Looking to clean your data?
Learn how to Remove Unwanted Variation with R

Brisbane useR! Conference 2018

Docker

Everything in this talk available at:

https://hub.docker.com/r/johanngb/ruv (https://hub.docker.com/r/johanngb/ruv)

Command:

```
docker run -it --rm -p 8000:8000 -p 8888:8888 -p 3838:3838 -p 3840:3840
johanngb/ruv:useR2018
```

Github

Additional code, data and workbooks are available at:

https://github.com/johanngb/ruv-useR2018 (https://github.com/johanngb/ruv-useR2018)

Outline

1. Example Dataset
2. The RUV Framework
3. The `ruv` Package
4. Examples with Shiny

Example: Gender in the Brain

Vawter, et al. *Neuropsychopharmacology* (2004)

Goal: Discover genes differentially expressed in the brains of men and women

- 5 men, 5 women
- Tissue taken from:
    1. Anterior Cingulate Cortex
    2. Dorsolateral Prefrontal Cortex
    3. Cerebellum
- Samples sent to:
    1. UC Davis
    2. UC Irvine
    3. University of Michigan
- Samples assayed by microarray. 12,600 genes.

The Data

```
In [1]: load("gender.rda")
        ls()
        # Y.raw:  Summarized by RMA, but otherwise not preprocessed
        # Y.norm: Background corrected and quantile normalized
```

```
'Y.norm'   'Y.raw'   'geneinfo'   'sampleinfo'
```

```
In [2]: Y = Y.norm
        Y[1:5, 1:5]
```

	1000_at	1001_at	1002_f_at	1003_s_at	1004_at
01_a_D_f_2.CEL	9.823395	6.258064	5.119432	7.053562	7.358204
01_a_I_f_2.CEL	9.598368	6.382745	5.052340	7.530220	7.244545
01_a_M_f_1.CEL	9.270307	5.633953	4.765587	7.695161	7.466540
01_c_D_f_1.CEL	8.180496	5.162317	4.653400	7.142755	6.679110
01_c_I_f_2.CEL	9.352611	6.569988	4.958501	7.460245	6.935908

```
In [3]: head(sampleinfo)
```

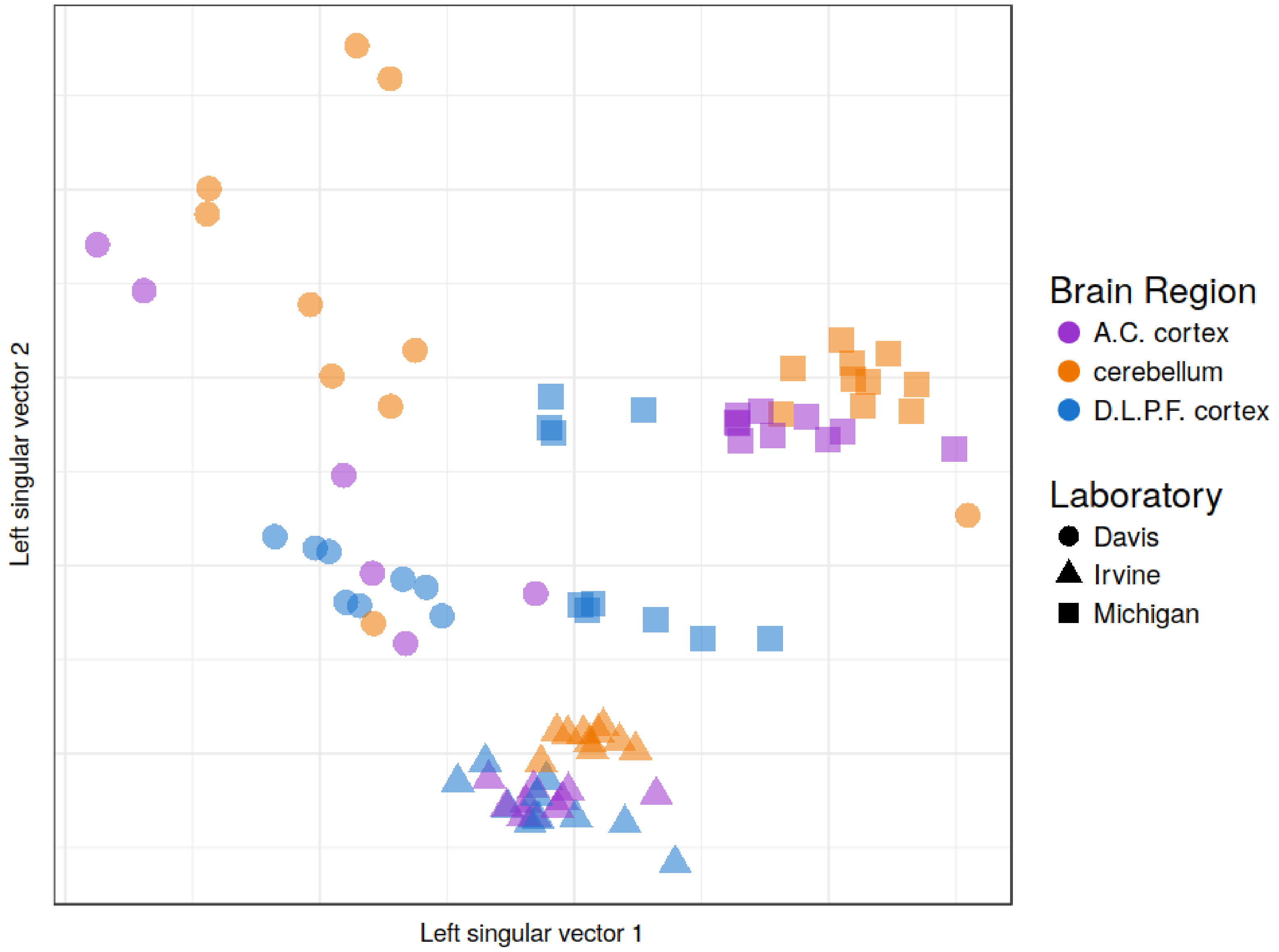
	patient	gender	region	lab	chip.version
01_a_D_f_2.CEL	patient_01	female	A.C. cortex	Davis	v2
01_a_I_f_2.CEL	patient_01	female	A.C. cortex	Irvine	v2
01_a_M_f_1.CEL	patient_01	female	A.C. cortex	Michigan	v1
01_c_D_f_1.CEL	patient_01	female	cerebellum	Davis	v1
01_c_I_f_2.CEL	patient_01	female	cerebellum	Irvine	v2
01_c_M_f_1.CEL	patient_01	female	cerebellum	Michigan	v1

```
In [4]: head(geneinfo)
```

