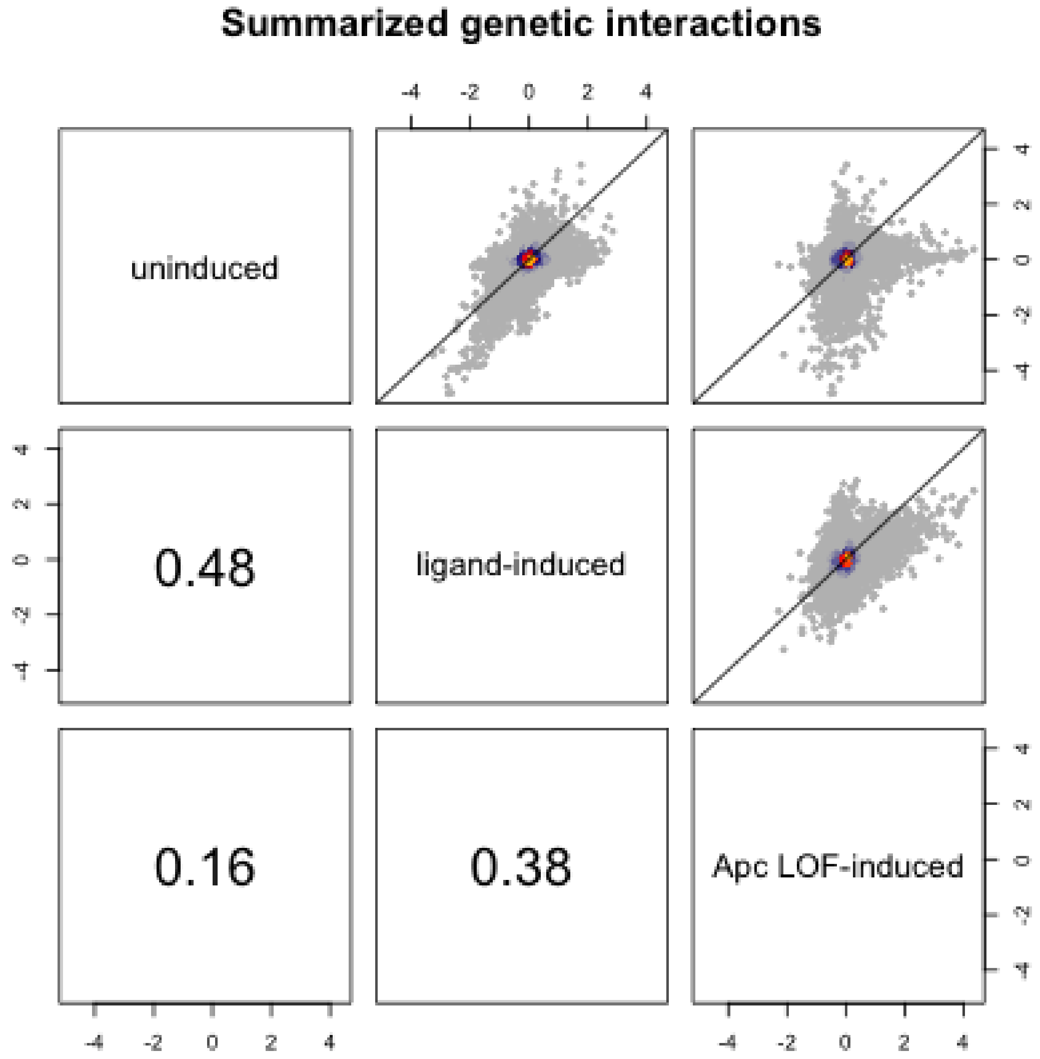
```
giDensity(geneticInteractions, geneticBackground)
```



State-dependency of the genetic interactions

Access the **similarity** of **all GIs** between the **three pathway states**. The similarity between all GI scores in a state are estimated a the Pearson correlation coefficient (PCC) and all scores are plotted.

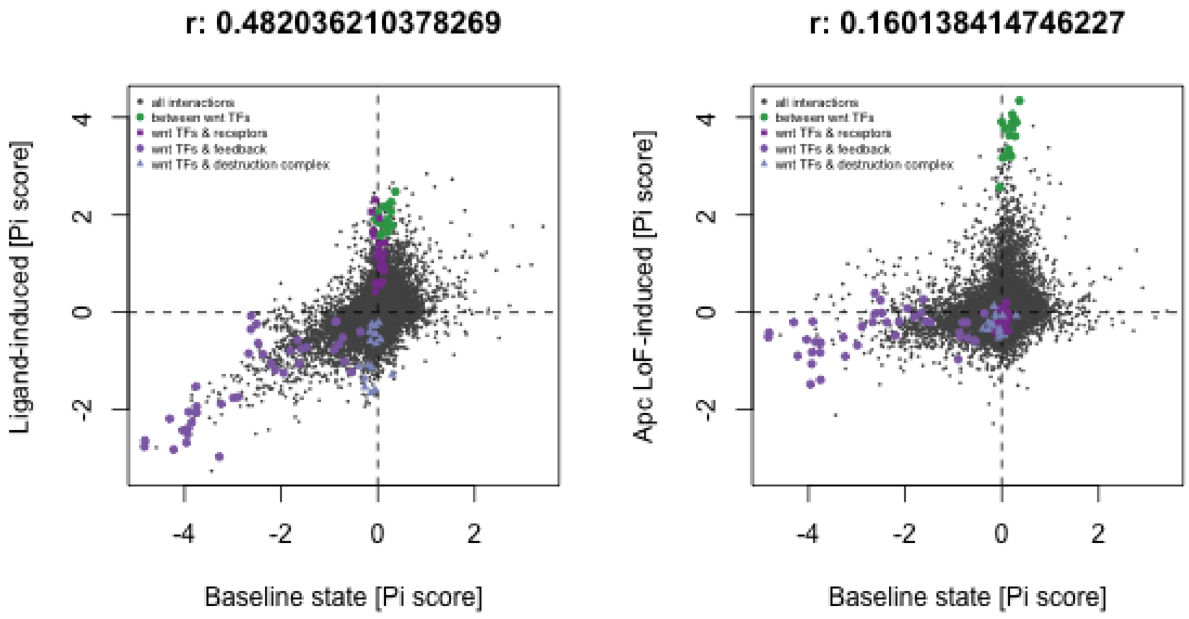
```
x = geneticInteractions
dim(x) = c(prod(dim(x)[1:2]),dim(x)[3])
heatpairs(x, labels=geneticBackground, method = "pearson",
          main="Summarized genetic interactions")
```



Plot the **change** of the **GI scores** upon pathway induction compared to the baseline state.

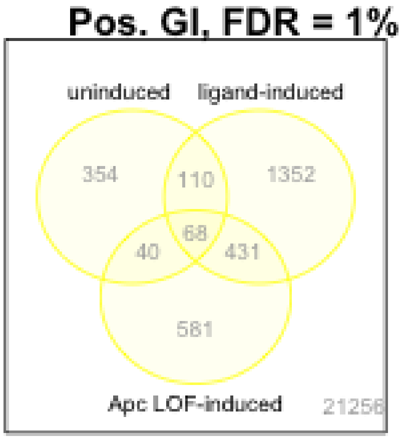
Color-code GI scores among **Wnt TFs** or between **Wnt TFs** and the functional groups **Receptors**, **Destruction complex** or **feedback regulators**.

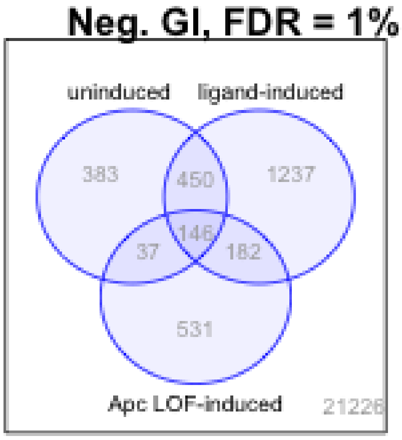
```
wntReceptor = c("fz2","arr","dsh")
destructionComplex = c("slmb","Axn","CkIalpha","sgg","skpA")
indReg = c("nkd","Notum")
wntTF = c("arm","pygo","lgs","pan")

giToState(geneticInteractions, gi = x, wntReceptor, destructionComplex, indReg, wntTF)
```



Show the overlap of statistically significant positive and negative genetic interactions between the three pathway states at a FDR of 1%.

```
a <- giFdr001[-ctrlMainEff,,]
ap <- geneticInteractions > 0
an <- geneticInteractions < 0

ap <- a & ap
an <- a & an

dim(ap) <- c(prod(dim(a)[1:2]), dim(a)[3])
colnames(ap) <- geneticBackground
dim(an) <- c(prod(dim(a)[1:2]), dim(a)[3])
colnames(an) <- geneticBackground

vennFunc(p = ap, n = an)
```





Correlation matrix of genetic interaction profiles

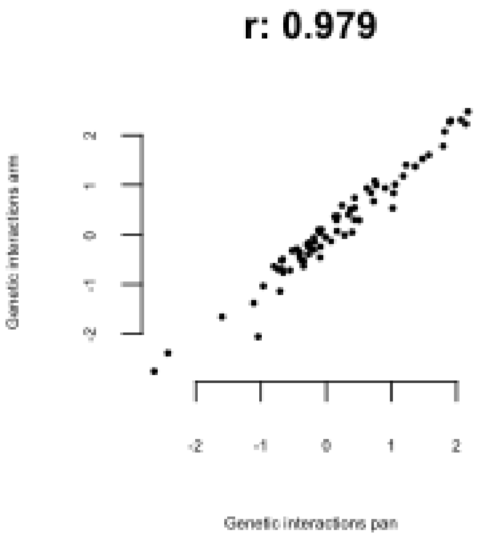
The functional similarity between two genes is estimated by calculating the Pearson correlation coefficient (PCC) between their GI profiles in each pathway state.

```
corSepGI = array(dim = c(dim(geneticInteractions)[1], dim(geneticInteractions)[1]
                         , length(geneticBackground)),
                 dimnames=list(row.names(geneticInteractions),
                               row.names(geneticInteractions),geneticBackground))
for(i in 1:length(geneticBackground)) {
  corSepGI[,,i] = cor(t(geneticInteractions[,,i]), use="complete.obs")
}
```

Functional implication of genetic interaction profile similarity

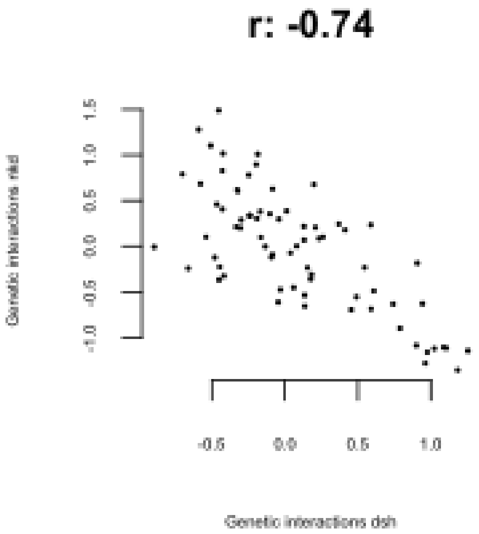
Example 1: *arm* and *pan* are core components of the Wnt target gene transcription machinery.

```
giProfileCor(x = geneticInteractions, gene1 = "pan", gene2 = "arm", states = 2, pcex = 0.5)
```



Example 2: **nkd** is a negative feedback regulator, which acts on the Wnt co-receptor **dsh**. *nkd* is induced by pathway activation, and thus facilitates negative feedback regulation.

```
giProfileCor(x = geneticInteractions, gene1 = "dsh", gene2 = "nkd", states = 2)
```

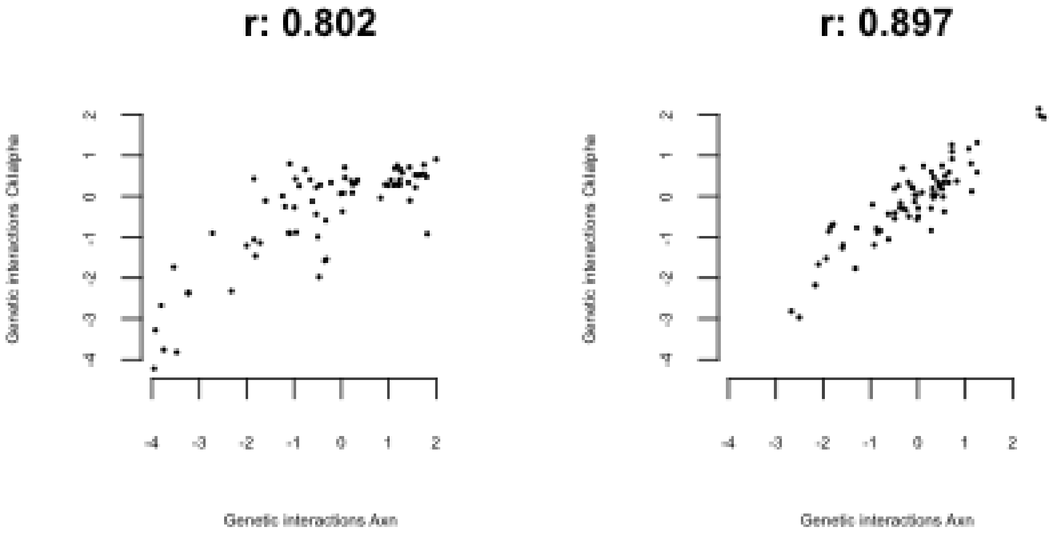


Conclusion: Positive and negative correlation between genetic interaction profiles reconstruct functional similarity (*arm* & *pan*) and inhibitory functional relations (*nkd* & *dsh*). As seen from example 2, state-dependent relationships can be captured.

