Reproduce paper plots with scPower

Matthias Heinig, Katharina Schmid

28 December, 2021

Contents

This vignette shows how the plots of the paper "Design and power analysis for multi-sample single cell genomics experiments" can be reproduced using scPower. For an introduction into the package with detailed explanations, please have a look at the second vignette "introduction-scPower". Due to runtime reasons, not the complete analysis is performed in this vignette and instead sometimes precalculated values provided from the package. However, for all steps it is explained how the precalculated values are generated. For more information on the modelling and all citations, please have a look into our publication.

We imported additionally the same color schemes and plotting packages (ggplot2 and ggpubr) to be able to create exactly the same plots as in the publication.

```
library(scPower)
library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 3.5.2
library(data.table)
#> Warning: package 'data.table' was built under R version 3.5.2
library(viridis)
```

```
#> Loading required package: viridisLite
#>
#> Attaching package: 'viridis'
#> The following object is masked from 'package:viridisLite':
#>
#>     viridis.map
library(ggpubr)
library(RColorBrewer)
library(gridExtra)
library(dplyr)
#> Warning: package 'dplyr' was built under R version 3.5.2
#>
#> Attaching package: 'dplyr'
#> The following object is masked from 'package:gridExtra':
#>
#>     combine
#> The following objects are masked from 'package:data.table':
#>
#>     between, first, last
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
```

```
#Color specifications
col.paired <- brewer.pal(n = 12, "Paired")
col.set <- col.paired[c(2,8,9)]
col.set2<- RColorBrewer::brewer.pal(n = 5, name = "Set1")[-1]
col.set3 <- brewer.pal(n = 11, "PRGn")[c(4,6,8)]
col.set3 <- c(col.set[3], "#F7F7F7", col.set[1])
col.set4 <- c(col.set[1], col.set[3])
col.set5<-col.paired[seq(2,12,2)]
col.dark2<-brewer.pal(n=7,"Dark2")
col.power<-c("#FF9900", "#9900CC","#FF5050")

#Large set with 10 colors for optimization plots
gg_color_hue <- function(n) {
  hues = seq(15, 375, length = n + 1)
  hcl(h = hues, l = 65, c = 100)[1:n]
}
col.set.new<- gg_color_hue(10)
#Order them so that similar colors are not close
col.set.new<-col.set.new[c(1,6,3,8,4,9,2,7,5,10)]
method.colors<-col.paired[c(1:4,7:9,6)]
```

Paper Plot

Figure 1: Dependence of experimental design parameters

Figure 1 is a schematic overview figure showing the dependence of experimental design parameters.

Figure 2: Expression probability model parameterized by UMI counts per cell

Figure 2a is a schematic overview figure visualizing the expression model

Figure 2b-c shows how the fit of our expression probabilty model is applied on two example data sets, our own data set and one of Kang et al, 2018. The model is fitted using our own PBMC data set. The fits are provided in the scPower package in the data frames “disp.fun.param” and “gamma.mixed.fits”. Also the used UMI-read fit was generated using our own PBMC data set, saved in the data frame “read.umi.fit” as “10X_PBMC_1”. The fitting itself is not shown here due to runtime limitations, but the fits were exactly produced as described in the introduction-scPower" vignette in section “Expression probability curves” (using the same functions in the same order). The only difference is that in the introduction vignette it is only shown for one cell type, we performed the fit separately for all celltypes with at least 50 cells.

The observed expression curves were calculated as described in section “Counting observed expressed genes”, the percalculated values are saved due to time reasons in the data frame “observed.gene.counts”.

```r
#Load observed gene counts (precalculated)
data(precalculatedObservedGeneCounts) #observed.gene.counts

#Selected run
run<-"Run 5"

estimates.allSamples<-NULL

################
# Fit for own data and count > 10
estimates<-observed.gene.counts[
  observed.gene.counts$evaluation=="own_count10" &
    observed.gene.counts$run==run,]

estimates$estimated.counts<-NA
for(i in 1:nrow(estimates)){
  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=14,
    nCellsCt=estimates$num.cells[i]/14,
    meanCellCounts=estimates$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=10,
    perc.indiv=0.5)

  estimates$estimated.counts[i]<-round(sum(exp.probs))

}
estimates$dataset<-"Training data set"
estimates$threshold<-"Training data set - Count > 10"
estimates.allSamples<-rbind(estimates.allSamples,estimates)

################
# Fit for own data and count > 0
estimates<-observed.gene.counts[
  observed.gene.counts$evaluation=="own_count0" &
    observed.gene.counts$run==run,]
```

```r
estimates$estimated.counts<-NA
for(i in 1:nrow(estimates)){
  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=14,
    nCellsCt=estimates$num.cells[i]/14,
    meanCellCounts=estimates$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=0,
    perc.indiv=0.5)

  estimates$estimated.counts[i]<-round(sum(exp.probs))

}
estimates$dataset<-"Training data set"
estimates$threshold<-"Training data set - Count > 0"
estimates.allSamples<-rbind(estimates.allSamples,estimates)

#Set batch variable
estimates.allSamples$run<-NULL
subsampled.names<-setNames(c("50000","37500","25000","12500"),
                           c("complete","subsampled75",
                             "subsampled50","subsampled25"))
estimates.allSamples$sample<-subsampled.names[estimates.allSamples$sample]
estimates.allSamples$sample<-paste0(estimates.allSamples$sample," - PBMC1")

###############
# Fit for Kang data and count > 10
estimates<-observed.gene.counts[
  observed.gene.counts$evaluation=="Kang_count10",]

personsPerRunKang<-list("A"=4,"B"=4,"C"=8)

estimates$estimated.counts<-NA
for(i in 1:nrow(estimates)){
  sampleSize<-personsPerRunKang[[estimates$run[i]]]

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates$num.cells[i]/sampleSize,
    meanCellCounts=estimates$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=10,
    perc.indiv=0.5)

  estimates$estimated.counts[i]<-round(sum(exp.probs))

}

estimates$dataset<-"Test data set"
```

```r
estimates$threshold<-"Test data set - Count > 10"
estimates$sample<-paste0(estimates$run," - PBMC2")
estimates$run<-NULL
estimates.allSamples<-rbind(estimates.allSamples,estimates)

###############
# Fit for Kang data and count > 10
estimates<-observed.gene.counts[
  observed.gene.counts$evaluation=="Kang_count0",]

personsPerRunKang<-list("A"=4,"B"=4,"C"=8)

estimates$estimated.counts<-NA
for(i in 1:nrow(estimates)){
  sampleSize<-personsPerRunKang[[estimates$run[i]]]

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates$num.cells[i]/sampleSize,
    meanCellCounts=estimates$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=0,
    perc.indiv=0.5)

  estimates$estimated.counts[i]<-round(sum(exp.probs))

}

estimates$dataset<-"Test data set"
estimates$threshold<-"Test data set - Count > 0"
estimates$sample<-paste0(estimates$run," - PBMC2")
estimates$run<-NULL
estimates.allSamples<-rbind(estimates.allSamples,estimates)

###############
# Get the correlation values
text_vals<-estimates.allSamples%>%
  group_by(threshold)%>%
  summarize(corr=cor(expressed.genes,estimated.counts)^2)
text_vals$corr<-paste0("r^2==",round(text_vals$corr,3))

###############
# Reformat data for plotting
estimates.allSamples$meanUMI<-NULL
estimates.allSamples$evaluation<-NULL

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("sample","cell.type","num.cells",
                                         "threshold","dataset"))

named_colvector<-c("12500 - PBMC1"=col.dark2[1],"25000 - PBMC1"=col.dark2[2],
                   "37500 - PBMC1"=col.dark2[3],"50000 - PBMC1"=col.dark2[4],
                   "A - PBMC2"=col.dark2[5],"B - PBMC2"=col.dark2[6],
                   "C - PBMC2"=col.dark2[7])

plot.estimates_train <- plot.estimates %>%
  dplyr::filter(dataset=="Training data set")
plot.estimates_test <- plot.estimates %>%
  dplyr::filter(dataset=="Test data set")

# New plot
gg_train <- ggplot(plot.estimates_train,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(shape=cell.type,color=sample,fill=sample),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_fill_manual(values=named_colvector)+
  scale_color_manual("Read depth",values=named_colvector)+
  scale_linetype_discrete(name="Curve",labels=c("Expressed","Estimated")) +
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  facet_wrap(~threshold,ncol=2,scales="free")+
  geom_text(data=text_vals[3:4,],aes(label=corr,x=Inf,y=-Inf,
                                     hjust=1.2, vjust=-0.8), parse = TRUE)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position = "bottom",
        legend.title=element_text(size=8),
        legend.text=element_text(size=8),
        aspect.ratio = 0.8)+
  guides(color = guide_legend(override.aes = list(size = 0.5), order=1, nrow=4),
         linetype=guide_legend(nrow=2,byrow=TRUE),
         shape=guide_legend(nrow=3,byrow=TRUE),
         fill=FALSE)

gg_test <- ggplot(plot.estimates_test,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(shape=cell.type,color=sample,fill=sample),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_fill_manual(values=named_colvector)+
  scale_color_manual("Batch",values=named_colvector)+
  scale_linetype_discrete(name="Curve",labels=c("Expressed","Estimated")) +
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  facet_wrap(~threshold,ncol=2,scales="free")+
  geom_text(data=text_vals[1:2,],aes(label=corr,x=Inf,y=-Inf,
                                     hjust=1.2, vjust=-0.8), parse=TRUE)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position = "bottom",
        legend.title=element_text(size=8),
        legend.text=element_text(size=8),
        aspect.ratio = 0.8)+
  guides(color = guide_legend(override.aes = list(size = 0.5), order=1, nrow=3),
         linetype=FALSE,
         shape=FALSE,
         fill=FALSE)

jointlegend<-gtable_cbind(get_legend(gg_train),get_legend(gg_test))

g<-ggarrange(gg_train,gg_test,ncol=2,
             labels=c("b","c"),
             common.legend = TRUE, legend="bottom", legend.grob = jointlegend)
print(g)
```

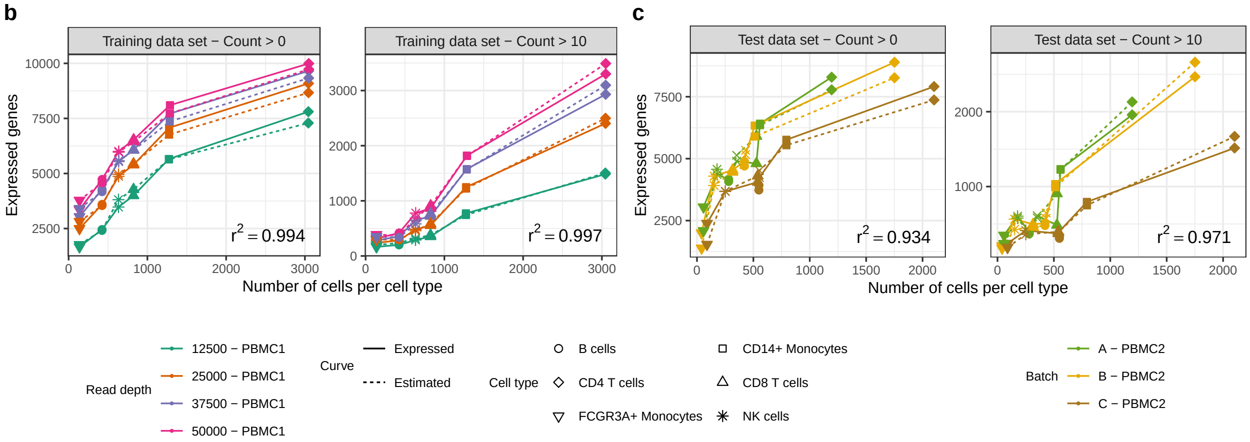


Figure 2: Expression probability model parameterized by UMI counts per cell.

a The expression probabilities for genes in pseudobulk of a newly planned experiment are estimated based on the expression prior and the planned experimental parameters. For this, the expression prior is derived from the mean and dispersion parameters of gene-wise negative binomial distributions fitted from a matching pilot data set.

b Using this approach, the number of expressed genes expected under our model (dashed line) closely matches the observed number of expressed genes (solid line) dependent on the number of cells per cell type (cell type indicated by point symbol) for one batch of the training PBMC data set. The data is subsampled to different read depths (indicated by color). The r2 values between estimated and expressed genes were highly significant for both expression thresholds.

c The model performed similarly well for the three batches of an independent validation PBMC data set.

Used expression threshold: count $>$ 10 (right panels of b,c) or count $>$ 0 (left panels of b,c) in more than 50% of the individuals.

Figure 3: Expression probability, DE/eQTL power and overall detection power and their validation in simulation studies.

