Statistical Analysis of Pathobiochemical Signatures in Bile Duct Ligated Mice

Matthias Koenig (2015-09-04)

Contents

Introduction

This document contains the statistical analysis performed in the publication *Pathobiochemical signatures of cholestatic liver disease in bile duct ligated mice.*

A comprehensive data set of serum markers, histological parameters and transcript profiles was compiled at 8 time points after bile duct ligation (BDL) in mice, comprising different stages of the disease. The data set consists of $N_r = 5$ repeats ($N_r = 3$ for the measured antibodies) for $N_t = 8$ time points denoted by $t_1, ..., t_{N_t}$ consisting of a total $N_f = 154$ measured parameters in the following referred to as factors (Fluidigm gene expression, antibodies, serum markers, histological measurements).

The main steps of the analysis comprise

- **Explorative data anaysis**
- **Dimension reduction via ANOVA**
- **Correlation analysis** based on time course correlation measure
- **Hierarchical clustering**
- **Decision trees** for prediction

The complete data set, source code and documentation of this analysis is available from https://github.com/matthiaskoenig/bdl-analysis .

The following naming conventions are used

- **factor** : one of the measured quantities/parameters over time, i.e. either
  - gene expression of a single gene (e.g. Actb);
  - one of the biomarkers (e.g. ALT, albumin, bilirubin)
  - one of the histological markers (e.g. BrdU-positive Kupffer cells)
  - one of the antibodies (e.g. CTGF, S100A4)
- **time point** : a single value t_i from the measured time points 0h (control), 6h, 12h, 18h, 30h, 2d, 5d, 14d
- **sample** : one of the $N_t \cdot N_r = 40$ mice, i.e. a one of the repeats for a given time point

The results of the analysis are written to the results directory defined via the `BDL_RESULTS` environment variable. To reproduce this analysis create the respective variable.

```
# read the folder for results from environment variable
resultsPath <- Sys.getenv("BDL_RESULTS")
if (identical(resultsPath, "")){
  stop("No results folder defined, set the BDL_RESULTS environment variable")
} else {
  print(resultsPath)
}
```

```
## [1] "/home/mkoenig/git/bdl-analysis/results"
```

```
# create data subfolder to store analysis results
dir.create(file.path(resultsPath, 'data'), showWarnings=FALSE)
```

Explorative data analysis

Data import

In a first step the processed data set is loaded from the `data` folder. The data consists of the time course data for all factors (`BDLdata`), additional information for the factors (`BDLfactors`), the sample definition, i.e. the assignment of sample ids to respective time point and repeat (`BDLsamples`), and a mapping of the Fluidigm (gene) probe ids to UniProt identifiers (`BDLprobes`).

```
suppressPackageStartupMessages(library(BDLanalysis))
suppressPackageStartupMessages(library(calibrate))
suppressPackageStartupMessages(library(pander))
dir.create(file.path(resultsPath, 'control'), showWarnings=FALSE)
# path definition
baseLoc <- system.file(package="BDLanalysis")
extPath <- file.path(baseLoc, "extdata")
# load data
data(BDLdata)
data(BDLsamples)
data(BDLfactors)
data(BDLprobes)
# counters
Nr <- 5  # repeats
Nt <- length(levels(BDLsamples$time_fac))  # time points
# store all data sets in the results folder
save(BDLdata, file=file.path(resultsPath, "data", "BDLdata.Rdata"))
save(BDLsamples, file=file.path(resultsPath, "data", "BDLsamples.Rdata"))
save(BDLfactors, file=file.path(resultsPath, "data", "BDLfactors.Rdata"))
save(BDLprobes, file=file.path(resultsPath, "data", "BDLprobes.Rdata"))
```

In addition to the individual data points per time point, the mean data averaged over the N_r repeats per time points is used in parts of the correlation analysis. This mean factor data set is calculated once at the beginning via

```
BDLmean <- bdl_mean_data(BDLdata, BDLsamples)
BDLmean.time <- as.numeric(levels(as.factor(BDLsamples$time)))
```

In total 153 factors were measured in the this BDL study falling in the categories: Antibodies, Biochemistry, GE_ADME, GE_Cytokines, GE_Fibrosis, Histology. The majority of factors belongs herby to the 3 fluidigm chips with 47 probes per chip.

An overview of the number of factors per category is provided in the following table

```
cat_table <- as.data.frame(table(BDLfactors$ftype))
colnames(cat_table) <- c("Category", "Freq")
set.caption(sub(".", " ", "Factors per category", fixed = TRUE))
pander(cat_table)
```

Table 1: Factors per category

Category	Freq
Antibodies	3

Category	Freq
Biochemistry	4
GE_ADME	47
GE_Cytokines	47
GE_Fibrosis	46
Histology	6

```
rm(cat_table)
```

Data visualization

Time course of single factors

In a first step overview plots of the raw and mean data for all individual factors are generated. These are available in the `resultsPath/factors` folder

```
# Create figures for all factors
factors_path <- file.path(resultsPath, 'factors')
dir.create(factors_path, showWarnings=FALSE)
plot_all_factors(path=factors_path)
rm(factors_path)
```

The example for a single factor is depicted below, here for the factor `bilirubin`.

```
plot_single_factor('bilirubin', path=NULL)
```

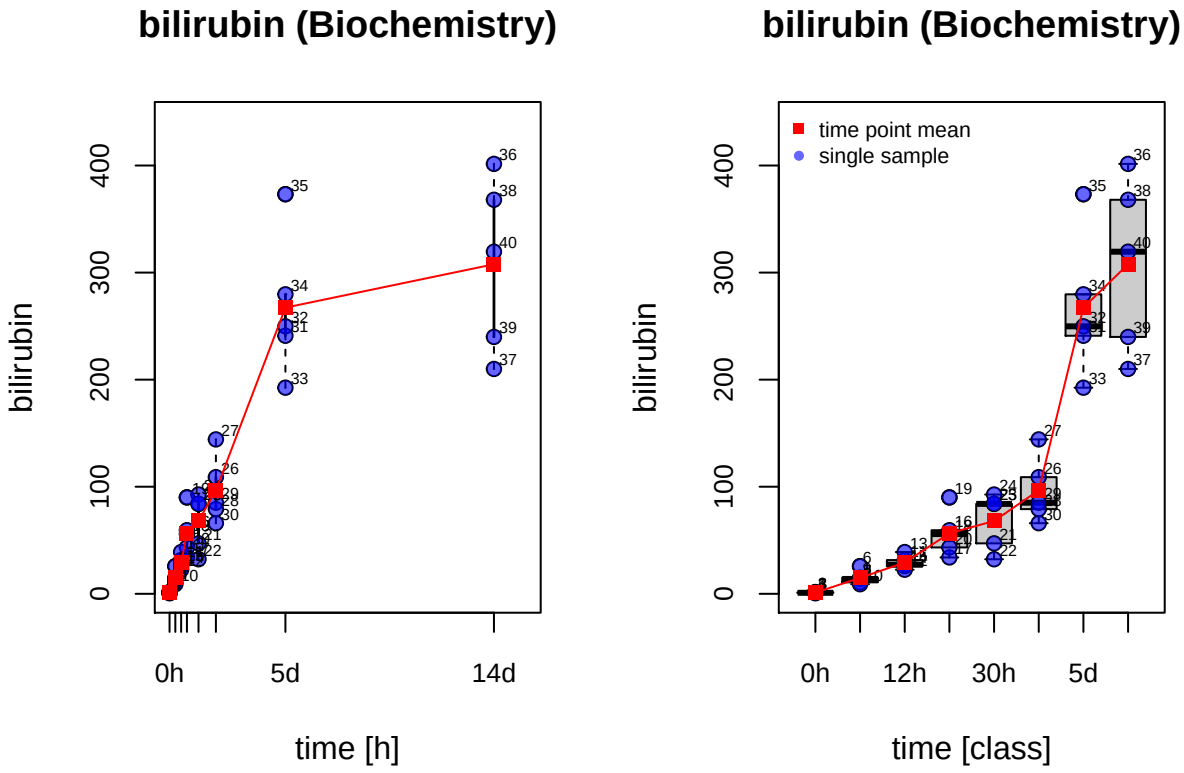


Figure Single factor: Plot of the raw time course data for a single factor, here for bilirubin. On the left the data is plotted against the time [h], on the right against the different time classes. Individual data points are depecticed in blue with the respective sample number shown next to the data points. The mean averaged of the repeats per time point are depicted in red. Box-and-whisker plots were added with default R parameters of boxwex=0.8, staplewex=0.5, outwex=0.5.

Time course of all factors (Heatmap)

In a next step the heatmap of the full data set was generated, i.e. of all time points and repeats. This provides a first overview over the complete data set. Rows correspond to the individual factors (factor order corresponding to the original data set: `GE_ADME`, `GE_Cytokines`, `GE_Fibrosis`, `Biochemistry`, `Histology`, `Antibodies`). Columns correspond to the 40 samples with 5 subsequent samples belonging to one of the 8 time points (with from left to right: 0h, 6h, 12h, 18h, 30h, 2d, 5d, 14d). The data is row scaled, i.e. every individual factor is scaled to have mean zero and standard deviation one.

```
suppressPackageStartupMessages(library(gplots))
suppressPackageStartupMessages(library("RColorBrewer"))

# define horizontal and vertical helper lines
v_lines <- ((1:Nt)*Nr+0.5)
factor_types <- c("Antibodies", "Histology", "Biochemistry", "GE_Fibrosis", "GE_Cytokines", "GE_ADME")
factor_table <- table(BDLfactors$ftype)
h_lines <- 0.5 + cumsum(factor_table[factor_types])

timecourse_heatmap <- function(){
  # create better row names
  dtmp <- BDLdata
  rownames(dtmp) <- paste(rownames(BDLsamples), BDLsamples$time_fac, sep=" ")
  # heatmap colors
  hmap_colors <- HeatmapColors()
  # colors for factor groups
  colorset <- brewer.pal(length(factor_types), "Set2")
  color.map <- function(factor_id) {return(colorset[ which(factor_types==BDLfactors$ftype[which(BDLfact
  factorColors <- unlist(lapply(BDLfactors$id, color.map))
  # heatmap
  heatmap.2(t(as.matrix(dtmp)), col=hmap_colors(100), scale="row", dendrogram="none", Rowv=NULL, Colv=NU
            key=TRUE, trace="none", cexRow=0.5, keysize=0.8, density.info="none",
            RowSideColors=factorColors,
            add.expr=abline(v=v_lines, h=h_lines, col="black", lwd=0.5),
            main="Heatmap of BDL time course data")
            # xlab="sample", ylab="factor")
  # legend
  legend("left",
      inset=c(-0.03,0),
      legend = rev(factor_types), # category labels
      col = rev(colorset),  # color key
      lty= 1, lwd = 10, cex = 0.7, bty="n"
  )
}
# plot to file
pdf(file.path(resultsPath, "control", "timecourse_heatmap.pdf"), width=10, height=10, pointsize=12)
timecourse_heatmap()
invisible(dev.off())
```

```
timecourse_heatmap()
```

```
# cleanup
rm(factor_table, h_lines, v_lines)
```

Figure All factors : Heatmap of the complete data set, i.e. all factors and repeats over time. The data is row scaled, i.e. every individual factor is scaled to have mean zero and standard deviation one, with positive Z-score in blue, negative Z-score in red. The row order is according to the the factor categories, the order within the fluidigm chips according to the order of the probes on the chip. the respective categories are depicted on the left.

Results: Various patterns are visible in the plotted raw data:

- **Two main classes of response can be observed**. One class with an increase in the early phase up to 6h after BDL (many of the ADME genes fall into this class) and a second class increasing in the later stage after 2-5 days after BDL. Many of the genes on the Cytokines and Fibrosis chips as well as some of the biochemical, histological and antibody fall in this second class.
- **The individual animals show heterogeneous responses to BDL**. Within one time point the 5 repeats can show very different patterns. For instance at time 6h after BDL 3/5 of the mice show a marked increase in the ADME genes, whereas 2/5 do not show such a marked increase. Another example is the mice sample 27 at time 2d, with a high increase in the genes on the Fibrosis chip, which is not observed in the other 4 samples at time 2d.

Actb quality control

Actb (Actin, cytoplasmic 1) probes were included on all Fluidigm chips (`GE_ADME`, `GE_Cytokines`, `GE_Fibrosis`) and not used in the normalization of the gene expression data. Hence, ActB can serve as quality control for the technical reproducibility of the Fluidigm chips. If the data is reproducible between chips the pairwise correlation between all individual Actb measurements should have high correlation coefficients close to 1. Plotting the data of the Actb measurements of two chips against each other should lie on a straight line

```
# Actb control figure
plot_actb_control <- function(){
  par(mfrow=c(2,3))
  plot_single("Actb")
  plot_single("Actb.x")
  plot_single("Actb.y")
  plot_cor_pair("Actb", "Actb.x", single_plots=FALSE)
  plot_cor_pair("Actb", "Actb.y", single_plots=FALSE)
  plot_cor_pair("Actb.x", "Actb.y", single_plots=FALSE)
  par(mfrow=c(1,1))
}

# plot to file
pdf(file.path(resultsPath, "control", "Actb_control.pdf"), width=10, height=6, pointsize=12)
plot_actb_control()
invisible(dev.off())

# calculate Spearman and Pearson correlation coefficients on N=8*5=40 data points
actb.spearman <- cor(data.frame(Actb=BDLdata$Actb,
```

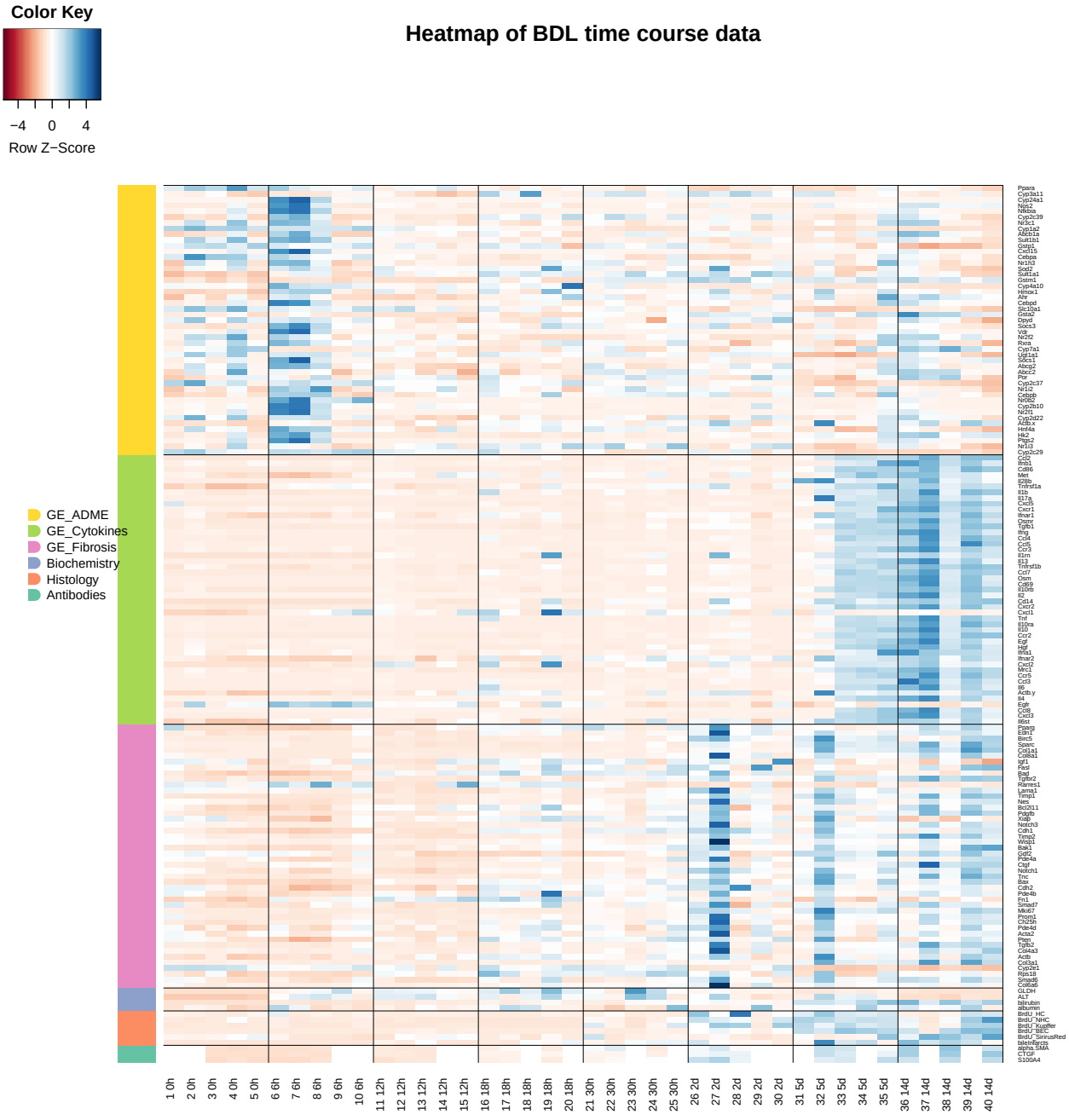

Color Key
−4
0
4
Row Z−Score
Heatmap of BDL time course data
GE_ADME
GE_Cytokines
GE_Fibrosis
Biochemistry
Histology
Antibodies
1 0h
2 0h
3 0h
4 0h
5 0h
6 6h
7 6h
8 6h
9 6h
10 6h
11 12h
12 12h
13 12h
14 12h
15 12h
16 18h
17 18h
18 18h
19 18h
20 18h
21 30h
22 30h
23 30h
24 30h
25 30h
26 2d
27 2d
28 2d
29 2d
30 2d
31 5d
32 5d
33 5d
34 5d
35 5d
36 14d
37 14d
38 14d
39 14d
40 14d

Figure 1:

```
                                Actb.x=BDLdata$Actb.x,
                                Actb.y=BDLdata$Actb.y), method="spearman")
actb.pearson <- cor(data.frame(Actb=BDLdata$Actb,
                               Actb.x=BDLdata$Actb.x,
                               Actb.y=BDLdata$Actb.y), method="pearson")

# table of correlation coefficients
set.caption(sub(".", " ", "Spearman correlation of Actb controls", fixed = TRUE))
pander(round(actb.spearman, digits=3))
```

Table 2: Spearman correlation of Actb controls

	Actb	Actb.x	Actb.y
Actb	1	0.908	0.944
Actb.x	0.908	1	0.917
Actb.y	0.944	0.917	1

```
set.caption(sub(".", " ", "Pearson correlation of Actb controls", fixed = TRUE))
pander(round(actb.pearson, digits=3))
```

Table 3: Pearson correlation of Actb controls

	Actb	Actb.x	Actb.y
Actb	1	0.945	0.938
Actb.x	0.945	1	0.935
Actb.y	0.938	0.935	1

```
plot_actb_control()
```

```
rm(actb.pearson, actb.spearman)
```

Figure Actb Control: Correlation plot of the Actb probes from the 3 Fluidigm chips: Actb (fibrosis), Actb.x (ADME), Actb.y (Cytokines). The top row shows the individual time courses, the bottom row the pair wise plot of individual data points.

Results: The Actb Fluidigm gene expression measurements are highly reproducible for the measured chips, with Spearman as well as Pearson correlation coefficients all > 0.9 for pairwise Actb comparison.

Dimension reduction via ANOVA

ANOVA for single factor

A one-way analysis of variance (ANOVA) was applied to reduce the factors to the subset showing significant ($p_{adjusted} < 0.05$) changes during the time course. In its simplest form, ANOVA provides a statistical test of whether or not the means of several groups are equal, and therefore generalizes the t-test to more than two groups, with the groups being the sampled time points. The Holm's procedure was used to correct the p-values for any artificial p-value inflation due to multiple testing.

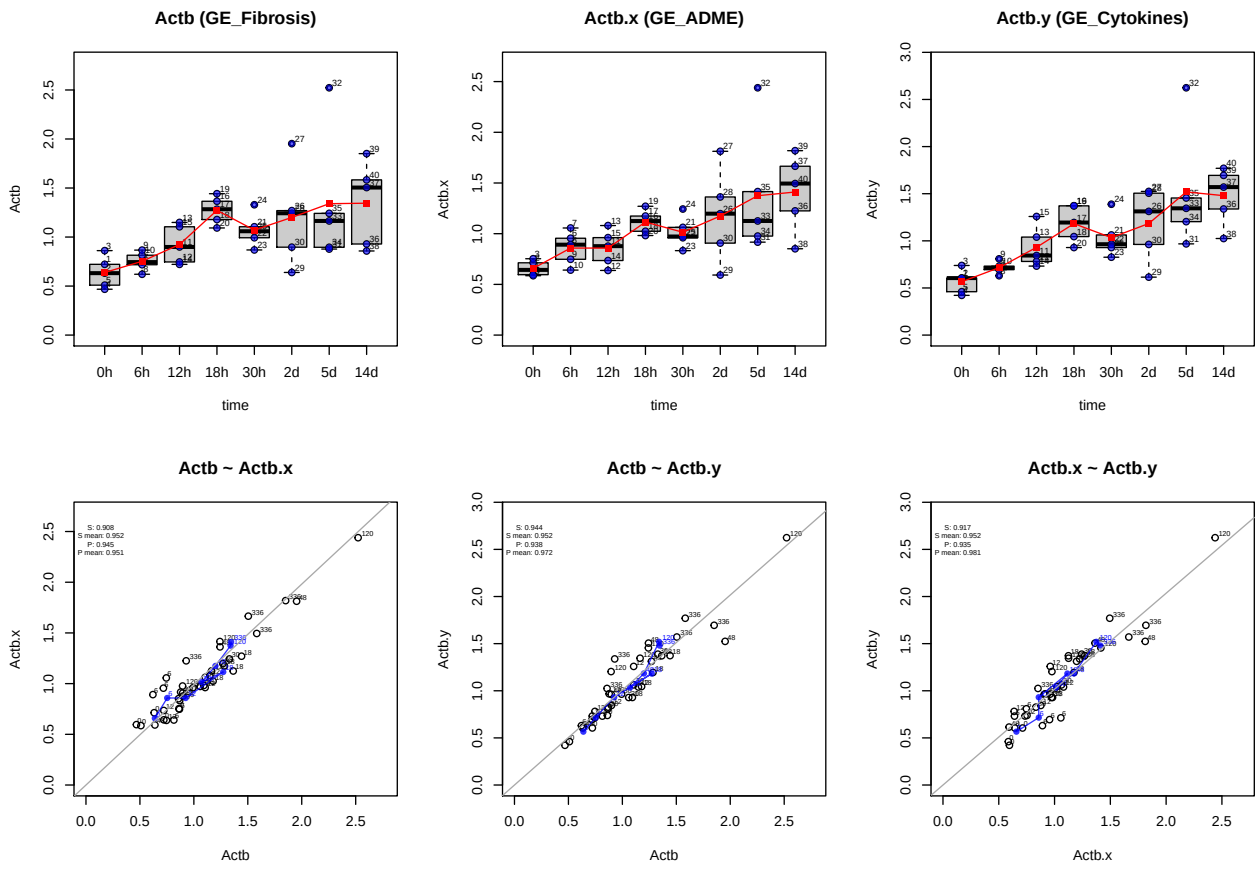

Actb (GE_Fibrosis)
Actb.x (GE_ADME)
Actb.y (GE_Cytokines)
Actb
Actb.x
Actb.y
time
0h
6h
12h
18h
30h
2d
5d
14d
Actb ~ Actb.x
Actb ~ Actb.y
Actb.x ~ Actb.y
S: 0.908
S mean: 0.952
P: 0.945
P mean: 0.951
S: 0.944
S mean: 0.952
P: 0.938
P mean: 0.972
S: 0.917
S mean: 0.952
P: 0.935
P mean: 0.981

Figure 2:

For every of the individual factors in the BDL data set an ANOVA was calculated. Dimension reduction of the BDL data set was than performed by filtering out factors which did not significantly changing over time.