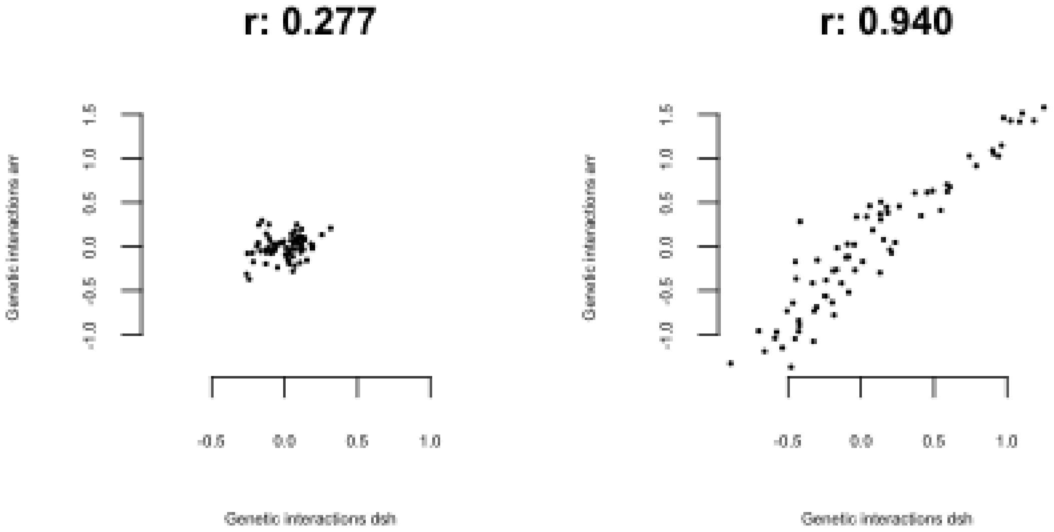
Pathway state dependency of genetic interaction profiles

Various examples: positive and negative correlation of genetic interaction profiles depend on the pathway state.

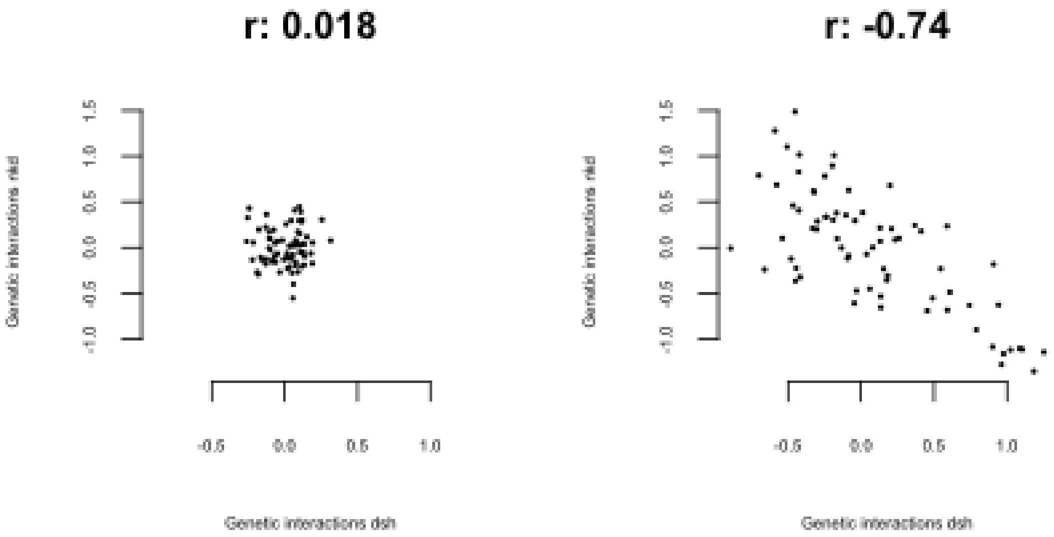
```
giProfileCor(x = geneticInteractions, gene1 = "Axn", gene2 = "CkIalpha", states = 1:2)
```



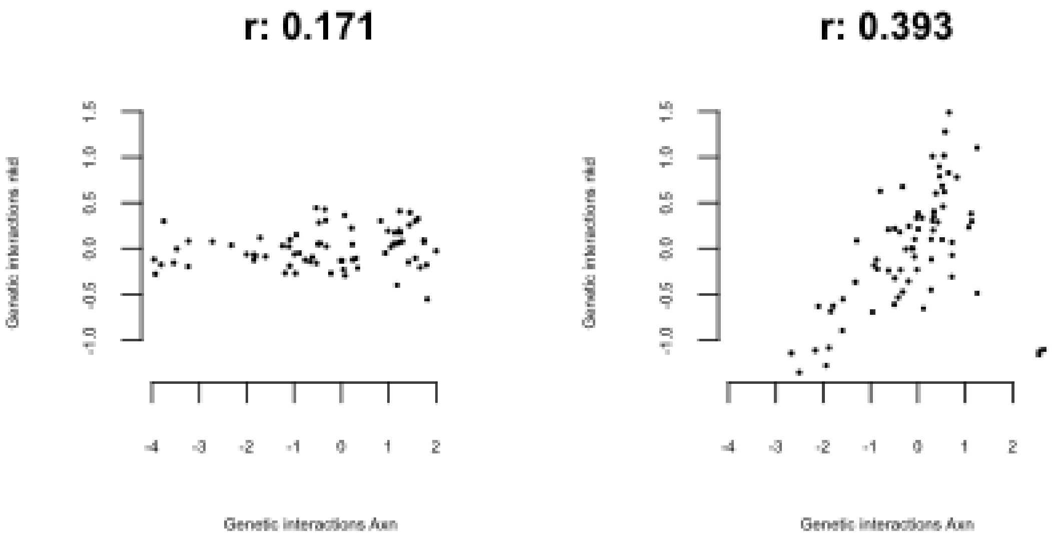
```
giProfileCor(x = geneticInteractions, gene1 = "dsh", gene2 = "arr", states = 1:2)
```



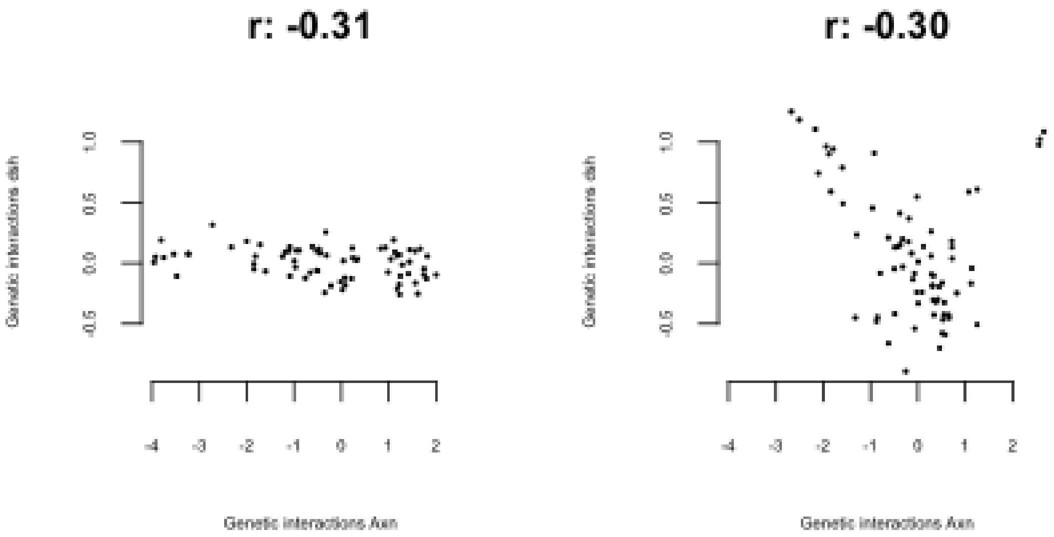
```
giProfileCor(x = geneticInteractions, gene1 = "dsh", gene2 = "nkd", states = 1:2)
```



```r
giProfileCor(x = geneticInteractions, gene1 = "Axn", gene2 = "nkd", states = 1:2)
```

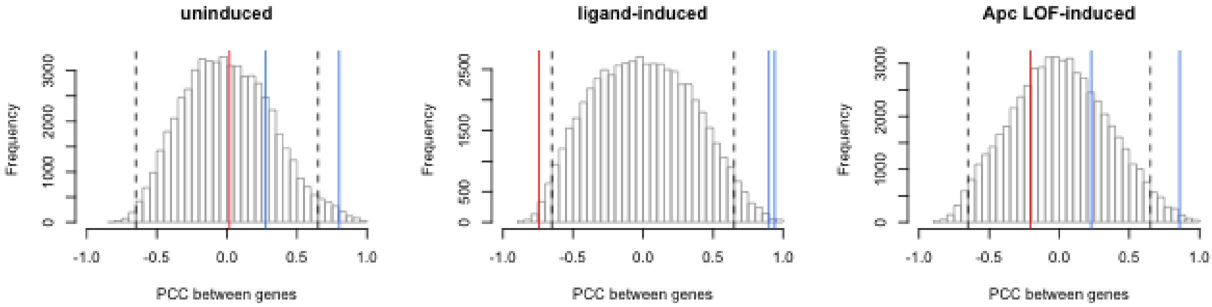


```r
giProfileCor(x = geneticInteractions, gene1 = "Axn", gene2 = "dsh", states = 1:2)
```



Indicate the state-specific PCC for three example gene pairs above in relation to all state-specific PCCs.

```
histStateExamples(x = corSepGI, pPair = list(p1 = c("Axn","CkIalpha"), p2 = c("dsh","arr")),
                  nPair = list(n1 = c("dsh","nkd")))
```



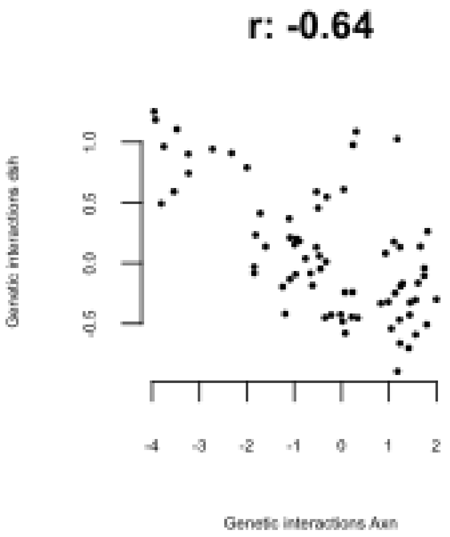
Between-state profile correlation

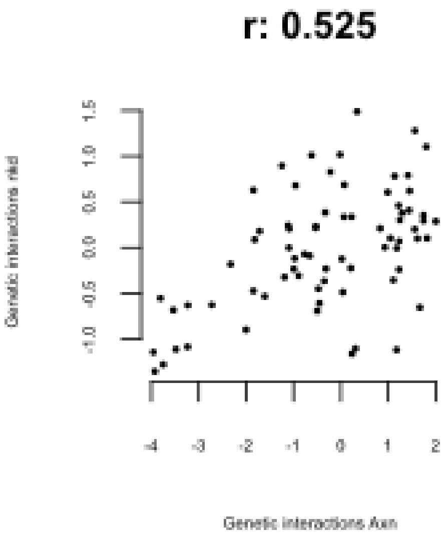
Case-study with the destruction complex component Axn

In the presence of ligands, the Wnt pathway is activated by a receptor complex, which senses ligand levels. **dsh** is a component of the receptor complex. This complex activates Wnt signaling by inhibiting the destruction complex; **Axn** is a component of this complex.