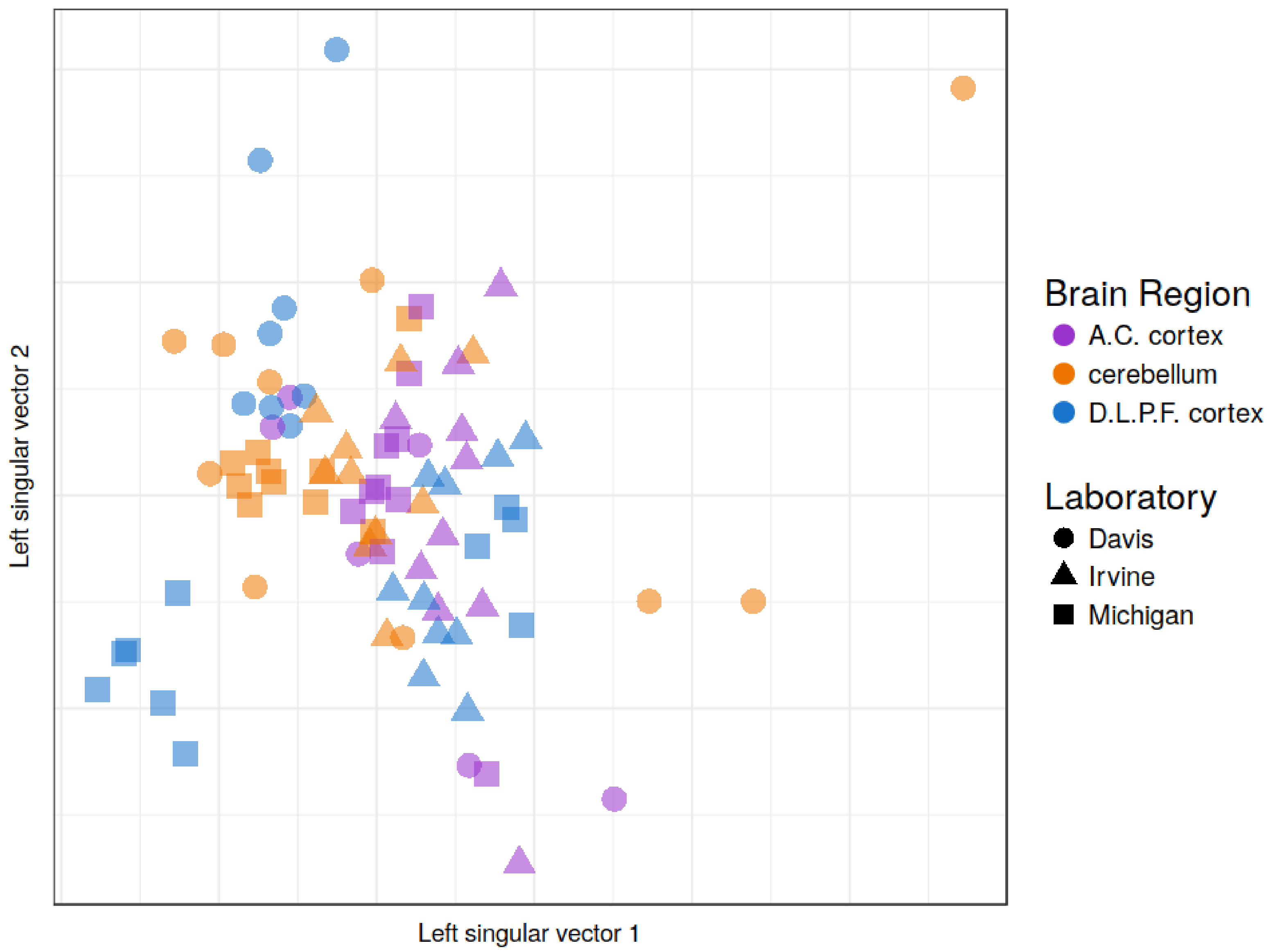
	genetype	sym	chrom	hkctl	spikectl	pctl
1000_at	other	MAPK3	16	FALSE	FALSE	FALSE
1001_at	other	TIE1	1	FALSE	FALSE	FALSE
1002_f_at	other	CYP2C19	10	FALSE	FALSE	FALSE
1003_s_at	other	CXCR5	11	FALSE	FALSE	FALSE
1004_at	other	CXCR5	11	FALSE	FALSE	FALSE
1005_at	other	DUSP1	5	FALSE	FALSE	FALSE

```
In [5]: # Load the ruv package
        library(ruv)
        # Graphics
        library(ggplot2)
        library(gridExtra)
        gg_additions = list(aes(color=sampleinfo$region,
                                shape=sampleinfo$lab,
                                size=5, alpha=.7),
                            labs(color="Brain Region",
                                 shape="Laboratory"),
                            scale_size_identity(guide="none"),
                            scale_alpha(guide="none"),
                            theme(legend.text=element_text(size=12),
                                  legend.title=element_text(size=16)),
                            guides(color = guide_legend(override.aes = list(size = 4)),
                                   shape = guide_legend(override.aes = list(size = 4
        ))),
                            scale_color_manual(values=c("darkorchid3", "darkorange2",
        "dodgerblue3"))
                           )
        options(repr.plot.width=8, repr.plot.height=6)
```