The `BDLdata` data set is reshaped into matrix format for the ANOVA calculation, with time points in rows and repeats as columns for every factor.

```
BDLmatrices <- bdl_matrix_data(BDLdata, BDLsamples)
```

The following shows the ANOVA calculation for a single factor, here for `bilirubin`.

```
  # example ANOVA for one factor
  mat.anova <- t(BDLmatrices[['bilirubin']])
  colnames(mat.anova) <- levels(BDLsamples$time_fac)

  # concatenate the data rows of df1 into a single vector r .
  r = c(t(as.matrix(mat.anova)))  # response data

  # assign new variables for the treatment levels and number of observations.
  f = levels(BDLsamples$time_fac)   # treatment levels
  k = 8                           # number of treatment levels
  n = 5                           # observations per treatment

  # create a vector of treatment factors that corresponds to each element of r in step 3 with the gl fu
  tm <- gl(k, 1, n*k, factor(f))   # matching treatments

  # apply the function aov to a formula that describes the response r by the treatment factor tm.
  # fit an analysis of variance model
  av <- aov(r ~ tm)

  # print out the ANOVA table with the summary function.
  summary(av)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## tm           7 479470   68496      41 3.27e-14 ***
## Residuals   32  53463    1671
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
  # print the corresponding p-value
  p.value <- summary(av)[[1]][["Pr(>F)"]][[1]]

  # show data matrix
  print(mat.anova)
```

```
##      0h    6h   12h   18h   30h     2d    5d   14d
## R1 1.49 25.80 31.51 59.40 47.11 108.95 241.0 401.5
## R2 0.93 13.87 22.51 33.95 32.26 144.20 249.8 210.0
## R3 1.65 10.70 38.99 55.80 83.98  79.10 192.5 368.0
## R4 0.85 15.08 25.81 90.05 92.72  85.10 279.7 239.9
## R5 0.53  8.96 27.21 43.30 84.02  65.95 373.2 319.5
```

```
  # cleanup
rm(r, f, k, n, tm, av, p.value, mat.anova)
```

ANOVA for all factors

Analog to the single factor ANOVA, the ANOVA is performed on all the factors. Hereby, a multitude of tests are performed, namely an ANOVA for every single factor. Consequently, the reported p-values of the ANOVA have to be adjusted via multiple testing procedures. Using the p.adjust function which given a set of p-values, returns p-values adjusted using one of several methods. The Bonferroni, Holm, Hochberg, Hommel are designed to give strong control of the family-wise error rate. We used the Holm's method for adjustment (*Holm, S. (1979). A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6, 65-70.*).

```
# Calculation of ANOVA for all factors
df.anova <- all_factor_anova()
df.anova$sig <- sapply(df.anova$p.value, significant_code)  # add significant codes

df.anova$p.holm <- p.adjust(df.anova$p.value, method ="holm" , n = length(df.anova$p.value))
df.anova$sig.holm <- sapply(df.anova$p.holm, significant_code)
df.anova$ftype <- BDLfactors$ftype
df.anova$fshort <- BDLfactors$ftype.short

# order factors by adjusted p-values
df.anova.ordered <- df.anova[with(df.anova, order(p.holm)), ]
df.anova.ordered
```

```
##             factors      p.value sig       p.holm sig.holm        ftype
## 8             Cyp1a2 1.913423e-16 *** 2.927537e-14      ***      GE_ADME
## 143       bilirubin 3.273960e-14 *** 4.976419e-12      *** Biochemistry
## 71            Il10rb 7.639824e-14 *** 1.153613e-11      *** GE_Cytokines
## 60             Tgfb1 2.197248e-13 *** 3.295871e-11      *** GE_Cytokines
## 48             Ccl2 2.320015e-13 *** 3.456823e-11      *** GE_Cytokines
## 50             Cd86 4.433981e-13 *** 6.562292e-11      *** GE_Cytokines
## 79             Ccr2 4.720799e-13 *** 6.939574e-11      *** GE_Cytokines
## 85             Mrc1 4.762017e-13 *** 6.952545e-11      *** GE_Cytokines
## 67         Tnfrsf1b 5.443030e-13 *** 7.892394e-11      *** GE_Cytokines
## 56            Cxcl5 4.345645e-12 *** 6.257729e-10      *** GE_Cytokines
## 152            CTGF 5.466353e-12 *** 7.816885e-10      ***   Antibodies
## 77            Il10ra 1.154771e-11 *** 1.639775e-09      *** GE_Cytokines
## 17             Gstm1 6.513806e-11 *** 9.184466e-09      ***      GE_ADME
## 68             Ccl7 2.430883e-10 *** 3.403237e-08      *** GE_Cytokines
## 86             Ccr5 3.139262e-10 *** 4.363574e-08      *** GE_Cytokines
## 81              Hgf 4.203344e-10 *** 5.800614e-08      *** GE_Cytokines
## 59             Osmr 7.389499e-10 *** 1.012361e-07      *** GE_Cytokines
## 62             Ccl4 7.670907e-10 *** 1.043243e-07      *** GE_Cytokines
## 38            Nr0b2 9.608293e-10 *** 1.297120e-07      ***      GE_ADME
## 104          Tgfbr2 1.276990e-09 *** 1.711167e-07      ***  GE_Fibrosis
## 148         BrdU_BEC 1.569640e-09 *** 2.087621e-07      ***    Histology
## 63             Ccl5 2.104076e-09 *** 2.777380e-07      *** GE_Cytokines
## 99           Col1a1 3.356632e-09 *** 4.397188e-07      ***  GE_Fibrosis
## 58           Ifnar1 5.933095e-09 *** 7.713023e-07      *** GE_Cytokines
## 153          S100A4 7.122988e-09 *** 9.188655e-07      ***   Antibodies
```

```
## 98            Sparc 8.383632e-09 *** 1.073105e-06      ***  GE_Fibrosis
## 137          Cyp2e1 1.181064e-08 *** 1.499951e-06      ***  GE_Fibrosis
## 74            Cxcr2 1.390313e-08 *** 1.751794e-06      *** GE_Cytokines
## 64            Ccr3 1.524189e-08 *** 1.905237e-06       *** GE_Cytokines
## 70            Cd69 2.200276e-08 *** 2.728342e-06       *** GE_Cytokines
## 47          Cyp2c29 2.384128e-08 *** 2.932477e-06      ***      GE_ADME
## 23            Gsta2 3.180374e-08 *** 3.880056e-06      ***      GE_ADME
## 76              Tnf 3.602497e-08 *** 4.359021e-06      *** GE_Cytokines
## 117           Gdf2 4.958043e-08 *** 5.949652e-06       ***  GE_Fibrosis
## 54            Il1b 6.128729e-08 *** 7.293187e-06       *** GE_Cytokines
## 61            Ifng 6.367290e-08 *** 7.512323e-06       *** GE_Cytokines
## 69             Osm 6.366375e-08 *** 7.512323e-06       *** GE_Cytokines
## 87            Ccl3 7.354920e-08 *** 8.531707e-06       *** GE_Cytokines
## 66            Il13 8.081570e-08 *** 9.293806e-06       *** GE_Cytokines
## 57            Cxcr1 9.412831e-08 *** 1.073063e-05      *** GE_Cytokines
## 35          Cyp2c37 9.650268e-08 *** 1.090480e-05      ***      GE_ADME
## 73            Cd14 1.028168e-07 *** 1.151549e-05       *** GE_Cytokines
## 136          Col3a1 1.816561e-07 *** 2.016383e-05      ***  GE_Fibrosis
## 53         Tnfrsf1a 3.028259e-07 *** 3.331085e-05      *** GE_Cytokines
## 72              Il2 4.437114e-07 *** 4.836454e-05       *** GE_Cytokines
## 49            Ifnb1 4.504193e-07 *** 4.864529e-05       *** GE_Cytokines
## 80              Egf 4.556800e-07 *** 4.875776e-05       *** GE_Cytokines
## 52            Il28b 4.645752e-07 *** 4.878116e-05       *** GE_Cytokines
## 78            Il10 4.666371e-07 *** 4.878116e-05        *** GE_Cytokines
## 90             Il4 4.601996e-07 *** 4.878116e-05        *** GE_Cytokines
## 22          Slc10a1 5.102225e-07 *** 5.255292e-05      ***      GE_ADME
## 114           Timp2 6.116144e-07 *** 6.238466e-05      ***  GE_Fibrosis
## 93            Cxcl3 6.750161e-07 *** 6.817662e-05      *** GE_Cytokines
## 92            Ccl8 1.232797e-06 *** 1.232797e-04       *** GE_Cytokines
## 119           Ctgf 1.336225e-06 *** 1.322862e-04       ***  GE_Fibrosis
## 11            Gstp1 1.409173e-06 *** 1.380990e-04      ***      GE_ADME
## 1             Ppara 1.683106e-06 *** 1.632613e-04      ***      GE_ADME
## 83           Ifnar2 1.898212e-06 *** 1.822284e-04      *** GE_Cytokines
## 88              Il6 2.352096e-06 *** 2.234491e-04       *** GE_Cytokines
## 55            Il17a 2.580378e-06 *** 2.425555e-04       *** GE_Cytokines
## 103             Bad 4.125226e-06 *** 3.836460e-04       ***  GE_Fibrosis
## 107           Timp1 4.678453e-06 *** 4.304177e-04       ***  GE_Fibrosis
## 113           Cdh1 4.919990e-06 *** 4.477191e-04        ***  GE_Fibrosis
## 13            Cebpa 5.473439e-06 *** 4.926095e-04       ***      GE_ADME
## 151       alpha.SMA 5.638004e-06 *** 5.017824e-04       ***   Antibodies
## 146        BrdU_NHC 6.005016e-06 *** 5.284414e-04       ***    Histology
## 123             Cdh2 6.179147e-06 *** 5.375858e-04       ***  GE_Fibrosis
## 149 BrdU_SirirusRed 8.819839e-06 *** 7.585062e-04       ***    Histology
## 110           Pdgfb 9.032454e-06 *** 7.677586e-04       ***  GE_Fibrosis
## 94            Il6st 1.037076e-05 *** 8.711441e-04       *** GE_Cytokines
## 125             Fn1 1.457762e-05 *** 1.209942e-03        **  GE_Fibrosis
## 127           Mki67 1.774623e-05 *** 1.455191e-03        **  GE_Fibrosis
## 82            Ifna1 1.805029e-05 *** 1.462074e-03        ** GE_Cytokines
## 91             Egfr 1.988312e-05 *** 1.590649e-03        ** GE_Cytokines
## 147    BrdU_Kupffer 2.470612e-05 *** 1.951783e-03        **    Histology
## 121             Tnc 2.611215e-05 *** 2.036748e-03        **  GE_Fibrosis
## 30           Ugt1a1 3.570685e-05 *** 2.749428e-03        **      GE_ADME
## 16          Sult1a1 3.922274e-05 *** 2.980928e-03        **      GE_ADME
## 141           GLDH 5.711856e-05 *** 4.283892e-03        ** Biochemistry
```

```
## 120          Notch1 6.016402e-05 *** 4.452138e-03       **  GE_Fibrosis
## 51             Met 6.803468e-05 *** 4.966531e-03       ** GE_Cytokines
## 29          Cyp7a1 1.365100e-04 *** 9.828721e-03       **      GE_ADME
## 3          Cyp24a1 1.391831e-04 *** 9.882000e-03       **      GE_ADME
## 133          Tgfb2 1.542364e-04 *** 1.079655e-02        *  GE_Fibrosis
## 97           Birc5 2.454637e-04 *** 1.693699e-02        *  GE_Fibrosis
## 89          Actb.y 2.806640e-04 *** 1.908515e-02        * GE_Cytokines
## 116           Bak1 4.077548e-04 *** 2.731957e-02        *  GE_Fibrosis
## 122            Bax 4.371327e-04 *** 2.885075e-02        *  GE_Fibrosis
## 105        Rarres1 5.491381e-04 *** 3.569397e-02        *  GE_Fibrosis
## 150    bileInfarcts 5.904178e-04 *** 3.778674e-02        *    Histology
## 2          Cyp3a11 8.410363e-04 *** 5.298529e-02        .      GE_ADME
## 10         Sult1b1 9.257764e-04 *** 5.739814e-02        .      GE_ADME
## 18         Cyp4a10 1.124446e-03  ** 6.792529e-02        .      GE_ADME
## 95           Pparg 1.113529e-03  ** 6.792529e-02        .  GE_Fibrosis
## 44             Hk2 1.271173e-03  ** 7.499918e-02        .      GE_ADME
## 142            ALT 1.515972e-03  ** 8.792635e-02        . Biochemistry
## 139          Smad6 1.665654e-03  ** 9.494230e-02        .  GE_Fibrosis
## 39         Cyp2b10 2.093404e-03  ** 1.172306e-01             GE_ADME
## 145        BrdU_HC 2.310442e-03  ** 1.270743e-01               Histology
## 65           Il1rn 2.507326e-03  ** 1.353956e-01           GE_Cytokines
## 108            Nes 2.590456e-03  ** 1.372942e-01           GE_Fibrosis
## 5           Nfkbia 3.024538e-03  ** 1.572760e-01               GE_ADME
## 138          Rps18 3.292204e-03  ** 1.679024e-01           GE_Fibrosis
## 12          Cxcl15 4.356699e-03  ** 2.178349e-01               GE_ADME
## 25           Socs3 4.981022e-03  ** 2.426040e-01               GE_ADME
## 26             Vdr 4.951101e-03  ** 2.426040e-01               GE_ADME
## 96           Edn1 5.022042e-03  ** 2.426040e-01           GE_Fibrosis
## 40          Nr2f1 5.717984e-03  ** 2.630272e-01               GE_ADME
## 32          Abcg2 6.283587e-03  ** 2.820514e-01               GE_ADME
## 46          Nr1i3 6.267808e-03  ** 2.820514e-01               GE_ADME
## 43          Hnf4a 6.700157e-03  ** 2.881068e-01               GE_ADME
## 45          Ptgs2 8.168008e-03  ** 3.430563e-01               GE_ADME
## 109        Bcl2l11 8.338430e-03  ** 3.430563e-01           GE_Fibrosis
## 31           Socs1 8.654611e-03  ** 3.461844e-01               GE_ADME
## 132           Pten 9.397086e-03  ** 3.664864e-01           GE_Fibrosis
## 42          Actb.x 1.025240e-02   * 3.895911e-01               GE_ADME
## 84          Cxcl2 1.098554e-02   * 4.064650e-01           GE_Cytokines
## 111           Xiap 1.124710e-02   * 4.064650e-01           GE_Fibrosis
## 118         Pde4a 1.263533e-02   * 4.422367e-01           GE_Fibrosis
## 24            Dpyd 1.337053e-02   * 4.432805e-01               GE_ADME
## 75          Cxcl1 1.303766e-02   * 4.432805e-01           GE_Cytokines
## 106         Lama1 1.428488e-02   * 4.571161e-01           GE_Fibrosis
## 100        Col8a1 1.634422e-02   * 5.066710e-01           GE_Fibrosis
## 128         Prom1 1.870355e-02   * 5.543027e-01           GE_Fibrosis
## 135           Actb 1.847676e-02   * 5.543027e-01           GE_Fibrosis
## 20            Ahr 1.989103e-02   * 5.569489e-01               GE_ADME
## 27          Nr2f2 2.048348e-02   * 5.569489e-01               GE_ADME
## 4           Nos2 2.237430e-02   * 5.817317e-01               GE_ADME
## 112         Notch3 2.905904e-02   * 7.264759e-01           GE_Fibrosis
## 21          Cebpd 3.036439e-02   * 7.287453e-01               GE_ADME
## 19          Hmox1 3.343725e-02   * 7.690567e-01               GE_ADME
## 41         Cyp2d22 3.800488e-02   * 8.361075e-01               GE_ADME
## 101           Igf1 4.658924e-02   * 9.783741e-01           GE_Fibrosis
```

```
## 102           Fasl 4.854110e-02   * 9.783741e-01          GE_Fibrosis
## 6          Cyp2c39 1.057596e-01     1.000000e+00              GE_ADME
## 7            Nr3c1 1.299624e-01     1.000000e+00              GE_ADME
## 9           Abcb1a 2.850937e-01     1.000000e+00              GE_ADME
## 14           Nr1h3 6.326127e-01     1.000000e+00              GE_ADME
## 15            Sod2 1.208096e-01     1.000000e+00              GE_ADME
## 28            Rxra 8.615924e-02   . 1.000000e+00              GE_ADME
## 33           Abcc2 5.803779e-02   . 1.000000e+00              GE_ADME
## 34             Por 4.054924e-01     1.000000e+00              GE_ADME
## 36           Nr1i2 6.889241e-02   . 1.000000e+00              GE_ADME
## 37           Cebpb 1.605796e-01     1.000000e+00              GE_ADME
## 115          Wisp1 1.651084e-01     1.000000e+00          GE_Fibrosis
## 124          Pde4b 2.641945e-01     1.000000e+00          GE_Fibrosis
## 126          Smad7 1.170115e-01     1.000000e+00          GE_Fibrosis
## 129          Ch25h 1.296172e-01     1.000000e+00          GE_Fibrosis
## 130          Pde4d 1.668398e-01     1.000000e+00          GE_Fibrosis
## 131          Acta2 2.365621e-01     1.000000e+00          GE_Fibrosis
## 134         Col4a3 6.313442e-02   . 1.000000e+00          GE_Fibrosis
## 140         Col6a6 1.370452e-01     1.000000e+00          GE_Fibrosis
## 144        albumin 3.546133e-01     1.000000e+00         Biochemistry
##     fshort
## 8
## 143      B
## 71
## 60
## 48
## 50
## 79
## 85
## 67
## 56
## 152      A
## 77
## 17
## 68
## 86
## 81
## 59
## 62
## 38
## 104
## 148      H
## 63
## 99
## 58
## 153      A
## 98
## 137
## 74
## 64
## 70
## 47
## 23
## 76
```

```
## 117
## 54
## 61
## 69
## 87
## 66
## 57
## 35
## 73
## 136
## 53
## 72
## 49
## 80
## 52
## 78
## 90
## 22
## 114
## 93
## 92
## 119
## 11
## 1
## 83
## 88
## 55
## 103
## 107
## 113
## 13
## 151      A
## 146      H
## 123
## 149      H
## 110
## 94
## 125
## 127
## 82
## 91
## 147      H
## 121
## 30
## 16
## 141      B
## 120
## 51
## 29
## 3
## 133
## 97
## 89
## 116
```

```
## 122
## 105
## 150      H
## 2
## 10
## 18
## 95
## 44
## 142      B
## 139
## 39
## 145      H
## 65
## 108
## 5
## 138
## 12
## 25
## 26
## 96
## 40
## 32
## 46
## 43
## 45
## 109
## 31
## 132
## 42
## 84
## 111
## 118
## 24
## 75
## 106
## 100
## 128
## 135
## 20
## 27
## 4
## 112
## 21
## 19
## 41
## 101
## 102
## 6
## 7
## 9
## 14
## 15
## 28
## 33
```

```
## 34
## 36
## 37
## 115
## 124
## 126
## 129
## 130
## 131
## 134
## 140
## 144      B
```

```
# save results
write.table(df.anova.ordered, file=file.path(resultsPath, "data", 'BDLanova.csv'), sep="\t", quote=FALSE
save(df.anova, file=file.path(resultsPath, "data", "BDLanova.Rdata"))
rm(df.anova.ordered)
```

Filter factors

The factors are filtered based on the respective acceptance level, with the cutoff for the adjusted p-value being p_{accept}, i.e. all factors with a ANOVA with $p_{adjusted} \geq p_{accept}$ are filtered out. The filtered raw data is available as `BDLdata.fil`, the filtered mean data set as `BDLmean.fil`. All subsequent analyses are performed on the filtered data set, which is depicted in the following heatmap

```
p.accept = 0.05  # acceptance level
anova.accept = (df.anova$p.holm < p.accept)  # accepted subset
# subset of filtered data
BDLdata.fil <- BDLdata[, anova.accept]
BDLmean.fil <- BDLdata[, anova.accept]

# accepted
table(anova.accept)  # 64 rejected / 90 accepted (adjusted)
```

```
## anova.accept
## FALSE  TRUE
##    63    90
```

```
# which factors were accepted in the various categories
fil_tab <- data.frame(
  table(BDLfactors$ftype[anova.accept]),
  table(BDLfactors$ftype),
  round(table(BDLfactors$ftype[anova.accept])/table(BDLfactors$ftype), 2)
)
fil_tab <- fil_tab[, c('Var1', 'Freq', 'Freq.1', 'Freq.2')]
names(fil_tab) <- c('Category', 'Accepted', 'All', 'Percent')
# overview of filtered factors
print(fil_tab)
```

```
##       Category Accepted All Percent
## 1   Antibodies        3   3    1.00
## 2 Biochemistry        2   4    0.50
```

```
## 3      GE_ADME       14  47    0.30
## 4 GE_Cytokines       44  47    0.94
## 5  GE_Fibrosis       22  46    0.48
## 6    Histology        5   6    0.83
```

```
rm(fil_tab)
```

Based on the adjusted p-values the data set was reduced from original 154 factors to 90. Almost all `Cytokines` genes (inflammation panel) were retained in the data set whereas many of the `ADME` and `Fibrosis` genes are filtered.

Heatmap of filtered time course data

The heatmap of the filtered raw data is depicted below

```
v_lines <- ((1:Nt)*Nr+0.5)

timecourse_heatmap_filtered <- function(){
  # prepare data with row names
  dtmp <- BDLdata.fil
  rownames(dtmp) <- paste(rownames(BDLsamples), BDLsamples$time_fac, sep=" ")

  # color definitions
  hmap_colors <- HeatmapColors()
  colorset <- brewer.pal(length(factor_types), "Set2")
  color.map <- function(factor_id) {return(colorset[ which(factor_types==BDLfactors$ftype[which(BDLfact
  factorColors <- unlist(lapply(colnames(BDLdata.fil), color.map))
  # heatmap
  heatmap.2(t(as.matrix(dtmp)), col=hmap_colors(100), scale="row", dendrogram="none",
            Rowv=NULL, Colv=NULL,
            key=TRUE, trace="none", cexRow=0.5, keysize=0.8, density.info="none",
            RowSideColors=factorColors,
            add.expr=abline(v=v_lines, col="black", lwd=0.5),
            main="ANOVA Filtered BDL factors")
            # xlab="sample", ylab="factor")
  # legend
  legend("left",
      inset=c(-0.03,0),
      legend = rev(factor_types), # category labels
      col = rev(colorset),   # color key
      lty= 1,                # line style
      lwd = 10,              # line width
      cex = 0.7,
      bty="n"
  )
}

# plot to file
pdf(file.path(resultsPath, "control", "timecourse_heatmap_filtered.pdf"),
    width=10, height=10, pointsize=12)
timecourse_heatmap_filtered()
invisible(dev.off())
```

```
# plot to report
timecourse_heatmap_filtered()
```

```
rm(v_lines)
```

Figure Timcourse Filtered: Plot of ANOVA filtered data set.