The baseline Wnt pathway is kept inactive by the destruction complex. In the active pathway, the destruction complex loses its inhibitory role and feedback regulators such as **nkd** inhibit pathway activity.

```
par.gi <- par(mfrow = c(1,2))
giAcrossProfileCor(x = geneticInteractions, gene1 = "Axn", gene2 = "dsh", states = 1:2)
giAcrossProfileCor(x = geneticInteractions, gene1 = "Axn", gene2 = "nkd", states = 1:2)
par(par.gi)
```





Conclusion: while destruction complex components were not connected to other components of the Wnt pathway in either state-specific network, between-state genetic interaction profile correlation suggests that this metric can be used to map functional relations of this complex.

Measure all between-state profile correlations

Estimate the correlation of genetic interaction profiles of all gene pairs **between pathway states**.

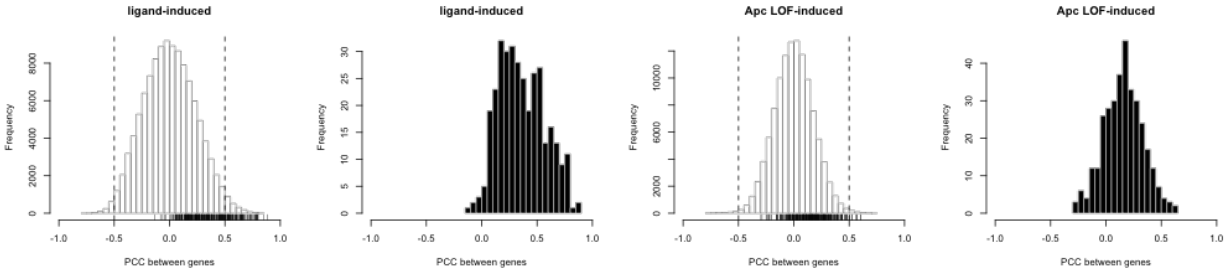
```
corAcross = array(data = NA, dim = c(336,336,3,3),
                  dimnames = list(row.names(geneticInteractions),
                                  row.names(geneticInteractions),
                                  geneticBackground, geneticBackground))

for(k in 1:length(geneticBackground)) {
  for(l in 1:length(geneticBackground)) {
    for(i in 1:dim(geneticInteractions)[1]) {
      for(j in 1:dim(geneticInteractions)[1]) {
        corAcross[i,j,k,l] <- cor(geneticInteractions[i,,k], geneticInteractions[j,,l])
      }
    }
  }
}
```

State-dependency of genetic interaction profile correlation

Between-state similarity distribution

Genetic interaction profiles are compared between the **baseline (uninduced)** and the respective **induced states**. All possible gene-gene relations are shown left and the gene-self correlations are shown right (also see black rug). Gene-self correlations compare the profile of each gene in the baseline state with its profile in a given induced state.

```
acrossHist(x = corAcross)
```

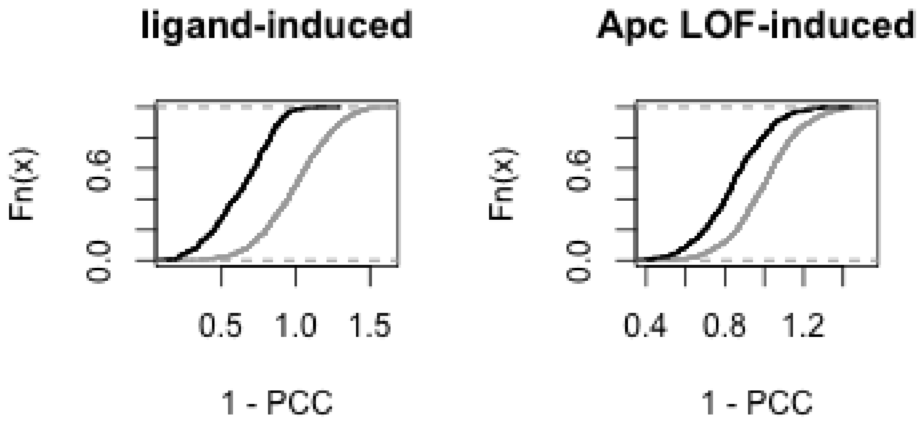


Comparison of all **between-gene** between-state PCCs (grey) with the **self-correlation** (black).

```
selfOthersAcross(X = corAcross, states = 2:3)
```

```
## [1] "ligand-induced"
## $p.value
## [1] 4.490049e-124
```

```
## [1] "Apc LOF-induced"
## $p.value
## [1] 6.259775e-52
```



fraction of relations beyond threshold

```
nrPairs = 336^2-336
th = 0.5

a = corAcross[,,"uninduced","ligand-induced"]; diag(a) <- NA
print(length(which(a > th)) / nrPairs)
```

```
## [1] 0.01799041
```

```
print(length(which(a < -th)) / nrPairs)
```

```
## [1] 0.008884151
```

```
a = corAcross[,,"uninduced","Apc LOF-induced"]; diag(a) <- NA
print(length(which(a > th)) / nrPairs)
```

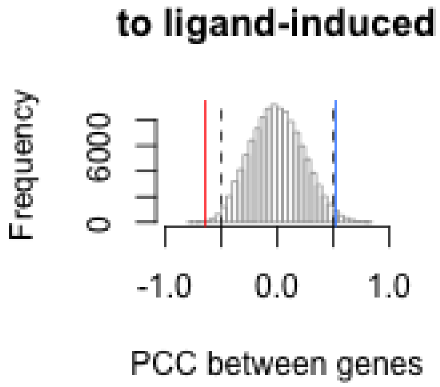
```
## [1] 0.003136105
```

```
print(length(which(a < -th)) / nrPairs)
```

```
## [1] 0.002798507
```

. . . and put between-state PCC for the two gene pairs shown above in context of all between-state PCCs.

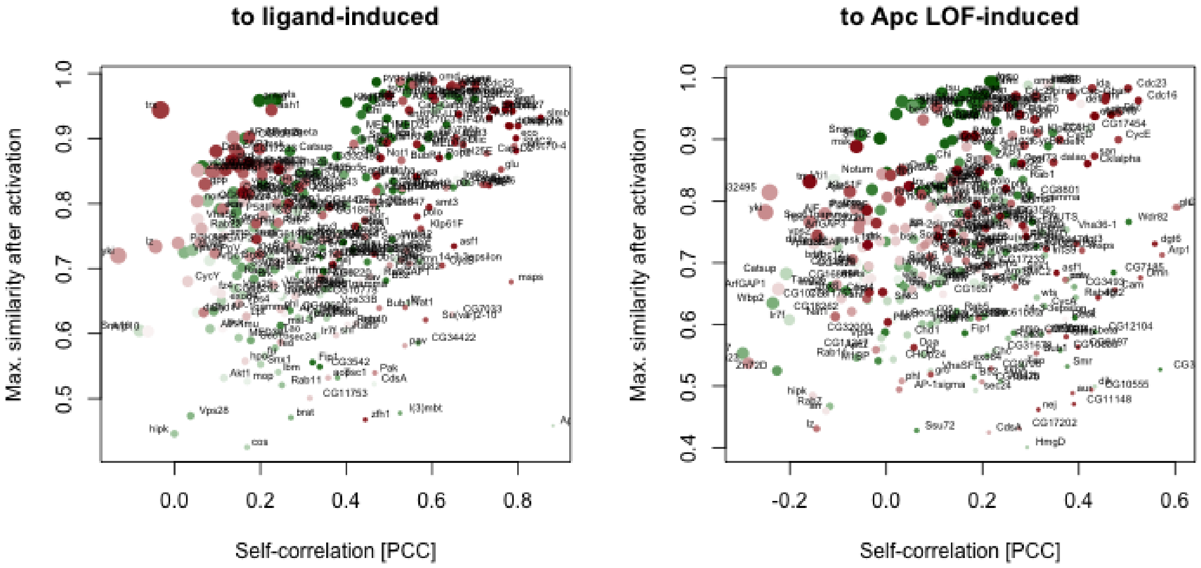
```
acrossHistExample(x = corAcross, states = 2,
                  pPair = c("Axn","nkd"), nPair = c("Axn","dsh"))
```



Change of functional similarity

Self-correlation between the baseline and induced state are plotted against the maximal similarity or dissimilarity (abs. PCC) of a given gene with any other gene in the respective induced state.

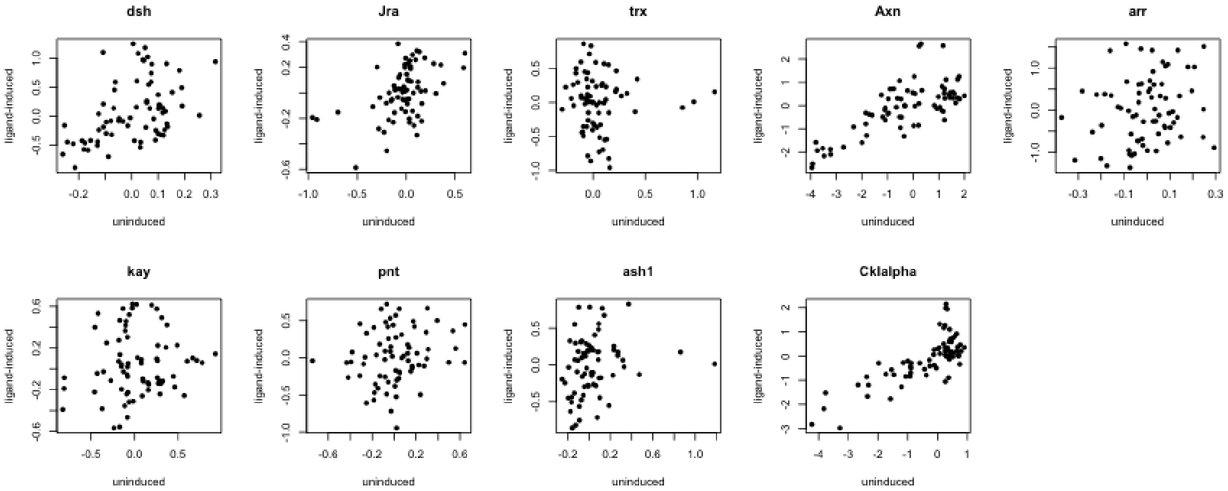
```
geneChange(X1 = singleKD, X2 = corSepGI, X3 = corAcross)
```



Change of gene-gene relations

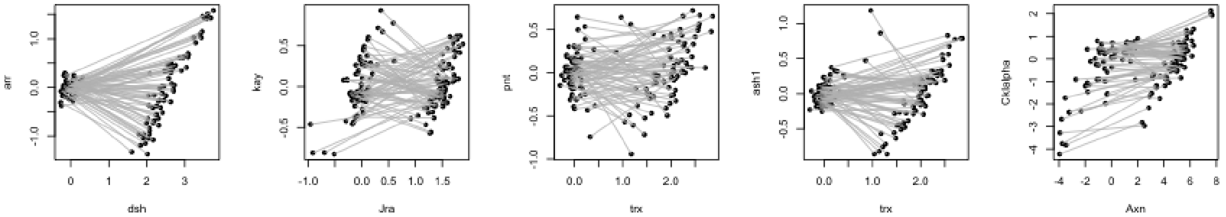
Examples show **self-correlation** for genes with **low** self-correlation.

```
geneChangeAcross(x = geneticInteractions,
                 a = c("dsh","Jra","trx","trx","Axn"),
                 b = c("arr","kay","pnt","ash1","CkIalpha"), states = 1:2)
```