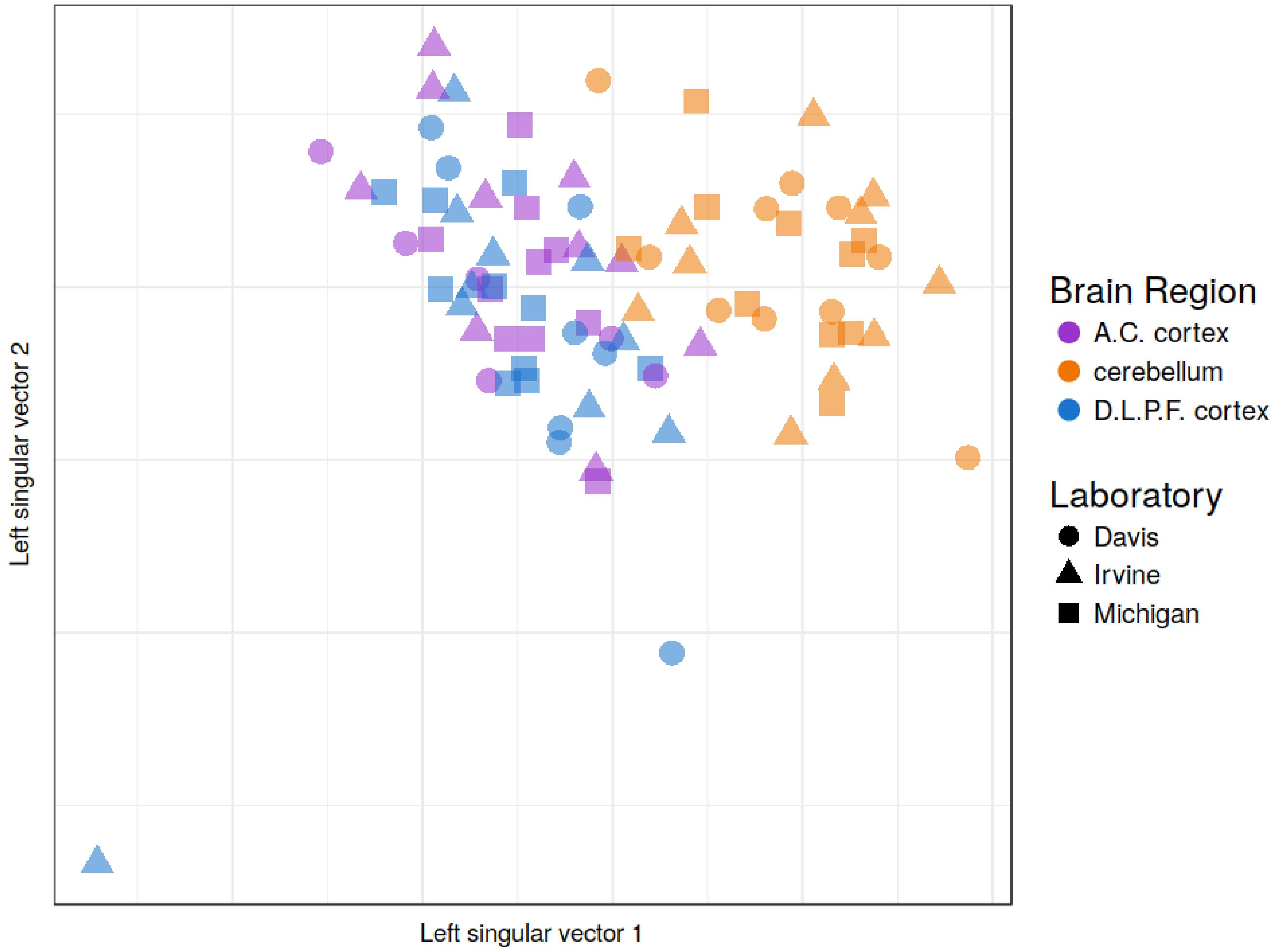
```
In [6]: ruv_svdplot(Y) + gg_additions # Technical note: centers columns by default
```



```
In [7]: ruv_svdplot(residop(scale(Y,scale=FALSE), svd(scale(Y,scale=FALSE))$u[,1:5])) +
         gg_additions
```



```
In [8]: ruv_svdplot(RUVIII(Y, replicate.matrix(sampleinfo[,c("patient", "region")]), ge
        neinfo$spikectl, k=10)) + gg_additions
```