Correlation analysis

For the correlation analysis between factors and the subsequent cluster analysis a correlation measure for time series data {Son2008} in combination with Complete-Linkage hierarchical clustering was used. This combination of methods provided the best enrichments on gene-expression time-series in a recent comparisons of methods {Jaskowiak2014, Jaskowiak2013} testing various correlation measures and clustering algorithms.

The calculation of correlation coefficients between factors i and j ($i, j = 1, ..., N_p$) was performed using the slightly modified correlation coefficient based similarity measure developed for clustering of time-course data ($Y_{i,j}^{S2}$ and $Y_{i,j}^{R2}$) {Son2008}. $Y_{i,j}^{S2}$ and $Y_{i,j}^{R2}$ are linear combinations of (i) a classical correlation part based on Spearman correlation $S_{i,j}^{*}$ in case of $Y_{i,j}^{S2}$ or Pearson $R_{i,j}^{*}$ in case of $Y_{i,j}^{R2}$, (ii) a component $A_{i,j}^{*}$ accounting for the similarity in changes between two time courses, (iii) a component $M_{i,j}^{*}$ comparing the location of minimum and maximum values of the time course (see {Son2008} for definitions)

$$Y_{i,j}^{S2} = w_1 S_{i,j}^{*} + w_2 A_{i,j}^{*} + w_3 M_{i,j}^{*}$$

$$Y_{i,j}^{R2} = w_1 R_{i,j}^{*} + w_2 A_{i,j}^{*} + w_3 M_{i,j}^{*}$$

$R_{i,j}^{*}$ and $S_{i,j}^{*}$ are hereby calculated on the individual data points for the factors i and j, $A_{i,j}^{*}$ and $M_{i,j}^{*}$ on the mean time courses averaged over the N_r repeated measurements. Throughout the analysis the following weights were used $w_1 = 0.5$, $w_2 = 0.3$, $w_3 = 0.2$.

In the calculation of the change component we used a Spearman correlation based measure (A^{**}) instead of the originally proposed Pearson measure (A^{*}) resulting in the correlation scores $Y_{i,j}^{S3}$ and $Y_{i,j}^{S3}$

$$Y_{i,j}^{S3} = w_1 S_{i,j}^{*} + w_2 A_{i,j}^{**} + w_3 M_{i,j}^{*}$$

$$Y_{i,j}^{R3} = w_1 R_{i,j}^{*} + w_2 A_{i,j}^{**} + w_3 M_{i,j}^{*}$$

Herein, $A_{i,j}^{**}$ calculates the correlation of changes between factors i and j based on Spearman correlation analog $A_{i,j}^{*}$ as

$$A_{i,j}^{**} = (S(d_i, d_j) + 1)/2$$

$$A_{i,j}^{*} = (R(d_i, d_j) + 1)/2$$

The reason for this adaption was that initial analysis showed a strong dependency of the change components on outliers.

All calculated correlation scores Y^S and Y^R are transformed from [0, 1] to [-1, 1] via

$$Y_{norm}^{S} = 2(Y^S - 0.5)$$

$$Y_{norm}^{R} = 2(Y^R - 0.5)$$

In addition to the used Y^S and Y^R correlation scores Pearson (R) and Spearman (S) correlations were calculated for comparison.


Color Key
−4 0 2 4
Row Z−Score
ANOVA Filtered BDL factors
GE_ADME
GE_Cytokines
GE_Fibrosis
Biochemistry
Histology
Antibodies
Ppara
Cyp24a1
Cyp1a2
Gstp1
Cebpa
Sult1a1
Gstm1
Slc10a1
Gsta2
Cyp7a1
Ugt1a1
Cyp2c37
Nr0b2
Cyp2c29
Ccl2
Ifnb1
Cd86
Met
Il28b
Tnfrsf1a
Il1b
Il17a
Cxcl5
Cxcr1
Ifnar1
Osmr
Tgfb1
Ifng
Ccl4
Ccl5
Ccr3
Il13
Tnfrsf1b
Ccl7
Osm
Cd69
Il10rb
Il2
Cd14
Cxcr2
Tnf
Il10ra
Il10
Ccr2
Egf
Hgf
Ifna1
Ifnar2
Mrc1
Ccr5
Ccl3
Il6
Actb.y
Il4
Egfr
Ccl8
Cxcl3
Ifi6st
Birc5
Sparc
Col1a1
Bad
Tgfbr2
Ramres1
Timp1
Pdgfb
Cdh1
Timp2
Bak1
Gdf2
Ctgf
Notch1
Tnc
Bax
Cdh2
Fn1
Mki67
Tgfb2
Col3a1
Cyp2e1
GLDH
bilirubin
BrdU_NHC
BrdU_Kupffer
BrdU_BEC
BrdU_SirirusRed
bileInfarcts
alpha.SMA
CTGF
S100A4
1 0h
2 0h
3 0h
4 0h
5 0h
6 6h
7 6h
8 6h
9 6h
10 6h
11 12h
12 12h
13 12h
14 12h
15 12h
16 18h
17 18h
18 18h
19 18h
20 18h
21 30h
22 30h
23 30h
24 30h
25 30h
26 2d
27 2d
28 2d
29 2d
30 2d
31 5d
32 5d
33 5d
34 5d
35 5d
36 14d
37 14d
38 14d
39 14d
40 14d

Figure 3:

```r
suppressPackageStartupMessages(library(corrplot))
dir.create(file.path(resultsPath, 'correlation'), showWarnings=FALSE)

# list of calculated correlation methods
correlation_methods <- c("pearson", "spearman", "ys1", "ys2", "ys3", "yr1", "yr2", "yr3")

# The calculated correlation matrices are stored in the `cor.matrices
# Provides simple access to the respective correlation matrix.
cor.matrices <- vector("list", length=length(correlation_methods))
names(cor.matrices) <- correlation_methods
```

Pearson & Spearman correlation

In a first step Pearson $R_{i,j}$ and Spearman $S_{i,j}$ correlation were calculated for the subset of filtered factors.

```r
# Spearman and Pearson on individual data points
cor.matrices$pearson <- cor(BDLdata.fil, method="pearson", use="pairwise.complete.obs")
cor.matrices$spearman <- cor(BDLdata.fil, method="spearman", use="pairwise.complete.obs")
```

Heatmap plot of the correlation matrices with factors in original order

```r
# Heatmap of correlation matrix
correlation_heatmap <- function(method){
  data <- cor.matrices[[method]]
  if(is.null(data)){
    stop("Correlation matrix does not exist for method: ", method)
  }
  cor_colors <- HeatmapColors() # color palette for correlation (red - white - blue)
  heatmap.2(data, col=cor_colors(10), scale="none",
          key=TRUE, symkey=FALSE, trace="none", cexRow=0.8, cexCol=0.8,
          main=method,
          density.info="none", dendrogram="none",
          Rowv=NULL, Colv=NULL,
          keysize=0.8, key.xlab = method,
          #revC=TRUE,
          sepwidth=c(0.01,0.01),
          sepcolor="black",
          colsep=1:ncol(data),
          rowsep=1:nrow(data))
}
```

```r
# Pearson correlation (no clustering)
correlation_heatmap(method="pearson")
```

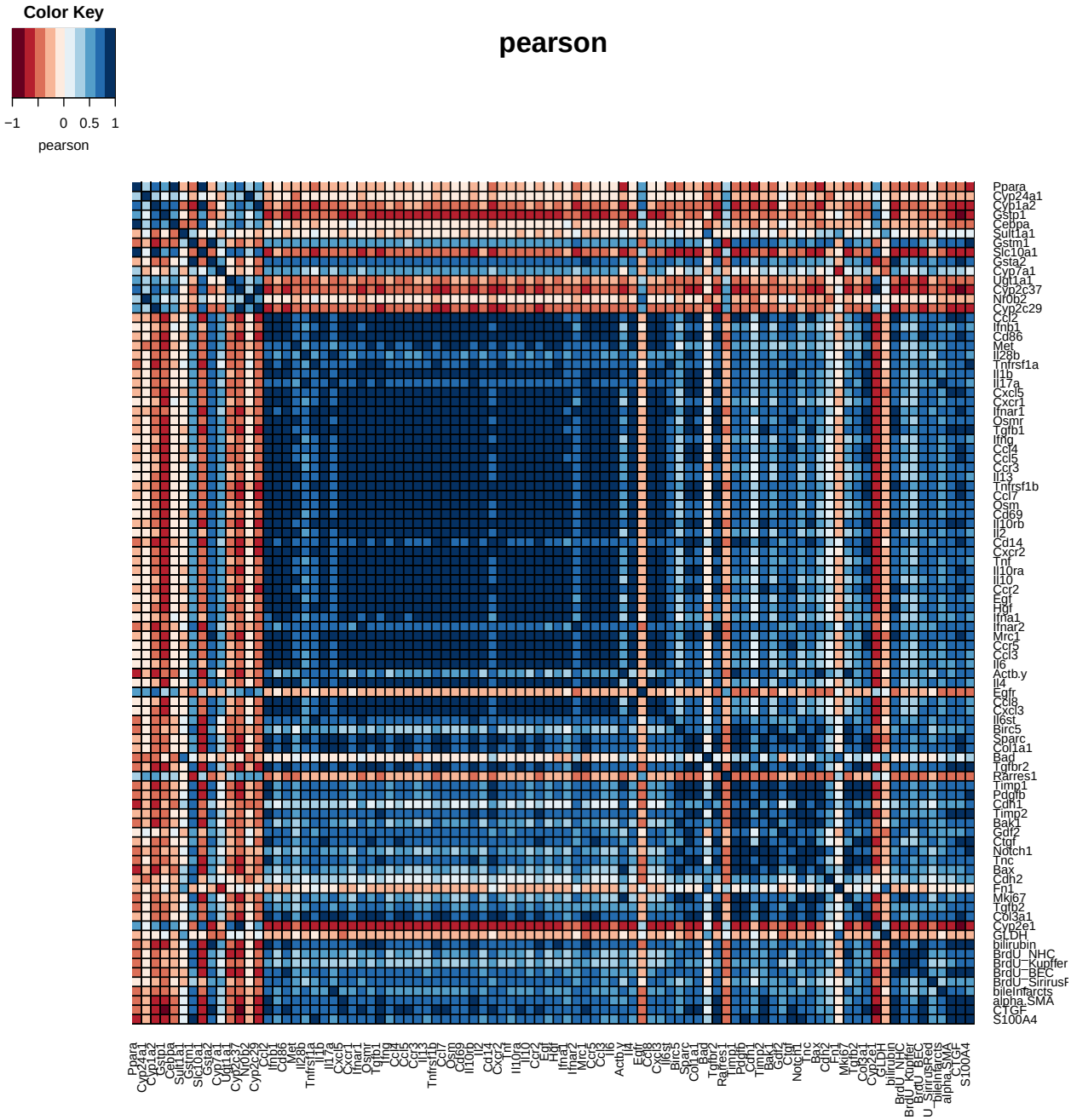


Figure Heatmap Pearson: Correlation matrix based on Pearson correlation.

```
# Spearman correlation (no clustering)
correlation_heatmap(method="spearman")
```

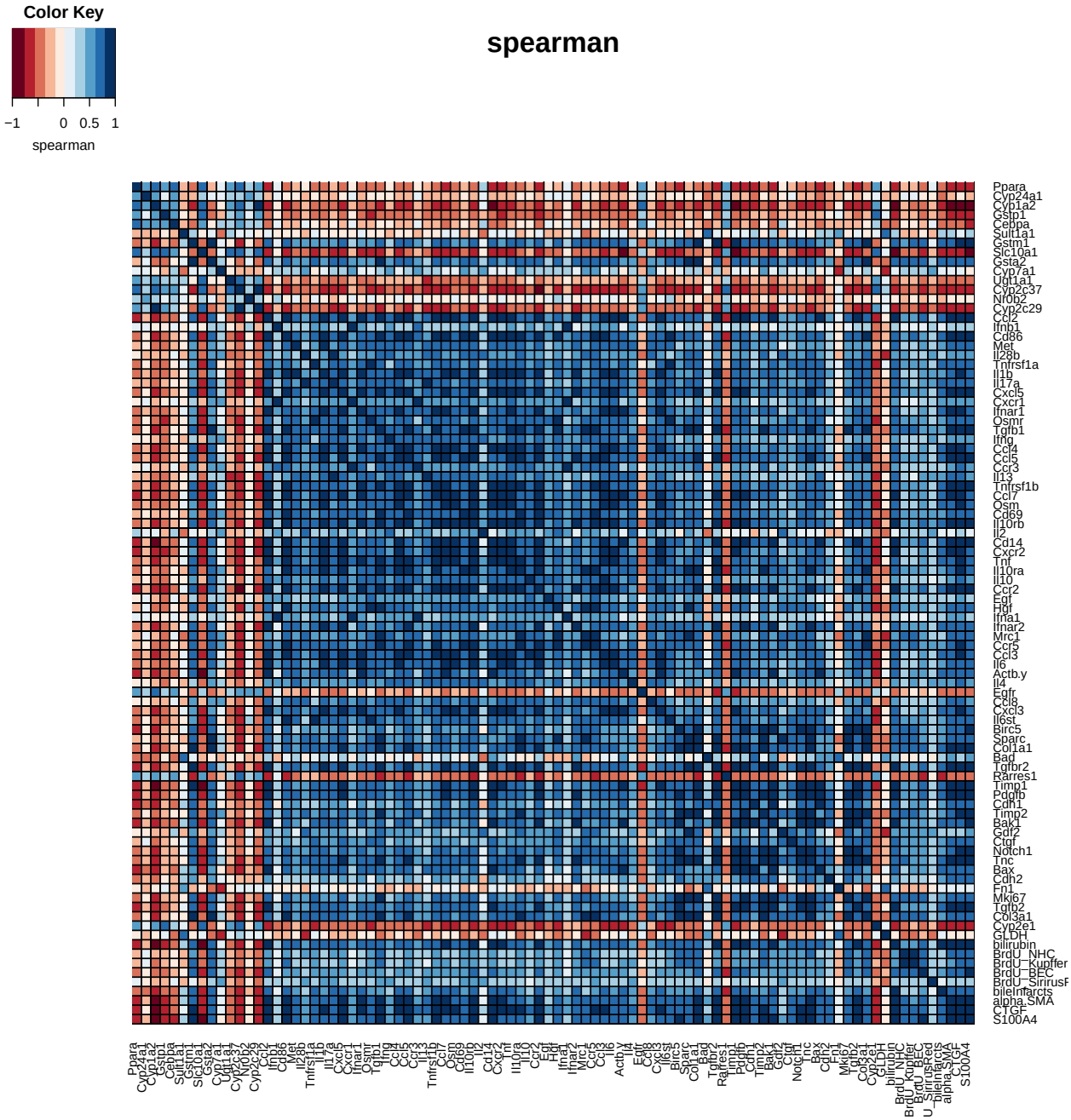


Figure Heatmap Spearman: Correlation matrix based on Spearman correlation.

The pearson correlation is highly sensitive to outliers (correlation between factors, but also in the changes between time points). Prelimary analysis of A^{*} showed similar problems, so we used A^{**} in Y^{S3} instead.

YS & YR correlation

Here, the time-course based correlation measurements Y^{S1}, Y^{S2}, Y^{S3}, Y^{R1}, Y^{R2} and Y^{R3} are calculated for the factors in the filtered BDL data set. To create the correlation matrix all pairwise correlations between all factors are calculated.

```
# weighting factors
w <- list(w1=0.5, w2=0.3, w3=0.2)

# calculate the YSR component matrices on the filtered data set (A, A*, A**, M, M*)
# all components are calculated on the mean data
ysr.res <- ysr.matrices(BDLmean.fil, BDLmean.time, use="pairwise.complete.obs")

# S* and R* (Pearson & Spearman correlation on individual data points)
cor.S_star <- (cor.matrices$spearman + 1)/2
cor.R_star <- (cor.matrices$pearson + 1)/2

# Calculate YS and YR scores based on the components
ysr_methods <- c("ys1", "ys2", "ys3", "yr1", "yr2", "yr3")
cor.ysr <- vector("list", length(ysr_methods))
names(cor.ysr) <- ysr_methods

# unnormalized correlations in [0, 1] as combination of weighted (S/R, A/A*/A**, M/M*)
cor.ysr$ys1 <- w$w1*cor.S_star + w$w2*ysr.res$A       + w$w3*ysr.res$M
cor.ysr$ys2 <- w$w1*cor.S_star + w$w2*ysr.res$A_star  + w$w3*ysr.res$M_star
cor.ysr$ys3 <- w$w1*cor.S_star + w$w2*ysr.res$A_star2 + w$w3*ysr.res$M_star

cor.ysr$yr1 <- w$w1*cor.R_star + w$w2*ysr.res$A       + w$w3*ysr.res$M
cor.ysr$yr2 <- w$w1*cor.R_star + w$w2*ysr.res$A_star  + w$w3*ysr.res$M_star
cor.ysr$yr3 <- w$w1*cor.R_star + w$w2*ysr.res$A_star2 + w$w3*ysr.res$M_star

# scaling of ysr correlation coefficient to interval [-1,1]
for (method in ysr_methods){
  cor.matrices[[method]] <- 2*(cor.ysr[[method]]-0.5)
}

# Save correlation matrices
save(cor.matrices, file=file.path(resultsPath, "data", "cor.matrices.Rdata"))
rm(cor.ysr, cor.R_star, cor.S_star, ysr.res, ysr_methods)
```

Create heatmap plots for all calculated correlation matrices

```
# create all correlation heatmaps on disk
hmap.settings <- list(width=10, height=10, pointsize=12)
for (method in correlation_methods){
  fname <- sprintf('cor.%s.pdf', method)
  print(fname)
  pdf(file.path(resultsPath, "correlation", fname),
      width=hmap.settings$width, height=hmap.settings$height, pointsize=hmap.settings$pointsize)
  correlation_heatmap(method=method)
invisible(dev.off())
rm(method, fname)
}
```

```
## [1] "cor.pearson.pdf"
```

```
## [1] "cor.spearman.pdf"
```

```
## [1] "cor.ys1.pdf"
```

```
## [1] "cor.ys2.pdf"

## [1] "cor.ys3.pdf"

## [1] "cor.yr1.pdf"

## [1] "cor.yr2.pdf"

## [1] "cor.yr3.pdf"
```

```
# ys3 heatmap in report
correlation_heatmap(method="ys3")
```

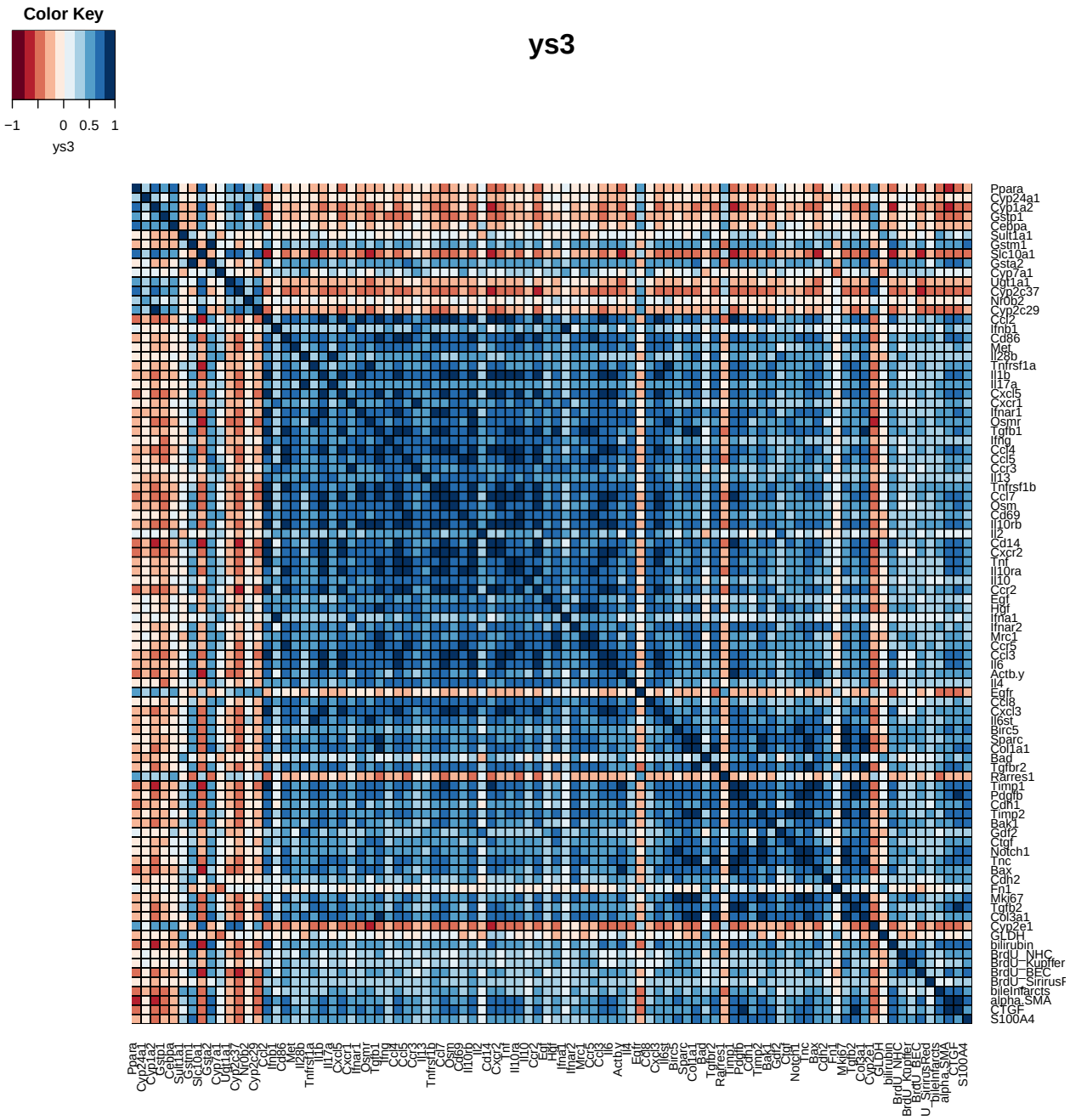


Figure Heatmap YS3: Correlation matrix based on YS3 correlation.