The genetic interaction profile of a gene pair involving the above-shown low self-correlation genes in the baseline state (left in each plot) and the ligand-induced state (right in each plot) are compared.

```
geneChangeCor(x = geneticInteractions,
              a = c("dsh","Jra","trx","trx","Axn"),
              b = c("arr","kay","pnt","ash1","CkIalpha"))
```



Genetic vs. protein-protein interactions

Load filtered protein-protein interaction (PPI) data from *Guruharsha et al., Cell 2011*, the Drosophila Protein Interaction Map (DPIM), which was also performed in S2R+ cells, which contain only gene pairs where both partners were tested in our study.

```
data("dpimPPI_overlap", package="WntSGI")
```

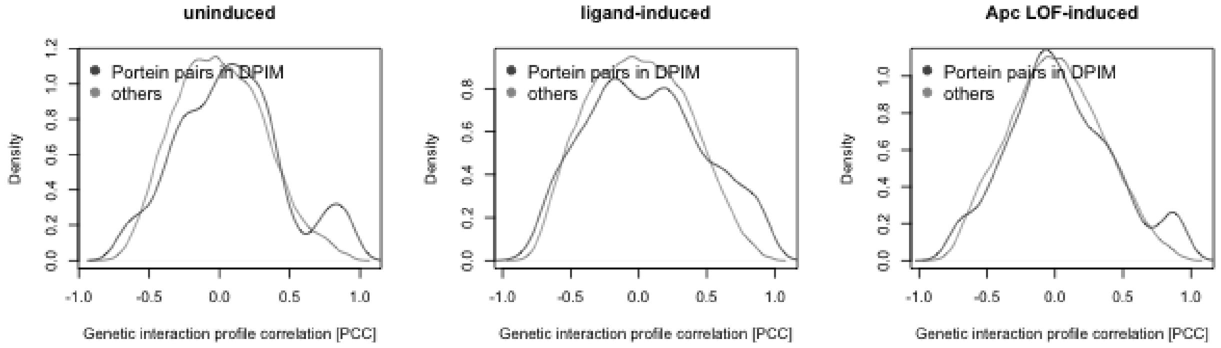
Generate binary array showing which gene pairs of our data had a PPI in the DPIM data.

```
ppi_binary <- array(FALSE, dim = dim(corSepGI[,,1]), dimnames = dimnames(corSepGI[,,1]))
for(i in 1:dim(dpimPPI_overlap)[1]) {
  ppi_binary[dpimPPI_overlap$Interactor_1[i], dpimPPI_overlap$Interactor_2[i]] <- TRUE
  ppi_binary[dpimPPI_overlap$Interactor_2[i], dpimPPI_overlap$Interactor_1[i]] <- TRUE
}
```

GI profile correlation of PPI gene pairs

Plot density of genetic interaction profile correlation between gene pairs that also show PPI vs pairs that do not share a PPI.

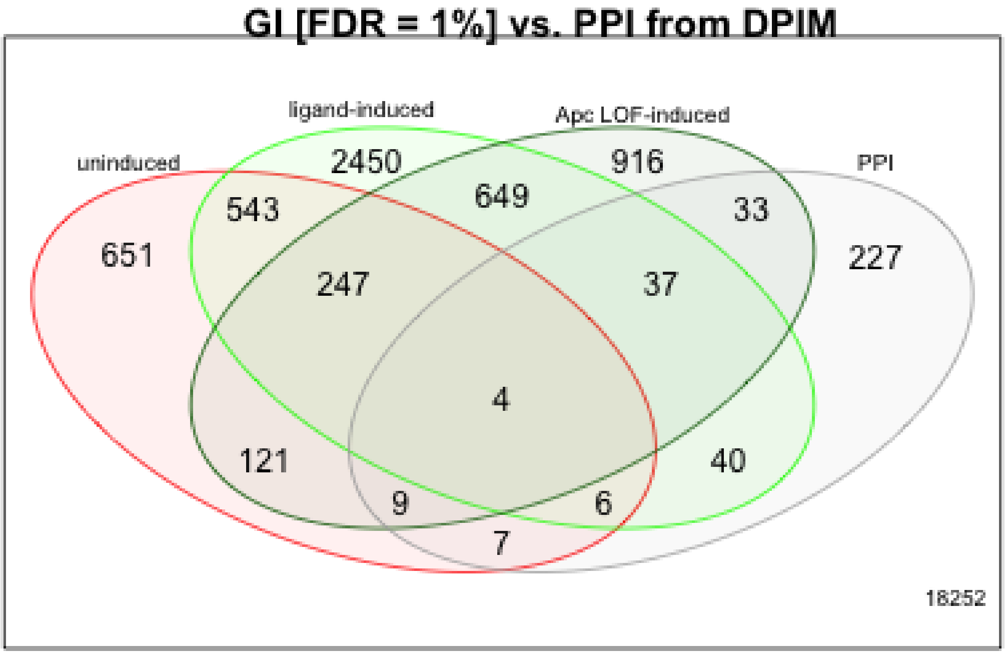
```
par.den <- par(mfrow=c(1,length(geneticBackground)))
for(i in 1:length(geneticBackground)) {
  a <- corSepGI[,,i]
  diag(a) <- NA

  plot(density(a, na.rm=TRUE), xlab="Genetic interaction profile correlation [PCC]",
       main=geneticBackground[i], col="#969696")
  lines(density(a[ppi_binary]), col="#525252")
  legend("topleft", c("Portein pairs in DPIM", "others"), col = c("#525252", "#969696"),
         cex=1.2, bty="n", pch = 19)
}
par(par.den)
```



GI vs. PPI

Illustrate overlap of significant genetic interactions (FDR of 1%) and PPI.

```
ppi_binary2 <- ppi_binary[,colnames(geneticInteractions)]

a <- abind(giFdr001[-ctrlMainEff,,], ppi_binary2, along = 3)

dim(a) <- c(prod(dim(a)[1:2]), dim(a)[3])
colnames(a) <- c(geneticBackground, "PPI")

vennDiagram(a, circle.col = c("red","green","darkgreen","darkgrey"), cex = 0.7)
title("GI [FDR = 1%] vs. PPI from DPIM")
```



Are gene pairs that genetically interact also enriched for protein-protein interactions?

```r
iJunction <- sum((a[,1] & a[,4]) | (a[,2] & a[,4]) | (a[,3] & a[,4]))
ppi <- sum(a[,4])
gi <- sum(a[,1] | a[,2] | a[,3])
allCombi <- prod(dim(geneticInteractions)[1:2]) - dim(geneticInteractions)[2]

fisher.test(matrix(c(iJunction, gi - iJunction,
                     ppi - iJunction, allCombi - gi - (ppi - iJunction)), nrow = 2))
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:
## p-value = 3.536e-09
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.563827 2.431447
## sample estimates:
## odds ratio
##   1.953057
```

Intra-module functional changes

Define clusters by computing the CSI

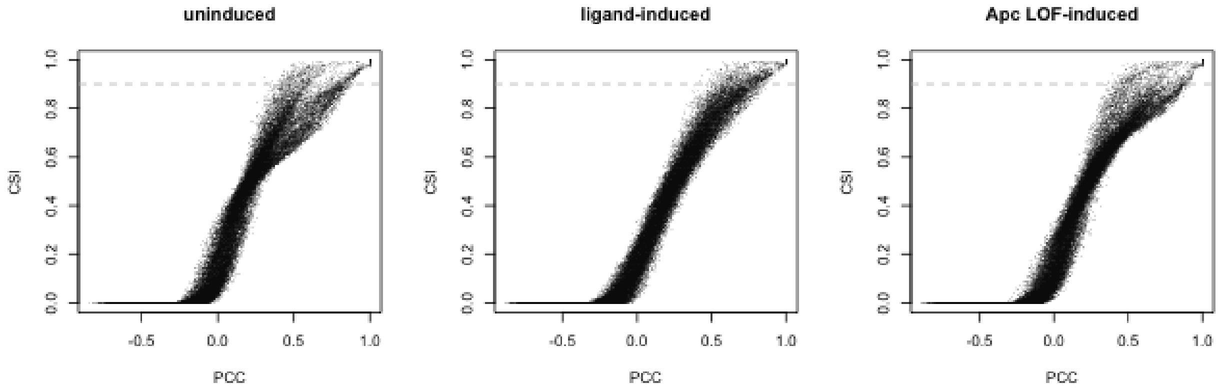
The connection specificity index (CSI) corrects the PCC profile of each gene with regard to the overall distribution of its PCCs. If a gene has many large PCCs each coefficient will be down-scaled.

Calculate the CSI.

```r
cf = 0.1
csi = corSepGI
for(j in 1:dim(csi)[3]) {
  for(k in 1:dim(csi)[1]) {
    for(i in 1:dim(csi)[2]) {
      csi[k,i,j] = 1 - ((length(unique(c(which(corSepGI[i,,j] >= corSepGI[k,i,j] - cf)
                                         , which(corSepGI[k,,j] >= corSepGI[k,i,j] - cf)
                                         )))) / dim(csi)[1])
    }
  }
}
```

How do PCC and CSI compare?

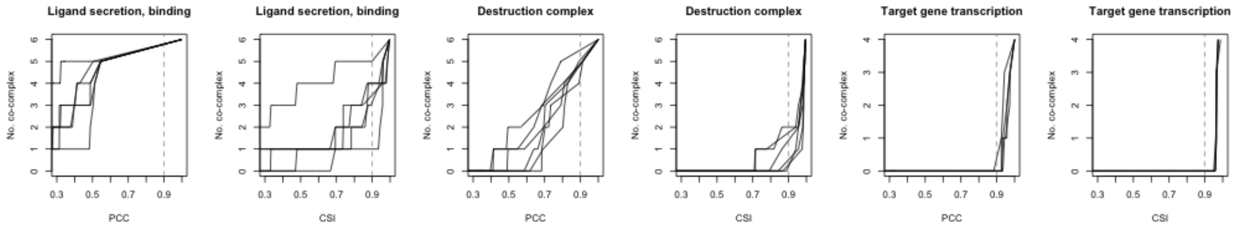
```r
par.csi <- par(mfrow = c(1, length(geneticBackground)))
for(i in 1:length(geneticBackground)) {
  a <- corSepGI[,,i]
  b <- csi[,,i]

  dim(a) <- c(prod(dim(a)[1:2]), 1)
  dim(b) <- c(prod(dim(b)[1:2]), 1)

  plot(a, b, pch = 16, cex=.1, xlab = "PCC", ylab = "CSI", main=geneticBackground[i])
  abline(h=.9, lty=2, col="grey")
}
par(par.csi)
```



The CSI enables modules detection

The CSI allows for better detection of known functional modules of the Wnt pathway using one global threshold of genetic interaction profile similarity (as compared to the PCC).

```
al <- list(c("wls","wg","CHOp24","fz2","dsh","arr"),
           c("Axn","CkIalpha","slmb","skpA","lin19","sgg"),
           c("arm","pan","lgs","pygo"))

par.csi <- par(mfrow = c(1, length(al) * 2))
for(m in 1:length(al)) {
  a <- al[[m]]
  for(j in 1:2) {
    for(k in 1:length(a)) {
      goi = a[k]
      x <- list(corSepGI, csi)[[j]]
      d <- sort(x[goi,,1])
      b <- rep(0, times=length(d))

      i = 1
      if(names(d)[i] %in% a) {
        b[i] <- 1
      }

      for(i in 2:length(b)) {
        if(names(d)[i] %in% a) {
          b[i] <- b[i-1] + 1
        } else {
          b[i] <- b[i-1]
        }
      }

      if(k == 1) {
        plot(d, b, xlim = c(0.3, 1), xlab = c("PCC", "CSI")[j], ylab = "No. co-complex",
             type = "l", main = c("Ligand secretion, binding","Destruction complex",
                                  "Target gene transcription")[m])
        abline(v=.9, lty=2, col="grey")
      } else {
        lines(d, b)
      }
    }
  }
}
par(par.csi)
```