The RUV Framework

Mini-Outline

1. Model
2. **Negative Controls**
3. Secondary Identifying Assumptions
   (Emphasis on **replicates**)

Model

$$Y_{m\times n} = X_{m\times p}\beta_{p\times n} + W_{m\times k}\alpha_{k\times n} + \epsilon_{m\times n}$$

Symbol	**Meaning**	**Example**
Y	Observed data	Microarray expression data
m	Number of Observations	84
n	Number of features (genes)	12600
X	Factor(s) of interest	gender, brain region
W	Unwanted factors	reagent quality
β, α	coefficients	
ϵ	random error	

Model due to Leek and Storey (2007) and Stegle, et al (2008)

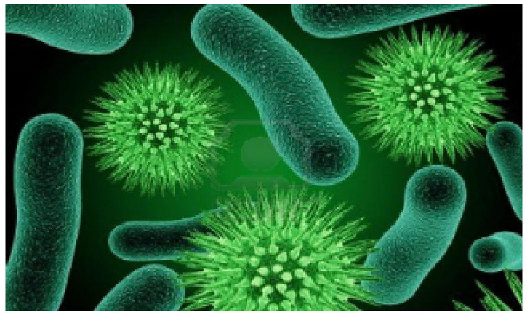
Negative Controls

Features (genes) that:

- Are unaffected by the factor of interest
- Are affected by the unwanted factors

Spike-in Controls

- Not all 12,600 “probes” measure human RNA
  33 probes measure bacterial RNA



- Bacterial RNA is spiked in at fixed, known quantities

- Used as a quality check

Negative Controls

Suppose n_c of the n columns of Y are negative controls.

Let Y_c be the $m \times n_c$ submatrix of Y containing only the columns of the negative controls.

Define β_c, α_c, and ϵ_c similarly.

$$Y_c = X\beta_c + W\alpha_c + \epsilon_c$$

Negative Controls

Suppose n_c of the n columns of Y are negative controls.

Let Y_c be the $m \times n_c$ submatrix of Y containing only the columns of the negative controls.

Define β_c, α_c, and ϵ_c similarly.

$$Y_c = X \underbrace{\beta_c}_{\text{Equal to 0 by assumption!}} + W\alpha_c + \epsilon_c$$

Negative Controls

Suppose n_c of the n columns of Y are negative controls.

Let Y_c be the $m \times n_c$ submatrix of Y containing only the columns of the negative controls.

Define β_c, α_c, and ϵ_c similarly.

$$Y_c = \cancel{X\beta_c} + W\alpha_c + \epsilon_c$$

Equal to 0 by assumption!

Negative Controls

Suppose n_c of the n columns of Y are negative controls.

Let Y_c be the $m \times n_c$ submatrix of Y containing only the columns of the negative controls.

Define β_c, α_c, and ϵ_c similarly.

$$Y_c = W\alpha_c + \epsilon_c$$

Negative Controls

Suppose n_c of the n columns of Y are negative controls.

Let Y_c be the $m \times n_c$ submatrix of Y containing only the columns of the negative controls.

Define β_c, α_c, and ϵ_c similarly.

$$Y_c = W\alpha_c + \epsilon_c$$

We can estimate *W* by factor analysis!

The Negative Control Assumption

The negative control assumption:

$$\beta_c = 0$$

Important:

- We do ***not*** assume the negative controls are uncorrelated with X
- Negative controls cannot be "discovered" *de novo*

Note:

- Negative controls need not be perfect
- Often, "sparse" or even "zero on average" will do.