The Figure 3a,b shows the influence of the experimental parameters on the detection power. The same expression probability model as in Figure 2 is used, the priors for the power, i.e. the effect sizes and expression ranks, are taken from different reference studies (see publication for citation and more explanation). The experimental parameters are set to realistic values. FDR adjustment is set as multiple testing correction for DE and FWER adjustment for eQTLs. In Figure 3c-f, the performance is validated against different simulation-based methods. Due to runtime reasons, all simulations (for powsimR, muscat and our own eQTL study) were already precalculated for the vignette here (for the DE studies for a smaller range shown than in the publication).

```
################################################################################
# Part a,b (DE and eQTL power for different parameter combis)

#General parameters
transcriptome.mapped.reads<-20000
nGenes<-21000
ct<-"CD4 T cells"
ct.freq<-1
cellsPerLane<-20000

########
# DE subplot
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(4,6,8,10,16,20)
ref.study.name<-"Blueprint (CLL) iCLL-mCLL"

#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize)
colnames(param.combis)<-c("cellsPerCelltype","sampleSize")

#Test results for each parameter combination
res<-lapply(1:nrow(param.combis),function(i)
  power.general.restrictedDoublets(param.combis$sampleSize[i],
                             param.combis$cellsPerCelltype[i],
                             transcriptome.mapped.reads,
                             ct.freq, "de",de.ref.study,ref.study.name,
                             cellsPerLane,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits,ct,
                             disp.fun.param,mappingEfficiency = 1,
                             multipletRate=0,multipletFactor=1,
                             min.UMI.counts = 10,
                             perc.indiv.expr = 0.5,
                             nGenes=21000,
                             sign.threshold = 0.05,
                             MTmethod = "FDR"))
#> Loading required package: MKmisc
#> Warning: package 'MKmisc' was built under R version 3.5.2

power.DE<-rbindlist(res)

#Plot DE results
power.DE.study<-reshape2::melt(power.DE,
                               id.vars=c("name","sampleSize","totalCells",
                                         "usableCells","multipletFraction",
                                         "ctCells","readDepth",
                                         "readDepthSinglet","mappedReadDepth",
                                         "expressedGenes"))
power.DE.study$variable<-factor(power.DE.study$variable,
                                levels=c("exp.probs","power","powerDetect"))
```

```r
var.labs<-c("Expression probability", "DE power",
            "Detection power")
names(var.labs)<-c("exp.probs","power","powerDetect")

power.DE.study$ctCells<-as.factor(power.DE.study$ctCells)
power.DE.study$sampleSize<-as.factor(power.DE.study$sampleSize)
g.DE<-ggplot(power.DE.study,aes(x=ctCells,y=sampleSize,fill=value))+
  geom_tile()+
  facet_wrap(~variable,ncol=3,labeller = labeller(variable=var.labs))+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Total sample size")+
  scale_fill_viridis("Probability",limits=c(0,1))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

########
# eQTL subplot
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(50,100,150,200)
ref.study.name<-"Blueprint (Tcells)"

#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize)
colnames(param.combis)<-c("cellsPerCelltype","sampleSize")

#Test results for each parameter combination
res<-lapply(1:nrow(param.combis),function(i)
  power.general.restrictedDoublets(param.combis$sampleSize[i],
                             param.combis$cellsPerCelltype[i],
                             transcriptome.mapped.reads,
                             ct.freq, "eqtl",eqtl.ref.study,ref.study.name,
                             cellsPerLane,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits,ct,
                             disp.fun.param,mappingEfficiency = 1,
                             multipletRate=0,multipletFactor=1,
                             min.UMI.counts = 10,
                             perc.indiv.expr = 0.5,
                             nGenes=21000,
                             sign.threshold = 0.05,
                             MTmethod = "Bonferroni"))
#> Loading required package: pwr
#> Warning: package 'pwr' was built under R version 3.5.2

power.eqtl<-rbindlist(res)

#Plot eQTL results
power.eqtl.study<-reshape2::melt(power.eqtl,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                            "ctCells","readDepth",
                                            "readDepthSinglet","mappedReadDepth",
                                            "expressedGenes"))
power.eqtl.study$variable<-factor(power.eqtl.study$variable,
                                  levels=c("exp.probs","power","powerDetect"))

var.labs<-c("Expression probability", "eQTL power",
            "Detection power")
names(var.labs)<-c("exp.probs","power","powerDetect")

power.eqtl.study$ctCells<-as.factor(power.eqtl.study$ctCells)
power.eqtl.study$sampleSize<-as.factor(power.eqtl.study$sampleSize)
g.eqtl<-ggplot(power.eqtl.study,aes(x=ctCells,y=sampleSize,fill=value))+
  geom_tile()+
  facet_wrap(~variable,ncol=3,labeller = labeller(variable=var.labs))+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Total sample size")+
  scale_fill_viridis("Probability",limits=c(0,1))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

g.ab<-ggarrange(g.DE,g.eqtl,ncol=1,nrow=2,labels=c("a","b"),
          align="hv",common.legend=TRUE,legend="right")

################################################################################
# Part c-e (Comparison with powsimR and edgeR)

#Rerun power calculation as different cutoffs and additional filtering is necessary!
cellsPerCelltype<-c(200,1000,3000)
sampleSize<-c(4,8,16)
ref.study.name<-"Blueprint (CLL) iCLL-mCLL"

#Filter ref study for the ones that are kept after gene filtering
ref.study<-de.ref.study[de.ref.study$name==ref.study.name,]
fitted.genes<-16197 #Hard-coded here, found in powsimR object

#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize)
colnames(param.combis)<-c("cellsPerCelltype","sampleSize")

power.DE<-NULL
for(i in 1:nrow(param.combis)){
  detailed<-power.general.restrictedDoublets(param.combis$sampleSize[i],
                             param.combis$cellsPerCelltype[i],
                             transcriptome.mapped.reads,
                             ct.freq, "de",de.ref.study,ref.study.name,
                             cellsPerLane,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits,ct,
                             disp.fun.param,mappingEfficiency = 1,
```

```r
                             multipletRate=0,multipletFactor=1,
                             min.UMI.counts = 0,
                             perc.indiv.expr = 0,
                             nGenes=21000,
                             returnResultsDetailed = TRUE,
                             MTmethod="FDR")

  temp.overview<-detailed$overview.df

  #Filter power for the expressed genes (for comparison with simulation methods)
  ref.study$not.filtered<-ref.study$rank<=min(fitted.genes,
                                              detailed$overview.df$expressedGenes)
  temp.overview$power.filtered<-mean(detailed$probs.df$power[ref.study$not.filtered])

  #Filter overall power for simulated genes (for comparison with simulation methods)
  ref.study$not.filtered<-ref.study$rank<=fitted.genes
  temp.overview$overall.power.filtered<-mean(detailed$probs.df$combined.prob[
    ref.study$not.filtered])

  power.DE<-rbind(power.DE,temp.overview)
}

#FDR corrected thresold (get only edgeR values)
power.fdr<-scPower::simulated.DE.values[simulated.DE.values$MT=="FDR",]
power.fdr<-power.fdr[startsWith(power.fdr$method,"edgeR"),]

power.fdr<-merge(power.fdr,power.DE[,c("sampleSize","totalCells","expressedGenes",
                                       "power.filtered","overall.power.filtered")],
           by=c("sampleSize","totalCells"))
power.fdr$variable<-ifelse(endsWith(power.fdr$method,"powsimR"),
                             "powsimR","muscat")
#Change variable ordering in plots
power.fdr$variable<-factor(power.fdr$variable,levels=c("powsimR","muscat"))

g.c<-ggplot(power.fdr,aes(x=expressed.genes,y=expressedGenes,
                           color=variable))+
  geom_point(alpha=0.7)+geom_abline()+geom_hline(yintercept = 16171)+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  xlab("Expressed genes - simulations")+
  ylab("Expressed genes - scPower")+
  theme_bw()

g.d<-ggplot(power.fdr,aes(x=power,y=power.filtered,
                           color=variable))+
  geom_point(alpha=0.7)+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  xlab("DE power - simulations")+
  ylab("DE power - scPower")+
  theme_bw()

g.e<-ggplot(power.fdr,aes(x=overall.power,y=overall.power.filtered,
                           color=variable))+
  geom_point(alpha=0.7)+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  xlab("Overall power - simulations")+
  ylab("Overall power - scPower")+
  theme_bw()
```

```
#Example code how the eQTL simulation is done (not run here due to runtime reasons)

power.eqtl.sim<-NULL

#Set specifications
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(50,100,150,200)
ref.study.name<-"Blueprint (Tcells)"

repetition.rounds<-25

combis<-expand.grid(sampleSize,cellsPerPerson,
                    KEEP.OUT.ATTRS = FALSE)
colnames(combis)<-c("sampleSize","cellsPerPerson")

for(c in 1:nrow(combis)){

  nSamples<-combis$sampleSize[c]
  nCells<-combis$cellsPerPerson[c]

  print(paste("Simulating parameter combination with",nSamples,
              "samples",nCells,"cells"))

  #Run expression probability part of scPower
  power.study<-power.general.restrictedDoublets(nSamples,
                                                nCells,
                                                transcriptome.mapped.reads,
                                                ct.freq=1,
                                                type="eqtl",
                                                scPower::eqtl.ref.study,
                                                study.name,
                                                20000,
                                                scPower::read.umi.fit[read.umi.fit$type==
                                                                        "10X_PBMC_1",],
                                                gamma.mixed.fits,
                                                ct,
                                                disp.fun.param,
                                                mappingEfficiency = 1,
                                                multipletRate=0,
                                                min.UMI.counts = 10,
                                                perc.indiv=0.5,
                                                nGenes=nGenes,
                                                useSimulatedPower = TRUE,
                                                returnResultsDetailed=TRUE)

  per.gene.estimates<-power.study$probs.df

  #Use the Bonferroni corrected threshold
  sign.threshold<-0.05/(10*power.study$overview.df$expressedGenes)
  #Start simulation
  for(j in 1:repetition.rounds){

    foundDEgenes<-sapply(1:nrow(per.gene.estimates), function(i)
        suppressWarnings(scPower:::power.eqtl.simulated(round(per.gene.estimates$mean.sum[i]),
                           per.gene.estimates$Rsq[i],
                           sign.threshold,
                           nSamples,
                           rep.times=1)))

    #Save results
    power.eqtl.sim<-rbind(power.eqtl.sim,
                          data.frame(nSamples,
                           nCells,
                           sim.round=j,
                           sim.power=mean(foundDEgenes))
  }
}

#Calculate mean and sd deviation over all simulation runs
power.eqtl.sim<-power.eqtl.sim%>%
  group_by(SampleSize,CellsPerson)%>%
  summarize(sim_power_mean=mean(Sim.Power),sim_power_sd=sd(Sim.Power))
```

```
################################################################################
# Part f (commparison with eQTL simulation)

#Load precalculated values for eQTL power simulation (generated with code above)
power.eqtl.sim<-scPower::simulated.eQTL.values
power.fdr<-power.fdr[startsWith(power.fdr$method,"edgeR"),]

power.eqtl.sim<-merge(power.eqtl.sim,power.eqtl[,c("sampleSize","totalCells","power")],
           by.x=c("SampleSize","CellsPerson"),
           by.y=c("sampleSize","totalCells"))

g.f<-ggplot(power.eqtl.sim,aes(sim_power_mean,power))+
  geom_point()+geom_abline()+xlim(0,1)+ylim(0,1)+
  xlab("eQTL power - simulation")+ylab("eQTL power - scPower")+
  theme_bw()

################################################################################
# Part g,h (Runtime and memory requirement)

#Create a table for the runtime (as a plot would not work visualizing the differences)
df<-data.frame(method=c("powsimR","muscat","scPower"),
               approach=c("Simulation","Simulation","Analytic"),
               runtime.num=c(192.06,75.34,0.14/60),
               runtime.text=c("192 h","75 h","< 1 min"),
               memory.num=c(48.03,34.75,1.12/1000),
               memory.text=c("48 GB","35 GB","1 MB"))
df$method<-factor(df$method,levels=c("powsimR","muscat","scPower"))

#Barplot with runtime
```

```r
g.g<-ggplot(df,aes(x=method,y=runtime.num,fill=method))+
  geom_bar(stat="identity") +
  geom_text(aes(x=method,y=runtime.num+10,label = runtime.text))+
  xlab("Method")+ylab("Runtime (h)")+
  scale_fill_manual(values=c("#F8766D","#00BFC4","#7CAE00"))+
  theme_bw()+
  theme(legend.position = "none")


#Barplot with memory
g.h<-ggplot(df,aes(x=method,y=memory.num,fill=method))+
  geom_bar(stat="identity") +
  geom_text(aes(x=method,y=memory.num+2,label = memory.text))+
  xlab("Method")+ylab("Memory (GB)")+
  scale_fill_manual(values=c("#F8766D","#00BFC4","#7CAE00"))+
  theme_bw()+
  theme(legend.position = "none")

g.cde<-ggarrange(g.c,g.d,g.e,ncol=3,nrow=1,labels=c("c","d","e"),
                 common.legend = TRUE,
                 legend="bottom")
g.fgh<-ggarrange(g.f,g.g,g.h,ncol=3,nrow=1,labels=c("f","g","h"),
                 common.legend = FALSE, align="hv")

g<-ggarrange(g.ab,g.cde,g.fgh,ncol=1,nrow=3, heights=c(1,0.7,0.6), align="hv")

print(g)
```

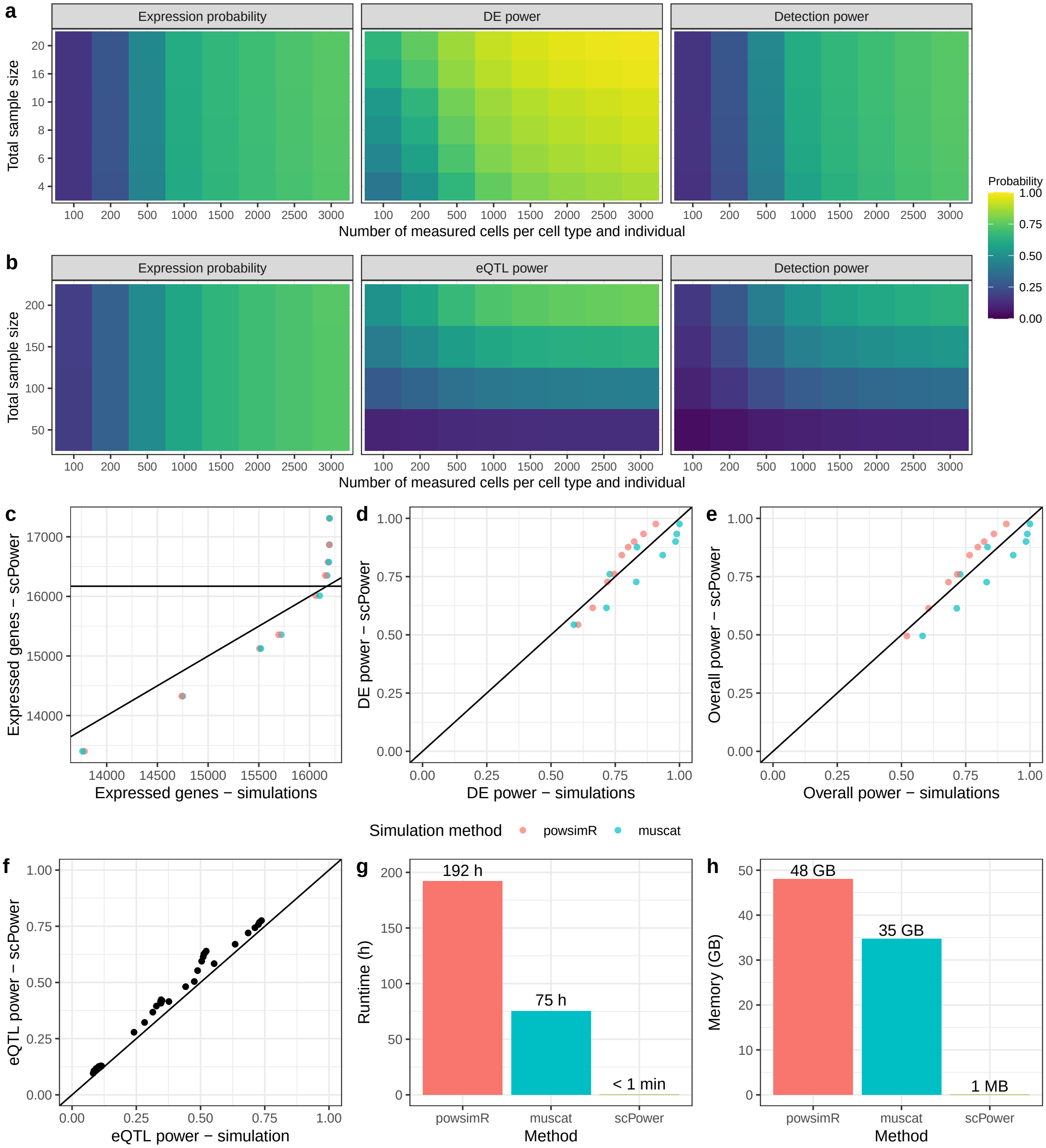


Figure 3: Expression probability, DE/eQTL power and overall detection power and their validation in simulation studies.

a,b Power estimation using data driven priors for DE genes (a) and eQTL genes (b) dependent on the total sample size and the number of measured cells per cell type. The detection power is the product of the expression probability and the power to detect the genes as DE or eQTL genes, respectively. The fold change for DEGs and the R2 for eQTL genes were taken from published studies, together with the expression rank of the genes. For (a), the Blueprint CLL study with comparison iCLL vs mCLL was used, for (b), the Blueprint T cell study. The expression profile and expression probabilities in a single cell experiment with a specific number of samples and measured cells was estimated using our expression prior, setting the definition for expressed to > 10 counts in more than 50% of the individuals. Multiple testing correction was performed by using FDR adjusted p-values for DE power and FWER adjusted p-values for eQTL power.

c-e The probabilities calculated in (a) were verified by the simulation-based methods powsimR and muscat with each point representing one parameter combination.

f The eQTL power of (b) could be replicated with a self-implemented simulation.

g,h Runtime (g) and memory requirements (h) were drastically higher in the simulations than for our tool scPower during the evaluations of (c-e), showing the strength of our analytic model.

Figure 4: Parameter optimization for constant budget.

Figure 4 shows two example calculations on how scPower selects the optimal experimental parameter combination for a specific budget, once for a DE study and once for an eQTL study. For an explanation of all the used parameters, please look into the introduction vignette.

In contrast to the original publication, the parameter useSimulatedPower is set to FALSE and the parameter speedPowerCalc to TRUE for the eQTL budget optimization. Both parameters speed the calculation in the vignette, while changing the overall power only slightly. We show in the publication the more accurate version including simulated eQTL power for lowly expressed genes. The difference between vignette and publication is especially visible in the eQTL power lines of Figure 4e,f, which have lower values here, as the power for genes with low expression values is set to 0 with speedPowerCalc=TRUE.

```
costKit<-5600
costFlowCell<-14032
readsPerFlowcell<-4100*10^6
cellsPerLane<-20000
ct<-"CD4 T cells"
ct.freq<-0.25
mappingEfficiency<-0.8

########
# DE calculation

comp.type<-"de"
ref.study.name<-"Blueprint (CLL) iCLL-mCLL"
readDepth<-c(2000,5000,10000,15000,20000,30000,50000,70000)
cellsPerIndividual<-c(100,200,500,700,seq(1000,12000,500))
totalBudget<-10000

power.study.de<-optimize.constant.budget.restrictedDoublets(totalBudget,
                                      comp.type,ct,ct.freq,
                                      costKit,costFlowCell,readsPerFlowcell,
                                      de.ref.study,ref.study.name,
                                      cellsPerLane,
                                      read.umi.fit[
                                        read.umi.fit$type=="10X_PBMC_1",],
                                      gamma.mixed.fits,
                                      disp.fun.param,
                                      nCellsRange=cellsPerIndividual,
                                      readDepthRange=readDepth,
                                      mappingEfficiency=mappingEfficiency,
                                      sign.threshold = 0.05,
                                      MTmethod="FDR")
```

```
########
# eQTL subplot

comp.type<-"eqtl"
ref.study.name<-"Blueprint (Tcells)"
readDepth<-c(2000,5000,10000,15000,20000,30000,50000)
cellsPerIndividual<-c(100,200,500,700,seq(1000,9000,500))
totalBudget<-30000

power.study.eqtl<-optimize.constant.budget.restrictedDoublets(totalBudget,
                                      comp.type,ct,ct.freq,
                                      costKit,costFlowCell,readsPerFlowcell,
                                      eqtl.ref.study,ref.study.name,
                                      cellsPerLane,
                                      read.umi.fit[
                                        read.umi.fit$type=="10X_PBMC_1",],
                                      gamma.mixed.fits,
                                      disp.fun.param,
                                      nCellsRange=cellsPerIndividual,
                                      readDepthRange=readDepth,
                                      mappingEfficiency=mappingEfficiency,
                                      sign.threshold = 0.05,
                                      MTmethod="Bonferroni",
                                      #These two parameters are set differently
                                      #in the publictions, but changed here to
                                      #speed the calculation
                                      useSimulatedPower = FALSE, #(in publication: TRUE)
                                      speedPowerCalc = TRUE) #(in publication: FALSE)



################################################################################
# Create grid plot

power.study.plot<-power.study.de
power.study.plot$totalCells<-as.factor(power.study.plot$totalCells)
power.study.plot$readDepth<-as.factor(power.study.plot$readDepth)

g.de<-ggplot(power.study.plot,aes(x=readDepth,y=totalCells,
                                  color=powerDetect,size=log10(sampleSize)))+
  geom_point()+ylab("Cells per individual")+xlab("Read depth")+
  ggtitle("DE study")+
  scale_color_viridis("Detection\npower",limits=c(0,1))+
  scale_size_continuous("Sample size",breaks=c(3,2,1),labels=c(1000,100,10))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8),
        plot.title = element_text(hjust = 0.5))

power.study.plot<-power.study.eqtl
power.study.plot$totalCells<-as.factor(power.study.plot$totalCells)
power.study.plot$readDepth<-as.factor(power.study.plot$readDepth)

g.eqtl<-ggplot(power.study.plot,aes(x=readDepth,y=totalCells,
                                    color=powerDetect,size=log(sampleSize)))+
  geom_point()+ylab("Cells per individual")+xlab("Read depth")+
  ggtitle("eQTL study")+
  scale_color_viridis("Detection\npower",limits=c(0,1))+
  scale_size_continuous("Sample size",breaks=c(3,2,1),labels=c(1000,100,10))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8),
        plot.title = element_text(hjust = 0.5))

g.a<-ggarrange(g.de,g.eqtl,ncol=2,nrow=1, labels=c("a","b"),
             common.legend = TRUE, legend="bottom")

################################################################################
# Create corresponding line plots

#Print maximal combination of values
max.study<-power.study.de[which.max(power.study.de$powerDetect),]

colnames(power.study.de)[2:4]<-c("Detection power","Expression probability",
                                 "DE power")

#Show two-dimensional plots with always one parameter fixed
#Fixed read depths
power.study.plot<-power.study.de[power.study.de$readDepth==max.study$readDepth,]
power.study.plot<-reshape2::melt(power.study.plot,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth","readDepthSinglet",
                                           "mappedReadDepth","expressedGenes"))
power.study.plot$variable<-factor(power.study.plot$variable,
                                  levels=c("Expression probability","DE power",
                                           "Detection power"))

g.rd<-ggplot(power.study.plot,aes(totalCells,value,color=variable))+geom_line()
labels<-as.numeric(ggplot_build(g.rd)$layout$panel_params[[1]]$x.labels)

g.rd<-g.rd +
  scale_x_continuous("Cells per individual",sec.axis=sec_axis(~., breaks=labels,
                     labels=sapply(labels, function(c) floor(
                     sampleSizeBudgetCalculation.restrictedDoublets(c,
                     max.study$readDepth, totalBudget, costKit, cellsPerLane,
                     costFlowCell,readsPerFlowcell))),
                     name="Sample size"))+
  geom_vline(xintercept = max.study$totalCells)+
  ylab("Probability")+
  scale_color_manual("",values=col.power)+
  ylim(0,1)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position=c(0.85,0.3),
        legend.title=element_blank(),
        legend.text=element_text(size=8))

#Fixed number of cells
power.study.plot<-power.study.de[power.study.de$totalCells==max.study$totalCells,]
power.study.plot<-reshape2::melt(power.study.plot,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth","readDepthSinglet",
                                           "mappedReadDepth","expressedGenes"))
power.study.plot$variable<-factor(power.study.plot$variable,
                                  levels=c("Expression probability","DE power",
                                           "Detection power"))

g.cp<-ggplot(power.study.plot,aes(readDepth,value,color=variable))+geom_line()
labels<-as.numeric(ggplot_build(g.cp)$layout$panel_params[[1]]$x.labels)

g.cp <- g.cp + scale_x_continuous("Read depth",sec.axis=sec_axis(~., breaks=labels,
                                  labels=sapply(labels,function(c) floor(
                                  sampleSizeBudgetCalculation.restrictedDoublets(
                                  max.study$totalCells, c,totalBudget, costKit,
                                  cellsPerLane, costFlowCell,readsPerFlowcell))),
  geom_vline(xintercept = max.study$readDepth)+
  ylab("Probability")+
  scale_color_manual("",values=col.power)+
  ylim(0,1)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position=c(0.85,0.3),
        legend.title=element_blank(),
        legend.text=element_text(size=8))

#Line graph for eQTL part

#Print maximal combination of values
max.study<-power.study.eqtl[which.max(power.study.eqtl$powerDetect),]

colnames(power.study.eqtl)[2:4]<-c("Detection power","Expression probability",
                                   "eQTL power")

#Show two-dimensional plots with always one parameter fixed
#Fixed read depths
power.study.plot<-power.study.eqtl[power.study.eqtl$readDepth==max.study$readDepth,]
power.study.plot<-reshape2::melt(power.study.plot,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth","readDepthSinglet",
                                           "mappedReadDepth","expressedGenes"))
```

```
power.study.plot$variable<-factor(power.study.plot$variable,
                                  levels=c("Expression probability","eQTL power",
                                           "Detection power"))

g.rd.b<-ggplot(power.study.plot,aes(totalCells,value,color=variable))+geom_line()
labels<-as.numeric(ggplot_build(g.rd)$layout$panel_params[[1]]$x.labels)
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf

g.rd.b<-g.rd.b +
  scale_x_continuous("Cells per individual",sec.axis=sec_axis(~., breaks=labels,
                     labels=sapply(labels,function(c) floor(
                     sampleSizeBudgetCalculation.restrictedDoublets(
                    max.study$totalCells, c,totalBudget, costKit, cellsPerLane,
                    costFlowCell,readsPerFlowcell))),
                    name="Sample size"))+
  geom_vline(xintercept = max.study$totalCells)+
  ylab("Probability")+
  scale_color_manual("",values=col.power)+
  ylim(0,1)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position=c(0.85,0.3),
        legend.title=element_blank(),
        legend.text=element_text(size=8))

#Fixed number of cells
power.study.plot<-power.study.eqtl[power.study.eqtl$totalCells==max.study$totalCells,]
power.study.plot<-reshape2::melt(power.study.plot,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth","readDepthSinglet",
                                           "mappedReadDepth","expressedGenes"))
power.study.plot$variable<-factor(power.study.plot$variable,
                                  levels=c("Expression probability","eQTL power",
                                           "Detection power"))

g.cp.b<-ggplot(power.study.plot,aes(readDepth,value,color=variable))+geom_line()
labels<-as.numeric(ggplot_build(g.cp)$layout$panel_params[[1]]$x.labels)
#> Warning in min(x): no non-missing arguments to min; returning Inf

#> Warning in min(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf

g.cp.b <- g.cp.b + scale_x_continuous("Read depth",sec.axis=sec_axis(~., breaks=labels,
                                      labels=sapply(labels, function(c)floor(
                                      sampleSizeBudgetCalculation.restrictedDoublets(
                                      max.study$totalCells, c,totalBudget, costKit,
                                      cellsPerLane, costFlowCell,readsPerFlowcell))),
  geom_vline(xintercept = max.study$readDepth)+
  scale_color_manual("",values=col.power)+
  ylab("Probability")+
  ylim(0,1)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position=c(0.85,0.3),
        legend.title=element_blank(),
        legend.text=element_text(size=8))

g.b<-ggarrange(g.rd,g.cp,g.rd.b,g.cp.b,ncol=4,nrow=1,labels=c("c","d","e","f"),
           align="hv",common.legend=TRUE,legend="bottom")
#> Warning in min(x): no non-missing arguments to min; returning Inf

#> Warning in min(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf

g<-ggarrange(g.a,g.b,ncol=1,nrow=2,heights=c(1.7,1))

print(g)
```