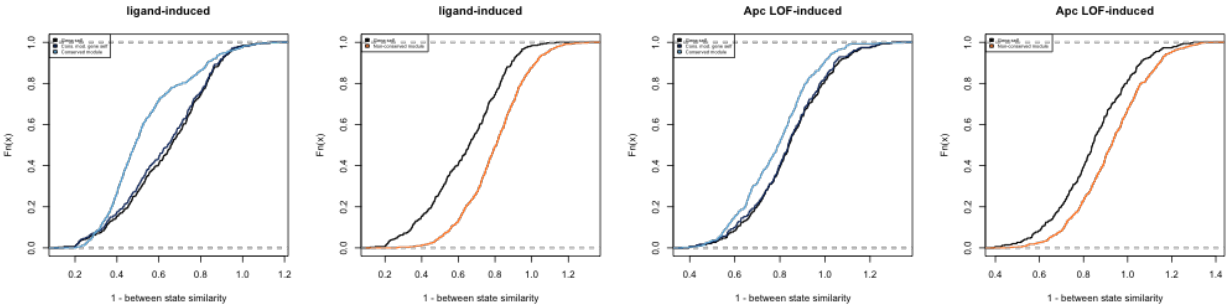
Identify functional modules

Define genes that are in the same functional module by applying a `CSI > 0.9` in the baseline and respective activated pathway state.

```
cs = csi
modGene = array(NA, dim=c(dim(cs)[1], dim(cs)[3]))
consModGene = array(NA, dim=c(dim(cs)[1], 2))
for(i in 1:dim(cs)[3]) {
  diag(cs[,,i]) <- NA
  for(j in 1:dim(cs)[1]) {
    modGene[j,i] <- any(cs[j,,i] > 0.9, na.rm=TRUE)
  }
}
for(i in 2:dim(cs)[3]) {
  for(j in 1:dim(cs)[1]) {
    consModGene[j,i-1] <- all(modGene[j,c(1,i)])==TRUE
  }
}
```

Compare how similar genes in a **conserved module** are to co-members between the states and test whether this similarity is higher than the self-correlation of those same genes. Next, the same comparison was done for functional connections that were **not conserved** in the specific states (as determined by a CSI > 0.9 in one state and < 0.7 in another state).

```
conservedVSnonModules(X1 = csi, X2 = corAcross)
```



Functional similarity within the destruction complex

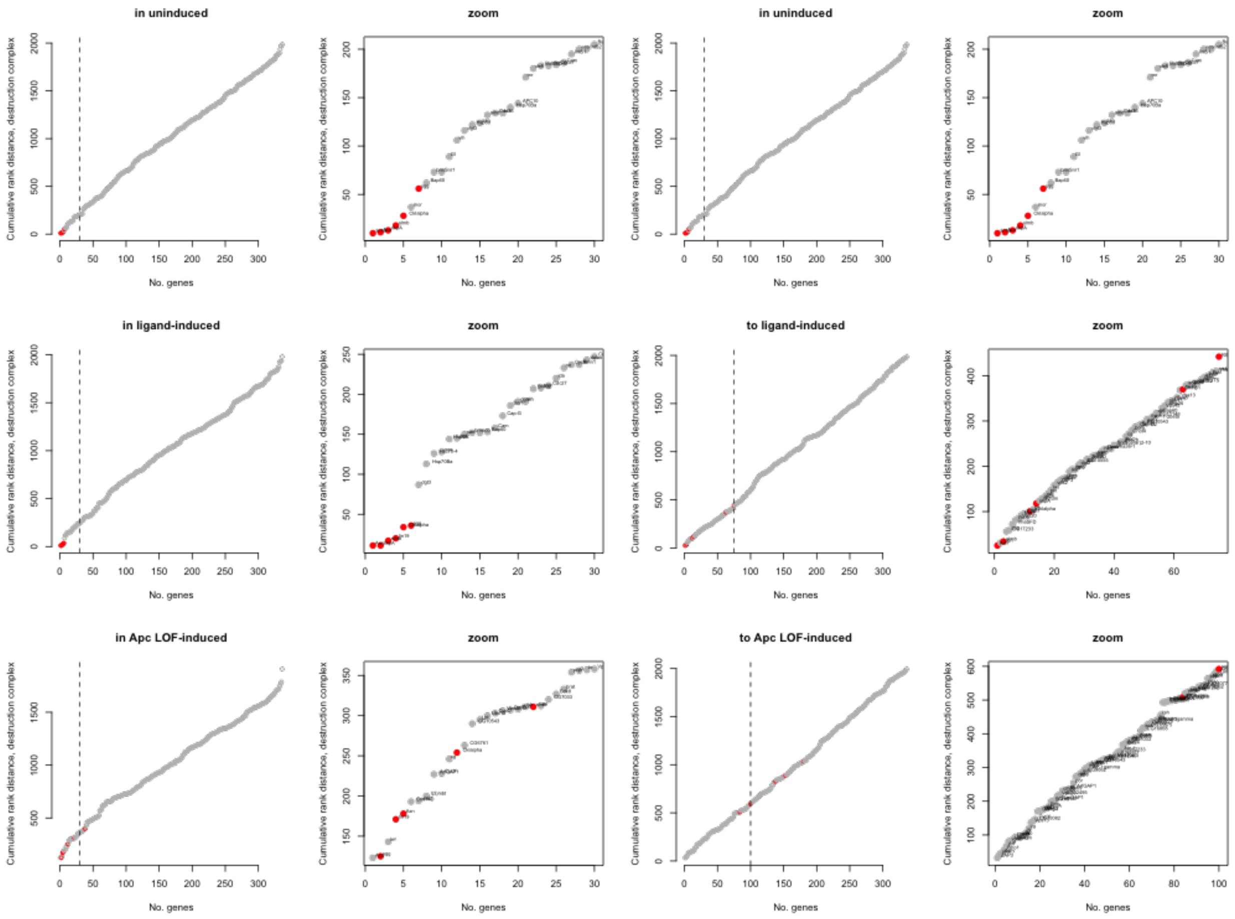
The functional similarity within e.g. the destruction complex can be determined by *i)* ranking the PCC of all genes with each gene from the complex, *ii)* add the ranks per gene to obtain the distance from the given complex. This can be done for **each pathway state** or for the **between state similarities**.

Calculate the cumulative rank distance.

```
sCl = c("Axn","CkIalpha","sgg","slmb","skpA","lin19")
cn = colnames(corSepGI)
rankComp = array(data=NA, dim=c(length(sCl), length(cn), length(geneticBackground), 2),
                 dimnames = list(sCl, cn, geneticBackground,c("within","between")))

for(i in 1:length(geneticBackground)) {
  a = corSepGI[sCl,,i]
  b = corAcross[sCl,,1,i]
  for(j in 1:length(sCl)) {
    for(k in 1:length(cn)) {
      rankComp[j,k,i,"within"] <- length(which(a[sCl[j],] > a[sCl[j],cn[k]]))
      rankComp[j,k,i,"between"] <- length(which(b[sCl[j],] > b[sCl[j],cn[k]]))
    }
  }
}
```

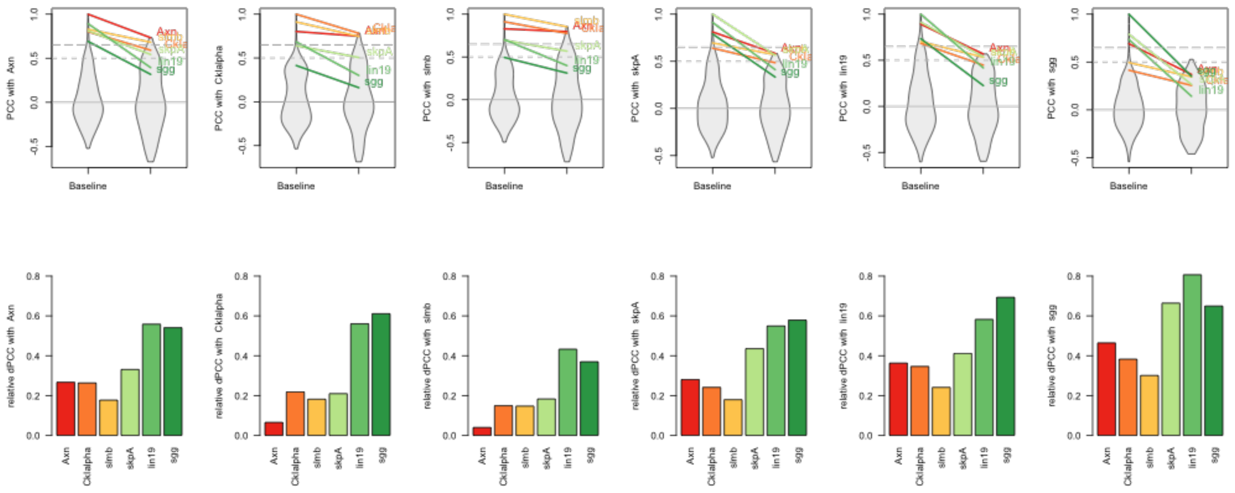
Plot the cumulative rank distance.

```
par.rank <- par(mfrow = c(3,4))
for(j in 1:dim(rankComp)[3]) {
  for(i in 1:dim(rankComp)[4]) {
    plotCumuRank(a = apply(rankComp[,,j,i],2,sum))
  }
}
par(par.rank)
```



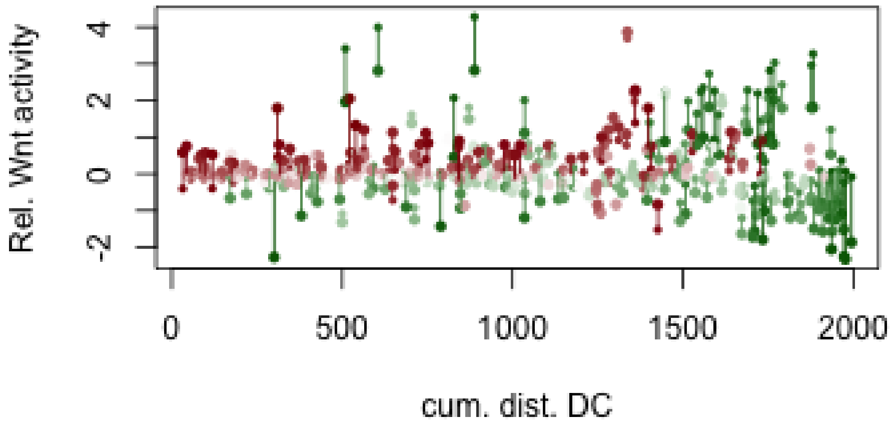
Baseline vs between-state in destruction complex

```r
acrossStateDivergence(x1 = corSepGI, x2 = corAcross,
                      n = c("Axn","CkIalpha","slmb","skpA","lin19","sgg"))
```



Sorted single gene effect differences after induction

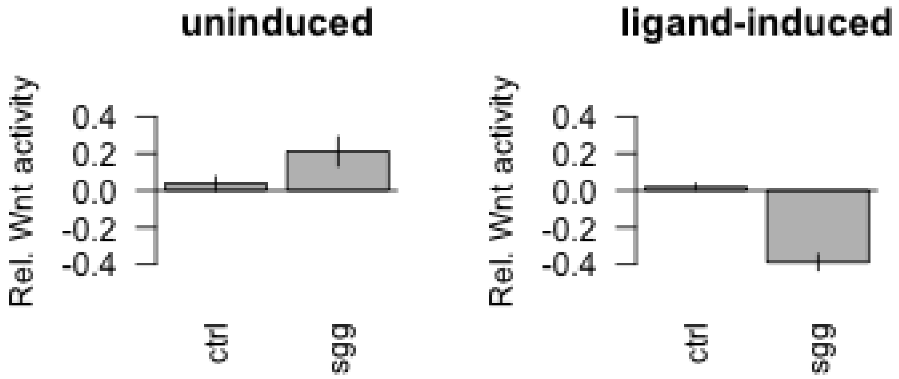
```
dcDistancePheno(a = apply(rankComp[,,j,i],2,sum),
                Y1 = singleKD[-ctrlMainEff,1], Y2 = singleKD[-ctrlMainEff,2])
```



Single gene effect of sgg upon induction

sgg (human: GSK3beta) is a component of the destruction complex, whose knockdown surprisingly showed a decrease of pathway activity (in Wnt-active state).

```
relWntGOI(x1 = singleKD, x2 = sdSingleKD, goi = "sgg", states = 1:2)
```



Genetic interaction networks of KNOWN Wnt signaling components

State-dependent network of core Wnt signaling components

Plot **network** including **Wnt receptors** and **transcription factors** as well as **destruction complex** members to illustrate how **known functional relations** can be displayed across the **3 distinct pathway states**.