Secondary Identifying Assumptions

Negative controls can identify W.

How to identify α?

"Secondary identifying assumptions":

- Have known factor of interest (X)
- Assume $X \perp W$
- Have gene-wise covariates
- Have replicates

Replicates

Suppose Y_1 and Y_2 are replicates. Then

$$Y_1 = X_1\beta + W_1\alpha + \epsilon_1$$

and

$$Y_2 = X_2\beta + W_2\alpha + \epsilon_2$$

where

$$X_1 = X_2$$

Replicates

Then:

$$Y_2 - Y_1 = (X_2 - X_1)\beta + (W_2 - W_1)\alpha + \epsilon_2 - \epsilon_1$$

Replicates

Then:

$$Y_2 - Y_1 = \underbrace{(X_2 - X_1)}_{= 0}\beta + (W_2 - W_1)\alpha + \epsilon_2 - \epsilon_1$$

Replicates

Then:

$$Y_2 - Y_1 = \underbrace{\cancel{(X_2 - X_1)}}_{= 0}\beta + (W_2 - W_1)\alpha + \epsilon_2 - \epsilon_1$$

Replicates

Then:

$$Y_2 - Y_1 = (W_2 - W_1)\alpha + \epsilon_2 - \epsilon_1$$

Replicates

Then:

$$Y_2 - Y_1 = (W_2 - W_1)\alpha + \epsilon_2 - \epsilon_1$$

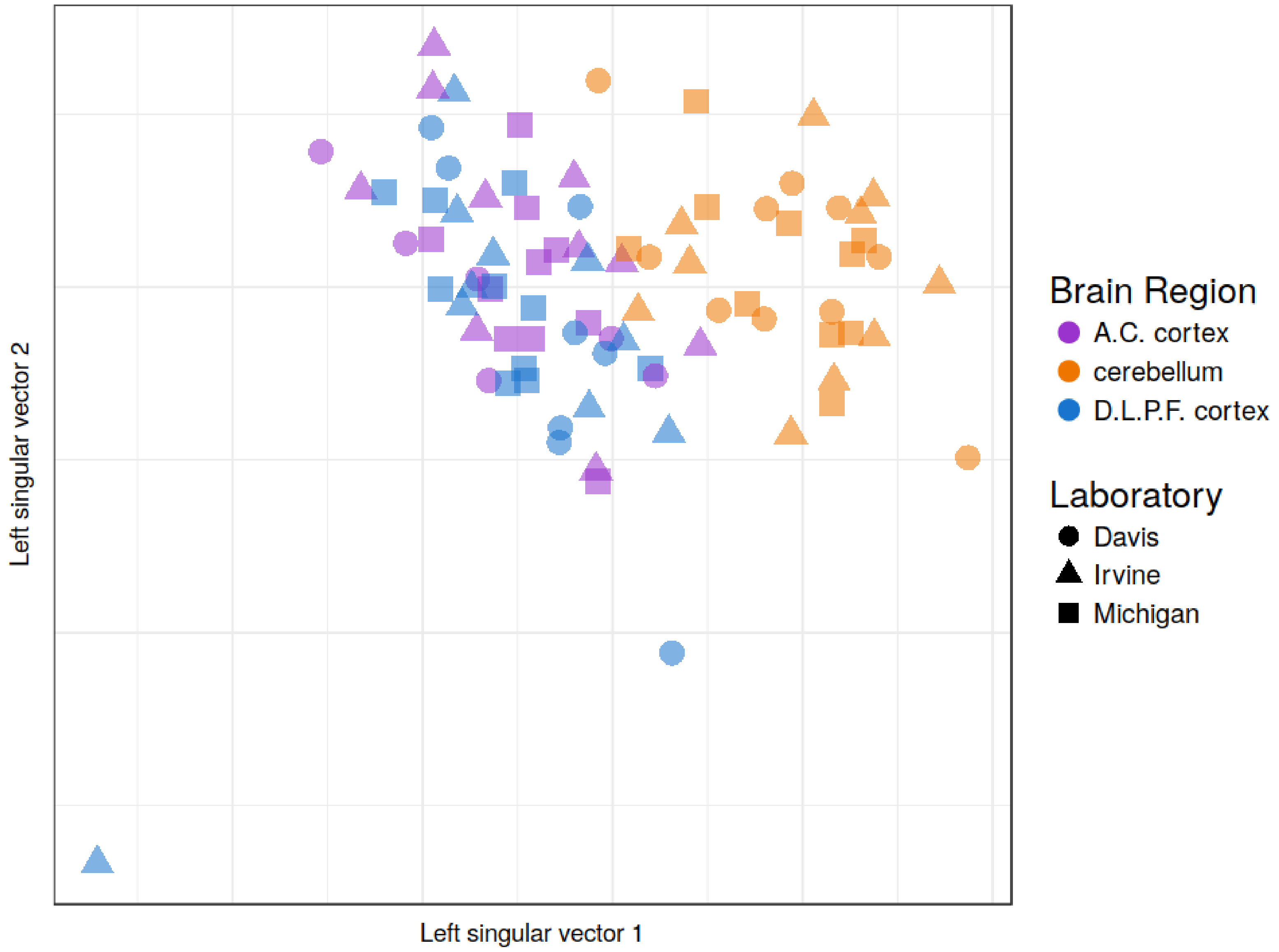
Can be estimated with factor analysis!