Hierarchical clustering

In the next step clusters are calculated on the correlation matrices using hierarchical clustering with complete linkage. Hereby, the factors are grouped into a number of clusters with the correlation within each cluster being larger than between clusters. This effectivly finds groups of factors which have similar time courses. The complete linkage method defines the cluster distance between two clusters to be the maximum distance between their individual components.

The hierarchical clustering is now calculated based on the respective correlation matrices. Hierarchical clustering based on the given correlation matrix is applied with `Ngroups=6` clusters. For every correlation measure the cluster plots are created. The number of clusters was choosen, so that at least `>1` members exist per cluster.

```
hclust.res <- vector("list", length=length(correlation_methods))
names(hclust.res) <- correlation_methods

# Calculate the hierarical clustering: N clusters for correlation based on method
calculate_clusters <- function(cor_method, N){
  # get correlation matrix
  cor.cluster <- cor.matrices[[cor_method]]
  # perform hierarchical clustering (complete linkage & Euclidion distance measure)
  hc <- hclust(dist(cor.cluster, method="euclidian"), method="complete")
  # cut into the clusters
  groups <- cutree(hc, k=Ngroups)
  groups.hc.order <- groups[hc$order]
  # store results
  return(list(hc=hc,
              groups=groups,
              groups.hc.order=groups.hc.order))
}

Ngroups <- 6   # number of clusters
for (method in correlation_methods){
  hclust.res[[method]] <- calculate_clusters(cor_method=method, N=Ngroups)
}
# save clustering
save(hclust.res, file=file.path(resultsPath, "data", "hclust.res.Rdata"))
rm(method)
```

In the next step the correlation matrices are plotted in combination with the clustering results

```
# Create correlation heatmap with hierachical clustering results
correlation_heatmap_cluster <- function(method){
  # matrix and cluster results
  cor.cluster <- cor.matrices[[method]]
  hc.res <- hclust.res[[method]]
  hc <- hc.res$hc
  groups <- hc.res$groups
  # colors
  hmap_colors <- HeatmapColors()
  colorset <- brewer.pal(Ngroups, "Set1")
```

```
  color.map <- function(cluster_id) {return(colorset[cluster_id])}
  clusterColors <- unlist(lapply(groups, color.map))

  heatmap.2(cor.cluster, col=hmap_colors(10), scale="none",
          key=TRUE, symkey=FALSE, trace="none", cexRow=0.8, cexCol=0.8,
          main=method,
          density.info="none", dendrogram="column", Rowv=as.dendrogram(hc), Colv=as.dendrogram(hc), keys
          key.xlab=method,
          ColSideColors=clusterColors, revC=TRUE,
          sepwidth=c(0.01,0.01),
          sepcolor="black",
          colsep=1:ncol(cor.cluster),
          rowsep=1:nrow(cor.cluster),
          margins=c(12,8))
  legend("left", legend=paste("c", 1:6, sep=""), col= unlist(lapply(1:6, color.map)), pch=15, bty="n")
}

# plot to files
for(method in correlation_methods){
  pdf(file.path(resultsPath, "correlation", sprintf("cor.%s.hclust.pdf", method)),
      width=10, height=10, pointsize=12)
  correlation_heatmap_cluster(method=method)
  invisible(dev.off())
}
rm(method)
```

Display the clusters with the heatmap

```
# plot ys3 to report
correlation_heatmap_cluster(method="ys3")
```

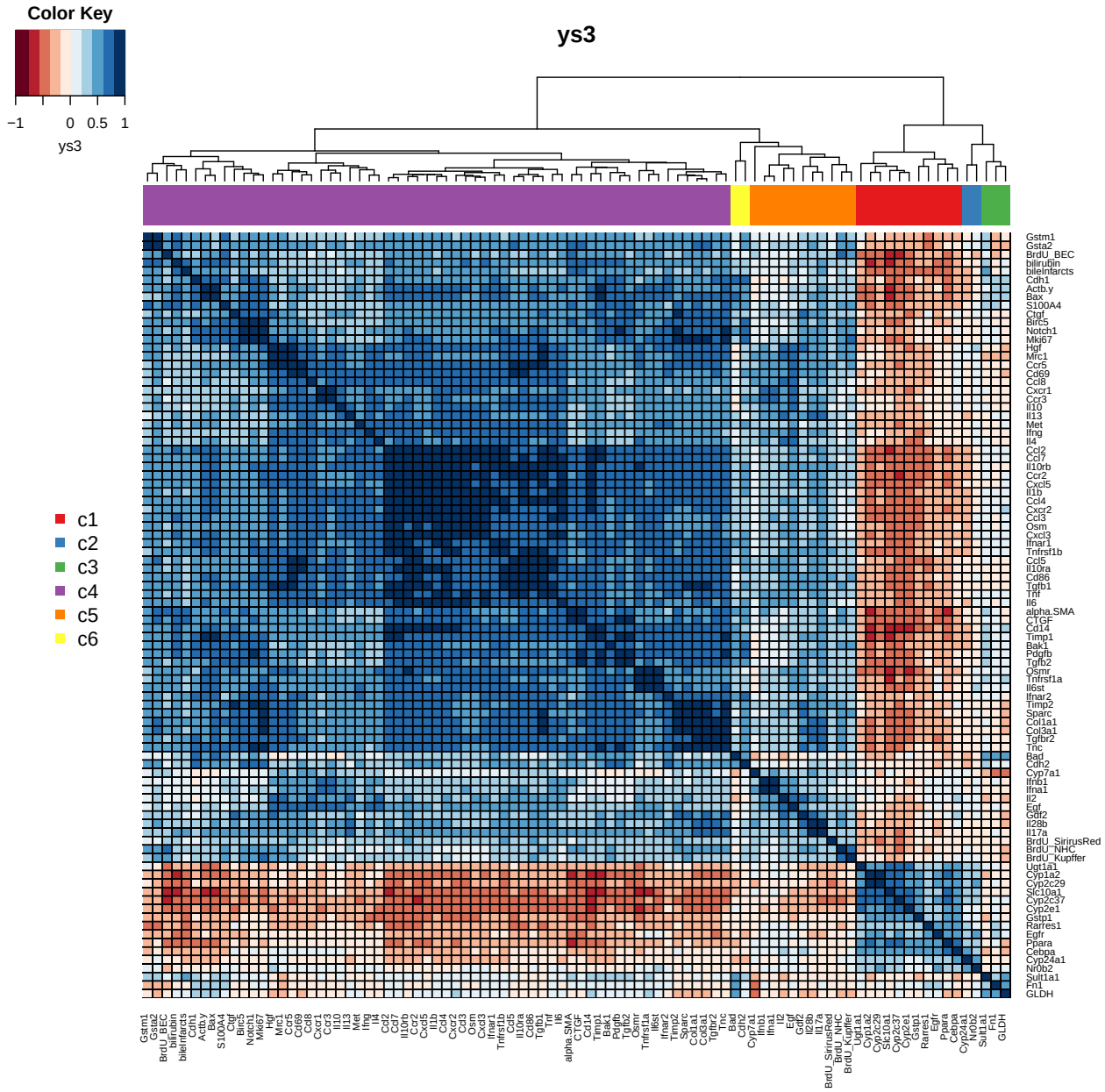


Figure Heatmap Cluster: Correlation matrix based on YS3 correlation with hierachical clustering infromation.

The following factors are in the `ys3` time course clusters

```
# print representatives of cluster
list_cluster_members <- function(method){
  groups <- hclust.res[[method]]$groups
  cat(sprintf("-----------------------------------------\n"))
  cat(sprintf("Correlation method: *** %s ***\n", method))
  cat(sprintf("-----------------------------------------\n"))
  for (k in 1:Ngroups){
    g <- groups[groups==k]
    cat(sprintf("Cluster %s (N=%s)\n", k, length(g)))
    print(names(g))
    cat("\n")
```

```
    # print(paste(names(g), sep=", ", collapse =", " ))
  }
}
list_cluster_members(method="ys3")
```

```
## ----------------------------------------
## Correlation method: *** ys3 ***
## ----------------------------------------
## Cluster 1 (N=11)
##  [1] "Ppara"   "Cyp1a2"  "Gstp1"   "Cebpa"   "Slc10a1" "Ugt1a1"  "Cyp2c37"
##  [8] "Cyp2c29" "Egfr"    "Rarres1" "Cyp2e1"
##
## Cluster 2 (N=2)
## [1] "Cyp24a1" "Nr0b2"
##
## Cluster 3 (N=3)
## [1] "Sult1a1" "Fn1"     "GLDH"
##
## Cluster 4 (N=61)
##  [1] "Gstm1"        "Gsta2"        "Ccl2"         "Cd86"
##  [5] "Met"          "Tnfrsf1a"     "Il1b"         "Cxcl5"
##  [9] "Cxcr1"        "Ifnar1"       "Osmr"         "Tgfb1"
## [13] "Ifng"         "Ccl4"         "Ccl5"         "Ccr3"
## [17] "Il13"         "Tnfrsf1b"     "Ccl7"         "Osm"
## [21] "Cd69"         "Il10rb"       "Cd14"         "Cxcr2"
## [25] "Tnf"          "Il10ra"       "Il10"         "Ccr2"
## [29] "Hgf"          "Ifnar2"       "Mrc1"         "Ccr5"
## [33] "Ccl3"         "Il6"          "Actb.y"       "Il4"
## [37] "Ccl8"         "Cxcl3"        "Il6st"        "Birc5"
## [41] "Sparc"        "Col1a1"       "Tgfbr2"       "Timp1"
## [45] "Pdgfb"        "Cdh1"         "Timp2"        "Bak1"
## [49] "Ctgf"         "Notch1"       "Tnc"          "Bax"
## [53] "Mki67"        "Tgfb2"        "Col3a1"       "bilirubin"
## [57] "BrdU_BEC"     "bileInfarcts" "alpha.SMA"    "CTGF"
## [61] "S100A4"
##
## Cluster 5 (N=11)
##  [1] "Cyp7a1"          "Ifnb1"           "Il28b"
##  [4] "Il17a"           "Il2"             "Egf"
##  [7] "Ifna1"           "Gdf2"            "BrdU_NHC"
## [10] "BrdU_Kupffer"    "BrdU_SirirusRed"
##
## Cluster 6 (N=2)
## [1] "Bad"  "Cdh2"
```

Correlations of histological, biochemical and antibody factors

Plot correlation matrix for histological, biochemical and antibody markers. Calculation of the largest correlation between the non-transcript factors and other factors. The columns in which no correlation coefficient has `abs(value) >= cor.cutoff` are filtered out, i.e. only factors are retained with a absolute correlation coeffient above this threshold.

```r
get_histological_correlations <- function(method="ys3", cor.cutoff=0.6){
  # correlation matrix
  cor_mat <- cor.matrices[[method]]

  # non RNA factors which were accepted by ANOVA
  hist_facs <- BDLfactors$id[anova.accept & (BDLfactors$ftype %in% c("Antibodies", "Biochemistry", "His
  print(hist_facs)

  # get the indices of these factors in the correlation matrix
  hist_idx <- rep(NA, length(hist_facs))
  for (k in 1:length(hist_facs)){
    hist_idx[k] <- which(colnames(cor_mat) == hist_facs[k])
  }
  # subset of correlation matrix for histological markers (corresponding rows)
  hist_data <-  cor_mat[hist_idx, ]

  # filter columns by correlation cutoff
  col.accept <- rep(NA, ncol(hist_data))
  for (k in 1:ncol(hist_data)){
    # if there is any correlation >= cutoff in the column, keep it
    col.accept[k] <- any( abs(hist_data[,k])>=cor.cutoff )
  }
  print(table(col.accept))  # how many factors are filtered based on correlation threshold
  hist_accept <- hist_data[, col.accept]

  # sort by the hierarchical cluster ordering
  hist_gene_names <- colnames(hist_accept)[1:(ncol(hist_accept)-nrow(hist_accept))]

  # correlation based cluster for ordering
  hc.res <- hclust.res[[method]]

  # create sort index for gene names based on clustering
  sort_idx <- rep(NA, length(hist_gene_names))
  for (k in 1:length(hist_gene_names)){
    sort_idx[k] <- which(names(hc.res$groups.hc.order) == hist_gene_names[k])
  }
  # first sorted genes than the histological factors
  hist_sorted <- hist_accept[, c(hist_gene_names[order(sort_idx)], rownames(hist_data))]
  return(hist_sorted)
}

histological_correlations <- get_histological_correlations(method="ys3")
```

```
##  [1] GLDH            bilirubin       BrdU_NHC        BrdU_Kupffer
##  [5] BrdU_BEC        BrdU_SirirusRed bileInfarcts    alpha.SMA
##  [9] CTGF            S100A4
## 153 Levels: Abcb1a Abcc2 Abcg2 Acta2 Actb Actb.x Actb.y Ahr ... Xiap
## col.accept
## FALSE  TRUE
##    42    48
```

```r
# plot the subset of the correlation matrix to file
plot_histological_correlations <- function(hist_data){
  hmap_colors <- HeatmapColors()
  corrplot(hist_data, method="circle", type="full",
           tl.cex=0.7, tl.col="black", col=hmap_colors(10))
}

# plot to file
pdf(file.path(resultsPath, "correlation", "histological_correlations.pdf"),
    width=10, height=4, pointsize=12)
plot_histological_correlations(histological_correlations)
invisible(dev.off())

# print correlation values
print(round(t(histological_correlations), digits=2))
```

```
##                 GLDH bilirubin BrdU_NHC BrdU_Kupffer BrdU_BEC
## Gstm1          -0.02      0.61     0.42         0.40     0.43
## Gsta2          -0.21      0.61     0.51         0.41     0.46
## Cdh1            0.24      0.46     0.25         0.34     0.35
## Actb.y          0.22      0.61     0.29         0.33     0.47
## Bax             0.23      0.61     0.31         0.35     0.42
## Birc5          -0.04      0.57     0.53         0.50     0.42
## Notch1         -0.04      0.52     0.43         0.50     0.31
## Mki67          -0.15      0.50     0.59         0.61     0.47
## Ccl2            0.17      0.66     0.25         0.25     0.52
## Ccl7            0.15      0.60     0.18         0.22     0.49
## Il10rb          0.00      0.57     0.31         0.33     0.50
## Ccr2            0.03      0.56     0.29         0.29     0.53
## Cxcl5           0.00      0.54     0.26         0.23     0.49
## Il1b            0.07      0.54     0.18         0.17     0.38
## Ccl4            0.13      0.55     0.16         0.22     0.45
## Cxcr2           0.22      0.55     0.15         0.15     0.44
## Ccl3            0.18      0.62     0.16         0.17     0.45
## Osm             0.19      0.59     0.15         0.14     0.38
## Cxcl3           0.18      0.61     0.12         0.18     0.41
## Ifnar1          0.11      0.56     0.21         0.23     0.36
## Il10ra         -0.15      0.46     0.31         0.30     0.39
## Cd86           -0.07      0.45     0.39         0.41     0.46
## Tgfb1          -0.14      0.48     0.41         0.40     0.46
## Tnf             0.00      0.49     0.25         0.25     0.34
## Il6             0.03      0.51     0.13         0.20     0.40
## Cd14            0.21      0.67     0.29         0.21     0.50
## Timp1           0.12      0.67     0.32         0.34     0.50
## Bak1            0.08      0.60     0.43         0.40     0.40
## Pdgfb           0.08      0.62     0.33         0.37     0.41
## Tgfb2           0.11      0.58     0.30         0.33     0.38
## Timp2          -0.24      0.51     0.47         0.39     0.37
## Col1a1         -0.22      0.55     0.53         0.38     0.48
## Tgfbr2         -0.10      0.59     0.47         0.31     0.37
## Tnc            -0.10      0.54     0.43         0.37     0.40
## Cyp1a2         -0.15     -0.60    -0.11        -0.05    -0.44
## Slc10a1        -0.17     -0.62    -0.38        -0.29    -0.66
```

```
## Cyp2c37          0.03     -0.55    -0.41        -0.25    -0.66
## Ppara           -0.22     -0.47    -0.18        -0.05    -0.49
## GLDH             1.00      0.18    -0.30        -0.20    -0.02
## bilirubin        0.18      1.00     0.39         0.26     0.49
## BrdU_NHC        -0.30      0.39     1.00         0.75     0.62
## BrdU_Kupffer    -0.20      0.26     0.75         1.00     0.57
## BrdU_BEC        -0.02      0.49     0.62         0.57     1.00
## BrdU_SirirusRed -0.10      0.35     0.32         0.30     0.36
## bileInfarcts     0.14      0.71     0.25         0.10     0.39
## alpha.SMA       -0.04      0.65     0.41         0.27     0.60
## CTGF            -0.06      0.69     0.46         0.25     0.56
## S100A4           0.08      0.62     0.54         0.47     0.62
##                BrdU_SirirusRed bileInfarcts alpha.SMA  CTGF S100A4
## Gstm1                     0.10         0.59      0.55  0.57   0.70
## Gsta2                     0.19         0.59      0.64  0.62   0.59
## Cdh1                      0.18         0.42      0.54  0.54   0.61
## Actb.y                    0.29         0.50      0.46  0.60   0.52
## Bax                       0.38         0.44      0.48  0.55   0.59
## Birc5                     0.36         0.41      0.61  0.59   0.72
## Notch1                    0.26         0.39      0.53  0.61   0.71
## Mki67                     0.35         0.39      0.55  0.57   0.68
## Ccl2                      0.36         0.54      0.61  0.72   0.46
## Ccl7                      0.32         0.48      0.62  0.67   0.48
## Il10rb                    0.40         0.46      0.64  0.70   0.50
## Ccr2                      0.33         0.49      0.59  0.74   0.46
## Cxcl5                     0.34         0.43      0.73  0.75   0.53
## Il1b                      0.37         0.37      0.63  0.71   0.48
## Ccl4                      0.33         0.42      0.61  0.68   0.46
## Cxcr2                     0.33         0.42      0.61  0.71   0.44
## Ccl3                      0.23         0.53      0.62  0.69   0.50
## Osm                       0.37         0.45      0.59  0.63   0.51
## Cxcl3                     0.36         0.44      0.53  0.57   0.47
## Ifnar1                    0.30         0.44      0.56  0.67   0.49
## Il10ra                    0.33         0.38      0.62  0.68   0.44
## Cd86                      0.40         0.32      0.62  0.72   0.44
## Tgfb1                     0.38         0.39      0.60  0.68   0.48
## Tnf                       0.31         0.40      0.60  0.69   0.49
## Il6                       0.25         0.37      0.65  0.61   0.52
## Cd14                      0.37         0.47      0.67  0.77   0.50
## Timp1                     0.27         0.57      0.73  0.73   0.64
## Bak1                      0.35         0.41      0.56  0.67   0.60
## Pdgfb                     0.30         0.48      0.69  0.80   0.72
## Tgfb2                     0.28         0.51      0.73  0.82   0.72
## Timp2                     0.42         0.38      0.63  0.62   0.68
## Col1a1                    0.38         0.42      0.59  0.61   0.66
## Tgfbr2                    0.31         0.46      0.69  0.78   0.59
## Tnc                       0.33         0.42      0.70  0.73   0.67
## Cyp1a2                   -0.20        -0.56     -0.70 -0.60  -0.45
## Slc10a1                  -0.37        -0.50     -0.54 -0.55  -0.44
## Cyp2c37                  -0.41        -0.49     -0.59 -0.55  -0.44
## Ppara                    -0.05        -0.41     -0.61 -0.59  -0.37
## GLDH                     -0.10         0.14     -0.04 -0.06   0.08
## bilirubin                 0.35         0.71      0.65  0.69   0.62
## BrdU_NHC                  0.32         0.25      0.41  0.46   0.54
```

```
## BrdU_Kupffer          0.30         0.10      0.27  0.25  0.47
## BrdU_BEC              0.36         0.39      0.60  0.56  0.62
## BrdU_SirirusRed       1.00         0.06      0.20  0.21  0.38
## bileInfarcts          0.06         1.00      0.61  0.63  0.44
## alpha.SMA             0.20         0.61      1.00  0.84  0.69
## CTGF                  0.21         0.63      0.84  1.00  0.60
## S100A4                0.38         0.44      0.69  0.60  1.00
```

```
# plot in report
plot_histological_correlations(histological_correlations)
```

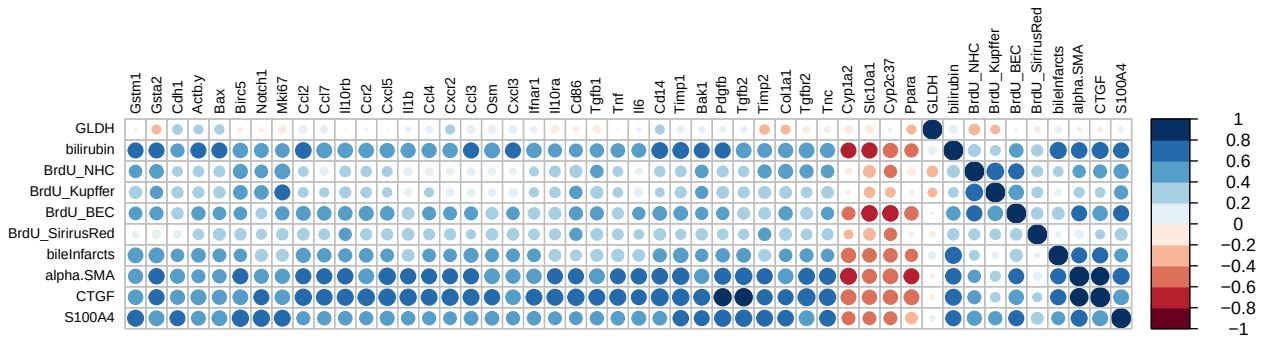


Figure Histological Correlations: Correlations between non-RNA factors and all other factors.