Define genes in groups for plotting.

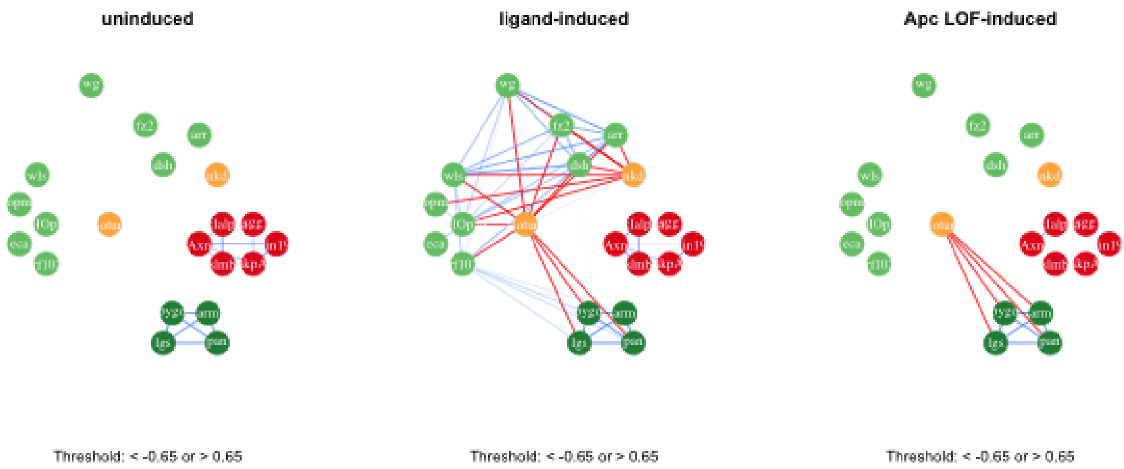
```
sn1 = c("CHOp24","opm","eca","Arf102F","wls","wg","fz2","arr","dsh")
sn2 = c("nkd","Notum")
sn3 = c("Axn","CkIalpha","sgg","slmb","skpA","lin19")
sn4 = c("arm","pan","lgs","pygo")

sn = c(sn1,sn2,sn3,sn4)
snl = list(sn1,sn2,sn3,sn4)
snCol = c(rep("#74c476", times=length(sn1)), rep("#feb24c", times=length(sn2)),
          rep("#e31a1c", times=length(sn3)), rep("#238b45", times=length(sn4)))
```

Define **fixed location** of **verteces** to visualize the reconstruction of functional relations between well-described Wnt signaling components.

```
x = c(1.5,0,0, 1.5, 1, 4, 7, 10, 8, 11, 5, 10, 11.3, 13, 11.3, 13, 14.3, 10.5, 11, 8, 8.5)
xd = c(x, x + 22)
y = c(5, 6, 4, 3, 7.5, 12, 10, 9.5, 8, 7.5, 5, 4, 5, 5, 3, 3, 4, 0.5, -1, -1, 0.5)
yd = c(y, y)
lo = cbind(x, y)
lod = cbind(xd, yd)
```

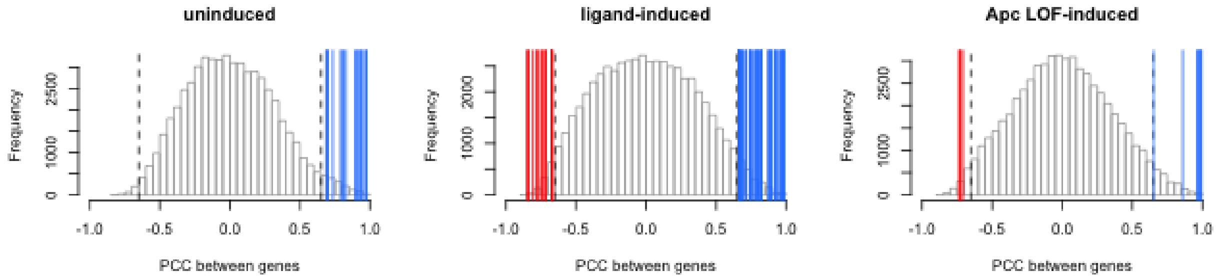
Draw **networks** by connecting known components of the Wnt signaling route by their positive (blue) or negative (red) **state-specific** genetic interaction profile corelation. The layout is fixed. Note state-specific edges.

```
wntCoreNw(sn, snl, snCol, lo, geneticBackground, corSepGI, thresh = 0.65, vsize = 18)
```



Visualize distribution of edges (PCC) between selected known Wnt signaling components (see above) in context of edges (PCC) between all gene pairs in each of the three pathway states.

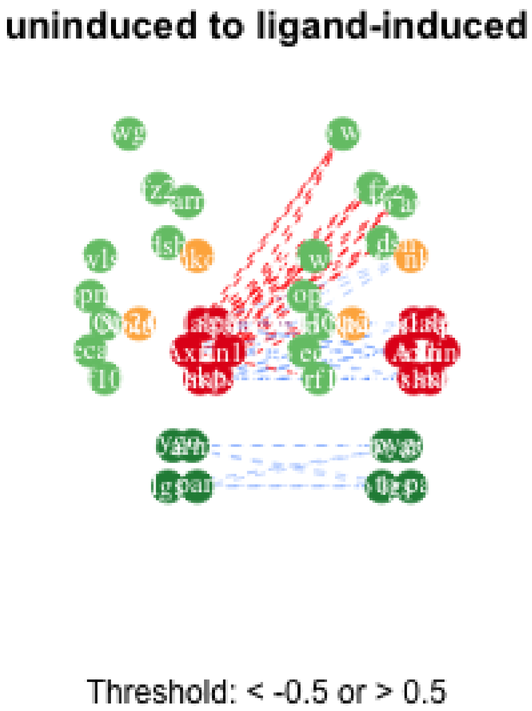
```r
histStateCore(x = corSepGI)
```



Between-state network of core Wnt signaling components

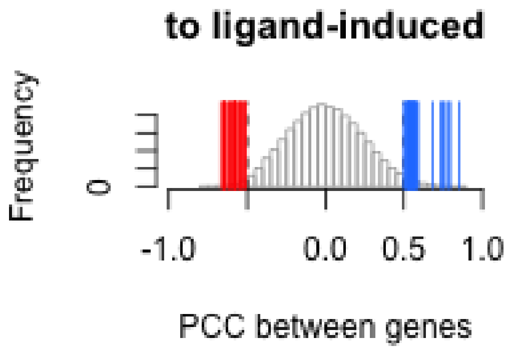
Plot **network** including **Wnt receptors** and **transcription factors** as well as **destruction complex** members to illustrate how **known functional relations** can be displayed between **baseline** and **Wg-induced states**.

```r
wntCoreNwBetween(sn, snl, snCol, lo=lod, from="uninduced", to="ligand-induced",
                 corAcross, thresh = 0.5, vsize = 18)
```



Visualize distribution of edges (PCC) between selected known Wnt signaling components (see above) in context of edges (PCC) between all gene pairs between the baseline and Wnt-active pathway state.

```r
acrossHistCore(x1 = corAcross)
```



Genetic interaction networks of NEW Wnt signaling components

New regulators at receptor level

In this multi-step process, *i)* **bait genes** are defined. Those are genes involved in ligand secretion and receptor binding. Next, *ii)* genes with high specific similarity to the bait genes are identified in the ligand-induced CSI-corrected data.

```
thresh = 0.9
ist = 2
recLevel = list()
baitG = c("CHOp24","opm","wls","fz","fz2","arr","dsh")
recLevel = names(which(csi[baitG[1],,ist] > thresh | csi[baitG[2],,ist] > thresh |
                         csi[baitG[3],,ist] > thresh | csi[baitG[4],,ist] > thresh |
                         csi[baitG[5],,ist] > thresh | csi[baitG[6],,ist] > thresh |
                         csi[baitG[7],,ist] > thresh))
```

Upon ligand binding, the receptors inhibit the destruction complex. Feedback regulators such as *nkd* take over destruction complex baseline state-function in the ligand-induced state. Therefore, *iii)* genes with **between-state** similarity to baseline destruction complex function are identified.

```
thresh = 0.5
ist = 2
negR = list()
negR$WntActive = names(which(corAcross["Axn",,1,ist] > thresh |
                               corAcross["CkIalpha",,1,ist] > thresh |
                               corAcross["slmb",,1,ist] > thresh))
ist = 3
negR$ApcLoss = names(which(corAcross["Axn",,1,ist] > thresh |
                             corAcross["CkIalpha",,1,ist] > thresh |
                             corAcross["slmb",,1,ist] > thresh))
```

. . . those genes should *iiii)* also neg. correlate with any bait gene.

```
csep = corSepGI[negR$WntActive, baitG,]

negRecL = array(NA, dim=c(dim(csep)[1], dim(csep)[3]))
for(i in 1:dim(csep)[3]) {
  for(j in 1:dim(csep)[1]) {
    negRecL[j,i] <- any(csep[j,,i] < -0.65, na.rm=TRUE)
  }
}
```

Conclusion: of 42 genes that take over DC function in the Wnt-active state, 27 neg. corr. with the 'baitG' and NONE do so in the two other states.

Finally, *v)* genes that share high similarity with newly identified genes can be added using a CSI > 0.9. For example:

```
goi="Sec61alpha"; csi[goi,,2][which(csi[goi,,2] > 0.9)]
```

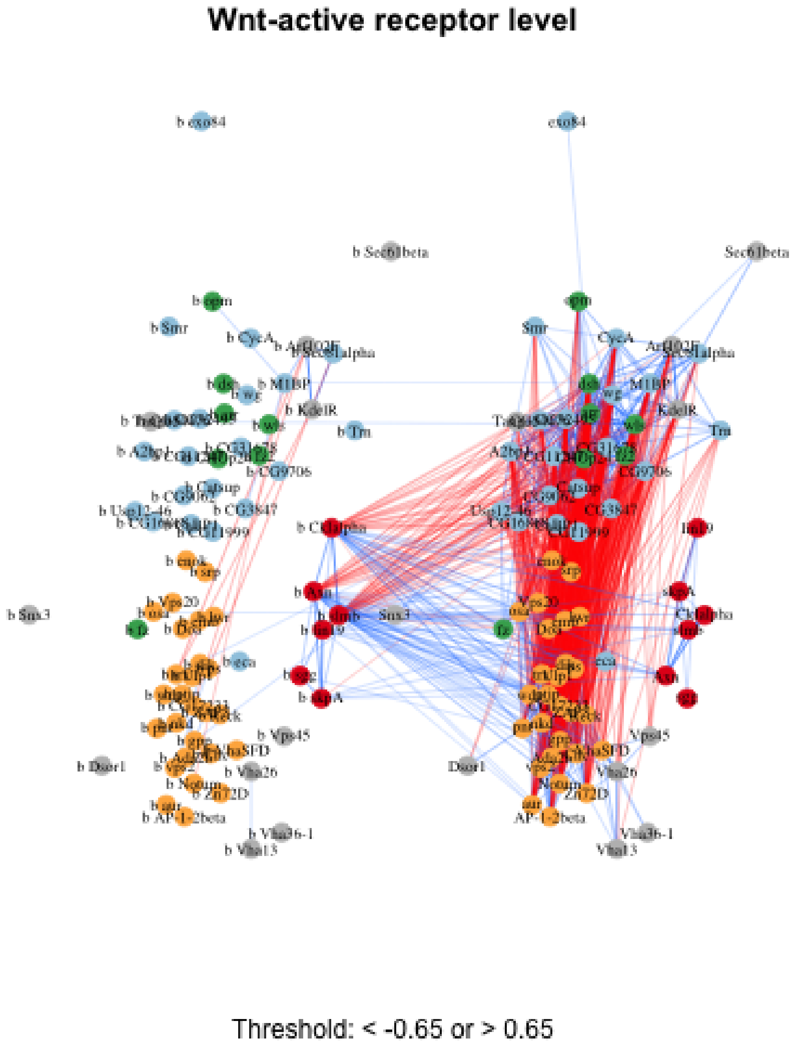
```
##    Arf102F        dsh      KdelR       M1BP Sec61alpha  Sec61beta 
##  0.9404762  0.9047619  0.9166667  0.9613095  0.9970238  0.9523810 
##        wls 
##  0.9077381
```