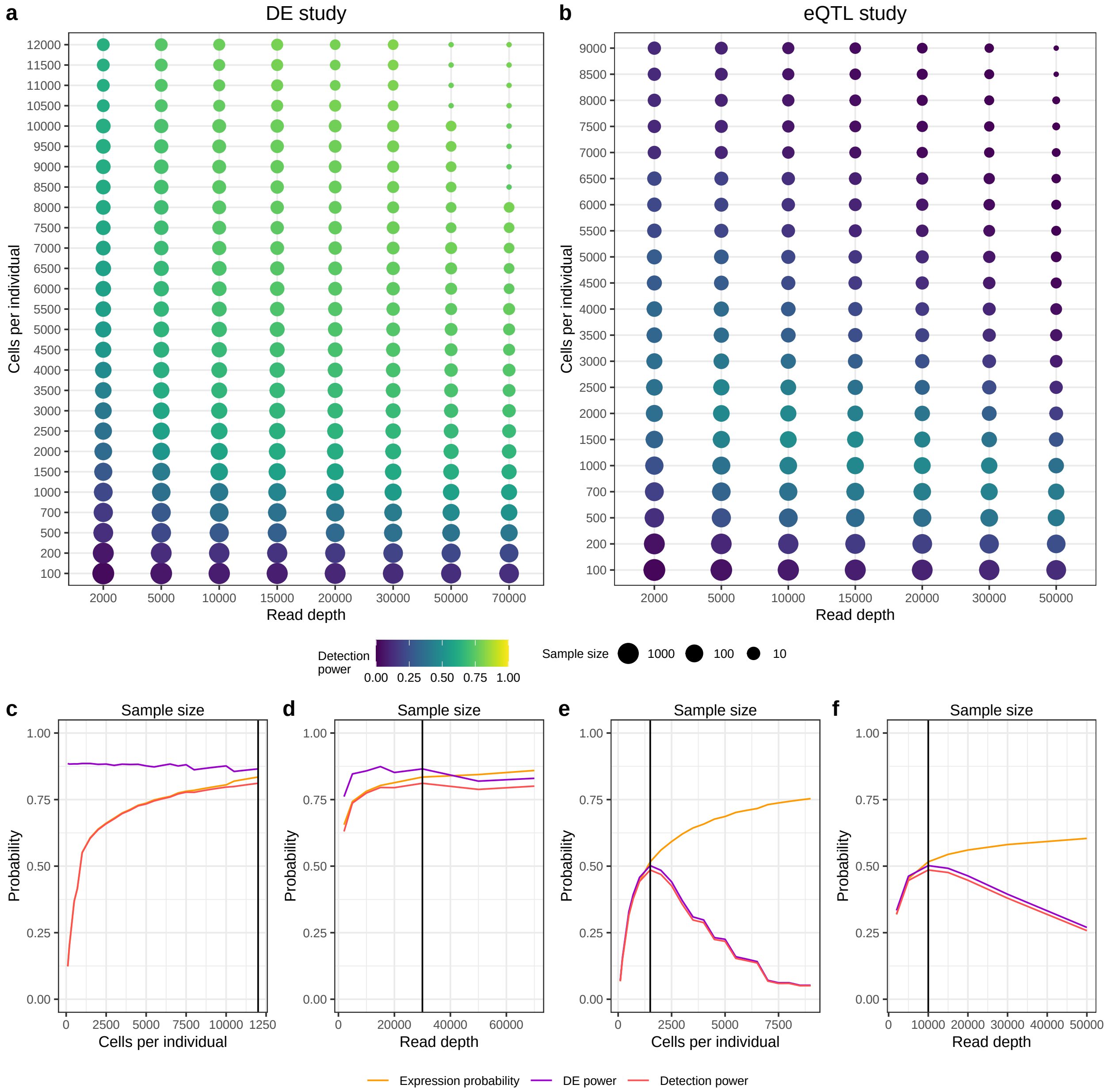


Figure 4: Parameter optimization for constant budget.

a,b Maximizing detection power by selecting the best combination of cells per individual and read depth for a DE study with a budget of 10,000€ (a) and an eQTL study with a budget of 30,000€ (b). Sample size is uniquely defined given the other two parameters due to the budget restriction and visualized using the point size.

c-f Overall detection power dependent on cost determining factors. Influence of the cells per individual given the optimized read depth (c,e) and of the read depth given the optimized number of cells per individual (d,f). (c,d) corresponds to the DE study in (a), visualized in (a) by the red frame around the row with the optimal number of cells (corresponding to (c)) and the red frame around the column with the optimal read depth (corresponding to (d)). Same frames for (e,f) in the eQTL study (b).The optimal sample size values are shown in the upper x axes for (c-f). Vertical line in the subplots marks the optimal parameter combination.

Effect sizes were chosen as in Figure 3. Gene expression is defined as detected in >50% (DE analysis) or >9.5% (eQTL analysis) of individuals.

Figure 5: Optimal parameters for varying budgets and 10X data

How the optimal parameters develop for an increasing experimental budget, is shown in Figures 5. As this means running the optimization function many times, this takes too long to run it in the vignette. Instead, the code is given, but the evaluation set to FALSE. Interested users can run the calculation, by setting it to TRUE. Calculations are done for DE and eQTL studies, with observed and simulated priors.

```
#####
#General parameters

ct<-"CD14+ Monocytes"
ct.freq<-0.2

cellsPerChanel<-20000
costKit<-5600
costFlowCell<-14032
readsPerFlowcell<-4100*10^6

mapping.efficiency<-0.8
minUMI<-3
perc.indiv<-0.5

budget<-seq(5000,100000,by=5000)

########
#Parameters for DE studies
readDepth<-c(2000,5000,10000,15000,20000,30000,40000,50000,60000,70000,100000)
cellsPerPerson<-c(100,200,500,700,seq(1000,9000,500))
```

```
#####
#Simulated DE studies - PBMC (10X)

simulated.parameters<-data.frame(mean.value=c(2,0.5,2,0.5),
                                 rank.max=c(20000,20000,10000,10000),
                                 name=c("highES_unifRank","lowES_unifRank",
                                        "highES_highRank","lowES_highRank"))

print(simulated.parameters)

#Number simulated DE genes
numGenes<-250

de.raw.values<-NULL
for(s in 1:nrow(simulated.parameters)){

  print(s)

  #Simulate prior for expression rank
  ranks<-uniform.ranks.interval(start=1,end=simulated.parameters$rank.max[s],
                                numGenes=numGenes)

  #Simulate prior for effect sizes
  set.seed(1)
  effectSizes<-effectSize.DE.simulation(mean=simulated.parameters$mean.value[s],
                                        sd=1,numGenes=numGenes)

  sample.DE.genes<-data.frame(ranks=ranks, FoldChange=effectSizes,
                              name="Simulated")

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.restrictedDoublets(i,"de",ct,ct.freq,
                             costKit,costFlowCell,readsPerFlowcell,
                             sample.DE.genes,"Simulated",
                             cellsPerChanel,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits,
                             disp.fun.param,
                             nCellsRange = cellsPerPerson,
                             readDepthRange = readDepth,
                             mappingEfficiency = mapping.efficiency,
                             min.UMI.counts = minUMI,
                             perc.indiv.expr = perc.indiv,
                             MTmethod = "FDR",
                             speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }
  max.values$budget<-budget
  max.values$parameters<-simulated.parameters$name[s]

  de.raw.values<-rbind(de.raw.values,max.values)
}

######
# DE studies with observed priors - PBMC (10X)

#Filter the DE reference study for PBMC data sets
de.ref.study<-scPower::de.ref.study[de.ref.study$type=="PBMC",]

de.real.values<-NULL
for(studyName in unique(de.ref.study$name)){

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.restrictedDoublets(i,"de",ct,ct.freq,
                   costKit,costFlowCell,readsPerFlowcell,
                   de.ref.study,studyName,
                   cellsPerChanel,read.umi.fit[read.umi.fit$type=="10X_PBMC_1",],
                   gamma.mixed.fits,
                   disp.fun.param,
                   nCellsRange = cellsPerPerson,
                   readDepthRange = readDepth,
                   mappingEfficiency = mapping.efficiency,
                   min.UMI.counts = minUMI, perc.indiv.expr = perc.indiv,
                   MTmethod = "FDR",
                   speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }

  max.values$budget<-budget
  max.values$parameters<-studyName
  de.real.values<-rbind(de.real.values,max.values)
}
```

```
#Parameters for eQTL studies
readDepth<-c(2000,5000,10000,15000,20000,30000,40000,50000,60000,70000,100000)
cellsPerPerson<-seq(500,9000,500)
```

```
#####
#Simulated eQTL studies - PBMC (10X)

simulated.parameters<-data.frame(mean.value=c(0.5,0.2,0.5,0.2),
                                 rank.max=c(20000,20000,10000,10000),
                                 name=c("highES_unifRank","lowES_unifRank",
                                        "highES_highRank","lowES_highRank"))

print(simulated.parameters)

#Number simulated eQTL genes
numGenes<-2000

eqtl.raw.values<-NULL
for(s in 1:nrow(simulated.parameters)){

  print(s)

  #Simulate prior for expression rank
  ranks<-uniform.ranks.interval(start=1,end=simulated.parameters$rank.max[s],
                                numGenes=numGenes)

  #Simulate prior for effect sizes
  set.seed(1)
  effectSizes<-effectSize.eQTL.simulation(mean=simulated.parameters$mean.value[s],
                                          sd=0.2,numGenes=numGenes)

  sample.eqtl.genes<-data.frame(ranks=ranks,Rsq=effectSizes,name="Simulated")

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.restrictedDoublets(i,"eqtl",ct,ct.freq,
                         costKit,costFlowCell,readsPerFlowcell,
                         sample.eqtl.genes,"Simulated",
                         cellsPerChanel,
                         read.umi.fit[read.umi.fit$type=="10X_PBMC_1",],
                         gamma.mixed.fits,
                         disp.fun.param,
                         nCellsRange = cellsPerPerson,
                         readDepthRange = readDepth,
                         mappingEfficiency = mapping.efficiency,
                         min.UMI.counts = minUMI, perc.indiv.expr = perc.indiv,
                         MTmethod = "Bonferroni",
                         speedPowerCalc=TRUE))

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }
  max.values$budget<-budget
  max.values$parameters<-simulated.parameters$name[s]

  eqtl.raw.values<-rbind(eqtl.raw.values,max.values)
}



######
# eQTL studies with observed priors - PBMC (10X)

eqtl.real.values<-NULL
for(studyName in unique(eqtl.ref.study$name)){

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.restrictedDoublets(i,"eqtl",ct,ct.freq,
                   costKit,costFlowCell,readsPerFlowcell,
                   eqtl.ref.study,studyName,
                   cellsPerChanel,read.umi.fit[read.umi.fit$type=="10X_PBMC_1",],
                   gamma.mixed.fits,
                   disp.fun.param,
                   nCellsRange = cellsPerPerson,
                   readDepthRange = readDepth,
                   mappingEfficiency = mapping.efficiency,
                   min.UMI.counts = minUMI, perc.indiv.expr = perc.indiv,
                   MTmethod = "Bonferroni",
                   speedPowerCalc=TRUE))

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }

  max.values$budget<-budget
  max.values$parameters<-studyName
  eqtl.real.values<-rbind(eqtl.real.values,max.values)
}
```

As mentioned above, percalculated values are loaded in the vignette instead, created with the code shown above (but set to eval=FALSE).

```
#Get precalculated values
data(precalculatedBudgetOptim) #budget.optimization
eqtl.raw.values<-budget.optimization[budget.optimization$type=="eQTL_simulated_PBMC",]
de.raw.values<-budget.optimization[budget.optimization$type=="DE_simulated_PBMC",]
eqtl.real.values<-budget.optimization[budget.optimization$type=="eQTL_observed_PBMC",]
de.real.values<-budget.optimization[budget.optimization$type=="DE_observed_PBMC",]

#Replace probability names
var.labs<-setNames(c("Detection power","Cells per individual",
                     "Read depth","Sample size"),
                   c("powerDetect","totalCells","readDepth","sampleSize"))

#Data frame with dummy values to scale the y axes
dummy.plot<-data.frame(budget=c(min(budget),max(budget),min(budget),max(budget),
                                min(budget),max(budget),min(budget),max(budget)),
                       variable=c("powerDetect","powerDetect","readDepth",
                                  "readDepth","totalCells","totalCells",
                                  "sampleSize","sampleSize"),
                       value=c(0,1,min(readDepth),max(readDepth),
                              min(cellsPerPerson),max(cellsPerPerson),0,400))

#eQTL simulated values
plot.values<-eqtl.raw.values[,c("budget","powerDetect","totalCells","readDepth",
                                "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.eqtl.1<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Simulated parameters",values=col.paired[c(2,4,1,3)])+
  theme_bw()+
  xlab("Budget")+ylab("eQTL study")

#DE simulated values
plot.values<-de.raw.values[,c("budget","powerDetect","totalCells","readDepth",
                              "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.de.1<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Simulated parameters",values=col.paired[c(2,4,1,3)])+
  theme_bw()+
  xlab("Budget")+ylab("DE study")

#Merge values from eQTL study and DE study
all.studies<-c(unique(de.real.values$parameters),
               unique(eqtl.real.values$parameters))
```

```
#eQTL real values
plot.values<-eqtl.real.values[,c("budget","powerDetect","totalCells","readDepth",
                                 "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.eqtl.2<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Observed parameters",values=col.set.new[5:10],
                     labels=all.studies,limits=all.studies)+
  theme_bw()+
  xlab("Budget")+ylab("eQTL study")

#DE real values
plot.values<-de.real.values[,c("budget","powerDetect","totalCells","readDepth",
                               "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.de.2<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Observed parameters",values=col.set.new[5:10],
                     labels=all.studies,limits=all.studies)+
  theme_bw()+
  xlab("Budget")+ylab("DE study")

#Overview plot
g.proto<-ggarrange(g.de.1,g.eqtl.1,nrow=2,common.legend = TRUE,
                   legend="bottom",labels=c("a","b"))

g.real<-ggarrange(g.de.2,g.eqtl.2,nrow=2,common.legend = TRUE,
                  legend="bottom",labels=c("c","d"))

row1 <- ggplot() + annotate(geom = 'text', x=1, y=1,
                            label="Prototypic scenarios",
                            size=6, angle = 90) + theme_void()

row2 <- ggplot() + annotate(geom = 'text', x=1, y=1,
                            label="Real world scenarios",
                            size=6,angle = 90) + theme_void()

g<-ggarrange(row1,g.proto,row2,g.real,nrow=2,ncol=2,
             widths=c(0.03,1))

print(g)
```