Top correlations for non-RNA factors

List the top correlations of every non-RNA factor, i.e. histological (H), biochemical (B) and immunohistochemical (A). The top correlations are sorted by absolute correlation values.

```
plot_hist_topcors <- function(method="ys3", labels=TRUE, mfrow=c(10,1),
                              only_RNA=FALSE, Ntop=10){
  hist_facs <- rownames(histological_correlations)
  cor.mat <- cor.matrices[[method]]

  par(mfrow=mfrow)
  hmap_colors <- HeatmapColors()
  for (name in hist_facs){
    if (only_RNA==TRUE){
      # get the elements which are not histological factors
      v <- cor.mat[!(colnames(cor.mat) %in% hist_facs), name]
    } else {
      # get all elements (including other non-RNA factors)
      v <- cor.mat[, name]
    }

    # sort by absolute correlation
    v.sorted <- rev(v[order(abs(v))])

    # and get the Ntop values without the self-correlation (idx=1)
    mv <- t(as.matrix(v.sorted[2:(Ntop+1)]))
    rownames(mv) <- c(name)

    # overview
```

```
    print(round(mv, digits=2))

    # plot without labels to have identical size of figure
    if (labels==FALSE){
      rownames(mv) <- NULL
      colnames(mv) <- NULL
    }
    corrplot(mv, method="pie", type="full",
             tl.cex=1.0, tl.col="black", col=hmap_colors(100), insig="p-value", sig.level=-1,
             p.mat=mv, cl.pos="n")
  }
  par(mfrow=c(1,1))
}

# plot to file
# necessary to plot once with and without labels so that all the plots are scaled equally
pdf(file.path(resultsPath, "correlation", "histological_topcors_1.pdf"),
    width=10, height=10, pointsize=12)
plot_hist_topcors(method="ys3", labels=TRUE)
```

```
##      Sult1a1  Bad  Fn1 Cyp7a1  Mrc1 Il28b  Gdf2   Il2 BrdU_NHC Cebpa
## GLDH    0.55 0.54 0.51  -0.47 -0.37 -0.35 -0.32 -0.31     -0.3  -0.3

##           bileInfarcts CTGF Timp1 Cd14 Ccl2 alpha.SMA Ccl3 Slc10a1 S100A4
## bilirubin         0.71 0.69  0.67 0.67 0.66      0.65 0.62   -0.62   0.62
##           Pdgfb
## bilirubin  0.62

##          BrdU_Kupffer BrdU_BEC Mki67 S100A4 Birc5 Col1a1 Sparc Gsta2 Mrc1
## BrdU_NHC         0.75     0.62  0.59   0.54  0.53   0.53  0.53  0.51  0.5
##          Col3a1
## BrdU_NHC   0.49

##              BrdU_NHC Mki67 BrdU_BEC Birc5 Notch1 S100A4 Ccr5 Mrc1 Sparc
## BrdU_Kupffer     0.75  0.61     0.57   0.5    0.5   0.47 0.45 0.44  0.43
##               Hgf
## BrdU_Kupffer 0.43

##          Cyp2c37 Slc10a1 BrdU_NHC S100A4 alpha.SMA BrdU_Kupffer CTGF
## BrdU_BEC   -0.66   -0.66     0.62   0.62       0.6         0.57 0.56
##          Cyp2e1 Ccr2 Cyp2c29
## BrdU_BEC  -0.56 0.53   -0.52

##                 Timp2 Cyp2c29 Cyp2c37 Tnfrsf1b Il10rb Ccr5 Cd86 Ifng Sparc
## BrdU_SiririusRed  0.42   -0.41   -0.41     0.41    0.4  0.4  0.4 0.39  0.38
##                 Col3a1
## BrdU_SiririusRed   0.38

##              bilirubin CTGF alpha.SMA Gsta2 Gstm1 Timp1 Cyp1a2 Ccl2 Cyp2e1
## bileInfarcts      0.71 0.63      0.61  0.59  0.59  0.57  -0.56 0.54  -0.53
##              Ccl3
## bileInfarcts 0.53
```

```
##           CTGF Tgfb2 Cxcl5 Timp1 Tnc Cyp1a2 S100A4 Pdgfb Tgfbr2 Cd14
## alpha.SMA 0.84  0.73  0.73  0.73 0.7   -0.7   0.69  0.69   0.69 0.67
```

```
##      alpha.SMA Tgfb2 Pdgfb Tgfbr2 Cd14 Cxcl5 Ccr2 Timp1  Tnc Cd86
## CTGF      0.84  0.82   0.8   0.78 0.77  0.75 0.74  0.73 0.73 0.72
```

```
##        Pdgfb Birc5 Tgfb2 Notch1 Gstm1 alpha.SMA Timp2 Mki67  Tnc Col1a1
## S100A4  0.72  0.72  0.72   0.71   0.7      0.69  0.68  0.68 0.67   0.66
```

```
dev.off()
```

```
## pdf
##   2
```

```
pdf(file.path(resultsPath, "correlation", "histological_topcors_2.pdf"),
    width=10, height=10, pointsize=12)
plot_hist_topcors(method="ys3", labels=FALSE)
```

```
##      Sult1a1  Bad  Fn1 Cyp7a1  Mrc1 Il28b  Gdf2   Il2 BrdU_NHC Cebpa
## GLDH    0.55 0.54 0.51  -0.47 -0.37 -0.35 -0.32 -0.31     -0.3  -0.3
```

```
##           bileInfarcts CTGF Timp1 Cd14 Ccl2 alpha.SMA Ccl3 Slc10a1 S100A4
## bilirubin         0.71 0.69  0.67 0.67 0.66      0.65 0.62   -0.62   0.62
##           Pdgfb
## bilirubin  0.62
```

```
##          BrdU_Kupffer BrdU_BEC Mki67 S100A4 Birc5 Col1a1 Sparc Gsta2 Mrc1
## BrdU_NHC         0.75     0.62  0.59   0.54  0.53   0.53  0.53  0.51  0.5
##          Col3a1
## BrdU_NHC   0.49
```

```
##              BrdU_NHC Mki67 BrdU_BEC Birc5 Notch1 S100A4 Ccr5 Mrc1 Sparc
## BrdU_Kupffer     0.75  0.61     0.57   0.5    0.5   0.47 0.45 0.44  0.43
##                Hgf
## BrdU_Kupffer 0.43
```

```
##          Cyp2c37 Slc10a1 BrdU_NHC S100A4 alpha.SMA BrdU_Kupffer CTGF
## BrdU_BEC   -0.66   -0.66     0.62   0.62       0.6         0.57 0.56
##          Cyp2e1 Ccr2 Cyp2c29
## BrdU_BEC  -0.56 0.53   -0.52
```

```
##                 Timp2 Cyp2c29 Cyp2c37 Tnfrsf1b Il10rb Ccr5 Cd86 Ifng Sparc
## BrdU_SiririusRed  0.42   -0.41   -0.41     0.41    0.4  0.4  0.4 0.39  0.38
##                 Col3a1
## BrdU_SiririusRed   0.38
```

```
##              bilirubin CTGF alpha.SMA Gsta2 Gstm1 Timp1 Cyp1a2 Ccl2 Cyp2e1
## bileInfarcts      0.71 0.63      0.61  0.59  0.59  0.57  -0.56 0.54  -0.53
##              Ccl3
## bileInfarcts 0.53
```

```
##           CTGF Tgfb2 Cxcl5 Timp1 Tnc Cyp1a2 S100A4 Pdgfb Tgfbr2 Cd14
## alpha.SMA 0.84  0.73  0.73  0.73 0.7   -0.7   0.69  0.69   0.69 0.67
```

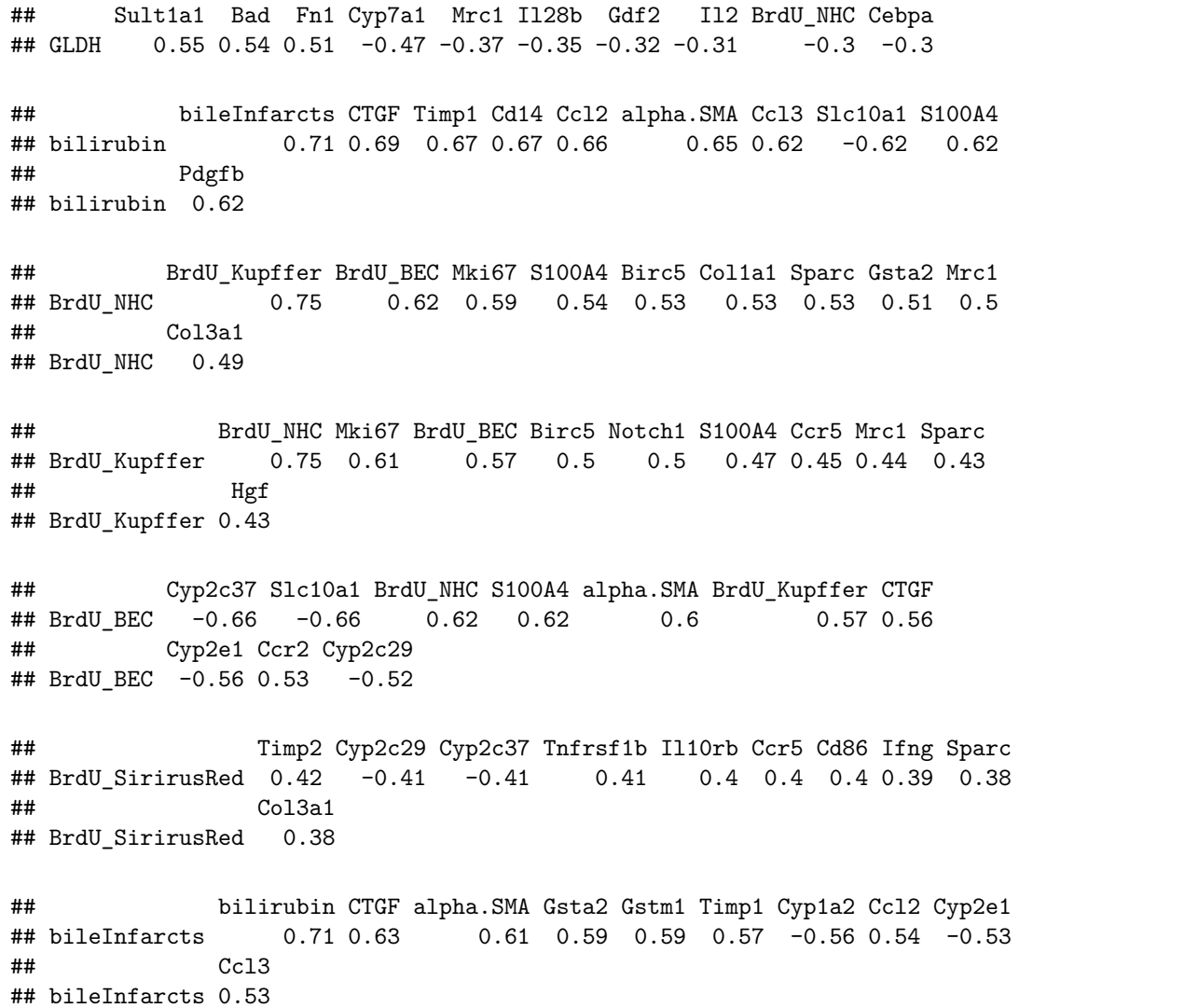
```
##      alpha.SMA Tgfb2 Pdgfb Tgfbr2 Cd14 Cxcl5 Ccr2 Timp1  Tnc Cd86
## CTGF      0.84  0.82   0.8   0.78 0.77  0.75 0.74  0.73 0.73 0.72
```

```
##        Pdgfb Birc5 Tgfb2 Notch1 Gstm1 alpha.SMA Timp2 Mki67  Tnc Col1a1
## S100A4  0.72  0.72  0.72   0.71   0.7      0.69  0.68  0.68 0.67   0.66
```

```
dev.off()
```

```
## pdf
##   2
```

```
# plot in report
plot_hist_topcors(method="ys3", labels=TRUE, mfrow=c(5,2))
```



```
##      Sult1a1  Bad  Fn1 Cyp7a1  Mrc1 Il28b  Gdf2   Il2 BrdU_NHC Cebpa
## GLDH    0.55 0.54 0.51  -0.47 -0.37 -0.35 -0.32 -0.31     -0.3  -0.3
```

```
##           bileInfarcts CTGF Timp1 Cd14 Ccl2 alpha.SMA Ccl3 Slc10a1 S100A4
## bilirubin         0.71 0.69  0.67 0.67 0.66      0.65 0.62   -0.62   0.62
##           Pdgfb
## bilirubin  0.62
```

```
##          BrdU_Kupffer BrdU_BEC Mki67 S100A4 Birc5 Col1a1 Sparc Gsta2 Mrc1
## BrdU_NHC         0.75     0.62  0.59   0.54  0.53   0.53  0.53  0.51  0.5
##          Col3a1
## BrdU_NHC   0.49
```

```
##              BrdU_NHC Mki67 BrdU_BEC Birc5 Notch1 S100A4 Ccr5 Mrc1 Sparc
## BrdU_Kupffer     0.75  0.61     0.57   0.5    0.5   0.47 0.45 0.44  0.43
##                Hgf
## BrdU_Kupffer 0.43
```

```
##          Cyp2c37 Slc10a1 BrdU_NHC S100A4 alpha.SMA BrdU_Kupffer CTGF
## BrdU_BEC   -0.66   -0.66     0.62   0.62       0.6         0.57 0.56
##          Cyp2e1 Ccr2 Cyp2c29
## BrdU_BEC  -0.56 0.53   -0.52
```

```
##                 Timp2 Cyp2c29 Cyp2c37 Tnfrsf1b Il10rb Ccr5 Cd86 Ifng Sparc
## BrdU_SirirusRed  0.42   -0.41   -0.41     0.41    0.4  0.4  0.4 0.39  0.38
##                 Col3a1
## BrdU_SirirusRed   0.38
```

```
##              bilirubin CTGF alpha.SMA Gsta2 Gstm1 Timp1 Cyp1a2 Ccl2 Cyp2e1
## bileInfarcts      0.71 0.63      0.61  0.59  0.59  0.57  -0.56 0.54  -0.53
##              Ccl3
## bileInfarcts 0.53
```

```
##           CTGF Tgfb2 Cxcl5 Timp1 Tnc Cyp1a2 S100A4 Pdgfb Tgfbr2 Cd14
## alpha.SMA 0.84  0.73  0.73  0.73 0.7   -0.7   0.69  0.69   0.69 0.67

##      alpha.SMA Tgfb2 Pdgfb Tgfbr2 Cd14 Cxcl5 Ccr2 Timp1  Tnc Cd86
## CTGF      0.84  0.82   0.8   0.78 0.77  0.75 0.74  0.73 0.73 0.72

##        Pdgfb Birc5 Tgfb2 Notch1 Gstm1 alpha.SMA Timp2 Mki67  Tnc Col1a1
## S100A4  0.72  0.72  0.72   0.71   0.7      0.69  0.68  0.68 0.67   0.66
```

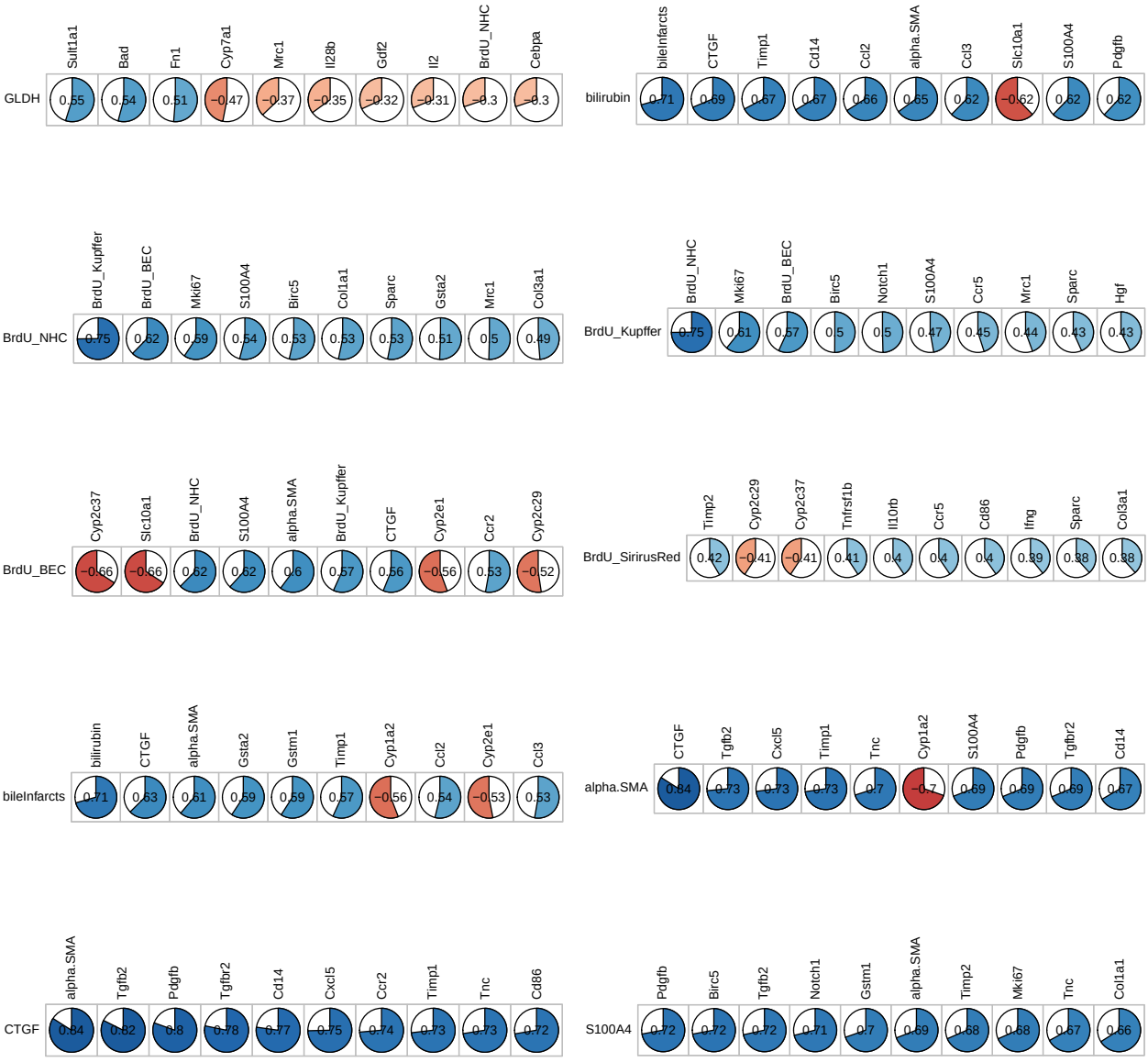


Figure Histological Top Correlations: Top correlations between non-RNA factors and all other factors.

Mean cluster time course

In the next step, we were interested in the time courses of the time course clusters: What are the typical profiles found in the different clusters and which factors are in these clusters. For the comparison of individual

factors against each other and against the mean time course of the cluster the factors were normalized. The normalization was hereby performed for every factor separately based on

$$f_k^{norm}(t_{i,r}) = \frac{f_k(t_{i,r}) - < f_k >}{max(f_x) - min(f_x)}$$

```
suppressPackageStartupMessages(library('matrixStats'))
dir.create(file.path(resultsPath, 'cluster'), showWarnings=FALSE)

# Normalize the individual factors
normalize_factor <- function(a, min.a, max.a, mean.a){
  res <- (a - mean.a)/(max.a - min.a)
}
# Calculate min, max and mean for all single factors (normalization constants)
factor.norm <- data.frame(min=apply(BDLdata, 2, min, na.rm=TRUE),
                          max=apply(BDLdata, 2, max, na.rm=TRUE),
                          mean=apply(BDLdata, 2, mean, na.rm=TRUE))
# Function for normalizing subset of BDL data with factor normalization constants.
normalize_BDLdata <- function(data, factor.norm){
  dnorm <- data
  for (name in colnames(data)){
    dnorm[, name] <- normalize_factor(a=dnorm[, name],
                                      min.a=factor.norm[name, "min"],
                                      max.a=factor.norm[name, "max"],
                                      mean.a=factor.norm[name, "mean"])
  }
  return(dnorm)
}

# Normalize the full data set
BDLdata.norm <- normalize_BDLdata(data=BDLdata, factor.norm=factor.norm)
# Calculate the mean of the normalized data set
BDLmean.norm <- bdl_mean_data(BDLdata.norm, BDLsamples)
```

Based on the normalized factor data the mean time course of the clusters were calculated, i.e. the time course due to averaging over all factors in the individual clusters. In addition the mean time course averaged over the repeats for all factors are plotted.

```
# Plot mean cluster with SD range and the individual representatives in the cluster.
plot_clusters_mean <- function(method){
  # clusters for correlation method
  hc.res <- hclust.res[[method]]
  groups.hc.order <- hc.res$groups.hc.order

  # par(mfrow=c(ceiling(sqrt(Ngroups)),ceiling(sqrt(Ngroups))))
  par(mfrow=c(2,3))
  steps <- 1:Nt  # time points
  for (k in 1:Ngroups){
    # get representatives of cluster
    g <- groups.hc.order[groups.hc.order==k]
    dgroup <- BDLmean.norm[names(g)]  # normalized mean data for group (normalized within factor)

    # mean and sd for timepoints (i.e. over all factors in the cluster)
    g.mean <- rowMeans(as.matrix(dgroup), na.rm=TRUE)
```

```
    g.sd <- rowSds(as.matrix(dgroup), na.rm=TRUE)

    # plot sd range
    plot(factor(levels(BDLsamples$time_fac), levels=levels(BDLsamples$time_fac)), rep(-2, 8), type="n",
         xlim=c(1, Nt), ylim=1.1*c( min(min(dgroup, na.rm=TRUE), na.rm=TRUE),
                                    max(max(dgroup, na.rm=TRUE), na.rm=TRUE) ),
         main=sprintf("%s : Cluster %s (N=%s)", method, k, ncol(dgroup)))
    polygon(c(steps, rev(steps)), c(g.mean+g.sd, rev(g.mean-g.sd)),
            col = rgb(0,0,1,0.2), border = NA)

    # individual data
    for (name in names(g)){
      points(steps, dgroup[, name], pch=16, col="black")
      lines(steps, dgroup[, name], col=rgb(0.5,0.5,0.5,1.0), lwd=1)
    }
    # mean over factors in cluster
    lines(steps, g.mean, col="blue", lwd=2)
  }
  par(mfrow=c(1,1))
}

# plot mean clusters to file
for (method in correlation_methods){
  pdf(file.path(resultsPath, 'cluster', sprintf("%s_cluster_mean.pdf", method)),
        width=10, height=7.5, pointsize=12)
  plot_clusters_mean(method=method)
  invisible(dev.off())
}
```

```
# plot ys3 clusters to report
plot_clusters_mean(method="ys3")
```

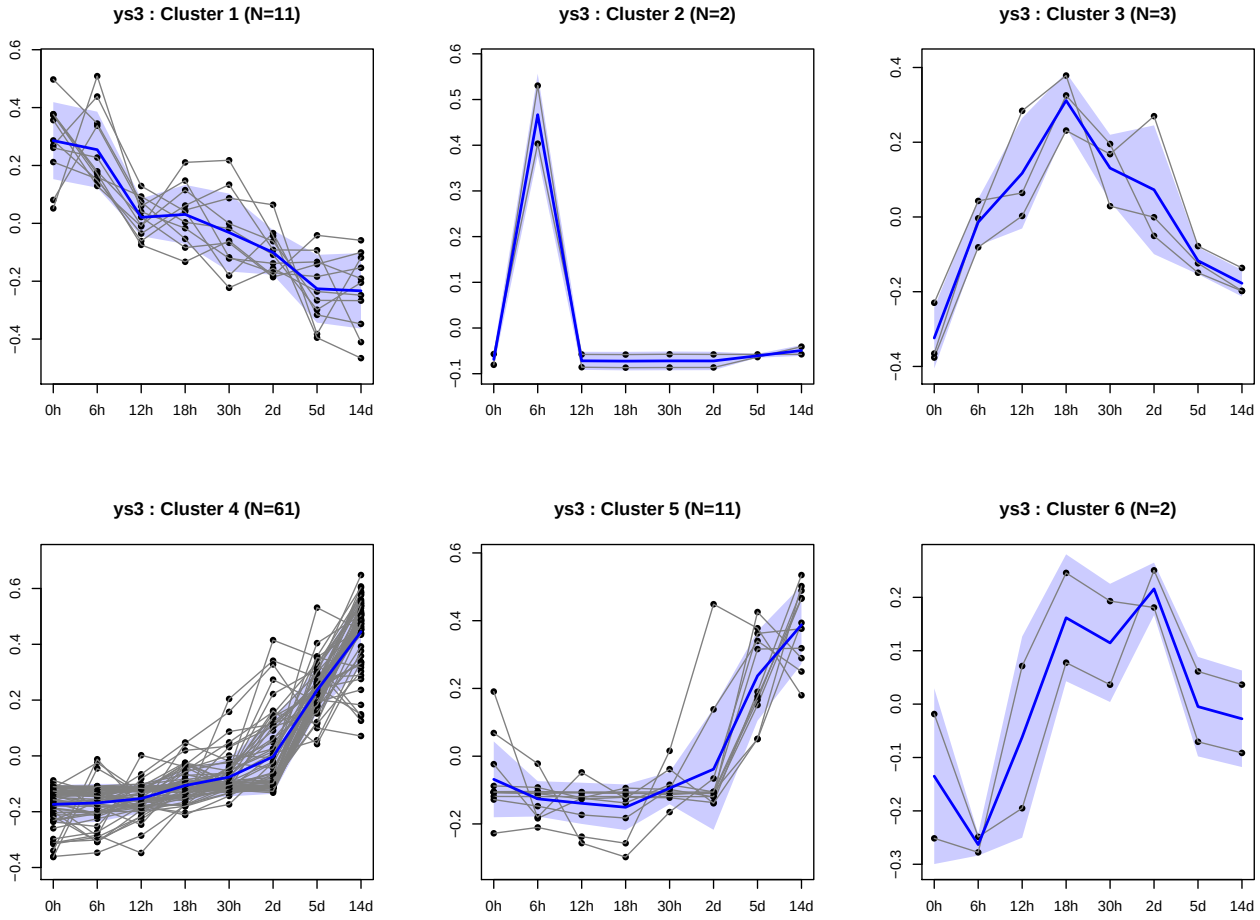


Figure Cluster mean: Plot of mean cluster data.