Define gene groups for drawing the network.

```
sn1 = baitG
sn2 = recLevel[(recLevel %in% baitG) == FALSE]
sn3 = c("Arf102F","KdelR","Sec61beta","Tango5","Snx3") #extended pos. reg.
sn4 = negR$WntActive[negRecL[,2]]
sn5 = c("Vha13","Vha26","Vha36-1","Vps45","Dsor1") #extended neg. reg.
sn6 = c("Axn","CkIalpha","sgg","slmb","skpA","lin19")

sn = c(sn1,sn2,sn3,sn4,sn5,sn6)
snB = paste("b",sn)
snl = list(sn1,sn2,sn3,sn4,sn5,sn6)
snlB = list(paste("b",sn1), paste("b",sn2), paste("b",sn3), paste("b",sn4),
            paste("b",sn5), paste("b",sn6))
snlCol = c("#41ab5d","#9ecae1","#bdbdbd","#feb24c","#bdbdbd","#e31a1c")
```

Draw network using a igraph-based function.

```
wntActiveModules(corSepGI, corAcross, pStatel =1, pState2 = 2, sn = sn, snl=snl,
                 thresh = 0.65, dthresh = 0.5, thCF=c(0.1,2.5), dthCF=c(0.25,0.8),
                 xShift1=13.2, yShift1=-12.4, xCo=-7, xShift2=9.8, yShift2=0.3)
```



New regulators at the TF level

Following a similar multi-step process, *i)* **bait genes** are defined. Those are genes involved in Wnt signaling target gene transcription. Next, *ii)* genes with high specific similarity to the bait genes are identified in the induced pathway CSI-corrected data.

```
thresh = 0.9
ist = 2
tfLevel = list()
baitG = c("arm","pan","pygo","lgs")
tfLevel$WntActive = names(which(csi[baitG[1],,ist] > thresh |
                                  csi[baitG[2],,ist] > thresh |
                                  csi[baitG[3],,ist] > thresh |
                                  csi[baitG[4],,ist] > thresh))
ist=3
tfLevel$ApcLoss = names(which(csi[baitG[1],,ist] > thresh |
                                csi[baitG[2],,ist] > thresh |
                                csi[baitG[3],,ist] > thresh |
                                csi[baitG[4],,ist] > thresh))
```

Which genes take over DC function upon induction

```
thresh = 0.5
ist = 2
negR = list()
negR$WntActive = names(which(corAcross["Axn",,1,ist] > thresh |
                               corAcross["CkIalpha",,1,ist] > thresh |
                               corAcross["slmb",,1,ist] > thresh))
ist = 3
negR$ApcLoss = names(which(corAcross["Axn",,1,ist] > thresh |
                             corAcross["CkIalpha",,1,ist] > thresh |
                             corAcross["slmb",,1,ist] > thresh))
```

Conjunction in Wnt high and Apc loss states

Which genes take over baseline destruction complex function in **BOTH activated states**.

```
coOccur = negR$ApcLoss[which(negR$ApcLoss %in% negR$WntActive)]
```

Which genes that take over DC function also neg. correlate with any bait gene.

```
csepCo = corSepGI[coOccur,
                  tfLevel$WntActive[which(tfLevel$WntActive %in% tfLevel$ApcLoss)],]

negRecL = array(NA, dim=c(dim(csepCo)[1], dim(csepCo)[3]))
for(i in 1:dim(csepCo)[3]) {
  for(j in 1:dim(csepCo)[1]) {
    negRecL[j,i] <- any(csepCo[j,,i] < -0.65, na.rm=TRUE)
  }
}
coOccGenes = coOccur[negRecL[,3] & negRecL[,2]] #genes that take over in BOTH states
```

Exclusive in Wnt active state

Which genes take over upon **WNT activation ONLY**.

```
exclWnt = negR$WntActive[-which(negR$WntActive %in% negR$ApcLoss)]
```

Which genes that take over DC function also neg. correlate with any bait gene.

```
csepW = corSepGI[exclWnt, baitG,]
negRecL = array(NA, dim=c(dim(csepW)[1], dim(csepW)[3]))
for(i in 1:dim(csepW)[3]) {
  for(j in 1:dim(csepW)[1]) {
    negRecL[j,i] <- any(csepW[j,,i] < -0.65, na.rm=TRUE)
  }
}
exclWntGenes = exclWnt[negRecL[,2] & negRecL[,3] == FALSE]
```

Exlcusive in Apc loss active state

Which genes take over upon **APC loss activation ONLY**.

```
exclApc = negR$ApcLoss[-which(negR$ApcLoss %in% negR$WntActive)]
```

Which genes that take over DC function also neg. correlate with any bait gene.

```
csepA = corSepGI[exclApc, baitG,]
negRecL = array(NA, dim=c(dim(csepA)[1], dim(csepA)[3]))
for(i in 1:dim(csepA)[3]) {
  for(j in 1:dim(csepA)[1]) {
    negRecL[j,i] <- any(csepA[j,,i] < -0.65, na.rm=TRUE)
  }
}
exclApcGenes = exclApc[negRecL[,3]] # & negRecL[,2] == FALSE]
```

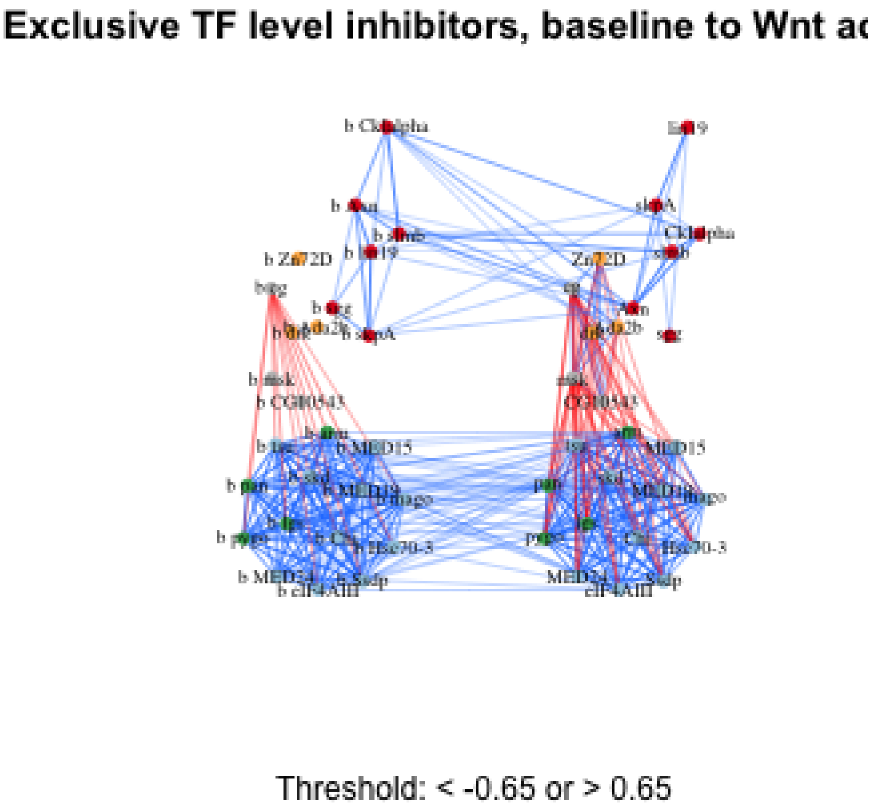
TF-level network between baseline and Wnt-induced pathway, state-EXCLUSIVE inhibitors

Define gene groups for drawing the network.

```
sn1 = baitG
sn2 = tfLevel$WntActive[(tfLevel$WntActive %in% baitG) == FALSE]
sn3 = c()
sn4 = exclWntGenes # state-relevant
sn5 = c("CG10543","msk","cg")
sn6 = c("Axn","CkIalpha","sgg","slmb","skpA","lin19")

sn = c(sn1,sn2,sn3,sn4,sn5,sn6)
snB = paste("b",sn)
snl = list(sn1,sn2,sn3,sn4,sn5,sn6)
snlB = list(paste("b",sn1), paste("b",sn2), paste("b",sn3), paste("b",sn4),
          paste("b",sn5), paste("b",sn6))
snlCol = c("#41ab5d","#9ecae1","#bdbdbd","#feb24c","#bdbdbd","#e31a1c")
```

Draw network using a igraph-based function.

```
wntActiveModules(corSepGI, corAcross, pStatel =1, pState2 = 2, sn = sn, snl=snl,
                 thresh = 0.65, dthresh = 0.5, thCF=c(0.15,2.5), dthCF=c(0.2,1.5),
                 xShift1=-7.5, yShift1=-10, xCo=-1, xShift2=-18, yShift2=0.3,
                 title="Exclusive TF level inhibitors, baseline to Wnt active")
```



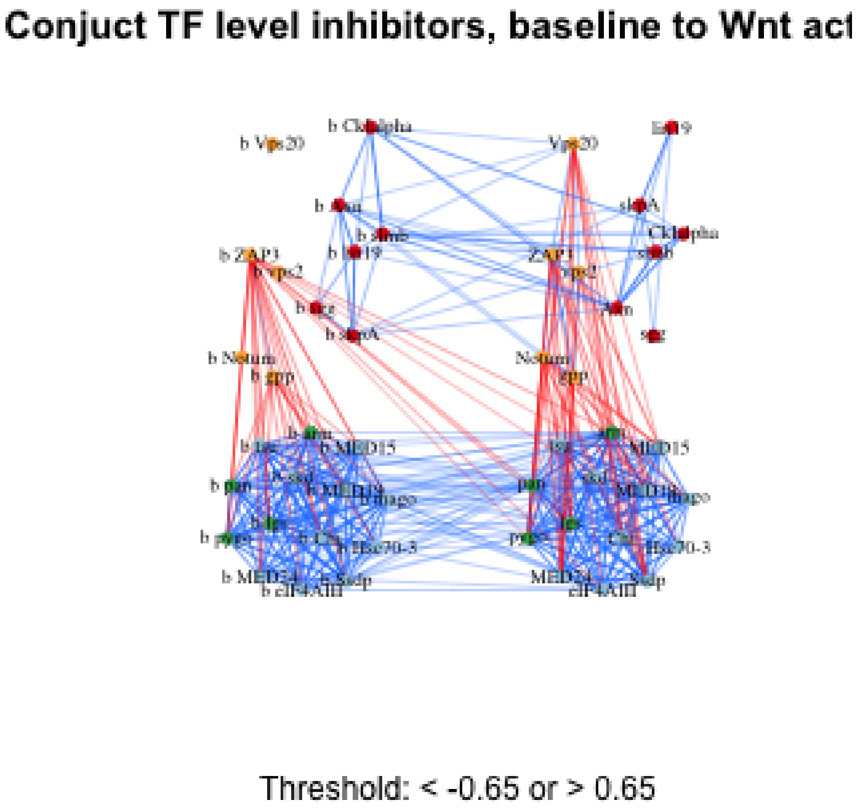
TF-level network between baseline and Wnt-induced pathway, CONJUCTION inhibitors