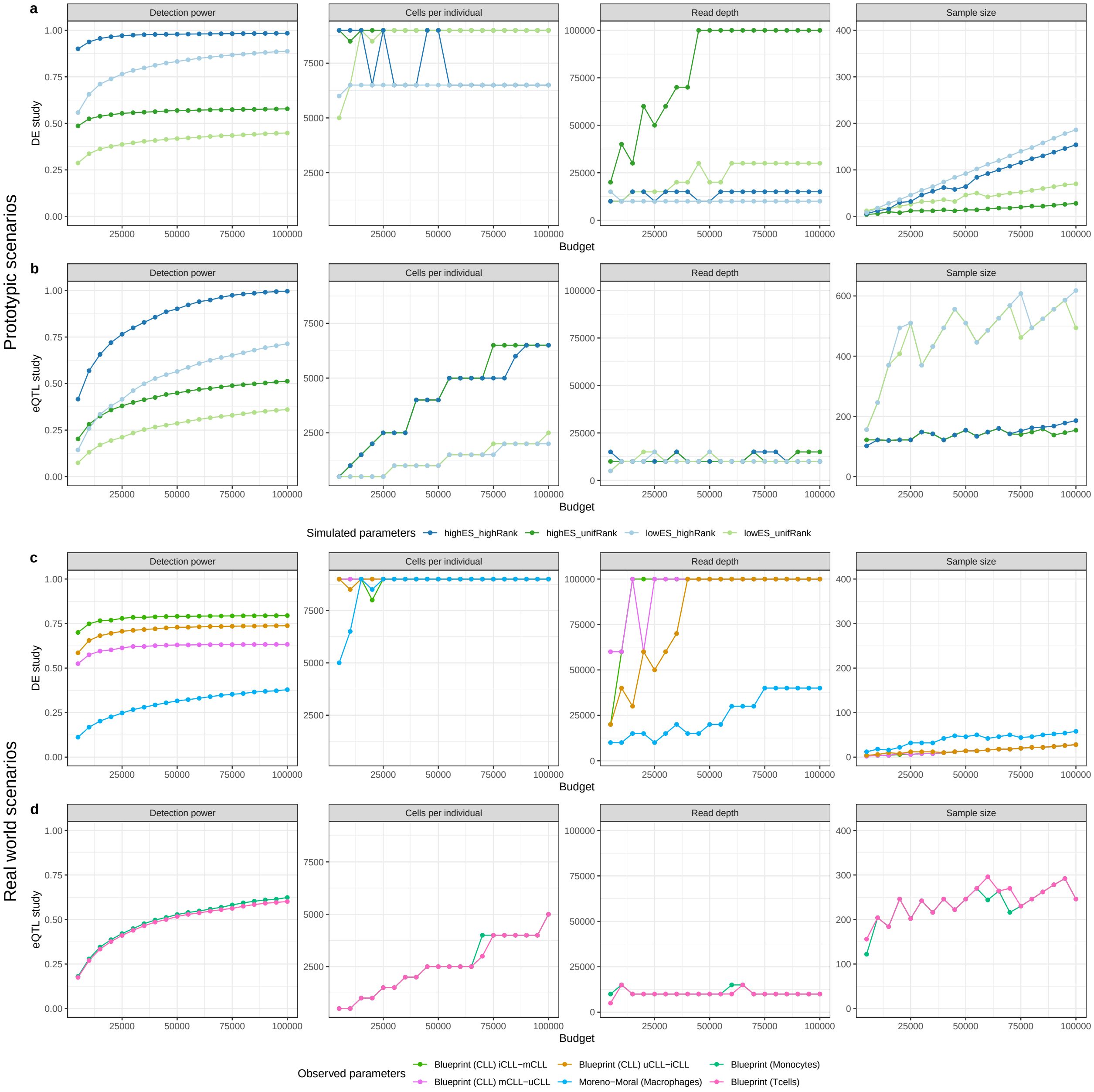


Figure 5: Optimal parameters for varying budgets and 10X Genomics data. The maximal reachable detection power (column 1) and the corresponding optimal parameter combinations (column 2-4) change depending on the given experimental budget (x-axis). The colored lines indicate different effect sizes and gene expression rank distributions.

a,b Different simulated effect sizes and rank distributions for DEG studies (a) and eQTL studies (b) with models fitted on 10X PBMC data. highES = high effect sizes, lowES = low effect sizes, highRank = high expression ranks and unifRank = uniformly distributed expression ranks (always relative to effect sizes observed in published studies).

c,d Effect sizes and rank distributions observed in cell type sorted bulk RNA-seq DEG studies (c) and eQTL studies (d) with model fits analogously to (a,b).

Expression thresholds were chosen as for Figure 4.

Supplementary Figures

Figure S1: Power to detect rare cell types

Model to calculate the power to detect rare cell types. Please look into the introduction vignette, section “Part 3: Power estimation to detect a sufficent number of cells in a specific cell types” for an explanation of all parameters. The cell type frequencies are taken from BioRad (see citation in the publication), all other parameters are set to realistic values.

```r
prob=0.95

#Sample size
Nind <- c(100, 150, 200, 300)

#Number of cells required
N.min.cells<-c(1,10,20,50)

#Take cell type frequencies estimated from BioRad (see above)
cell.types.biorad<-data.frame(ct=c("B cells", "Monocytes", "NK cells",
                                   "Dendritic cells","CD4 T cells","CD8 T cells"),
                              freq=c(13,13,7,1.5,30,13)/100)


filter.ct<-c("CD4 T cells", "NK cells", "Dendritic cells", "Monocytes")
cell.type.comp<-cell.types.biorad[cell.types.biorad$ct %in% filter.ct,]

#Set frequency behind the cell type labels (add a small number to round up at 0.5)
cell.type.comp$ct<-paste0(cell.type.comp$ct,
                                   " (",round(cell.type.comp$freq*100+0.01,1),"%)")


#Arrange the cell types in the order of frequency
cell.type.comp<-cell.type.comp[order(cell.type.comp$freq),]
cell.type.comp$ct<-factor(cell.type.comp$ct,levels=cell.type.comp$ct)

#Test each possible parameter combination
parameter.combinations<-expand.grid(prob,N.min.cells,cell.type.comp$ct,Nind)
colnames(parameter.combinations)<-c("prob.cutoff","min.num.cells","ct","num.indivs")

#Merge cell types with frequencies
parameter.combinations<-merge(parameter.combinations,cell.type.comp)

parameter.combinations$result.ssize<-mapply(scPower::number.cells.detect.celltype,
                                           parameter.combinations$prob.cutoff,
                                           parameter.combinations$min.num.cells,
                                           parameter.combinations$freq,
                                           parameter.combinations$num.indivs)




#Labels samples
axis.labeller<-function(variable,value){
  return(paste(value,"individuals"))
}

g<-ggplot(parameter.combinations,aes(x=min.num.cells,y=result.ssize,col=ct))+
  geom_line()+geom_point()+facet_wrap(~num.indivs,ncol=2, labeller = axis.labeller)+
  xlab("Minimal number of cells from target cell type per individual")+
  ylab("Cells per individual")+
  scale_color_manual("Cell type",values=col.set2)+
  scale_y_log10()+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        aspect.ratio = 0.8,
        legend.position = "bottom")+
  guides(col=guide_legend(nrow=2,byrow=TRUE))
#> Warning: The labeller API has been updated. Labellers taking `variable` and
#> `value` arguments are now deprecated. See labellers documentation.

print(g)
```

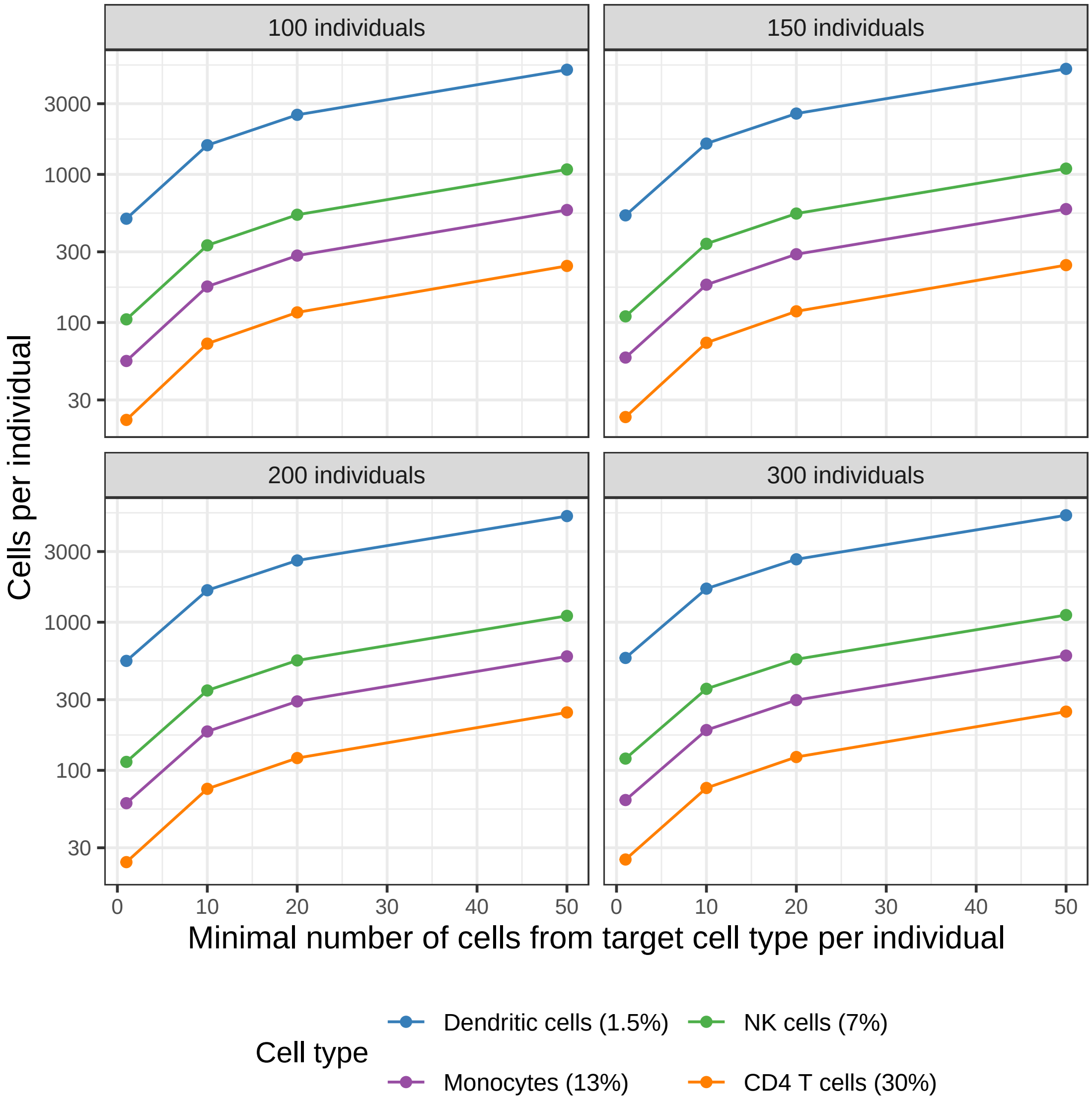


Figure S1: Power to detect rare cell types. Required number of cells per individual (y-axis, log scale) to detect the minimal number of cells from a target cell type per individual (x-axis) with a probability of 95%. The probability depends on the total number of individuals and the frequency of the target cell type (purple, red, yellow, green). Note that the required number of cells per sample only counts "correctly measured" cells (no doublets etc), so the number is a lower bound for the required cells to be sequenced.

Figure S2: PBMC data set

Figure S2 shows the UMAP of the PBMC data set with cell type annotation and marker gene expression. Due to runtime reasons, the preprocessing is not performed here again. It is in detail described in the methods part of the publication.

Figure S3 - S5: Steps fitting the gamma mixed model

Fitting the gamma mixed model is too runtime intensive for the vignette. The fitting for a small data set is shown in the introduction vignette. For our large data set, it was done exactly the same way, using the same functions. Also the corresponding plots to Figure S3 to S5 are shown in the introduction vignette and are therefore not reproduced here again. Figure S3 shows how mean values sampled for the fitted gamma mixture distribution are compared to the observed mean values, this is shown in the introduction vignette, section "Comparison of gamma mixed fits with original means". The next section "Parameterization of the parameters of the gamma fits by the mean UMI counts per cell" shows the relationship between the gamma mixed parameters and the mean UMI counts per cell, as depicted in Figure S4. And the relationship between UMI counts and mapped read depth is shown in section "Fitting a functions for UMI counts dependent on read depth", as in Figure S5.

Figure S6: Expression probability model for a count threshold > 0

The supplementary figure shows the expression probability model for all runs of our own data set. It is an extension to Figure 2, where only the results for "Run 5" are shown. For an explanation of the calculation, please look at the description text for Figure 2.

```r
#Load observed gene counts (precalculated)
data(precalculatedObservedGeneCounts) #observed.gene.counts

#Number of samples per run
personsPerRun<-list("Run 1"=14,"Run 2"=7, "Run 3"=7, "Run 4"=14,
                    "Run 5"=14, "Run 6"=14)

#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="own_count0",]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){

  sampleSize<-personsPerRun[[estimates.allSamples$run[[i]]]]

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=0,
    perc.indiv=0.5)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Replace subsampling
subsampled.names<-setNames(c("50000","37500","25000","12500"),
                           c("complete","subsampled75","subsampled50",
                             "subsampled25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]

#Rename runs
plot.estimates$run<-as.character(plot.estimates$run)

g<-ggplot(plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(fill=sample,color=sample,shape=cell.type),size=2)+
  facet_wrap(~run,ncol=2,scales="free")+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_color_manual("Target read depth",values=col.set2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  scale_fill_manual(values=col.set2)+
  scale_linetype_discrete(name="Curve",labels=c("Observed","Estimated")) +
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.position = "bottom",
        legend.direction = "horizontal", legend.box = "vertical")+
  guides(fill=FALSE)

print(g)
```

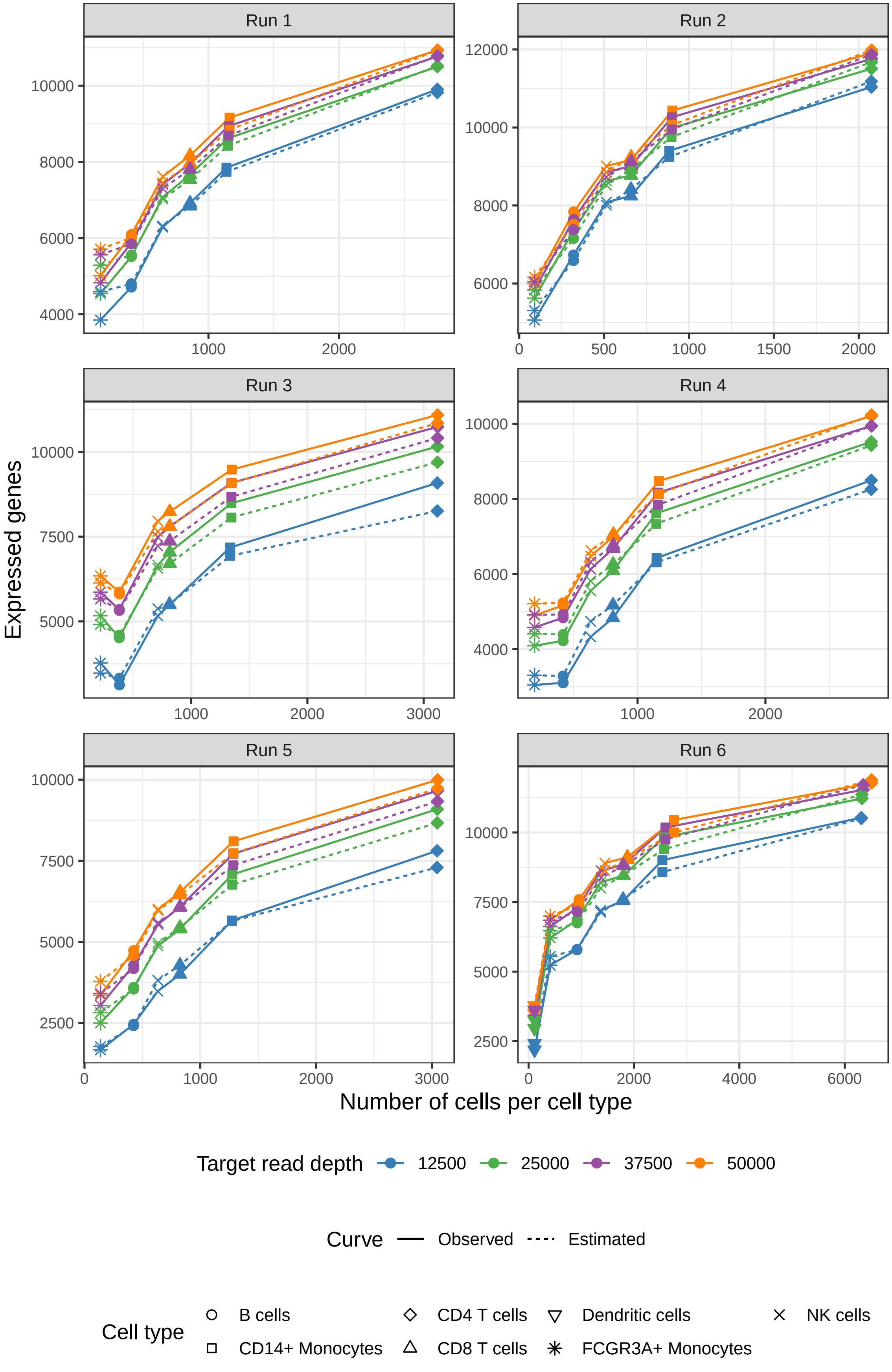

Run 1
Run 2
Run 3
Run 4
Run 5
Run 6
Expressed genes
Number of cells per cell type
4000
6000
8000
10000
1000
2000
6000
8000
10000
12000
0
500
1000
1500
2000
5000
7500
10000
1000
2000
3000
4000
6000
8000
10000
1000
2000
2500
5000
7500
10000
0
1000
2000
3000
2500
5000
7500
10000
0
2000
4000
6000
Target read depth
12500
25000
37500
50000
Curve
Observed
Estimated
Cell type
B cells
CD14+ Monocytes
CD4 T cells
CD8 T cells
Dendritic cells
FCGR3A+ Monocytes
NK cells

Figure S6: Expression probability model for a count threshold > 0. The number of expressed genes expected under our model (dashed line) closely matches the observed number of expressed genes (solid line) dependent on the number of cells per cell type (cell type indicated by point symbol) for all 6 batches (see panel titles) of the training PBMC data set (Supplementary Table S2). A gene is defined as expressed if it has > 0 counts in more than 50% of the individuals.