In addition the plots of the individual factors in the clusters are generated. Despite being clustered together still quit large variance exists between the different factors in the clusters.

```
# Plot individual time courses in cluster
plot_clusters_items <- function(method, toFile=FALSE){
  # get the cluster assignment for the given method
  hc.res <- hclust.res[[method]]
  groups.hc.order <- hc.res$groups.hc.order

  for (k in 1:Ngroups){
    if (toFile==TRUE){
      pdf(file.path(resultsPath, "cluster", sprintf("%s_cluster_%s.pdf", method, k)),
          width=10, height=10, pointsize=8)
    }

    g <- groups.hc.order[groups.hc.order==k]
    N <- ceiling(sqrt(length(g)))
    par(mfrow=c(N,N))
    for (name in names(g)){
      plot_single(name_A=name)
    }
    par(mfrow=c(1,1))
    if (toFile==TRUE){
```

```
      invisible(dev.off())
    }
  }
}

# plot individual factors in clusters on disk
for (method in correlation_methods){
  plot_clusters_items(method=method, toFile=TRUE)
}
rm(method)
```

Top cluster representatives

Here, the top cluster representatives are calculated for every cluster. The evaluation of all factors in a cluster is hereby based on the correlation between the factors and the mean cluster time course.

```
# Calculate and plot the top cluster representatives.
# Currently only ys3 based calculation is supported. This function should be refactored
# to support all correlation measures.
plot_top_cluster_representatives <- function(method="ys3", Ntop=11, labels=TRUE){
  if (!identical(method, "ys3")){
    stop("Only ys3 top correlation is currently supported")
  }
  # get the clusters
  hc.res <- hclust.res[[method]]
  groups.hc.order <- hc.res$groups.hc.order

  # calculate and plot the top correlations for every cluster
  par(mfrow=c(Ngroups, 1))
  hmap_colors <- HeatmapColors()
  for (k in 1:Ngroups){
      name_cluster <- sprintf("cluster.mean.%s", k)

      # get members of cluster
      g <- groups.hc.order[groups.hc.order==k]
      # get normalized data for members of cluster
      dgroup <- BDLdata.norm[names(g)]
      # calculate cluster mean and add as factor (mean of all factors in cluster for given time point a
      # (for S*)
      g.mean <- rowMeans(as.matrix(dgroup), na.rm=TRUE)
      dgroup[[name_cluster]] <- g.mean

      # calculate mean averaged over repeats (for A** and M*)
      dgroup.mean <- bdl_mean_data(dgroup, BDLsamples)

      # now calculate the ys3 correlation with the created data.frame containing the mean cluster facto
      ysr.res <- ysr.matrices(dgroup.mean, BDLmean.time, use="pairwise.complete.obs")
      # Spearman correlation on individual data points
      cor.S_star <- ( cor(dgroup, method="spearman", use="pairwise.complete.obs") + 1 )/2
      # YS3 in [-1,1]
      ys3 <- 2*( (w$w1*cor.S_star + w$w2*ysr.res$A_star2 + w$w3*ysr.res$M_star) - 0.5)

      # correlation for the cluster mean
```

```
      v <- ys3[, which(names(dgroup)==name_cluster)]
      # sort by absolute correlation
      v.sorted <- rev(v[order(abs(v))])

      # reduce to Ntop candidates and fill short clusters with zeros tto Ntop
      if (length(v.sorted)>=(Ntop+1)){
        v.sorted <- v.sorted[2:(Ntop+1)]
      } else {
        v.sorted <- c(v.sorted[2:length(v.sorted)], rep(0,Ntop-length(v.sorted)+1))
      }
      # prepare data for corrplot
      mv <- t(as.matrix(v.sorted))
      rownames(mv) <- c(name_cluster)
      if (labels == FALSE){
        colnames(mv) <- NULL
      }
      corrplot(mv, method="pie", type="full",
             tl.cex=1.0, tl.col="black", col=hmap_colors(100),
             insig="p-value", sig.level=-1, p.mat=mv,
             cl.pos="n")
  }
  par(mfrow=c(1,1))
}

# plot to file (with and without names)
pdf(file.path(resultsPath, "cluster", "cluster_top_representatives_01.pdf"),
    width=5, height=5, pointsize=12)
plot_top_cluster_representatives(labels=FALSE)
dev.off()
```

```
## pdf
##   2
```

```
pdf(file.path(resultsPath, "cluster", "cluster_top_representatives_02.pdf"),
    width=10, height=10, pointsize=12)
plot_top_cluster_representatives(labels=TRUE)
dev.off()
```

```
## pdf
##   2
```

```
# plot in report
plot_top_cluster_representatives(labels=TRUE)
```

Decision Trees

For the prediction of disease progression after BDL a decision tree was used. The regression tree was fitted with the R package `rpart`, being the open-source implementation of CART, implementing algorithms for recursive partitioning for classification following in most details closely Breiman et. al (1984) (Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) Classification and Regression Trees. Wadsworth).

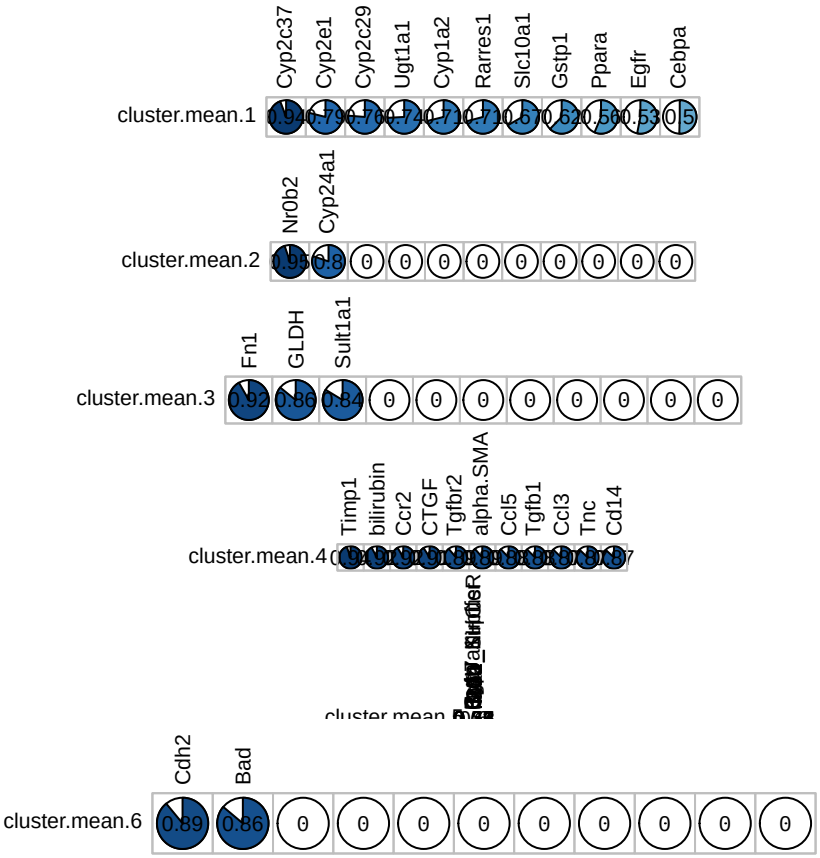

Cyp2c37
Cyp2e1
Cyp2c29
Ugt1a1
Cyp1a2
Rarres1
Slc10a1
Gstp1
Ppara
Egfr
Cebpa
cluster.mean.1
Nr0b2
Cyp24a1
cluster.mean.2
Fn1
GLDH
Sult1a1
cluster.mean.3
Timp1
bilirubin
Ccr2
CTGF
Tgfbr2
alpha.SMA
Ccl5
Tgfb1
Ccl3
Tnc
Cd14
cluster.mean.4
Cdh2
Bad
cluster.mean.6

Figure 4:

The predictor variables are the 6 mean time courses of the clusters, the dependent variable is the log transformed time class, to get aprroximately equidistant intervals between the trainings classes in the regression models. The tree is hereby built in a two-step process: First the single variable is found which best splits the data into two groups. The data than separated based on the split, and the splitting process is applied separately to each sub-group, and so on recursively until the subgroups either reach a minimum size (`minbucket`) or until no improvement can be made. The second stage of the procedure consists of using cross-validation to trim back the full tree. The splitting criterion, which is used to decide which variable gives the best split for nodes in the regression trees is $SS_T - (SS_L + SS_R)$, with $SS_T = \sum (y_i - < y >)^2$ the sum of squares for the node and SS_R and SS_L the sums of squares for the left and right son. This is equivalent to choosing the split ot maximize the between-groups sum-of-squares in a simple analysis of variance (see {Therneau2015}). Two important parameters controlling the resulting tree are

- `minsplit` : The minimum number of observations in a node for which the routine will even try to compute a split. The default is 20 and is set to 6 in the tree calculation ($N_r = 5$ repeats per time point).
- `minbucket` : The minimum number of observations in a terminal node. This defaults to minsplit/3.

```
suppressPackageStartupMessages(library(rpart))
suppressPackageStartupMessages(library(rpart.plot))
suppressPackageStartupMessages(library(caret))
dir.create(file.path(resultsPath, 'decision_tree'), showWarnings=FALSE)
```

To generate approximatly equally distant time classes for training the regression tree the log transformed time values are used. The transformation used is

$$\tilde{t_i} = log(t_i + 1)$$

```
# Transform data to log scale (for comparable time intervals)
log_transform <- function(data){
  log(data+1)
}
# Back transformation
log_transform_back <- function(log_data){
  exp(log_data)-1
}
# Resulting time transformations
time_transformation <- data.frame(time=BDLmean.time, log_time=log_transform(BDLmean.time))
print(round(time_transformation,2))
```

```
##   time log_time
## 1    0     0.00
## 2    6     1.95
## 3   12     2.56
## 4   18     2.94
## 5   30     3.43
## 6   48     3.89
## 7  120     4.80
## 8  336     5.82
```

Trainings data

In a first step the mean cluster trainings data for fitting the decision tree is prepared. The predictor variables are the mean samples of the clusters, the dependent variable is the log transformed time of the respective sample. The trainings set consists of the $N_t * N_r = 40$ samples.

```
  prepare_treedata_mean <- function(method="ys3"){
  # Hierarchical clusters based on ys3 to fit the regression tree
  hc.res <- hclust.res[[method]]
  groups <- hc.res$groups

  # Prepare training set for fitting the decision trees (mean cluster data set,
  # i.e. mean over normalized factors in cluster).
  na.vec <- rep(NA, Nt*Nr)
  treedata.mean <- data.frame(c1=na.vec, c2=na.vec, c3=na.vec, c4=na.vec, c5=na.vec, c6=na.vec)
  # for every sample
  for (ks in 1:(Nt*Nr)){
    # create the mean over the cluster
    for (kgroup in 1:Ngroups){
      # get factors in the cluster
      factors <- names(groups[groups==kgroup])
      # calculate mean over normalized data
      treedata.mean[ks, kgroup] <- mean(as.numeric(BDLdata.norm[ks, factors]), na.rm=TRUE)
    }
  }
  # add log transformed time [h] for regression
  treedata.mean$logtime <- log_transform(BDLsamples$time)
  # add experimental time class
  treedata.mean$class <- BDLsamples$time_fac

  return(treedata.mean)
}
treedata.mean <- prepare_treedata_mean()

# mean cluster data set for model fitting
print(round(treedata.mean[, -8], digits=2))
```

```
##       c1    c2    c3    c4    c5    c6 logtime
## 1   0.22 -0.07 -0.36 -0.14 -0.11 -0.03    0.00
## 2   0.50 -0.06 -0.29 -0.14 -0.07 -0.07    0.00
## 3   0.26 -0.07 -0.32 -0.19 -0.10 -0.17    0.00
## 4   0.34 -0.07 -0.31 -0.19 -0.04 -0.23    0.00
## 5   0.12 -0.07 -0.33 -0.19 -0.02 -0.18    0.00
## 6   0.31  0.69 -0.01 -0.17 -0.09 -0.29    1.95
## 7   0.21  0.93 -0.02 -0.18 -0.13 -0.36    1.95
## 8   0.24  0.27  0.03 -0.18 -0.15 -0.30    1.95
## 9   0.19  0.12  0.05 -0.15 -0.14 -0.25    1.95
## 10  0.33  0.33 -0.12 -0.14 -0.11 -0.12    1.95
## 11  0.01 -0.07  0.23 -0.13 -0.15 -0.01    2.56
## 12  0.08 -0.07  0.01 -0.15 -0.11 -0.07    2.56
## 13 -0.07 -0.07  0.12 -0.17 -0.15 -0.18    2.56
## 14  0.07 -0.07  0.09 -0.15 -0.14  0.03    2.56
## 15  0.01 -0.07  0.13 -0.15 -0.15 -0.08    2.56
## 16  0.13 -0.07  0.30 -0.05 -0.16  0.17    2.94
## 17 -0.03 -0.07  0.39 -0.12 -0.15  0.24    2.94
## 18  0.02 -0.07  0.22 -0.13 -0.15  0.07    2.94
## 19  0.02 -0.07  0.43 -0.07 -0.15  0.20    2.94
## 20  0.01 -0.07  0.22 -0.15 -0.14  0.12    2.94
## 21  0.05 -0.07 -0.01 -0.09 -0.04  0.14    3.43
```

```
## 22 -0.02 -0.07 -0.03 -0.07 -0.10  0.08    3.43
## 23 -0.11 -0.07  0.41 -0.11 -0.16  0.03    3.43
## 24 -0.11 -0.07  0.13 -0.03 -0.12  0.09    3.43
## 25  0.03 -0.07  0.17 -0.08 -0.05  0.23    3.43
## 26 -0.13 -0.07  0.01  0.02 -0.04  0.11    3.89
## 27 -0.17 -0.07  0.26  0.13  0.00  0.43    3.89
## 28 -0.09 -0.07 -0.15  0.00 -0.04  0.28    3.89
## 29 -0.08 -0.07  0.12 -0.11 -0.14  0.15    3.89
## 30 -0.03 -0.07  0.12 -0.08 -0.02  0.12    3.89
## 31 -0.23 -0.07 -0.27 -0.01  0.14 -0.09    4.80
## 32 -0.26 -0.07  0.11  0.27  0.17  0.27    4.80
## 33 -0.28 -0.07 -0.27  0.29  0.19 -0.13    4.80
## 34 -0.26 -0.07 -0.23  0.28  0.23 -0.13    4.80
## 35 -0.10 -0.01  0.07  0.33  0.44  0.06    4.80
## 36 -0.16  0.03 -0.17  0.45  0.44 -0.06    5.82
## 37 -0.22 -0.06 -0.14  0.64  0.49 -0.01    5.82
## 38 -0.23 -0.07 -0.13  0.23  0.26 -0.07    5.82
## 39 -0.24 -0.07 -0.17  0.54  0.44  0.12    5.82
## 40 -0.32 -0.07 -0.27  0.36  0.30 -0.11    5.82
```

```
# save the trainings data
save(treedata.mean, file=file.path(resultsPath, "data", "treedata.mean.Rdata"))
```

Fit regression tree

Fit the log transformed time against the mean cluster data for the time points. Set control parameters for the algorithm are `minsplit=6` and `minbucket=2` to account for the low number of samples (`N=40`).

```
# formula for regression tree
formula.reg = paste("logtime ~ c1 + c2 + c3 + c4 + c5 + c6")

# fit regression tree with mean cluster data
tree.reg <- rpart(formula=formula.reg,
                  data=treedata.mean,
                  method="anova",
                  control=rpart.control(minsplit=6, minbucket=2))

# overview resulting tree
print(tree.reg)
```

```
## n= 40
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 40 1.111047e+02 3.174622
##    2) c4< -0.1174939 18 2.310447e+01 1.743756
##      4) c3< -0.2081362 5 0.000000e+00 0.000000 *
##      5) c3>=-0.2081362 13 2.053563e+00 2.414432
##       10) c1>=0.1331057 5 2.465190e-31 1.945910 *
##       11) c1< 0.1331057 8 2.700232e-01 2.707258 *
##    3) c4>=-0.1174939 22 2.099513e+01 4.345331
```

```
##      6) c5< 0.07192204 12 1.384347e+00 3.543160 *
##      7) c5>=0.07192204 10 2.622937e+00 5.307937
##       14) c4< 0.3463259 6 8.743124e-01 4.966506 *
##       15) c4>=0.3463259 4 0.000000e+00 5.820083 *
```

```
summary(tree.reg)
```

```
## Call:
## rpart(formula = formula.reg, data = treedata.mean, method = "anova",
##     control = rpart.control(minsplit = 6, minbucket = 2))
##   n= 40
##
##           CP nsplit  rel error     xerror       xstd
## 1 0.60308067      0 1.00000000 1.05279859 0.20917846
## 2 0.18946910      1 0.39691933 0.51512113 0.07352301
## 3 0.15289945      2 0.20745023 0.37354418 0.05432524
## 4 0.01605279      3 0.05455078 0.11121777 0.02198016
## 5 0.01573853      4 0.03849799 0.10117463 0.01942152
## 6 0.01000000      5 0.02275946 0.08936793 0.02000611
##
## Variable importance
## c4 c5 c1 c3 c6 c2
## 27 19 19 15 12  8
##
## Node number 1: 40 observations,    complexity param=0.6030807
##   mean=3.174622, MSE=2.777617
##   left son=2 (18 obs) right son=3 (22 obs)
##   Primary splits:
##       c4 < -0.1174939   to the left,  improve=0.6030807, (0 missing)
##       c1 < -0.07537026  to the right, improve=0.5729499, (0 missing)
##       c5 < -0.008508656 to the left,  improve=0.5487412, (0 missing)
##       c3 < -0.2829117   to the left,  improve=0.5183388, (0 missing)
##       c6 < -0.149531    to the left,  improve=0.3184694, (0 missing)
##   Surrogate splits:
##       c1 < -0.07537026  to the right, agree=0.850, adj=0.667, (0 split)
##       c5 < -0.0625688   to the left,  agree=0.800, adj=0.556, (0 split)
##       c6 < -0.149531    to the left,  agree=0.750, adj=0.444, (0 split)
##       c2 < 0.07279386   to the right, agree=0.675, adj=0.278, (0 split)
##       c3 < -0.2829117   to the left,  agree=0.675, adj=0.278, (0 split)
##
## Node number 2: 18 observations,    complexity param=0.1894691
##   mean=1.743756, MSE=1.283582
##   left son=4 (5 obs) right son=5 (13 obs)
##   Primary splits:
##       c3 < -0.2081362   to the left,  improve=0.9111184, (0 missing)
##       c5 < -0.1122942   to the right, improve=0.5991729, (0 missing)
##       c1 < 0.09730379   to the right, improve=0.5785906, (0 missing)
##       c4 < -0.1838753   to the left,  improve=0.4737815, (0 missing)
##       c6 < -0.01908181  to the left,  improve=0.3296556, (0 missing)
##   Surrogate splits:
##       c4 < -0.1838753   to the left,  agree=0.889, adj=0.6, (0 split)
##       c5 < -0.1122942   to the right, agree=0.889, adj=0.6, (0 split)
##       c1 < 0.3351088    to the right, agree=0.833, adj=0.4, (0 split)
##
```

```
## Node number 3: 22 observations,    complexity param=0.1528994
##   mean=4.345331, MSE=0.9543241
##   left son=6 (12 obs) right son=7 (10 obs)
##   Primary splits:
##       c5 < 0.07192204   to the left,  improve=0.8091327, (0 missing)
##       c4 < 0.1840943    to the left,  improve=0.7540271, (0 missing)
##       c1 < -0.1437557   to the right, improve=0.6642991, (0 missing)
##       c3 < -0.08194432  to the right, improve=0.6127738, (0 missing)
##       c6 < 0.007982308  to the right, improve=0.5246106, (0 missing)
##   Surrogate splits:
##       c4 < 0.1840943    to the left,  agree=0.955, adj=0.9, (0 split)
##       c1 < -0.1936876   to the right, agree=0.909, adj=0.8, (0 split)
##       c3 < -0.08194432  to the right, agree=0.864, adj=0.7, (0 split)
##       c6 < 0.007982308  to the right, agree=0.864, adj=0.7, (0 split)
##       c2 < -0.07256249  to the right, agree=0.727, adj=0.4, (0 split)
##
## Node number 4: 5 observations
##   mean=0, MSE=0
##
## Node number 5: 13 observations,    complexity param=0.01605279
##   mean=2.414432, MSE=0.1579664
##   left son=10 (5 obs) right son=11 (8 obs)
##   Primary splits:
##       c1 < 0.1331057    to the right, improve=0.8685099, (0 missing)
##       c2 < 0.02364916   to the right, improve=0.8685099, (0 missing)
##       c3 < 0.06868251   to the left,  improve=0.7242745, (0 missing)
##       c6 < -0.09746977  to the left,  improve=0.7242745, (0 missing)
##       c5 < -0.1383548   to the right, improve=0.4701461, (0 missing)
##   Surrogate splits:
##       c2 < 0.02364916   to the right, agree=1.000, adj=1.0, (0 split)
##       c3 < 0.06868251   to the left,  agree=0.923, adj=0.8, (0 split)
##       c6 < -0.2167742   to the left,  agree=0.923, adj=0.8, (0 split)
##       c4 < -0.153395    to the left,  agree=0.846, adj=0.6, (0 split)
##       c5 < -0.1383548   to the right, agree=0.846, adj=0.6, (0 split)
##
## Node number 6: 12 observations
##   mean=3.54316, MSE=0.1153623
##
## Node number 7: 10 observations,    complexity param=0.01573853
##   mean=5.307937, MSE=0.2622937
##   left son=14 (6 obs) right son=15 (4 obs)
##   Primary splits:
##       c4 < 0.3463259    to the left,  improve=0.6666667, (0 missing)
##       c5 < 0.2491244    to the left,  improve=0.6666667, (0 missing)
##       c6 < -0.1217972   to the left,  improve=0.2500000, (0 missing)
##       c3 < -0.0328633   to the right, improve=0.2500000, (0 missing)
##       c1 < -0.247735    to the left,  improve=0.1666667, (0 missing)
##   Surrogate splits:
##       c5 < 0.2821139    to the left,  agree=0.9, adj=0.75, (0 split)
##       c1 < -0.2249667   to the left,  agree=0.7, adj=0.25, (0 split)
##       c2 < -0.06371928  to the left,  agree=0.7, adj=0.25, (0 split)
##       c3 < -0.1376999   to the right, agree=0.7, adj=0.25, (0 split)
##       c6 < -0.06444851  to the left,  agree=0.7, adj=0.25, (0 split)
##
```

```
## Node number 10: 5 observations
##   mean=1.94591, MSE=4.930381e-32
##
## Node number 11: 8 observations
##   mean=2.707258, MSE=0.0337529
##
## Node number 14: 6 observations
##   mean=4.966506, MSE=0.1457187
##
## Node number 15: 4 observations
##   mean=5.820083, MSE=0
```

```
printcp(tree.reg)
```

```
##
## Regression tree:
## rpart(formula = formula.reg, data = treedata.mean, method = "anova",
##     control = rpart.control(minsplit = 6, minbucket = 2))
##
## Variables actually used in tree construction:
## [1] c1 c3 c4 c5
##
## Root node error: 111.1/40 = 2.7776
##
## n= 40
##
##         CP nsplit rel error   xerror     xstd
## 1 0.603081      0  1.000000 1.052799 0.209178
## 2 0.189469      1  0.396919 0.515121 0.073523
## 3 0.152899      2  0.207450 0.373544 0.054325
## 4 0.016053      3  0.054551 0.111218 0.021980
## 5 0.015739      4  0.038498 0.101175 0.019422
## 6 0.010000      5  0.022759 0.089368 0.020006
```

```
tree.reg$frame
```

```
##       var  n wt          dev     yval  complexity ncompete nsurrogate
## 1      c4 40 40 1.111047e+02 3.174622 0.603080672        4          5
## 2      c3 18 18 2.310447e+01 1.743756 0.189469098        4          3
## 4  <leaf>  5  5 0.000000e+00 0.000000 0.010000000        0          0
## 5      c1 13 13 2.053563e+00 2.414432 0.016052788        4          5
## 10 <leaf>  5  5 2.465190e-31 1.945910 0.010000000        0          0
## 11 <leaf>  8  8 2.700232e-01 2.707258 0.010000000        0          0
## 3      c5 22 22 2.099513e+01 4.345331 0.152899447        4          5
## 6  <leaf> 12 12 1.384347e+00 3.543160 0.007176239        0          0
## 7      c4 10 10 2.622937e+00 5.307937 0.015738533        4          5
## 14 <leaf>  6  6 8.743124e-01 4.966506 0.010000000        0          0
## 15 <leaf>  4  4 0.000000e+00 5.820083 0.010000000        0          0
```

```
# pretty plot of tree to file
pdf(file.path(resultsPath, 'decision_tree', "regression_tree.pdf"),
    width=10, height=5, pointsize=12)
```

```
prp(tree.reg, type=0, extra=101, yesno=TRUE)
invisible(dev.off())

# and report
prp(tree.reg, type=0, extra=101, yesno=TRUE)
```