Define gene groups for drawing the network.

```r
sn4 = coOccGenes
sn5 = c()

sn = c(sn1,sn2,sn3,sn4,sn5,sn6)
snB = paste("b",sn)
snl = list(sn1,sn2,sn3,sn4,sn5,sn6)
snlB = list(paste("b",sn1), paste("b",sn2), paste("b",sn3), paste("b",sn4),
            paste("b",sn5), paste("b",sn6))
```

Draw network using a igraph-based function.

```r
wntActiveModules(corSepGI, corAcross, pState1 =1, pState2 = 2, sn = sn, snl=snl,
                 thresh = 0.65, dthresh = 0.5, thCF=c(0.15,2.5), dthCF=c(0.2,1.5),
                 xShift1=-7.5, yShift1=-10, xCo=-1, xShift2=-12, yShift2=-3.7,
                 title="Conjuct TF level inhibitors, baseline to Wnt active")
```



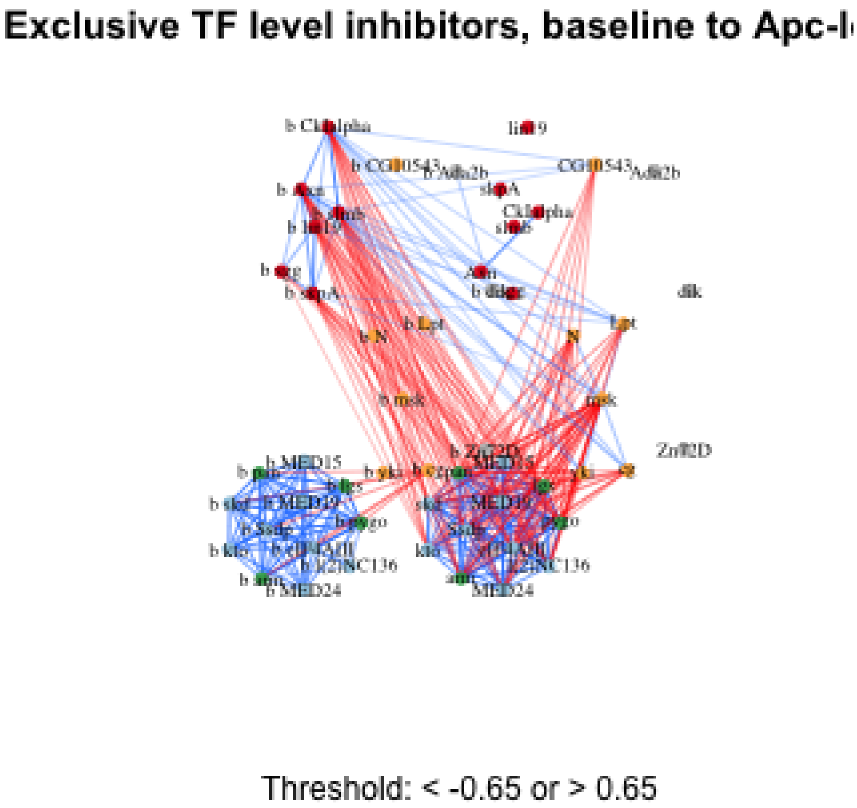
TF-level network between baseline and Apc loss-induced pathway, state-EXCLUSIVE inhibitors

Define gene groups for drawing the network.

```
sn1 = baitG
sn2 = tfLevel$ApcLoss[(tfLevel$ApcLoss %in% baitG) == FALSE]
sn3 = c()
sn4 = c("CG10543","msk","cg","yki","N","Lpt") # state-relevant
sn5 = c("Ada2b","dik","Zn72D")
sn6 = c("Axn","CkIalpha","sgg","slmb","skpA","lin19")

sn = c(sn1,sn2,sn3,sn4,sn5,sn6)
snB = paste("b",sn)
snl = list(sn1,sn2,sn3,sn4,sn5,sn6)
snlB = list(paste("b",sn1), paste("b",sn2), paste("b",sn3), paste("b",sn4),
            paste("b",sn5), paste("b",sn6))
snlCol = c("#41ab5d","#9ecae1","#bdbdbd","#feb24c","#bdbdbd","#e31a1c")
```

Draw network using a igraph-based function.

```
wntActiveModules(corSepGI, corAcross, pState1 = 1, pState2 = 3, sn = sn, snl=snl,
                 thresh = 0.65, dthresh = 0.5, xCo=-2, thCF = c(0.15, 2.5),
                 dthCF = c(0.4, 0.7), xShift1 = -5, yShift1 = -7.5, xShift2=-33,
                 yShift2=-67, title = "Exclusive TF level inhibitors, baseline to Apc-loss")
```


Exclusive TF level inhibitors, baseline to Apc-l
b MED15
b MED19
b MED24
MED19
MED24
dlk
Threshold: < -0.65 or > 0.65

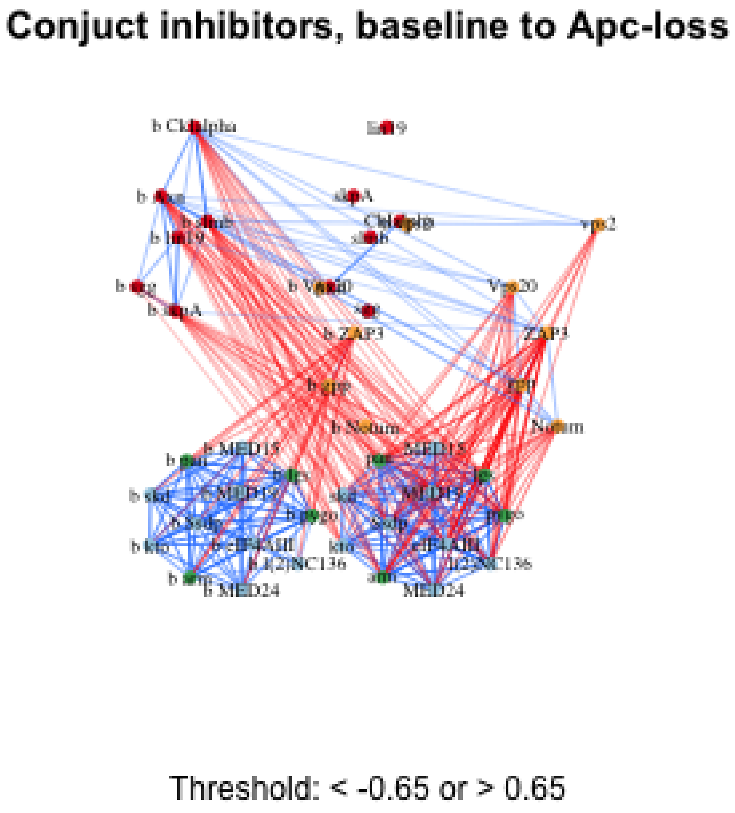
TF-level network between baseline and Apc loss-induced pathway, CONJUCTION inhibitors

Define gene groups for drawing the network.

```
sn4 = coOccGenes
sn5 = c()

sn = c(sn1,sn2,sn3,sn4,sn5,sn6)
snB = paste("b",sn)
snl = list(sn1,sn2,sn3,sn4,sn5,sn6)
snlB = list(paste("b",sn1), paste("b",sn2), paste("b",sn3), paste("b",sn4),
            paste("b",sn5), paste("b",sn6))
```

Draw network using a igraph-based function.

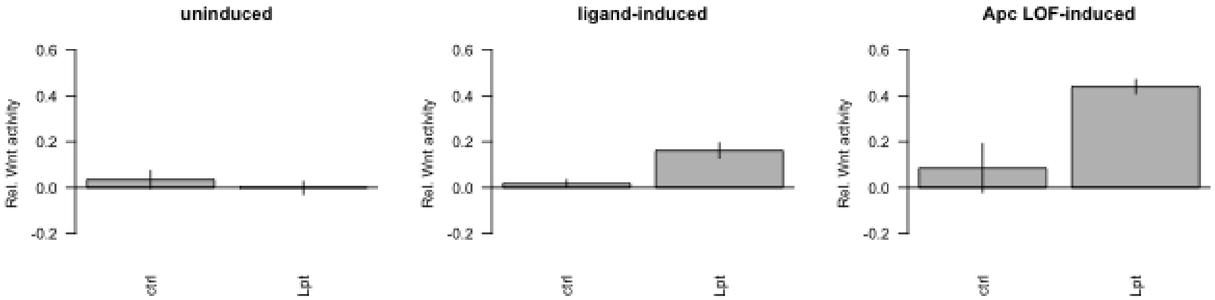
```
wntActiveModules(corSepGI, corAcross, pState1 = 1, pState2 = 3, sn = sn, snl=snl,
                 thresh = 0.65, dthresh = 0.5, xCo=-2, thCF = c(0.15, 2.5),
                 dthCF = c(0.4, 0.7), xShift1 = -7, yShift1 = -8, xShift2=-15.5,
                 yShift2=-7, title = "Conjuct inhibitors, baseline to Apc-loss")
```



Effects of Lpt

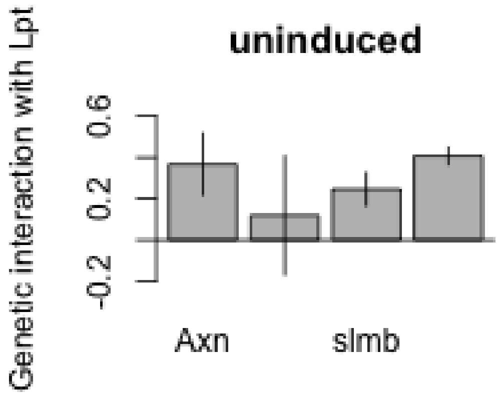
Single gene effect of Lpt on Wnt signaling activity in baseline and the induced states.

```
relWntGOI(x1 = singleKD, x2 = sdSingleKD, goi = "Lpt", states = 1:3, yl = c(-0.3,0.6))
```



Genetic interactions of Lpt with destruction complex components in the baseline state.

```
giGOIpartners(x1 = geneticInteractions, x2 = giSD, goi = "Lpt",
              partners = c("Axn","CkIalpha","slmb","skpA"), states = 1)
```



END OF VIGNETTE

```
sessionInfo()
```

```
## R version 3.3.1 (2016-06-21)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.11.6 (El Capitan)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] WntSGI_0.0.1         Hmisc_4.0-2          ggplot2_2.2.1
##  [4] Formula_1.2-1        survival_2.40-1      geneplotter_1.50.0
##  [7] annotate_1.50.0      XML_3.98-1.4         AnnotationDbi_1.34.4
## [10] IRanges_2.6.1        S4Vectors_0.10.3     lattice_0.20-33
## [13] Biobase_2.32.0       BiocGenerics_0.18.0  calibrate_1.7.2
## [16] MASS_7.3-45          vioplot_0.2          sm_2.2-5.4
## [19] igraph_1.0.1         plotrix_3.6-3        boot_1.3-18
## [22] limma_3.28.20        LSD_3.0              RColorBrewer_1.1-2
## [25] gplots_3.0.1         abind_1.4-5          knitr_1.14
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.6         gtools_3.5.0        assertthat_0.1
##  [4] rprojroot_1.2       digest_0.6.10       plyr_1.8.4
##  [7] backports_1.0.5     acepack_1.4.1       RSQLite_1.0.0
## [10] evaluate_0.9        lazyeval_0.2.0      data.table_1.10.0
## [13] gdata_2.17.0        rpart_4.1-10        Matrix_1.2-6
## [16] checkmate_1.8.2     rmarkdown_1.5       splines_3.3.1
## [19] stringr_1.1.0       foreign_0.8-66      htmlwidgets_0.8
## [22] munsell_0.4.3       base64enc_0.1-3     htmltools_0.3.5
## [25] nnet_7.3-12         tibble_1.2          gridExtra_2.2.1
## [28] htmlTable_1.9       codetools_0.2-14    bitops_1.0-6
## [31] grid_3.3.1          xtable_1.8-2        gtable_0.2.0
## [34] DBI_0.5-1           magrittr_1.5        formatR_1.4
## [37] scales_0.4.1        KernSmooth_2.23-15  stringi_1.1.1
## [40] latticeExtra_0.6-28 tools_3.3.1         beeswarm_0.2.3
## [43] yaml_2.1.13         colorspace_1.2-6    cluster_2.0.4
## [46] caTools_1.17.1
```