Figure S7: Expression probability model for a count threshold > 10

The supplementary figure shows the same as Figure S6, but with a count threshold of 10 instead of 0

```r
#Load observed gene counts (precalculated)
data(precalculatedObservedGeneCounts) #observed.gene.counts

#Number of samples per run
personsPerRun<-list("Run 1"=14,"Run 2"=7, "Run 3"=7, "Run 4"=14,
                    "Run 5"=14, "Run 6"=14)

#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="own_count10",]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){

  sampleSize<-personsPerRun[[estimates.allSamples$run[[i]]]]

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=10,
    perc.indiv=0.5)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Replace subsampling
subsampled.names<-setNames(c("50000","37500","25000","12500"),
                           c("complete","subsampled75","subsampled50",
                             "subsampled25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]

#Rename runs
plot.estimates$run<-as.character(plot.estimates$run)
```

```
g<-ggplot(plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(fill=sample,color=sample,shape=cell.type),size=2)+
  facet_wrap(~run,ncol=2,scales="free")+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_color_manual("Target read depth",values=col.set2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  scale_fill_manual(values=col.set2)+
  scale_linetype_discrete(name="Curve",labels=c("Observed","Estimated")) +
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.position = "bottom",
        legend.direction = "horizontal", legend.box = "vertical")+
  guides(fill=FALSE)

print(g)
```

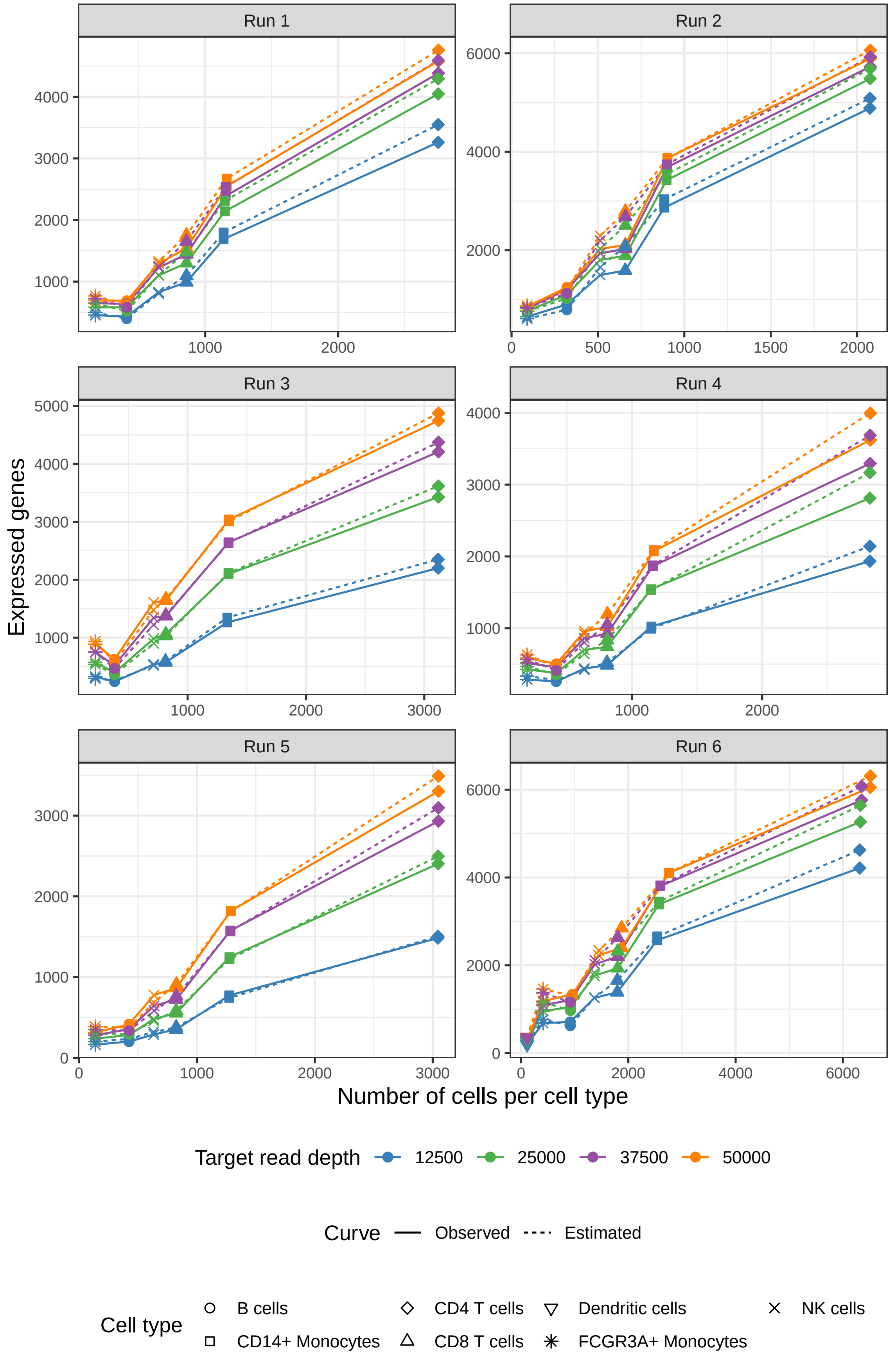

Run 1
Run 2
Run 3
Run 4
Run 5
Run 6
Expressed genes
Number of cells per cell type
1000
2000
3000
4000
5000
6000
0
500
1500
Target read depth
12500
25000
37500
50000
Curve
Observed
Estimated
Cell type
B cells
CD14+ Monocytes
CD4 T cells
CD8 T cells
Dendritic cells
FCGR3A+ Monocytes
NK cells

Figure S7: Expression probability model for a count threshold > 10. Variant of Figure S6 with a count threshold of 10 instead of 0.

Figure S8: Expression rank priors from cell type sorted bulk studies

Figure S8 shows the expression rank priors taken from all analysed DE and eQTL studies. The ranks were calculated using the function "gene.rank.calculation", as described in the introduction vignette in section "Priors from DE / eQTL studies". For citations of all used studies, please look into the publication.

```
#Load precalculated results
eqtl.ranks<-scPower::eqtl.ref.study
de.ranks<-scPower::de.ref.study[de.ref.study$type=="PBMC",]
de.ranks.lung<-scPower::de.ref.study[de.ref.study$type=="lung",]
de.ranks.panc<-scPower::de.ref.study[de.ref.study$type=="pancreas",]

#Add on the y-axis also absolute numbers (second plot)
eqtl.ranks$name<-paste0("Blueprint (",eqtl.ranks$cellType,")")
eqtl.g<-ggplot(eqtl.ranks, aes(rank, col=name)) + stat_ecdf(geom = "step") +
  xlab("Number of expressed genes") +ylab("Percentage of eQTL genes")+
  scale_color_manual("Study", values=col.set.new[1:2])+
  theme_bw() + xlim(0,30000) +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.position = "bottom",
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8)+
  guides(col=guide_legend(nrow=2,byrow=TRUE))

de.g<-ggplot(de.ranks, aes(rank, col=name)) + stat_ecdf(geom = "step") +
  xlab("Number of expressed genes") +ylab("Percentage of DE genes")+
  scale_color_manual("Study", values=col.set.new[3:6])+
  theme_bw() + xlim(0,30000) +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.position = "bottom",
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8)+
  guides(col=guide_legend(nrow=4,byrow=TRUE))

de.ranks.lung$name<-gsub("_"," ",de.ranks.lung$name)
de.lung<-ggplot(de.ranks.lung, aes(rank, col=name)) + stat_ecdf(geom = "step") +
  xlab("Number of expressed genes") +ylab("Percentage of DE genes")+
  scale_color_manual("Study", values=col.set.new[7])+
  theme_bw() + xlim(0,30000) +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.position = "bottom",
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8)+
  guides(col=guide_legend(nrow=1,byrow=TRUE))
```

```
de.ranks.panc$name<-gsub("_"," ",de.ranks.panc$name)
de.panc<-ggplot(de.ranks.panc, aes(rank, col=name)) + stat_ecdf(geom = "step") +
  xlab("Number of expressed genes") +ylab("Percentage of DE genes")+
  scale_color_manual("Study", values=col.set.new[8:9])+
  theme_bw() + xlim(0,30000) +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.position = "bottom",
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8)+
  guides(col=guide_legend(nrow=2,byrow=TRUE))

g<-ggarrange(eqtl.g,de.g,de.lung,de.panc,ncol=2,nrow=2,labels=c("a","b","c","d"),
             align="hv")
#> Warning: Removed 1 rows containing non-finite values (stat_ecdf).

print(g)
```

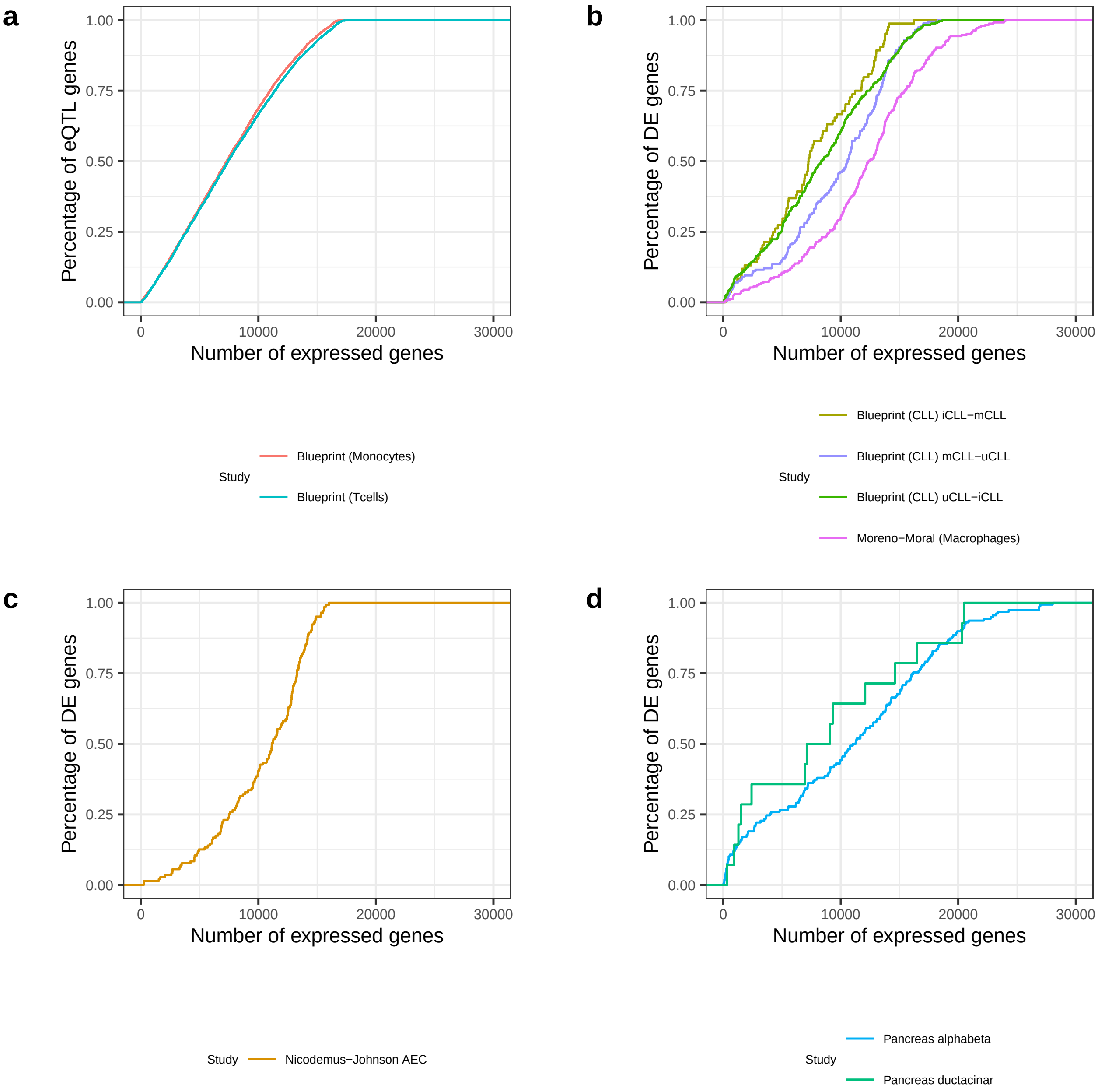


Figure S8: Expression rank priors from cell type sorted bulk studies. Gene expression level of DEGs / eQTL genes (y axis) relative to all genes (x axis) (i.e. gene expression ranks) gained from cell type sorted bulk studies for eQTL studies of PBMCs (a) and for DE studies of PBMCs (b), lung tissue (c) and pancreas tissue (d).

Figure S9: Effect size priors from cell type sorted bulk studies

Figure S9 shows effect sizes taken directly from all analysed DE and eQTL studies, defining significant genes to be below an FDR threshold of 0.05. For citations of all used studies, please look into the publication.

```
eqtl.ranks$name<-paste0("Blueprint (",eqtl.ranks$cellType,")")
eqtl.g<-ggplot(eqtl.ranks, aes(x=beta, fill=name))+
  geom_histogram(position="dodge")+
  xlab("Beta values") +ylab("Counts")+
  scale_fill_manual("Study", values=col.set.new[1:2])+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        legend.position = "bottom")+
  guides(fill=guide_legend(nrow=2,byrow=TRUE,
                           override.aes = list(size = 0.5)))

de.ranks$log2FoldChange<-log2(de.ranks$FoldChange)
de.g<-ggplot(de.ranks, aes(x=log2FoldChange, fill=name)) +
  geom_histogram(position="dodge")+
  xlab("Log fold change") +ylab("Counts")+
  scale_fill_manual("Study", values=col.set.new[3:6])+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        legend.position = "bottom")+
  guides(fill=guide_legend(nrow=4,byrow=TRUE,
                           override.aes = list(size = 0.5)))

de.ranks.lung$log2FoldChange<-log2(de.ranks.lung$FoldChange)
de.lung<-ggplot(de.ranks.lung, aes(x=log2FoldChange, fill=name)) +
  geom_histogram(position="dodge")+
  xlab("Log fold change") +ylab("Counts")+
  scale_fill_manual("Study", values=col.set.new[7])+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        legend.position = "bottom")+
  guides(fill=guide_legend(nrow=1,byrow=TRUE,
                           override.aes = list(size = 0.5)))

de.ranks.panc$log2FoldChange<-log2(de.ranks.panc$FoldChange)
de.panc<-ggplot(de.ranks.panc, aes(x=log2FoldChange, fill=name)) +
  geom_histogram(position="dodge")+
  xlab("Log fold change") +ylab("Counts")+
  scale_fill_manual("Study", values=col.set.new[8:9])+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        legend.position = "bottom")+
  guides(fill=guide_legend(nrow=2,byrow=TRUE,
                           override.aes = list(size = 0.5)))
```

```
g<-ggarrange(eqtl.g,de.g,de.lung,de.panc,ncol=2,nrow=2,
             labels=c("a","b","c","d"),align="hv")
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

print(g)
```

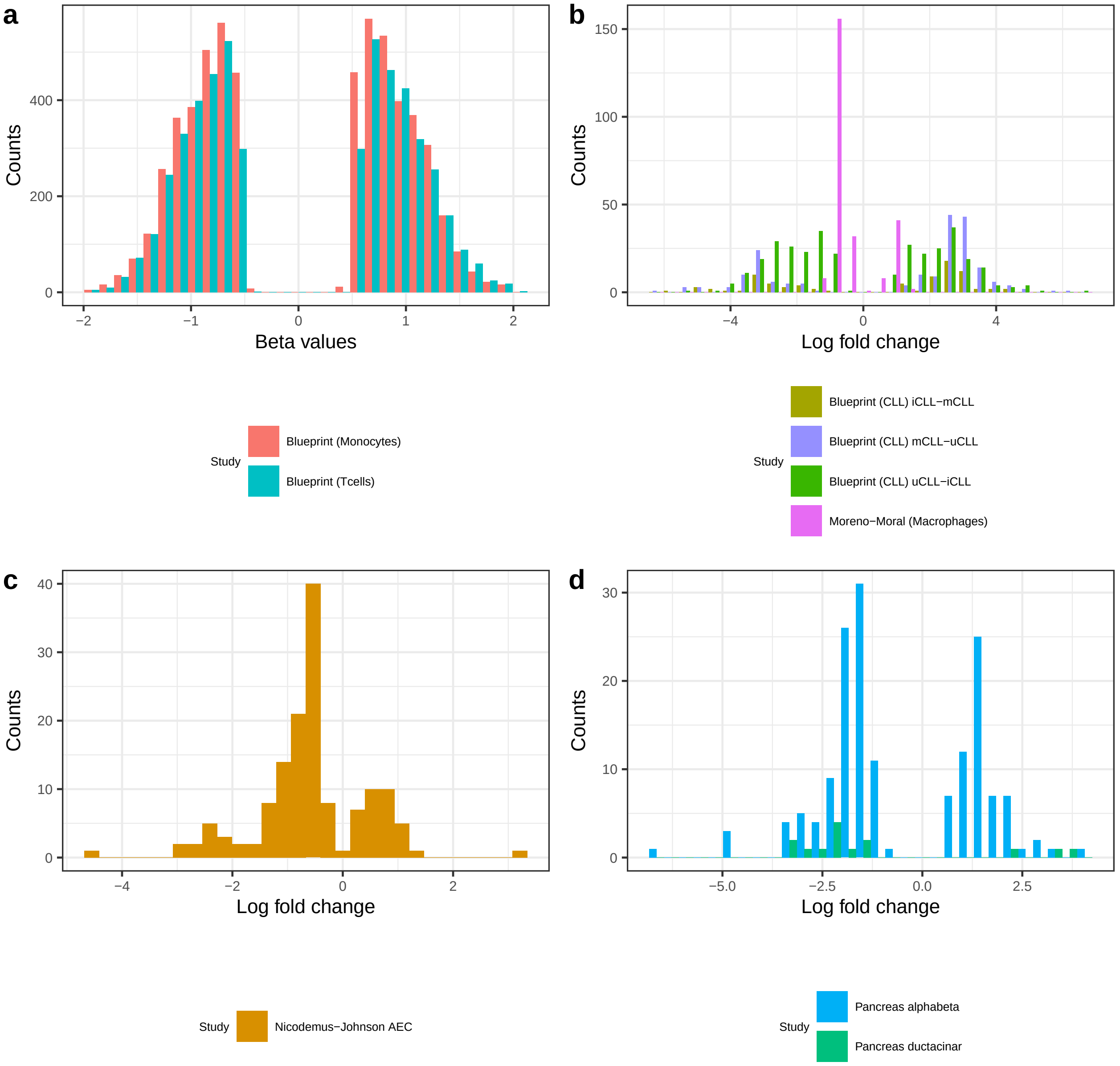


Figure S9: Effect size priors from cell type sorted bulk studies. Effect sizes gained from cell type sorted bulk studies for eQTL studies of PBMCs (a) and for DE studies of PBMCs (b), lung tissue (c) and pancreas tissue (d). The effect size is quantified as beta values for eQTL studies and as log fold changes for DE studies.