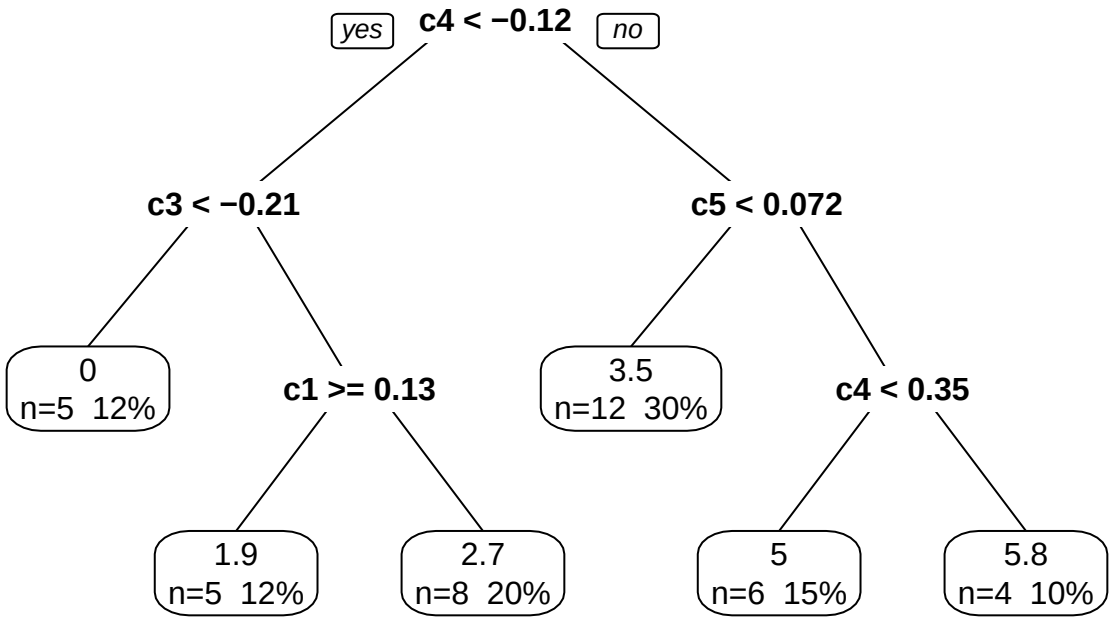
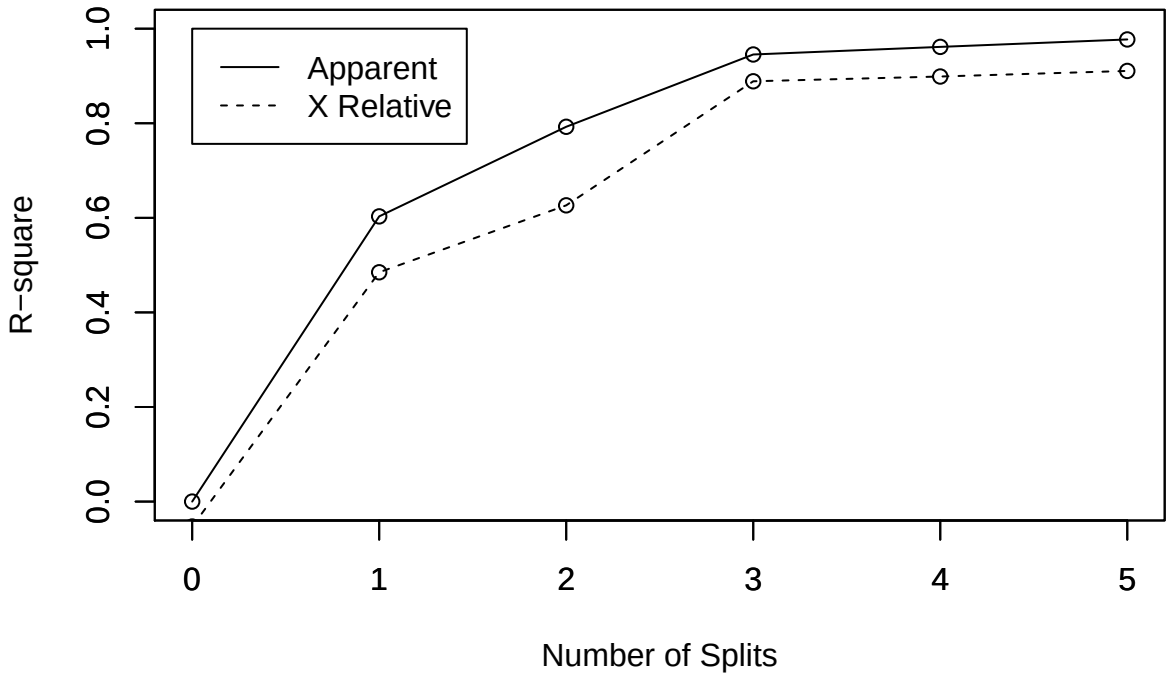
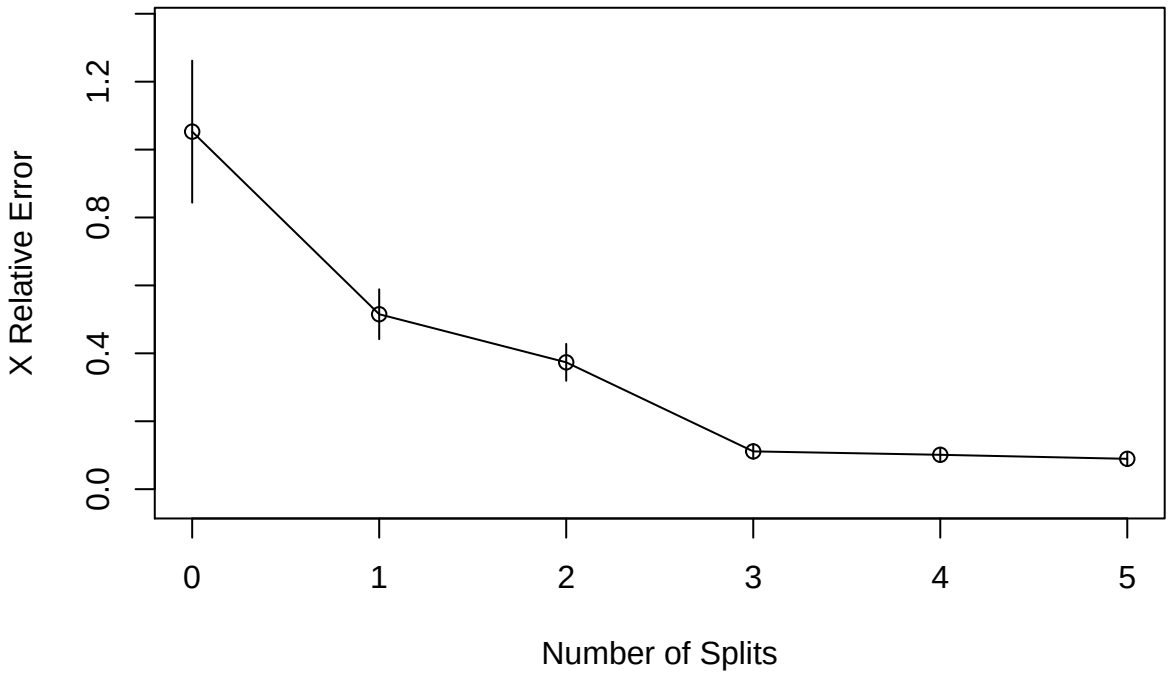


Figure 5:

```
# visualize cross-validation results
rsq.rpart(tree.reg)
```

```
##
## Regression tree:
## rpart(formula = formula.reg, data = treedata.mean, method = "anova",
##     control = rpart.control(minsplit = 6, minbucket = 2))
##
## Variables actually used in tree construction:
## [1] c1 c3 c4 c5
##
## Root node error: 111.1/40 = 2.7776
##
## n= 40
##
##         CP nsplit rel error   xerror     xstd
## 1 0.603081      0  1.000000 1.052799 0.209178
## 2 0.189469      1  0.396919 0.515121 0.073523
## 3 0.152899      2  0.207450 0.373544 0.054325
## 4 0.016053      3  0.054551 0.111218 0.021980
## 5 0.015739      4  0.038498 0.101175 0.019422
## 6 0.010000      5  0.022759 0.089368 0.020006
```






```
# variables used for splitting in the decision tree
tree.nodes <- (tree.reg$frame)$var
tree.nodes <- tree.nodes[tree.nodes != "<leaf>"]
tree.vars <- as.character(sort(unique(tree.nodes)))
rm(tree.nodes)
# variables used for splitting decisions in tree
print(tree.vars)
```

```
## [1] "c1" "c3" "c4" "c5"
```

Prediction on trainings data

Predict data with the tree, here the time class leaves based on the mean cluster data. The regression tree predicts log classes, which are back-transformed to time in [h]. So how good is the tree performing on the trainings data set, i.e. the mean cluster data. Predictons are evaluated based on the distance between the predicted and the experimental time classes based on the following distance measure on log scale

$$d = \frac{1}{N_s}\sqrt{\sum TODO}$$

```
# L2 (euclidian) distance measurement on the log transformed data. Analog to the distance measurement i
# the regression tree.
log_distance <- function(d1, d2){
  # sums over all the distances of the samples in log space
  log_rmsd <- sqrt(sum( (log_transform(d1)-log_transform(d2) )^2 ))/length(d1)
  return(log_rmsd)
}
```

Prediction on trainings data

```
# mean cluster predictions
pred.mean.log <- predict(tree.reg, newdata=treedata.mean, type="vector")
# transformation to time in [h]
pred.mean <- log_transform_back( pred.mean.log )

# TODO: create the overview matrix which experimental time classes are predicted in which
# tree time classes (on normal and log scale)
# TODO: heatmap of the matrix
# pred.matrix.mean <- matrix(NA, nrow=nrow(node_ranges), ncol=length(levels(as.factor(BDLsamples$time))

# Distance calculation (predicted to experimental)
dist.mean.all <- treedata.mean$logtime - pred.mean.log
dist.mean <- log_distance(pred.mean, BDLsamples$time)

# plot predicted ~ experimentell
plot_mean_prediction <- function(){
  par(mfrow=c(1,2))
  plot(BDLsamples$time, pred.mean, pch=15, col=rgb(0,0,1, 0.2), main="Regression Tree:\nPredicted ~ exp
       xlab="experimentell time [h]", ylab="predicted time [h]")
  abline(a=0, b=1, col=rgb(0.5,0.5,0.5,0.5))
  hist(dist.mean.all, breaks=seq(from=-1.05, to=1.05, by=0.1),
       main="Histogram prediction error:\nmean cluster data",
       col=rgb(0.5,0.5,0.5,0.5),
       xlab="logtime(exp)-logtime(pred)")
  par(mfrow=c(1,1))
}
# plot to file
pdf(file.path(resultsPath, 'decision_tree', "prediction_mean.pdf"),
    width=10, height=5, pointsize=12)
plot_mean_prediction()
invisible(dev.off())
# and report
plot_mean_prediction()
```

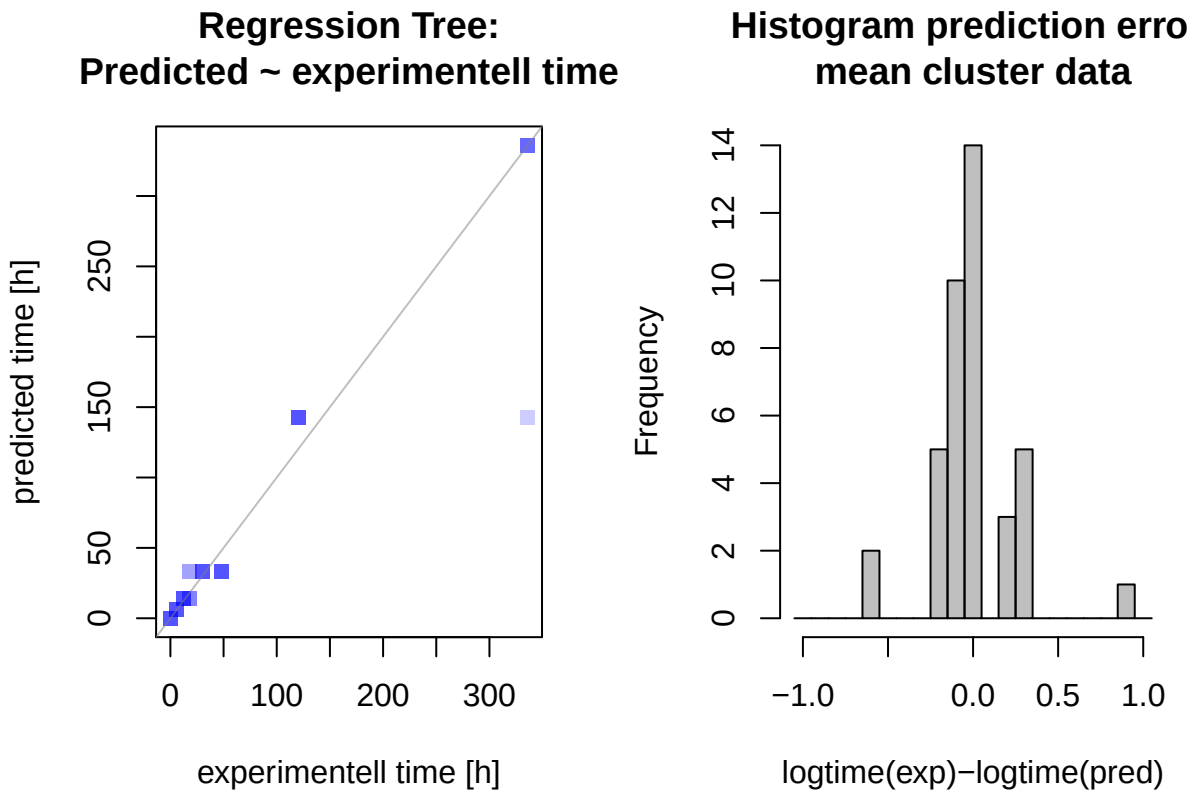


Figure 6:

The ranges of the predicted classes by the regression trees can be calculated based on the split points on log scale. These provides the information which time points would be classified in which class in the regression tree.

```
calculate_node_ranges <- function(){
  # classes via predicted classes for cluster data
  node_levels <- levels(as.factor(pred.mean))
  # These are the predicted classes
  node_classes <- round(as.numeric(node_levels), digits=1)

  # get the intervals of the time classes
  node_mean <- as.numeric(levels(as.factor(pred.mean.log)))
  node_midpoints <- (node_mean[2:length(node_mean)] + node_mean[1:(length(node_mean)-1)])/2
  # minimum of range
  node_min <- node_mean
  node_min[2:length(node_min)] <- node_midpoints
  # maximum of range
  node_max <- node_mean
  node_max[1:(length(node_min)-1)] <- node_midpoints
  # ranges in log scale
  node_ranges.log <- data.frame(node_mean, node_min, node_max)

  node_ranges <- data.frame(mean=log_transform_back(node_mean),
                            min=log_transform_back(node_min),
                            max=log_transform_back(node_max))
  rownames(node_ranges) <- paste("class", 1:nrow(node_ranges))
```

```
  return(node_ranges)
}

# predicted time classes by decision tree
node_ranges <- calculate_node_ranges()
print(round(node_ranges, digits=1))
```

```
##          mean   min   max
## class 1   0.0   0.0   1.6
## class 2   6.0   1.6   9.2
## class 3  14.0   9.2  21.8
## class 4  33.6  21.8  69.4
## class 5 142.5  69.4 218.9
## class 6 336.0 218.9 336.0
```

Test data for tree evaluation

Single factor per cluster

Prepare the evaluation data sets for the fitted trees consisting of all single factor combinations from the clusters used in the decision tree. Only the combinations are created which are from clusters used for decisions in the tree

```
# names of factors in the clusters
cluster_names <- paste('c', 1:Ngroups, sep="")
cluster.factors <- vector("list", length=Ngroups)
groups <- (hclust.res$ys3)$groups  # get the ys3 groups
for (k in 1:Ngroups){
  cluster.factors[[k]] <- as.character(names(groups[groups==k]))
}
names(cluster.factors) <- cluster_names

# create data.frame of all single combinations from clusters
single.combinations <- expand.grid(cluster.factors, stringsAsFactors=TRUE)
names(single.combinations) <- names(cluster.factors)
# number of single combinations
Nsingle <- nrow(single.combinations)
print(Nsingle)  # 88572 combinations
```

```
## [1] 88572
```

```
# create all single factor data
print("Calculating single factor data (~ 3min) ... ")
```

```
## [1] "Calculating single factor data (~ 3min) ... "
```

```
ptm <- proc.time()
treedata.single <- vector("list", Nsingle)  # list for all combinations
for (k in 1:Nsingle){
  # ------------------------------------------------------------------
```

```
  # THIS HAS TO BE FAST (<0.005 s)
  # ------------------------------------------------------------------
  # ptm <- proc.time() # Start the clock!

  # get factor data
  tmp <- BDLdata.norm[, t(single.combinations[k, ]) ]
  # add regression values
  tmp[c("class", "logtime")] <- treedata.mean[ c("class", "logtime")]

  # add factor fields
  tmp[, paste(cluster_names, '.id', sep="")] <- single.combinations[k,]
  colnames(tmp) <- c(cluster_names, 'class', 'logtime', paste(cluster_names, '.id', sep=""))

  # store data
  treedata.single[[k]] <- tmp

  # if (k%%500 == 0){print(k)}
  # print(proc.time()-ptm)  # Stop the clock
}
# Stop the clock
print(proc.time() - ptm)
```

```
##    user  system elapsed
## 119.615   0.080 119.625
```

```
print("... calculated.")
```

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

```
rm(tmp,k)
# treedata.single[[1]]

# which factor combinations only use genes
factor_is_gene <- BDLfactors$ftype %in% c("GE_ADME", "GE_Cytokines", "GE_Fibrosis")
names(factor_is_gene) <- BDLfactors$id

# vector for lookup if only genes were used
gene_only.single <- vector("logical", Nsingle)
for (k in 1:Nsingle){
  gene_only.single[k] <- all(factor_is_gene[t(single.combinations[k,])])
}
rm(k)
```