Negative Controls vs Replicates

Compare:

- Negative controls:
    - Some genes where $\beta = 0$
    - Allows identification of W
- Replicates:
    - Some array differences where $X = 0$
    - Allows identification of α

```
In [9]: ruv_svdplot(RUVIII(Y, replicate.matrix(sampleinfo[,c("patient", "region")]), ge
        neinfo$spikectl, k=10)) + gg_additions
```



The RUV Packages

Our focus today:

"`ruv`" on CRAN

Other RUV packages:

"`RUVnormalize`" on Bioconductor

"`RUVseq`" on Bioconductor

Mini-Outline

1. Overview of Functions
2. Regression Methods
    a. Description of Arguments
    b. Comparison of Methods
    c. Example Analysis
3. Global Adjustments
    a. RUVI
    b. RUVIII
    c. Example analyses

And then on to the case studies...

Overview of Functions

RUV Methods

Global Adjustments	Regression Methods
`RUVI`	`RUV2`
`RUVIII`	`RUV4`
	`RUVinv`
	`RUVrinv`

Helper Functions

For RUVIII	For Regression Methods	Other
`replicate.matrix`	`ruv_summary`	`design.matrix`
`collapse.replicates`	`ruv_residuals`	`residop`
	`variance_adjust`	

Plot Functions

General	For Regression Methods
`ruv_cancorplot`	`ruv_hist`
`ruv_rle`	`ruv_ecdf`
`ruv_svdplot`	`ruv_rankplot`
`ruv_svdgridplot`	`ruv_projectionplot`
`ruv_scree`	`ruv_volcano`
	`ruv_varianceplot`

Typically Not Used

getK	(Possibly useful for RUV4)
get_empirical_variances	
google_search	
inputcheck1	
invvar	
projectionplotvariables	
randinvvar	
sigmashrink	

Shiny App

`ruv_shiny`

Regression Methods

Regression Methods

Common syntax:

```
RUV2    (Y, X, ctl, k, Z = 1, eta = NULL             )
RUV4    (Y, X, ctl, k, Z = 1, eta = NULL             )
RUVinv  (Y, X, ctl,    Z = 1, eta = NULL             )
RUVrinv (Y, X, ctl,    Z = 1, eta = NULL, lambda=NULL)
```

Regression Methods

Function Arguments:

Argument	Meaning	Example	Data Type	Notes
Y	Expression data		Matrix	row = sample, column = gene
X	Factor of interest	gender	matrix, factor, vector, or data frame	
ctl	Neg. Controls	spike-ins	index (logical or integer vector)	
Z	array-wise covariates	batch	matrix, factor, vector, or data frame	1 for intercept
eta	gene-wise covariates	GC content	matrix, factor, vector, or data frame	1 for intercept
k	# of unwanted factors		integer	0 for no adjustment
lambda	ridge parameter		numeric	NULL for sensible default

Y

- Expression Data

- $m \times n$ matrix, where
    - m is the number arrays
    - n is the number of genes

- Should be log transformed

- Often best **not** to preprocess (quantile normalize, etc.)

X

- Factor of interest (gender, brain region, etc.)
- Should **not** include the intercept
- Rule of thumb: "The fewer factors, the better"
  - More factors in $X \implies$ fewer factors estimated in $\hat{W}$
  - Better to repeat analysis for each factor of interest separately

Z

- Additional covariates (batch, etc.)
- Should include the intercept (if desired)
- Rule of thumb: "The fewer factors, the better"
  - More factors in $Z \implies$ fewer factors estimated in $\hat{W}$
  - $\hat{W}$ often captures unwanted variation better than Z
  - Exception: Z is a factor that affects only a small number of genes, and likely the same genes as X.
    Example: X is a disease that affects a small number of genes; Z is a drug that affects those same genes

eta (η)

- Gene-wise covariates ***associated with unwanted factors*** (GC content, etc.)
- Included for convenience; equivalent to preprocessing by

```
Y = RUVI(Y, eta, ctl)
```

- eta = 1 (for intercept) typically recommended, but **not** default

ctl

- Crucial to success

- Ideally:
  - Unaffected by factor of interest
  - Affected by unwanted factors
  - "representative" of other genes
    (similar range of expressions, not affected by their own unwanted factors, etc.)