Figure S10: Detection power using observed priors from reference studies

Figure S10 is an extension of Figure 3a,b, showing the detection power for all analysed reference studies with PBMC cell types and matched cell type expression model, while in Figure 3a,b it is only shown for one DE and one eQTL study.

In contrast to the original publication, the parameter useSimulatedPower is set to FALSE for the eQTL budget optimization. Both parameters speed the calculation in the vignette, while changing the overall power only slightly. We show in the publication the more accurate version including simulated eQTL power for lowly expressed genes.

```
#General parameters
transcriptome.mapped.reads<-20000
nGenes<-21000
ct.freq<-1
cellsPerLane<-20000

########
# DE subplot
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(4,6,8,10,16,20)

#Build a frame of all possible combinations and reference studies
param.combis<-expand.grid(cellsPerCelltype,sampleSize,
                          unique(de.ref.study$name[de.ref.study$type=="PBMC"]))
colnames(param.combis)<-c("cellsPerCelltype","sampleSize","refStudyName")

#Match for each study the corresponding cell type
matched.ct<-data.frame(refStudyName=unique(de.ref.study$name[de.ref.study$type==
                                                               "PBMC"]),
                       ct=c(rep("CD4 T cells",3),"CD14+ Monocytes"),
                       stringsAsFactors=FALSE)
print(matched.ct)
#>                 refStudyName              ct
#> 1  Blueprint (CLL) iCLL-mCLL     CD4 T cells
#> 2  Blueprint (CLL) mCLL-uCLL     CD4 T cells
#> 3  Blueprint (CLL) uCLL-iCLL     CD4 T cells
#> 4 Moreno-Moral (Macrophages) CD14+ Monocytes
param.combis<-merge(param.combis,matched.ct,by="refStudyName")
param.combis$refStudyName<-as.character(param.combis$refStudyName)

#Test results for each parameter combination
res<-lapply(1:nrow(param.combis),function(i)
  power.general.restrictedDoublets(param.combis$sampleSize[i],
                             param.combis$cellsPerCelltype[i],
                             transcriptome.mapped.reads,
                             ct.freq, "de",de.ref.study,
                             param.combis$refStudyName[i],
                             cellsPerLane,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits,
                             param.combis$ct[i],
                             disp.fun.param,mappingEfficiency = 1,
                             multipletRate=0,multipletFactor=1,
                             min.UMI.counts = 10,
                             perc.indiv.expr = 0.5,
                             nGenes=nGenes,
                             sign.threshold = 0.05,
                             MTmethod= "FDR"))

power.DE.study<-rbindlist(res)

#Convert axes to factors
power.DE.study$sampleSize<-as.factor(power.DE.study$sampleSize)
power.DE.study$totalCells<-as.factor(power.DE.study$totalCells)

g.DE<-ggplot(power.DE.study,aes(x=totalCells,y=sampleSize,fill=powerDetect))+
  geom_tile()+facet_wrap(~name,ncol=2)+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Total sample size")+
  scale_fill_viridis("Detection power",limits=c(0,1))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

##########
#eQTL subplot

cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(50,100,150,200)

#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize,unique(eqtl.ref.study$name))
colnames(param.combis)<-c("cellsPerCelltype","sampleSize","refStudyName")

#Match for each study the corresponding cell type
matched.ct<-data.frame(refStudyName=unique(eqtl.ref.study$name),
                       ct=c("CD14+ Monocytes","CD4 T cells"),
                       stringsAsFactors=FALSE)
print(matched.ct)
#>            refStudyName              ct
#> 1 Blueprint (Monocytes) CD14+ Monocytes
#> 2    Blueprint (Tcells)     CD4 T cells
param.combis<-merge(param.combis,matched.ct,by="refStudyName")
param.combis$refStudyName<-as.character(param.combis$refStudyName)

#Test results for each parameter combination
res<-lapply(1:nrow(param.combis),function(i)
  power.general.restrictedDoublets(param.combis$sampleSize[i],
                             param.combis$cellsPerCelltype[i],
                             transcriptome.mapped.reads,
                             ct.freq, "eqtl",eqtl.ref.study,
                             param.combis$refStudyName[i],
                             cellsPerLane,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits,
                             param.combis$ct[i],
                             disp.fun.param,mappingEfficiency = 1,
                             multipletRate=0,multipletFactor=1,
                             min.UMI.counts = 10,
                             perc.indiv.expr = 0.5,
                             nGenes=21000,
                             sign.threshold = 0.05,
                             MTmethod= "Bonferroni",
                             #The parameter is set differently
                             #in the publictions, but changed here to
                             #speed the calculation
                             useSimulatedPower = FALSE)) #(in publication: TRUE)

power.eqtl.study<-rbindlist(res)

#Convert axes to factors
power.eqtl.study$sampleSize<-as.factor(power.eqtl.study$sampleSize)
power.eqtl.study$totalCells<-as.factor(power.eqtl.study$totalCells)

g.eQTL<-ggplot(power.eqtl.study,aes(x=totalCells,y=sampleSize,fill=powerDetect))+
  geom_tile()+facet_wrap(~name,ncol=2)+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Total sample size")+
  scale_fill_viridis("Detection power",limits=c(0,1))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

g<-ggarrange(g.DE,g.eQTL,ncol=1,nrow=2,heights=c(2,1),labels=c("a","b"),
          align="hv",common.legend = TRUE,legend="right")
#> Warning: Graphs cannot be horizontally aligned unless the axis parameter is set.
#> Placing graphs unaligned.

print(g)
```

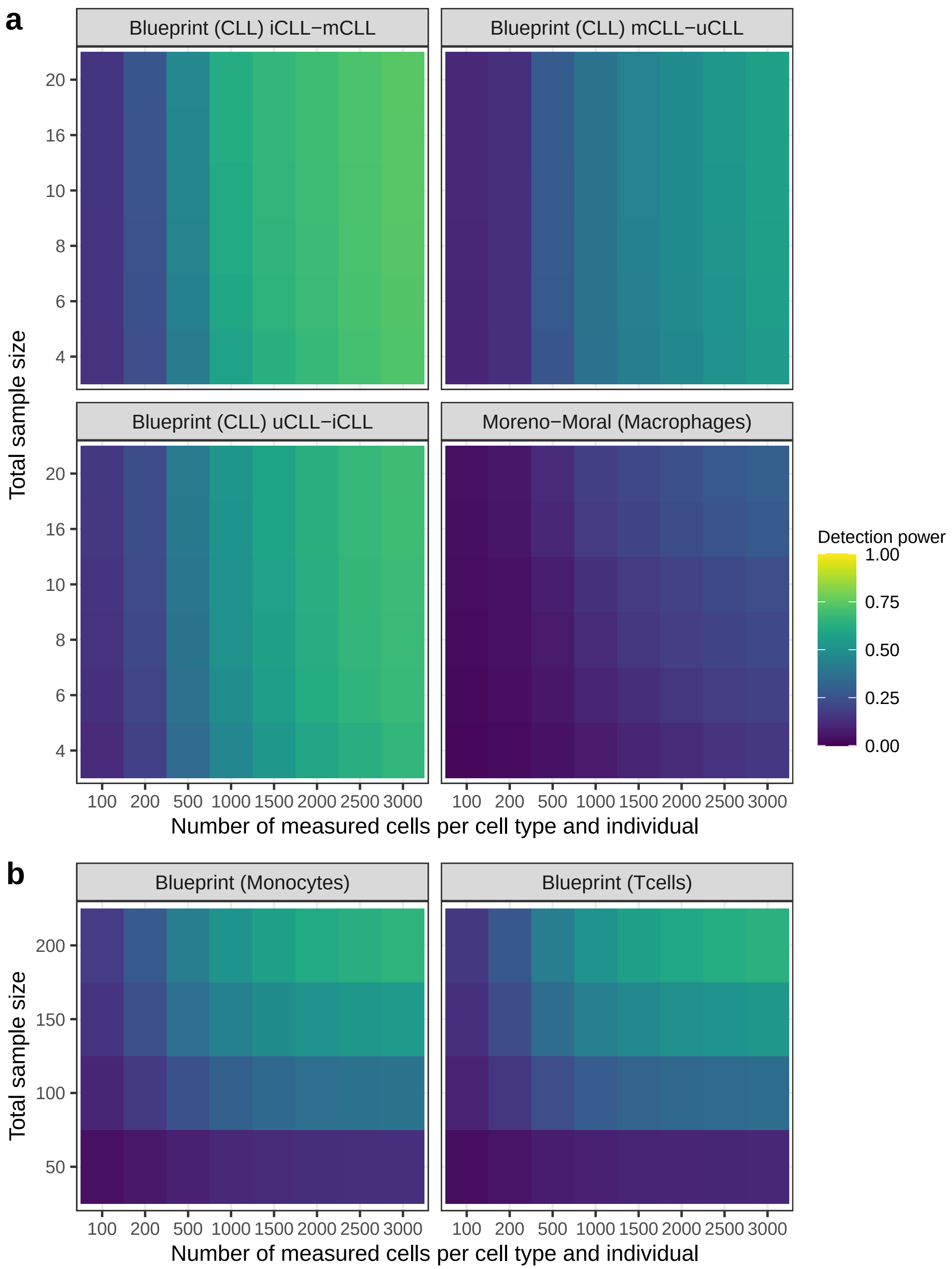


Figure S10: Detection power using observed priors from reference studies. Detection power for DE genes (a) and eQTL genes (b) dependent on the study, total sample size and the number of measured cells per cell type for a transcriptome mapped read depth per cell of 20,000. The detection power is the product of the probability that the gene is expressed and the power to detect it as a DE or eQTL gene, respectively, assuming that it is expressed. The fold change for DE genes and the R2 for eQTL genes is taken from published studies, together with the expression rank of the genes (studies shown in panel titles). The expression profile and

expression probabilities in a single cell experiment with a specific number of samples and measured cells was estimated using our expression prior, setting the definition for expressed to > 10 counts in more than 50% of the individuals. Multiple testing correction was performed by using FDR adjusted p-values for DE analysis and FWER adjusted p-values for eQTL analysis.

Figure S11: Variant of Main Figure 3 with FWER adjusted significance thresholds

This figure was generated with exactly the same code as Main Figures 3a,b, only setting the multiple testing method (MTmethod) to “FWER” instead of “FDR” for the DE power.

Figure S12: Relation between eQTL power and expression mean in a simulation study

Figure S12 shows how large the deviation between simulated and analytic mean values is for small mean values, using a range of typical parameter combinations for sample size, effect size (Rsq), mean and Bonferroni adjustetd p-values.

To speed up calculations, the plot is only shown for a reduced number of parameter combinations in this vignette.

```r
#Parameters used for the original plot
# count.mean<-c(seq(1,20),50,100,150)
# nGenes<-c(1000,2000,5000,10000)

#Reduced parameters used here to accelerate
count.mean<-seq(1,9)
nGenes<-1000

#Simulate look-up table of typical parameter combinations
sampleSize<-c(20,50,100,200)
Rsq<-quantile(abs(scPower::eqtl.ref.study$Rsq))

#Check also different p-values (all approximating Bonferroni corrected p-values)
pvals<-0.05/(10*nGenes)

param.combis<-expand.grid(sampleSize,Rsq,count.mean,pvals,KEEP.OUT.ATTRS = FALSE)
colnames(param.combis)<-c("sampleSize","Rsq","count.mean","pvals")

#Get simulated power
param.combis$power.simulated<-sapply(1:nrow(param.combis),function(i)
    scPower:::power.eqtl.simulated(param.combis$count.mean[i],
                                   param.combis$Rsq[i],
                                   param.combis$pvals[i],
                                   param.combis$sampleSize[i]))
#> Warning in summary.lm(model): essentially perfect fit: summary may be unreliable

#Get analytic power
param.combis$power.analytic<-sapply(1:nrow(param.combis),function(i)
    scPower:::power.eqtl.ftest(param.combis$Rsq[i],
                               param.combis$pvals[i],
                               param.combis$sampleSize[i]))
```

```
#Calculate deviation
param.combis$deviation<-param.combis$power.analytic-param.combis$power.simulated

g.a<-ggplot(param.combis[param.combis$count.mean<=20,],
            aes(x=power.simulated,y=power.analytic,color=count.mean))+
  geom_point()+geom_abline()+
  xlab("Power - eQTL simulation") + ylab("Power - scPower")+
  theme_bw()+
  scale_color_viridis("Mean")

g.b<-ggplot(param.combis,aes(x=as.factor(count.mean),y=deviation))+geom_boxplot()+
  xlab("Deviation of analytic power and simulated power")+xlab("Mean")+
  theme_bw()

g<-ggarrange(g.a,g.b,ncol=2,labels=c("a","b"))

print(g)
```

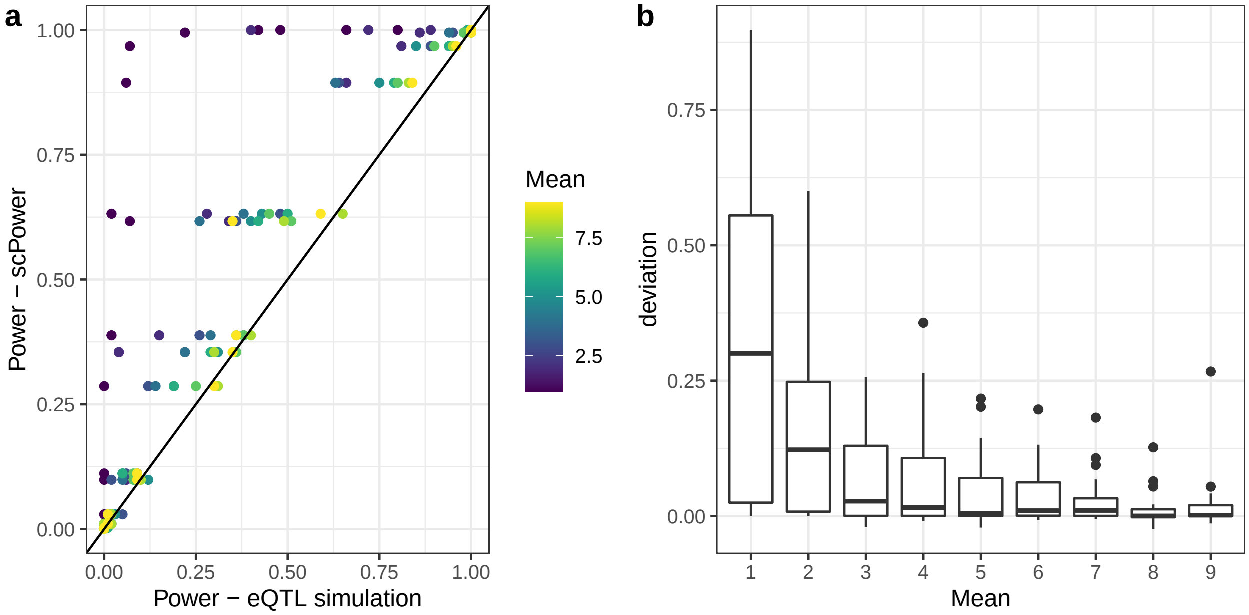


Figure S12: Relation between eQTL power and expression mean in a simulation study.

a The simulated eQTL power is compared to the analytic power calculated with scPower for a range of effect sizes (R2 between 0.1 and 0.6), sample sizes (between 20 and 200) and Bonferroni-corrected p-values (here only 0.05/(10*1,000) to speed calculations). The color coding shows the mean count value.

b The deviation between the analytic power and the simulated power is stratified by the expression mean used in the simulation. Boxplots show medians (centre lines), first and third quartiles (lower and upper box limits, respectively), 1.5-fold interquartile ranges (whisker extents) and outliers (black circles).

Figure S13: Comparison of scPower with the simulation-based methods powsimR and muscat in combination with different DE methods.