Double factor per cluster

Create a sample of double combinations from the various clusters.

```
print("Calculating double factor data (~ 1min) ... ")
```

```
## [1] "Calculating double factor data (~ 1min) ... "
```

```
ptm <- proc.time()
set.seed(123456)
Ndouble <- 10000
treedata.double <- vector("list", Ndouble)  # list for sampled double combinations
for (k in 1:Ndouble){
  # sample from the 4 clusters without replacement
  n1 = sample(cluster.factors[[1]], 2, replace=FALSE)
  n2 = sample(cluster.factors[[2]], 2, replace=FALSE)
  n3 = sample(cluster.factors[[3]], 2, replace=FALSE)
  n4 = sample(cluster.factors[[4]], 2, replace=FALSE)
  n5 = sample(cluster.factors[[5]], 2, replace=FALSE)
  n6 = sample(cluster.factors[[6]], 2, replace=FALSE)

  # The mean of the combination is used (handle NAs)
  tmp <- 0.5 * (  BDLdata.norm[, c(n1[1], n2[1], n3[1], n4[1], n5[1], n6[1])]
                + BDLdata.norm[, c(n1[2], n2[2], n3[2], n4[2], n5[2], n6[2])] )

   # add class and regression values
  tmp[c("class", "logtime")] <- treedata.mean[ c("class", "logtime")]

  # add factor fields
  tmp[ , paste(cluster_names, '.id', sep="")] <- data.frame(paste(n1, collapse="__"),
                                                paste(n2, collapse="__"),
                                                paste(n3, collapse="__"),
                                                paste(n4, collapse="__"),
                                                paste(n5, collapse="__"),
                                                paste(n6, collapse="__"))
  colnames(tmp) <- c(cluster_names, 'class', 'regvalue', paste(cluster_names, '.id', sep=""))
  # store data
  treedata.double[[k]] <- tmp
}
print(proc.time() - ptm)
```

```
##    user  system elapsed
##  37.282   0.068  37.460
```

```
print("... calculated.")
```

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

```
rm(k, tmp)
# treedata.double[[5]]
```

Predicting test data

Single representative predictions

Time class prediction with regression tree for single representative from each cluster

```
print("Predicting single factor data (~ 2min) ... ")
```

```
## [1] "Predicting single factor data (~ 2min) ... "
```

```
pred.single.all <- vector("list", length(treedata.single))
for (k in (1:length(treedata.single))){
  # prediction and back transformation
  pred.single.all[[k]] <- log_transform_back( predict(tree.reg, newdata=treedata.single[[k]], method="a
}
pred.single <- do.call("rbind", pred.single.all)

# distance for all predictions on single factor per cluster
dist.single <- rep(NA, Nsingle)
for (k in 1:Nsingle){
  dist.single[k] <- log_distance(pred.single[k,], BDLsamples$time)
}
```

Random double representative predictions

Time class prediction with regression tree for random selection of two representatives from each clusters

```
print("Predicting double factor data (~ 1min) ... ")
```

```
## [1] "Predicting double factor data (~ 1min) ... "
```

```
pred.double.all <- vector("list", length(treedata.double))
for (k in (1:length(treedata.double))){
  pred.double.all[[k]] <- log_transform_back( predict(tree.reg, newdata=treedata.double[[k]], method="a
}
pred.double <- do.call("rbind", pred.double.all)

# distance for predictions on 2 sampled factors per cluster
dist.double <- rep(NA, Ndouble)
for (k in 1:Ndouble){
  dist.double[k] <- log_distance(pred.double[k,], BDLsamples$time)
}
```

Best factor combinations for predicting time classes

Finding the best regression trees based on i) all single representatives from the clusters; and ii) single representatives only consisting of gene probes. The best is defined by minimal euclidian distance between experimental classes and predicted classes on the log scale. The best tree is not refitted with the respective factors in the tree, but the mean cluster tree uses the respective factor data for prediction.

```
# Best decision tree using all factors
# Find unique combinations with minimal distance
dist.rep.best <- min(dist.single)
# best combination of representatives for the clusters (remove duplicates)
rep.best <- unique(single.combinations[which(dist.single==dist.rep.best), c("c1", "c3", "c4", "c5")])
rep.best.idx <- rownames(rep.best)[1]
```

```r
# predictions of best representative
pred.rep.best <- pred.single[as.numeric(rep.best.idx), ]

print("Best single representatives for decision tree:")
```

```
## [1] "Best single representatives for decision tree:"
```

```r
print(rep.best)
```

```
##           c1   c3     c4    c5
## 16062 Cyp1a2  Fn1 S100A4 Il17a
## 16084 Cyp1a2 GLDH S100A4 Il17a
```

```r
print(dist.rep.best)
```

```
## [1] 0.05554854
```

```r
# -------------------------
# Best decision tree using only gene factors, i.e. first reduce to the gene combinations
dist.single.genes <- dist.single[gene_only.single]
single.combinations.genes <- single.combinations[gene_only.single,]
dist.gene.best <- min(dist.single.genes)
# find best gene combination
gene.best <- unique(single.combinations.genes[which(dist.single.genes==dist.gene.best), c("c1", "c3", "
# predictions with best representative
gene.best.idx <- rownames(gene.best)
pred.gene.best <- pred.single[as.numeric(gene.best.idx), ]
pred.gene.best
```

```
##         1         2         3         4         5         6         7
##   0.00000   0.00000   0.00000   0.00000   0.00000   6.00000   6.00000
##         8         9        10        11        12        13        14
##   6.00000   6.00000   6.00000  13.98812   6.00000  13.98812  13.98812
##        15        16        17        18        19        20        21
##  13.98812  13.98812  13.98812  13.98812  13.98812  13.98812  13.98812
##        22        23        24        25        26        27        28
##  33.57599  13.98812  33.57599  13.98812  33.57599  33.57599  33.57599
##        29        30        31        32        33        34        35
##  33.57599  33.57599  33.57599 336.00000 142.52453 142.52453 142.52453
##        36        37        38        39        40
## 142.52453 336.00000 142.52453 336.00000 336.00000
```

```r
print("Best single representatives based on genes for decision tree:")
```

```
## [1] "Best single representatives based on genes for decision tree:"
```

```r
print(gene.best)
```

```
##           c1  c3     c4    c5
## 14808 Cyp1a2 Fn1 Col1a1 Il17a
```

```
print(dist.gene.best)
```

```
## [1] 0.06672908
```

```
# -------------------------
# Plot time courses of the representatives/factors in provided tree combinations
plot_tree_representatives <- function(combination){
  Nc <- ncol(combination)
  Nr <- nrow(combination)
  par(mfrow=c(Nr, Nc))
  for (kr in 1:Nr){
    for (kc in 1:Nc){
      cluster <- colnames(combination)[kc]
      name <- as.character(combination[kr, kc])
      plot_single(name)
    }
  }
  par(mfrow=c(1,1))
}

# plot to file
pdf(file.path(resultsPath, "decision_tree", "rep.best.representatives.pdf"),
    width=10, height=6, pointsize=12)
plot_tree_representatives(rep.best)
invisible(dev.off())
pdf(file.path(resultsPath, "decision_tree", "gene.best.representatives.pdf"),
    width=10, height=3, pointsize=12)
plot_tree_representatives(gene.best)
invisible(dev.off())

# plot to report
plot_tree_representatives(rep.best)
```

```
plot_tree_representatives(gene.best)
```

Distributions of prediction distance

Plot the distance distributions of single and double representatives and the best gene and representative trees.

```
# distance histogram
plot_tree_distance_dist <- function(){
  breaks <- seq(from=0, to=0.40, by=0.0125)
  hist(dist.single, freq=FALSE, breaks=breaks,
      main="Distance between predicted and experimentell time",
      xlab="RMSD(time.predicted, time.exp)",
      col=rgb(0.7,0.7,0.7, 1),
      ylim=c(0,15)
  )
  hist(dist.double, freq=FALSE, breaks=breaks,
       col=rgb(1,0,0, 0.5), add=TRUE)
```

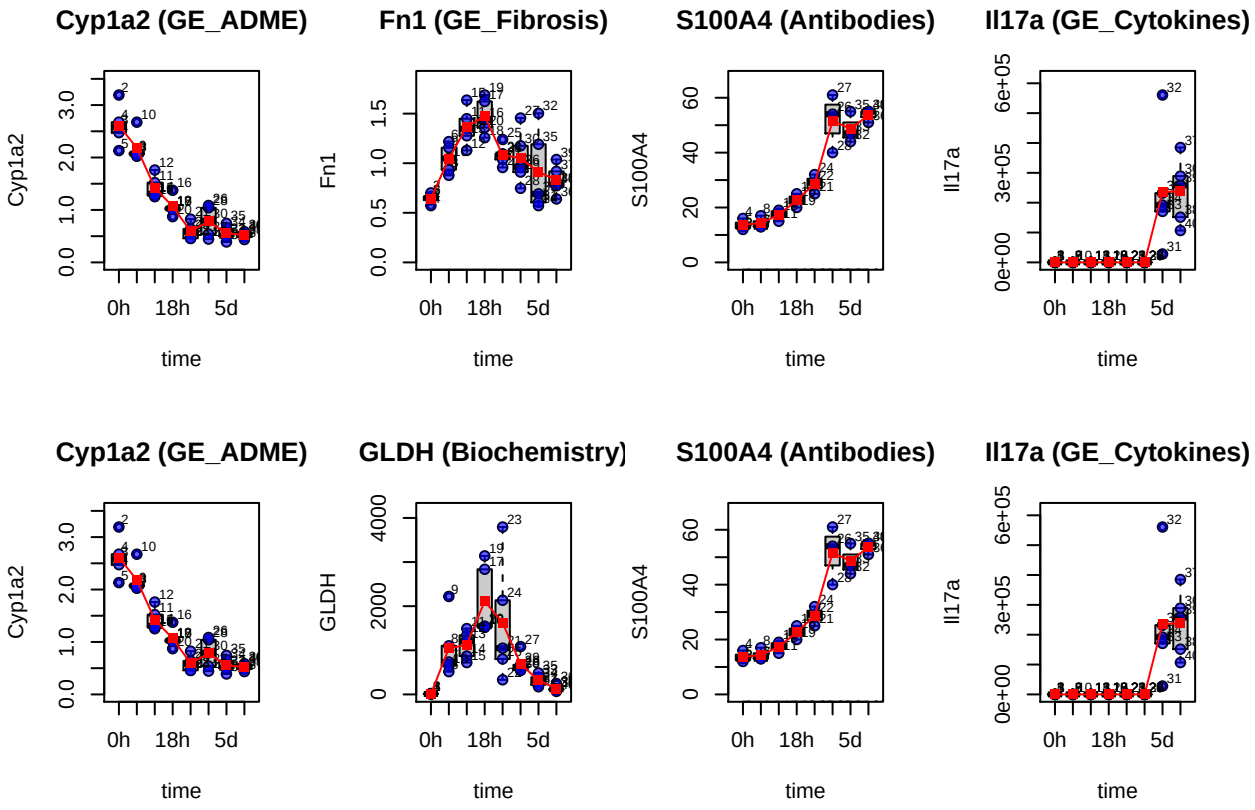


Figure 7:

```
  abline(v=dist.mean, col=rgb(0,0,1, 0.5), lwd=2)
  abline(v=dist.rep.best, col="black", lwd=2)
  abline(v=dist.gene.best, col="grey", lwd=2)

  legend("topright", legend=c("single factor", "double factor", "mean cluster"),
            col=c(rgb(0.7, 0.7, 0.7, 1),rgb(1, 0, 0, 0.5),rgb(0, 0, 1, 0.5)),
            bty="n", cex=1.0, pch=15)
}
# plot to file
pdf(file.path(resultsPath, "decision_tree", "tree_distance_distribution.pdf"),
    width=10, height=6, pointsize=14)
plot_tree_distance_dist()
invisible(dev.off())

# plot in report
plot_tree_distance_dist()
```

Evaluation of the predictions. Which experimental classes were predicted in which time classes of the regression tree for the mean cluster data (trainings data), single representative from each clusters and double representatives from each cluster.

```
# Predicted classes of the regression tree
node_levels <- levels(as.factor(pred.mean))
node_classes <- round(as.numeric(node_levels), digits=1)
```

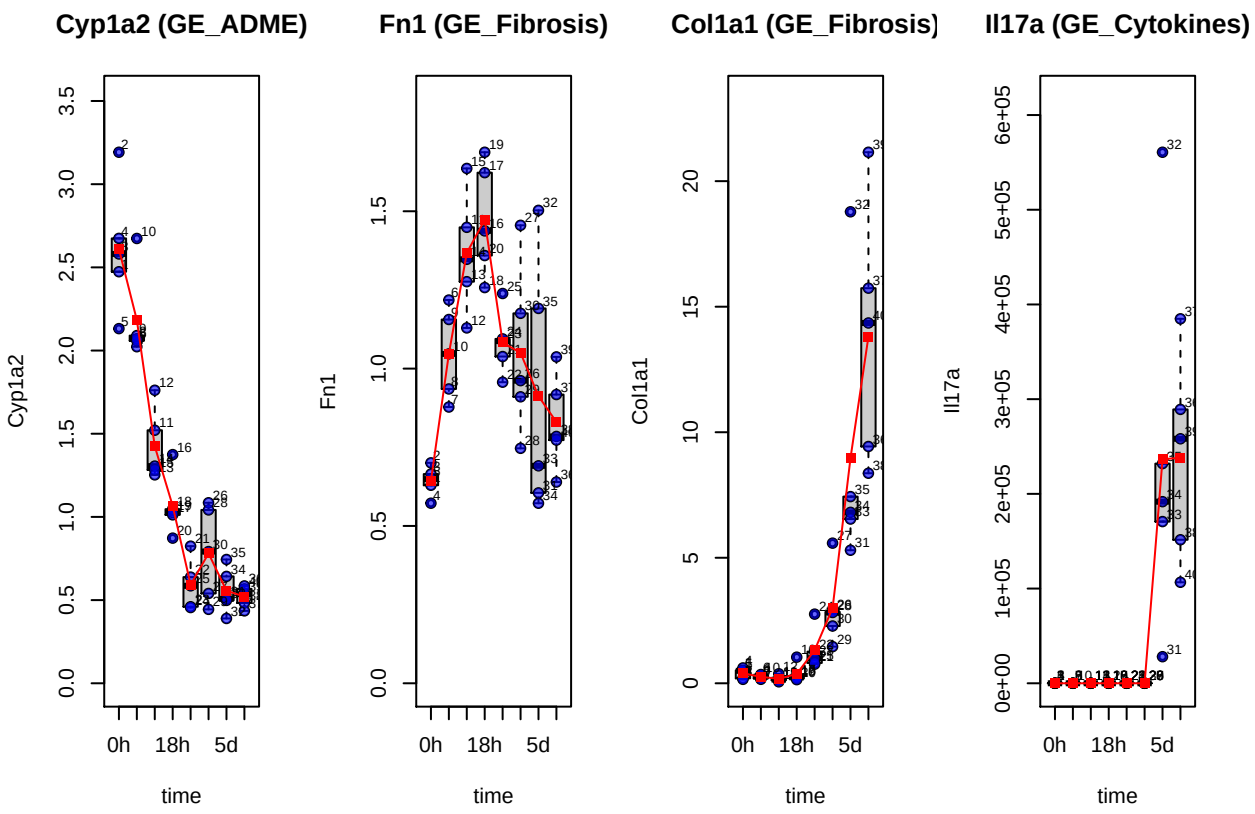


Figure 8:

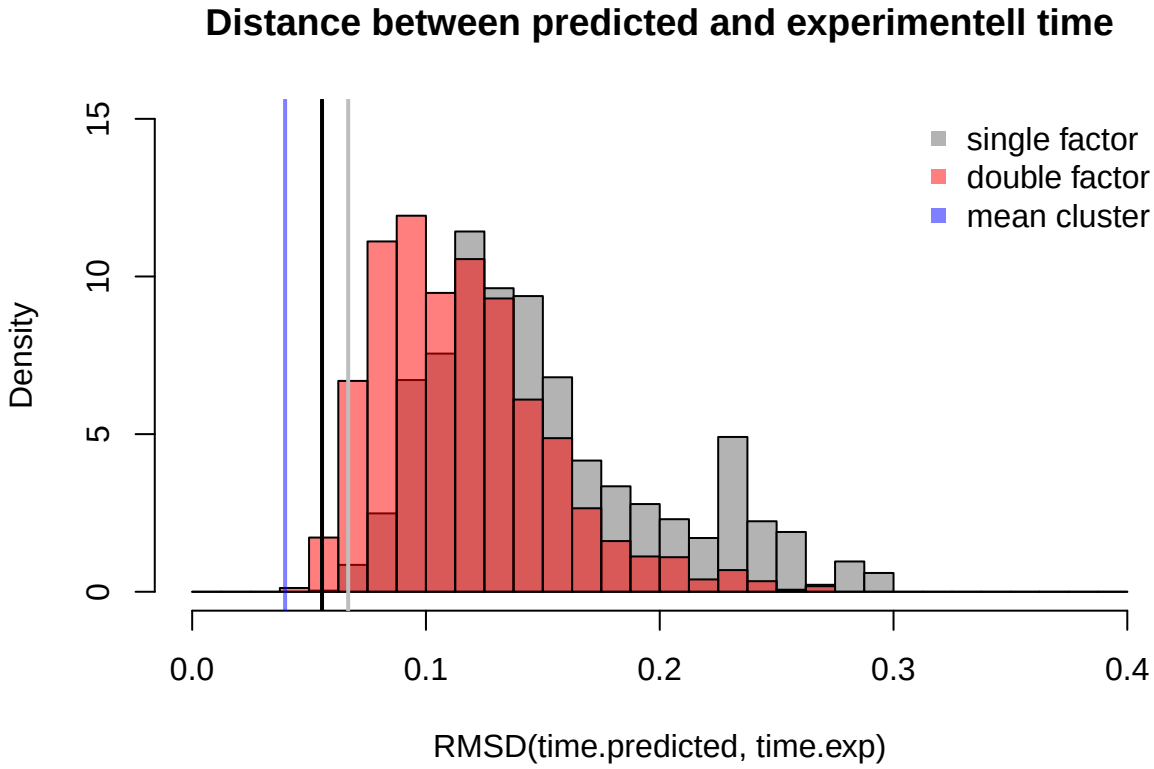


Figure 9:

```
node_classes
```

```
## [1]   0.0   6.0  14.0  33.6 142.5 336.0
```

```
# Plot of the predicted classes with the decision tree
plot_predicted_classes <- function(){
  bar_colors <- brewer.pal(4, "Set3")
  par(mfrow=c(2,4))
  for (k in 1:Nt){
    # single factor predictions
    data <- as.vector(pred.single[, ((1:Nr)+Nr*(k-1))])
    tab.single <- table(factor(data, levels=node_levels))/length(data)

    # two factor predictions
    data <- as.vector(pred.double[, ((1:Nr)+Nr*(k-1))])
    tab.double <- table(factor(data, levels=node_levels))/length(data)

    # mean cluster predictions
    data <- as.vector(pred.mean[((1:Nr)+Nr*(k-1))])
    tab.mean <- table(factor(data, levels=node_levels))/length(data)

    # best single representative
    # data <- as.vector(pred.rep.best[((1:Nr)+Nr*(k-1))])
    # tab.rep.best <- table(factor(data, levels=node_levels))/length(data)

    # best single gene representative
    data <- as.vector(pred.gene.best[((1:Nr)+Nr*(k-1))])
    tab.gene.best <- table(factor(data, levels=node_levels))/length(data)

    # combined table
    tab <- rbind(tab.single, tab.double, tab.gene.best, tab.mean)
    colnames(tab) <- round(as.numeric(colnames(tab)), digits=1)

    # create the bar plot
    name <- sprintf("Time after BDL: %sh", levels(as.factor(BDLsamples$time))[k])
    barplot(tab, beside=TRUE,
            main=name,
            xlab="predicted time class [h]", ylab="fraction of predictions",
            ylim=c(0,1), col=bar_colors)
    if (k==1){
      legend("topright", legend=c("single factors", "double factors",
                                  "best single gene", "mean cluster"),
             col=bar_colors,
             bty="n", cex=1.0, pch=15)
    }
  }
  par(mfrow=c(1,1))
}

# barplot of the predicted classes
plot_predicted_classes()
```

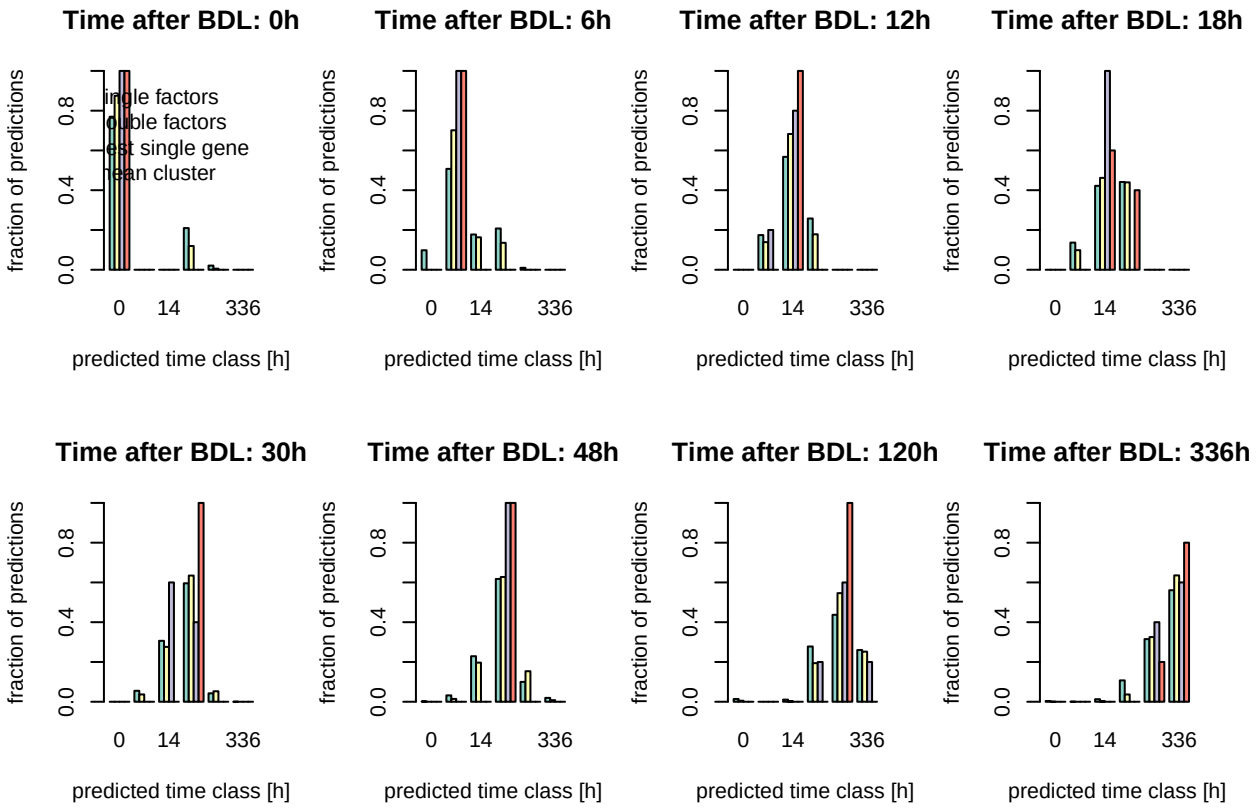


Figure 10:

```
# barplot to file
pdf(file.path(resultsPath, "decision_tree", "predicted_classes.pdf"),
    width=14, height=8, pointsize=14)
plot_predicted_classes()
invisible(dev.off())
```