- **Cannot be automatically "discovered" from the data**
  (at least not naively)

- **Need not be perfect**
  RUV methods are robust (to varying degrees, and in different ways)

k

- Number of unwanted factors. For RUV2 and RUV4 only.
- Useful when negative controls may contain biology.
  Keeping k small reduces risk of overadjusting.
- Best chosen "by hand".
  ("getK" function not ideal)

Comparison of Regression Methods

Method	Strengths	Weaknesses	Notes
RUV2	• Simple and Interpretable • Not too sensitive to "nonrepresentative" NCs	• Sensitive to misspecified NCs	• Good for spike-in controls • Keep k small if NCs may be misspecified
RUV4	• Robust to misspecified NCs	• Sensitive to "nonrepresentative" NCs • Anti-conservaitve for large k	• RUV(r)inv usually a better option • Good when NCs highly misspecified; keep k small
RUVinv	• Robust to misspecified NCs • No tuning parameter • Well calibrated p-values	• Requires large number of NCs • Somewhat sensitive to "nonrepresentative" NCs	
RUVrinv	• Robust to misspecified NCs • Reasonable default for lambda	• Somewhat sensitive to "nonrepresentative" NCs	• Good compromise of features

Technical Note

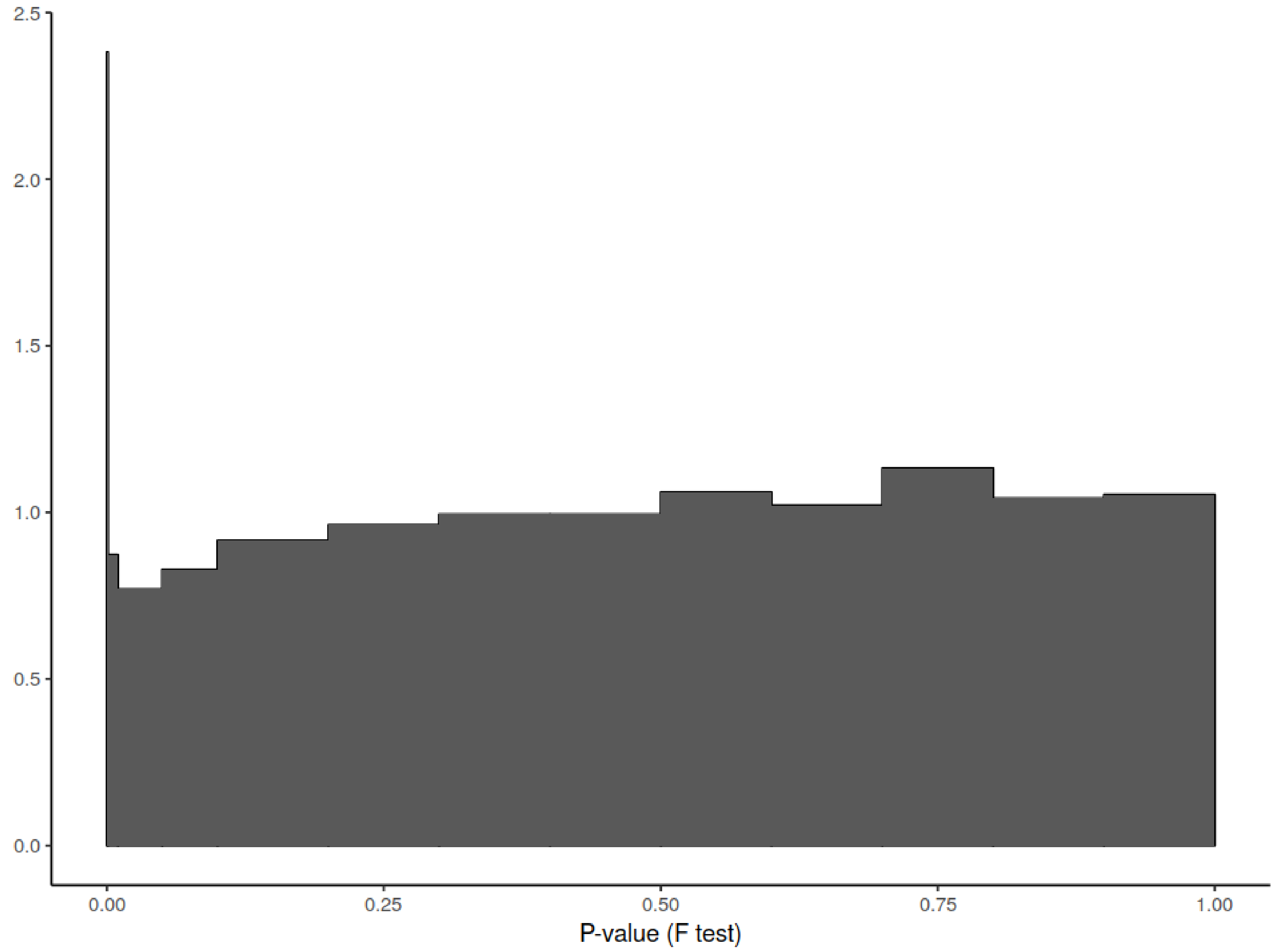
- RUV2 requires
$$\beta_c = 0$$
- RUV4, RUVinv, and RUVrinv require
$$\beta_c \alpha_c' (\alpha_c \alpha_c')^{-1} \approx 0$$