Extension of Figure 3c-e: more extensive comparison between scPower and simulation-based methods combining the simulation with different DE methods (in Figure 3 only showing the results for edgeR).

```r
#General parameters
transcriptome.mapped.reads<-25000
nGenes<-21000
ct<-"CD4 T cells"
ct.freq<-1
cellsPerLane<-20000
cellsPerCelltype<-c(200,1000,3000)
sampleSize<-c(4,8,16)
ref.study.name<-"Blueprint (CLL) iCLL-mCLL"

#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize,KEEP.OUT.ATTRS = FALSE)
colnames(param.combis)<-c("cellsPerCelltype","sampleSize")

#Filter ref study for the ones that are kept after gene filtering
ref.study<-de.ref.study[de.ref.study$name==ref.study.name,]
fitted.genes<-16197 #Hard-coded here, found in powsimR object

######################################################################
#Test results for each parameter combination with MT method FWER

overview.results<-NULL
for(i in 1:nrow(param.combis)){
  detailed<-power.general.restrictedDoublets(param.combis$sampleSize[i],
                             param.combis$cellsPerCelltype[i],
                             transcriptome.mapped.reads,
                             ct.freq, "de",de.ref.study,ref.study.name,
                             cellsPerLane,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits,ct,
                             disp.fun.param,mappingEfficiency = 1,
                             multipletRate=0,multipletFactor=1,
                             min.UMI.counts = 0,
                             perc.indiv.expr = 0,
                             nGenes=21000,
                             returnResultsDetailed = TRUE,
                             MTmethod="Bonferroni")

  temp.overview<-detailed$overview.df

  #Filter power for the expressed genes
  ref.study$not.filtered<-ref.study$rank<=min(fitted.genes,
                                              detailed$overview.df$expressedGenes)
  temp.overview$power.filtered<-mean(detailed$probs.df$power[ref.study$not.filtered])

  #Filter overall power for simulated genes
  ref.study$not.filtered<-ref.study$rank<=fitted.genes
  temp.overview$overall.power.filtered<-mean(detailed$probs.df$combined.prob[
    ref.study$not.filtered])

  overview.results<-rbind(overview.results,temp.overview)
}
```

```
overview.results$method<-"scPower"
overview.results$power.sd<-0
overview.results$overall.power.sd<-0
overview.results$expressed.genes.sd<-0
overview.results<-overview.results[,c("sampleSize","totalCells","method",
                                      "power.filtered","power.sd",
                                      "overall.power.filtered","overall.power.sd",
                                      "expressedGenes","expressed.genes.sd")]
overview.results$MT<-"FWER"

#Bonferroni corrected values
power.fwer<-scPower::simulated.DE.values[simulated.DE.values$MT=="FWER",]
colnames(overview.results)<-colnames(power.fwer)
power.fwer<-rbind(power.fwer,overview.results)
power.fwer$totalCells<-as.factor(power.fwer$totalCells)

#Format all methods as factors so that points are sorted correctly
method_ordering<-sort(unique(power.fwer$method))
#switch limma-voom and trend to match the color conding better
method_ordering<-method_ordering[c(1,2,3,4,6,7,5,8)]
power.fwer$method<-factor(power.fwer$method,levels=method_ordering)

g.a<-ggplot(power.fwer,aes(x=totalCells,y=expressed.genes,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=expressed.genes-expressed.genes.sd,
                      ymax=expressed.genes+expressed.genes.sd), width=.2,
                 position=position_dodge(.9))+
  ylab("Expressed genes")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+
  theme_bw()+
  scale_fill_manual("Method",values=method.colors)

g.b<-ggplot(data=power.fwer,aes(x=totalCells,y=power,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=power-power.sd, ymax=power+power.sd), width=.2,
                 position=position_dodge(.9))+
  ylab("DE power")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+ylim(0,1)+
  theme_bw()+
  scale_fill_manual("Method",values=method.colors)

g.c<-ggplot(data=power.fwer,aes(x=totalCells,y=overall.power,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=overall.power-overall.power.sd,
                    ymax=overall.power+overall.power.sd), width=.2,
                 position=position_dodge(.9))+
  ylab("Overall detection power")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+ylim(0,1)+
  theme_bw()+
  scale_fill_manual("Method",values=method.colors)

######################################################################
#Test results for each parameter combination with MT method FDR
```

```
overview.results<-NULL
for(i in 1:nrow(param.combis)){
  detailed<-power.general.restrictedDoublets(param.combis$sampleSize[i],
                             param.combis$cellsPerCelltype[i],
                             transcriptome.mapped.reads,
                             ct.freq, "de",de.ref.study,ref.study.name,
                             cellsPerLane,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits,ct,
                             disp.fun.param,mappingEfficiency = 1,
                             multipletRate=0,multipletFactor=1,
                             min.UMI.counts = 0,
                             perc.indiv.expr = 0,
                             nGenes=21000,
                             returnResultsDetailed = TRUE,
                             MTmethod="FDR")

  temp.overview<-detailed$overview.df

  #Filter power for the expressed genes
  ref.study$not.filtered<-ref.study$rank<=min(fitted.genes,
                                              detailed$overview.df$expressedGenes)
  temp.overview$power.filtered<-mean(detailed$probs.df$power[ref.study$not.filtered])

  #Filter overall power for simulated genes
  ref.study$not.filtered<-ref.study$rank<=fitted.genes
  temp.overview$overall.power.filtered<-mean(detailed$probs.df$combined.prob[
    ref.study$not.filtered])

  overview.results<-rbind(overview.results,temp.overview)
}

overview.results$method<-"scPower"
overview.results$power.sd<-0
overview.results$overall.power.sd<-0
overview.results$expressed.genes.sd<-0
overview.results<-overview.results[,c("sampleSize","totalCells","method",
                                      "power.filtered","power.sd",
                                      "overall.power.filtered","overall.power.sd",
                                      "expressedGenes","expressed.genes.sd")]
overview.results$MT<-"FDR"

#Bonferroni corrected values
power.fdr<-scPower::simulated.DE.values[simulated.DE.values$MT=="FDR",]
colnames(overview.results)<-colnames(power.fdr)
power.fdr<-rbind(power.fdr,overview.results)
power.fdr$totalCells<-as.factor(power.fdr$totalCells)

#Format all methods as factors so that points are sorted correctly
power.fdr$method<-factor(power.fdr$method,levels=method_ordering)

g.d<-ggplot(power.fdr,aes(x=totalCells,y=expressed.genes,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=expressed.genes-expressed.genes.sd,
                      ymax=expressed.genes+expressed.genes.sd), width=.2,
                 position=position_dodge(.9))+
  ylab("Expressed genes")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+
  theme_bw()+
  scale_fill_manual("Method",values=method.colors)

g.e<-ggplot(data=power.fdr,aes(x=totalCells,y=power,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=power-power.sd, ymax=power+power.sd), width=.2,
                 position=position_dodge(.9))+
  ylab("DE power")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+ylim(0,1)+
  theme_bw()+
  scale_fill_manual("Method",values=method.colors)

g.f<-ggplot(data=power.fdr,aes(x=totalCells,y=overall.power,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=overall.power-overall.power.sd,
                    ymax=overall.power+overall.power.sd), width=.2,
                 position=position_dodge(.9))+
  ylab("Overall detection power")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+ylim(0,1)+
  theme_bw()+
  scale_fill_manual("Method",values=method.colors)

g<-ggarrange(g.a,g.d,
             g.b,g.e,
             g.c,g.f,
             ncol=2,nrow=3,
             labels=c("a","b","c","d","e","f"),
             common.legend=TRUE,
             legend="bottom",align="hv")

print(g)
```

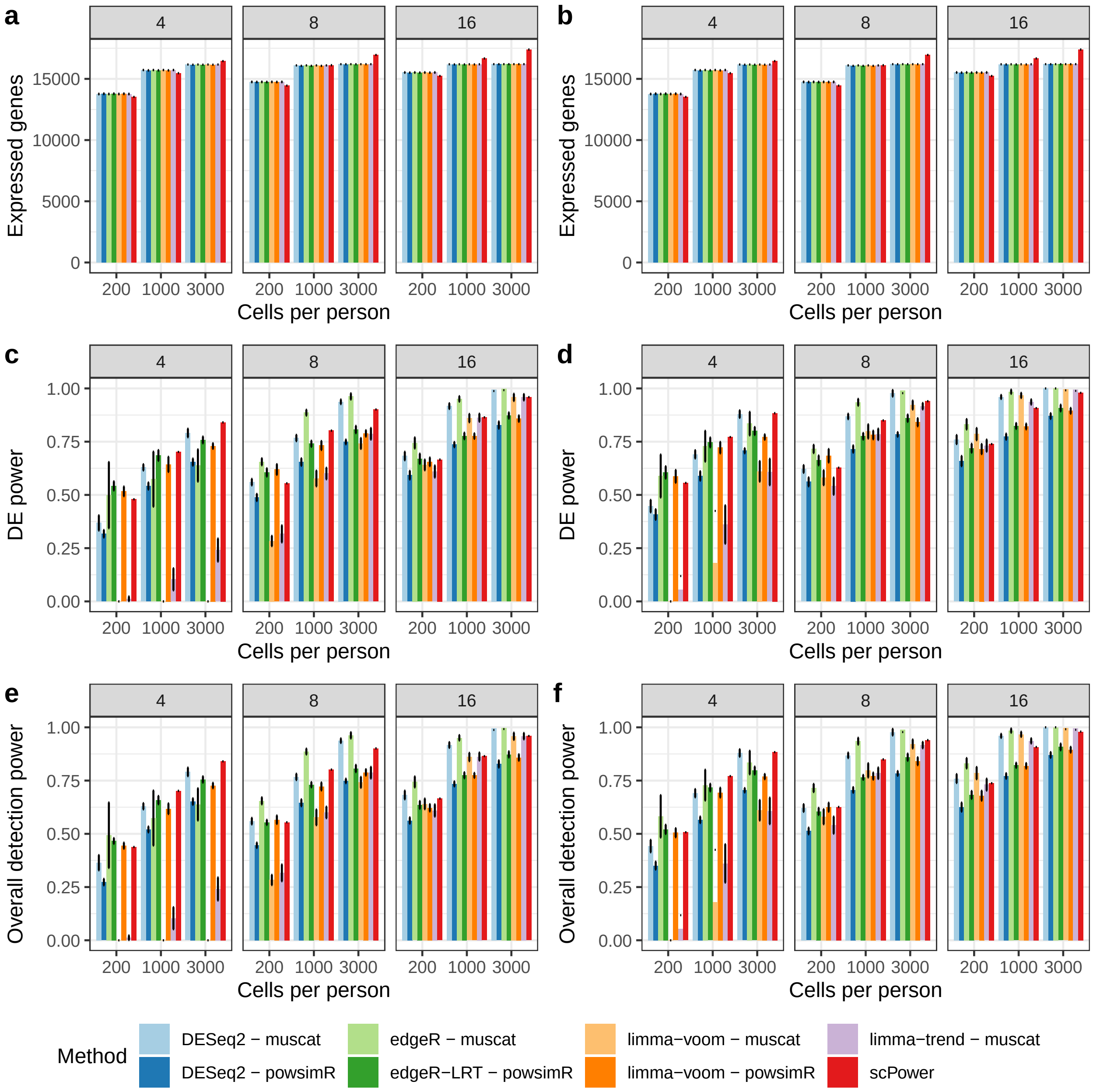


Figure S13: Comparison of scPower with the simulation-based methods powsimR and muscat in combination with different DE methods. Difference in number of expressed genes (a,b), DE power (c,d) and overall detection power (e,f) between scPower and simulations. The adapted version of powsimR (see methods) was run with DESeq2, edgeR-LRT and limma-voom, using the mean-ratio method (MR) for normalization. The adapted version of muscat (see methods) was run with DESeq2, edgeR, limma-trend and limma-voom. The power was evaluated for sample sizes of 4, 8 and 16 (see panel titles) and for 200, 1000 and 3000 cells per person (x axis). Both FWER adjusted power (a,c,e) and FDR adjusted power (b,d,f) were evaluated. The barplots represent the mean power over n=25 simulation runs of powsimR and muscat and the error bar shows the standard deviation.

Figure S14: Comparison between scPower with the simulation-based methods powsimR and muscat over a large range of experimental design

Extensing of Figure 3c-e: showing the comparison of scPower with the simulation-based methods also for FWER correction.

```
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(4,6,8,10,16,20)
```

The final plots (restricted for the parameter combinations generated in Figure S13) are generated as:

```
#Create plot showing reproduction of main figure 3

#FWER corrected threshold
power.fwer$method<-as.character(power.fwer$method)
power.fwer.edgeR<-power.fwer[startsWith(power.fwer$method,"edgeR"),]
power.fwer.scpower<-power.fwer[power.fwer$method == "scPower",]
power.fwer<-merge(power.fwer.edgeR,power.fwer.scpower,
                  by=c("sampleSize","totalCells"),
                  suffixes=c("",".scpower"))
power.fwer$variable<-ifelse(endsWith(power.fwer$method,"powsimR"),
                            "powsimR","muscat")
#Reorder variable for plotting
power.fwer$variable<-factor(power.fwer$variable,levels=c("powsimR","muscat"))

g.a<-ggplot(power.fwer,aes(x=expressed.genes,y=expressed.genes.scpower,
                           color=variable))+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  geom_point(alpha=0.7)+geom_abline()+geom_hline(yintercept = 16171)+
  xlab("Expressed genes - simulations")+ylab("Expressed genes - scPower")+
  theme_bw()

g.b<-ggplot(power.fwer,aes(x=power,y=power.scpower,
                           color=variable))+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  geom_point(alpha=0.7)+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  xlab("DE power - simulations")+ylab("DE power - scPower")+
  theme_bw()

g.c<-ggplot(power.fwer,aes(x=overall.power,y=overall.power.scpower,
                           color=variable))+
  geom_point(alpha=0.7)+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  xlab("Overall power - simulations")+
  ylab("Overall power - scPower")+
  theme_bw()

#FDR corrected thresold
power.fdr$method<-as.character(power.fdr$method)
power.fdr.edgeR<-power.fdr[startsWith(power.fdr$method,"edgeR"),]
power.fdr.scpower<-power.fdr[power.fdr$method == "scPower",]
power.fdr<-merge(power.fdr.edgeR,power.fdr.scpower,
                  by=c("sampleSize","totalCells"),
                  suffixes=c("",".scpower"))
```

```r
power.fdr$variable<-ifelse(endsWith(power.fdr$method,"powsimR"),
                            "powsimR","muscat")
#Reorder variable for plotting
power.fdr$variable<-factor(power.fdr$variable,levels=c("powsimR","muscat"))

g.d<-ggplot(power.fdr,aes(x=expressed.genes,y=expressed.genes.scpower,
                           color=variable))+
  geom_point(alpha=0.7)+geom_abline()+geom_hline(yintercept = 16171)+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  xlab("Expressed genes - simulations")+
  ylab("Expressed genes - scPower")+
  theme_bw()

g.e<-ggplot(power.fdr,aes(x=power,y=power.scpower,
                           color=variable))+
  geom_point(alpha=0.7)+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  xlab("DE power - simulations")+
  ylab("DE power - scPower")+
  theme_bw()

g.f<-ggplot(power.fdr,aes(x=overall.power,y=overall.power.scpower,
                           color=variable))+
  geom_point(alpha=0.7)+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  scale_color_discrete("Simulation method",labels=c("powsimR","muscat"))+
  xlab("Overall power - simulations")+
  ylab("Overall power - scPower")+
  theme_bw()

g<-ggarrange(g.a,g.b,g.c,
             g.d,g.e,g.f,
             ncol=3,nrow=2,
             labels=c("a","b","c","d","e","f"),align="hv",
             common.legend = TRUE,legend="bottom")

print(g)
```

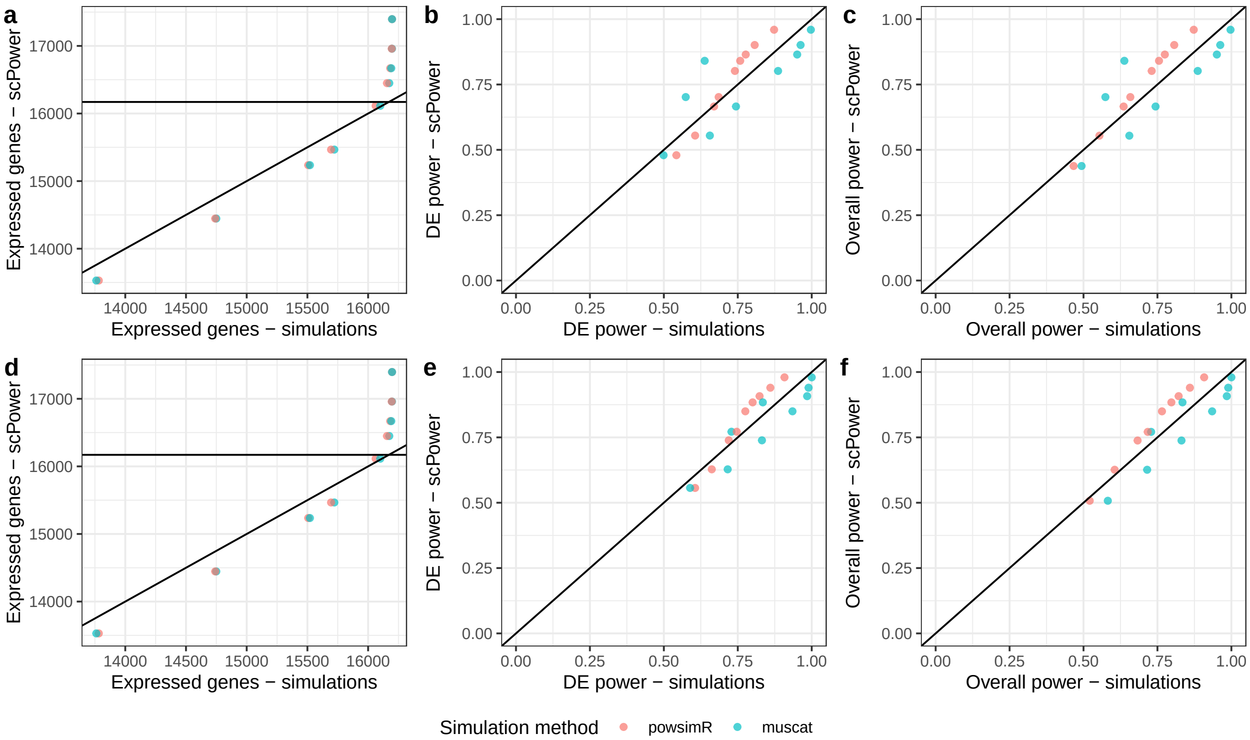


Figure S14: Comparison between scPower with the simulation-based methods powsimR and muscat over a large range of experimental design. Difference in number of expressed genes (a,d), DE power (b,e) and overall detection power (c,f) between scPower (y axis) and simulations of powsimR and muscat (x axis). The adapted version of powsimR (see Methods) was run with edgeR-LRT using the mean-ratio method (MR) for normalization and the adapted version of muscat with edgeR. The power was evaluated for all combinations of main Figure 3 (sample sizes of 4, 6, 8, 16 and 20 and cells per person of 100, 200, 500, 1000, 1500, 2000, 2500 and 3000). Both FWER adjusted p-values (a-c) and FDR adjusted p-values (d-f) were evaluated. In (a) and (d), the horizontal line represents the maximum number of possible simulated genes for powsimR .

Figure S15&S16: Comparison between scPower with the simulation-based methods with specific simulation setting (batch effects and imbalanced cell proportions)

In Figure S15, we evaluated the influence of batch effects using powsimR, and respectively, in Figure S16, the influence of imbalanced cell proportions between the DE groups using muscat. In both cases, the parameters of the simulation tools were set correspondingly, but the comparison with scPower including the plotting is analogously done as in Figure S13.

Figure S17: Variant of Main Figure 5 with FWER adjusted significance thresholds

This figure was generated with exactly the same code as Main Figures 5, only setting the multiple testing method (MTmethod) to “FWER” instead of “FDR” for the DE power.

Figure S18: Gene curve fits for different single cell technologies

The figure shows the gene expression model for a Drop-seq lung and a Smart-seq pancreas data set, corresponding to the models shown Figure 2, S5 and S6 for a 10X Genomics PBMC data set.

```r
#Load observed gene counts (precalculated)
data(precalculatedObservedGeneCounts) #observed.gene.counts

################
#Drop-seq data

#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="Dropseq_count10",]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){

  sampleSize<-1

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits.drop,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param.drop,
    min.counts=10,
    perc.indiv=0.5,
    nGenes=24000)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Replace subsampling
subsampled.names<-setNames(c("20000","15000","10000","5000"),
                           c("complete","75","50","25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]
plot.estimates$sample<-factor(plot.estimates$sample,
                              levels=c("20000","15000","10000","5000"))

#Order variable level, first expressed.genes than estimated
plot.estimates$variable<-factor(plot.estimates$variable,
                                levels=c("expressed.genes","estimated.counts"))

g.drop<-ggplot(data=plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(shape=cell.type,color=sample,fill=sample),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_fill_manual(values=col.set2)+
  scale_shape_manual("Cell type",values=seq(1,13))+
  scale_color_manual("Read depth",values=col.set2)+
  scale_linetype_discrete(name="Curve",labels=c("Observed","Estimated")) +
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8,
        legend.direction = "vertical", legend.box = "horizontal",
        legend.position="bottom")+
  guides(fill=FALSE,shape=guide_legend(ncol=3))

################
#Smart-seq data

#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="Smartseq_count10",]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){

  sampleSize<-1

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits.smart,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param.smart,
    min.counts=10,
    perc.indiv=0.5,
    nGenes=21000,
    countMethod="read")

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Replace subsampling
subsampled.names<-setNames(c("500000","375000","250000","125000"),
                           c("complete","subsampling0.75","subsampling0.5",
                             "subsampling0.25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]
plot.estimates$sample<-factor(plot.estimates$sample,
                              levels=c("500000","375000","250000","125000"))

#Order variable level, first expressed.genes than estimated
plot.estimates$variable<-factor(plot.estimates$variable,
                                levels=c("expressed.genes","estimated.counts"))

g.smart<-ggplot(data=plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(shape=cell.type,color=sample,fill=sample),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_fill_manual(values=col.set2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  scale_color_manual("Read depth",values=col.set2)+
  scale_linetype_discrete(name="Curve",labels=c("Observed","Estimated")) +
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8,
        legend.direction = "vertical", legend.box = "horizontal",
        legend.position="bottom")+
  guides(fill=FALSE)

#####
#Combine both
g<-ggarrange(g.drop,g.smart,ncol=2,labels=c("a","b"),align="hv")

print(g)
```

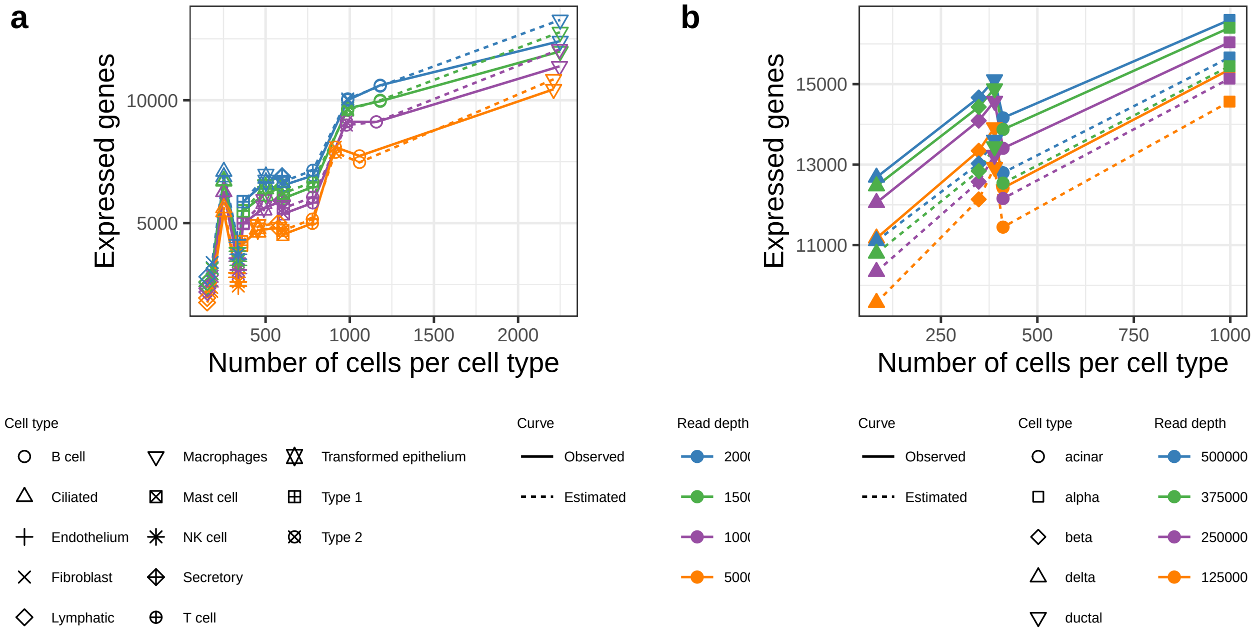


Figure S18: Gene curve fits for different single cell technologies. Evaluation of the expression probability from scPower for a lung data set measured with Drop-seq (a) and for a pancreas data set measured with Smart-seq2 (b), both subsampled to different read depths (represented by line color). The solid lines represent the observed gene curves, the dashed lines the fitted curves. The point symbol visualizes the cell type. Gene expression criteria are chosen as UMI counts > 10 in all cells for Drop-seq (a) and read counts > 10 per kilobase transcript in all cells for Smart-Seq2 (b).

Figure S19: Comparison between powsimR and scPower for Drop-Seq and Smart-seq

Same as in Figure S14, only run for Drop-Seq and Smart-Seq data.

Figure S20: Optimal parameters for varying budgets and Drop-seq and Smart-seq data

The same evaluation is done in Figure 5 than in Figure 4, only this time with priors of Drop-seq lung data and Smart-seq pancreas data instead of 10X Genomics PBMC data. Similar to Figure 4, the code for the calculation is given, but set to eval=FALSE due to long runtime. Instead, precalculated values are taken for the plotting.

```
#Drop-seq parameters

#Cell type
ct<-"NK cell"
ct.freq<-0.2

cellsPerChanel<-NA
libPrepCell<-0.09
costFlowCell<-14032
readsPerFlowcell<-4100*10^6

readDepth<-c(2000,5000,10000,15000,20000,30000,40000,50000,60000,70000,100000)
cellsPerPerson<-c(100,200,500,700,seq(1000,9000,500))

mapping.efficiency<-0.8
nFittedGenes<-24000

#Model growth rate constant
multipletRateGrowth<-"constant"
multipletRate<-0.05

#Number of DE genes to be modeled
numGenes<-250
```

```
#####
#Simulated DE studies - lung (drop-seq)

simulated.parameters<-data.frame(mean.value=c(2,0.5,2,0.5),
                                 rank.max=c(20000,20000,10000,10000),
                                 name=c("highES_unifRank","lowES_unifRank",
                                        "highES_highRank","lowES_highRank"))
```

```
print(simulated.parameters)

de.raw.values.lung<-NULL
for(s in 1:nrow(simulated.parameters)){

  print(s)

  #Simulate prior for expression rank
  ranks<-uniform.ranks.interval(start=1,end=simulated.parameters$rank.max[s],
                                numGenes=numGenes)

  #Simulate prior for effect sizes
  set.seed(1)
  effectSizes<-effectSize.DE.simulation(mean=simulated.parameters$mean.value[s],
                                        sd=1, numGenes=numGenes)

  sample.DE.genes<-data.frame(ranks=ranks, FoldChange=effectSizes,
                              name="Simulated")

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.libPrepCell(i,"de",ct,ct.freq,
                             libPrepCell,costFlowCell,readsPerFlowcell,
                             sample.DE.genes,"Simulated",
                             cellsPerChanel,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits.drop,
                             disp.fun.param.drop,
                             nCellsRange = cellsPerPerson,
                             readDepthRange = readDepth,
                             mappingEfficiency = mapping.efficiency,
                             multipletRate = multipletRate,
                             min.UMI.counts = minUMI,
                             perc.indiv.expr = perc.indiv,
                             nGenes=nFittedGenes,
                             multipletRateGrowth = multipletRateGrowth,
                             MTmethod = "FDR",
                             speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }
  max.values$budget<-budget
  max.values$parameters<-simulated.parameters$name[s]

  de.raw.values.lung<-rbind(de.raw.values.lung,max.values)
}

#####
#Observed DE studies - lung (drop-seq)

de.priors.lung<-de.ref.study[de.ref.study$type=="lung",]
de.real.values.lung<-NULL
for(studyName in unique(de.priors.lung$name)){

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.libPrepCell(i,"de",ct,ct.freq,
                           libPrepCell,costFlowCell,readsPerFlowcell,
                           de.priors.lung,studyName,
                           cellsPerChanel,read.umi.fit[
                             read.umi.fit$type=="10X_PBMC_1",],
                           gamma.mixed.fits.drop,
                           disp.fun.param.drop,
                           nCellsRange = cellsPerPerson,
                           readDepthRange = readDepth,
                           mappingEfficiency = mapping.efficiency,
                           multipletRate = multipletRate,
                           min.UMI.counts = minUMI, perc.indiv.expr = perc.indiv,
                           nGenes=nFittedGenes,
                           multipletRateGrowth = multipletRateGrowth,
                           MTmethod = "FDR",
                           speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }

  max.values$budget<-budget
  max.values$parameters<-studyName
  de.real.values.lung<-rbind(de.real.values.lung,max.values)
}
```

```
#Smart-seq parameters

#Parameters
readDepth.smart<-c(20000,50000,seq(100000,1000000,100000),1500000,2000000)
cellsPerPerson.smart<-c(50,100,200,500,700,seq(1000,9000,500))

mapping.efficiency<-0.8
min.norm.count<-minUMI
perc.indiv.expr<-perc.indiv

ct<-"alpha"
ct.freq<-0.2

prepCostCell<-13
costFlowCell<-14032
readsPerFlowcell<-4100*10^6

nFittedGenes<-22000

#Number DE genes to be modeled
numGenes<-250
```

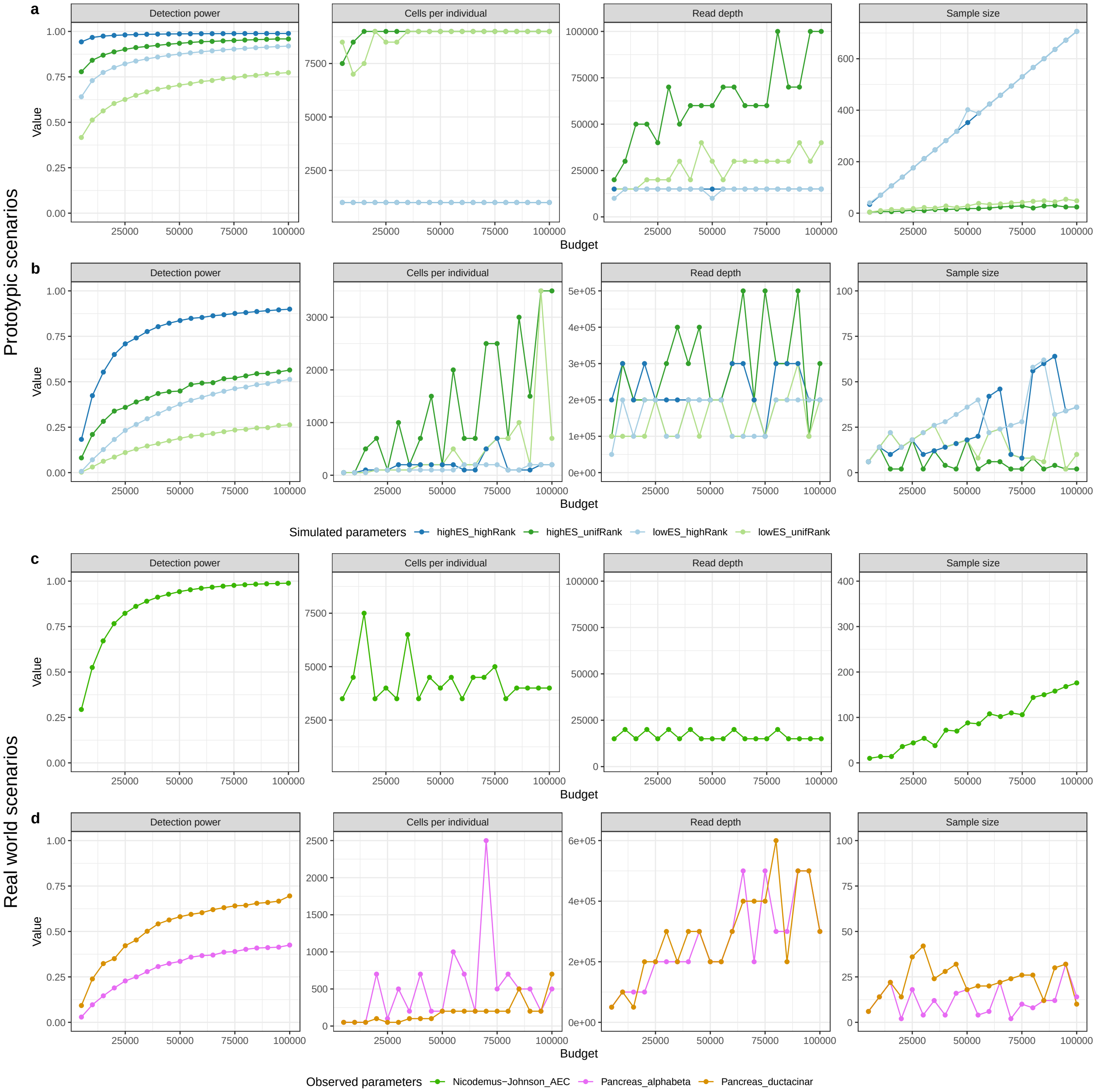
```r
#####
#Simulated DE studies - pancreas (smart-seq)

simulated.parameters<-data.frame(mean.value=c(2,0.5,2,0.5),
                                 rank.max=c(20000,20000,10000,10000),
                                 name=c("highES_unifRank","lowES_unifRank",
                                        "highES_highRank","lowES_highRank"))

print(simulated.parameters)

de.raw.values.panc<-NULL
for(s in 1:nrow(simulated.parameters)){

  print(s)

  #Simulate prior for expression rank
  ranks<-uniform.ranks.interval(start=1,end=simulated.parameters$rank.max[s],
                                numGenes=numGenes)

  #Simulate prior for effect sizes
  set.seed(1)
  effectSizes<-effectSize.DE.simulation(mean=simulated.parameters$mean.value[s],
                                        sd=1,numGenes=numGenes)

  sample.DE.genes<-data.frame(ranks=ranks, FoldChange=effectSizes,
                              geneLength=5000, name="Simulated")

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.smartseq(i,"de",ct,ct.freq,
                                         prepCostCell,costFlowCell,
                                         readsPerFlowcell,
                                         sample.DE.genes,"Simulated",
                                         gamma.mixed.fits.smart,
                                         disp.fun.param.smart,
                                         nCellsRange = cellsPerPerson.smart,
                                         readDepthRange = readDepth.smart,
                                         mappingEfficiency = mapping.efficiency,
                                         min.norm.count = min.norm.count,
                                         perc.indiv.expr = perc.indiv.expr,
                                         nGenes=nFittedGenes,
                                         MTmethod = "FDR",
                                         speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }
  max.values$budget<-budget
  max.values$parameters<-simulated.parameters$name[s]

  de.raw.values.panc<-rbind(de.raw.values.panc,max.values)
}
```

```
#####
#Observed DE studies - pancreas (smart-seq)
de.priors.panc<-de.ref.study[de.ref.study$type=="pancreas",]

de.real.values.panc<-NULL
for(studyName in unique(de.priors.panc$name)){

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.smartseq(i,"de",ct,ct.freq,
                                   prepCostCell,costFlowCell,readsPerFlowcell,
                                   de.priors.panc,studyName,
                                   gamma.mixed.fits.smart,
                                   disp.fun.param.smart,
                                   nCellsRange = cellsPerPerson.smart,
                                   readDepthRange = readDepth.smart,
                                   mappingEfficiency = mapping.efficiency,
                                   min.norm.count = min.norm.count,
                                   perc.indiv.expr = perc.indiv.expr,
                                   nGenes=nFittedGenes,
                                   MTmethod = "FDR",
                                   speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }

  max.values$budget<-budget
  max.values$parameters<-studyName
  de.real.values.panc<-rbind(de.real.values.panc,max.values)
}
```

```
#Get precalculated values
data(precalculatedBudgetOptim)
de.raw.values.lung<-budget.optimization[budget.optimization$type=="DE_simulated_lung",]
de.raw.values.panc<-budget.optimization[budget.optimization$type=="DE_simulated_pancreas",]
de.real.values.lung<-budget.optimization[budget.optimization$type=="DE_observed_lung",]
de.real.values.panc<-budget.optimization[budget.optimization$type=="DE_observed_pancreas",]

dummy.plot.smart<-data.frame(budget=c(min(budget),max(budget),min(budget),max(budget),
                                min(budget),max(budget),min(budget),max(budget)),
                       variable=c("powerDetect","powerDetect","readDepth","readDepth",
                                  "totalCells","totalCells","sampleSize","sampleSize"),
                       value=c(0,1,0,500000,
                               min(cellsPerPerson.smart),1000,0,100))

#DE lung simulated values
plot.values<-de.raw.values.lung[,c("budget","powerDetect","totalCells","readDepth",
                                   "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
```

```
g.lung<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Simulated parameters",values=col.paired[c(2,4,1,3)])+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#DE pancreas simulated values
plot.values<-de.raw.values.panc[,c("budget","powerDetect","totalCells","readDepth",
                                   "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot.smart
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.panc<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Simulated parameters",values=col.paired[c(2,4,1,3)])+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#Merge values from eQTL study and DE study
all.studies<-c(unique(de.real.values.lung$parameters),
               unique(de.real.values.panc$parameters))

#De lung real values
plot.values<-de.real.values.lung[,c("budget","powerDetect","totalCells","readDepth",
                                    "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-de.real.values.lung$parameters[1]
g.lung.2<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Observed parameters",values=col.set.new[5:10],
                     labels=all.studies,limits=all.studies)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#De pancreas real values
plot.values<-de.real.values.panc[,c("budget","powerDetect","totalCells","readDepth",
                                    "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot.smart
dummy.plot.temp$parameters<-de.real.values.panc$parameters[1]
g.panc.2<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Observed parameters",values=col.set.new[5:10],
                     labels=all.studies,limits=all.studies)+
```

```r
  theme_bw()+
  xlab("Budget")+ylab("Value")

g.proto<-ggarrange(g.lung,g.panc,nrow=2,common.legend = TRUE,
                   legend="bottom",labels=c("a","b"))

g.real<-ggarrange(g.lung.2,g.panc.2,nrow=2,common.legend = TRUE,
                  legend="bottom",labels=c("c","d"))

row1 <- ggplot() + annotate(geom = 'text', x=1, y=1,
                            label="Prototypic scenarios",
                            size=6, angle = 90) + theme_void()

row2 <- ggplot() + annotate(geom = 'text', x=1, y=1,
                            label="Real world scenarios",
                            size=6,angle = 90) + theme_void()

g<-ggarrange(row1,g.proto,row2,g.real,nrow=2,ncol=2,
             widths=c(0.03,1))

print(g)
```


a
b
c
d
Prototypic scenarios
Real world scenarios
Detection power
Cells per individual
Read depth
Sample size
Value
Budget
1.00
0.75
0.50
0.25
0.00
25000
50000
75000
100000
7500
5000
2500
3000
2000
1000
0
600
400
200
5e+05
4e+05
3e+05
2e+05
1e+05
0e+00
6e+05
4e+05
2e+05
100
75
50
25
300
1500
500
Simulated parameters
highES_highRank
highES_unifRank
lowES_highRank
lowES_unifRank
Observed parameters
Nicodemus−Johnson_AEC
Pancreas_alphabeta
Pancreas_ductacinar