Example Analysis

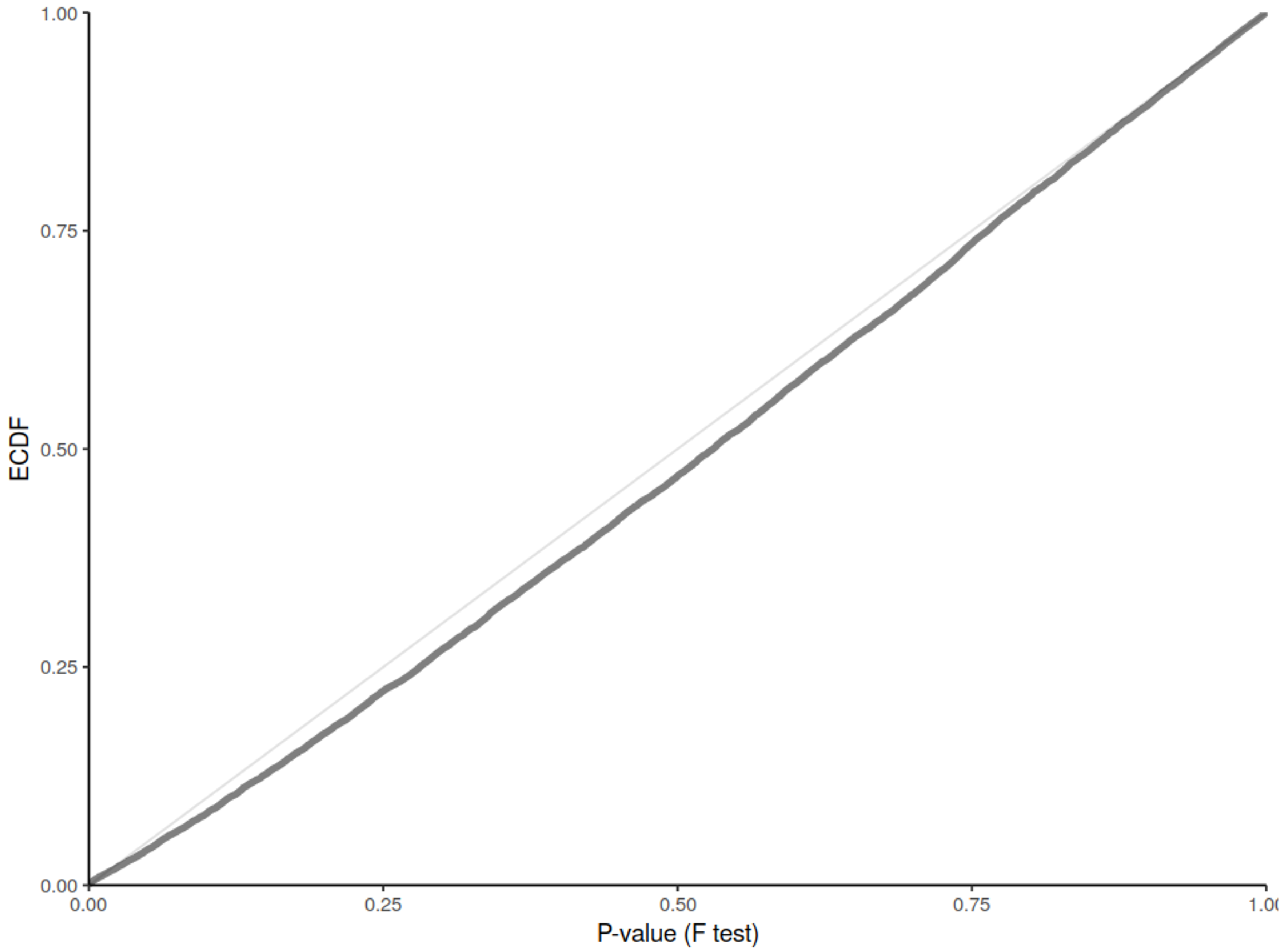
In [10]:
```
fit = RUVrinv(Y, sampleinfo$gender, geneinfo$spikectl)
fit.summary = ruv_summary(Y, fit, sampleinfo, geneinfo)
head(fit.summary$C)
```

	F.p	F.p.BH	p_X1.male	p.BH_X1.male	b_X1.male	s
41214_at	1.000000e-24	1.000000e-24	1.000000e-24	1.000000e-24	2.6714931	(
37583_at	1.000000e-24	1.321247e-23	1.000000e-24	1.321247e-23	0.7328333	(
38355_at	1.000000e-24	2.757869e-21	1.000000e-24	2.757869e-21	2.0265366	(
38446_at	4.418298e-18	1.391764e-14	4.418298e-18	1.391764e-14	-0.9738340	(
35885_at	7.721272e-16	1.945760e-12	7.721272e-16	1.945760e-12	0.7532324	(
34477_at	2.796912e-12	5.873516e-09	2.796912e-12	5.873516e-09	0.5352809	(

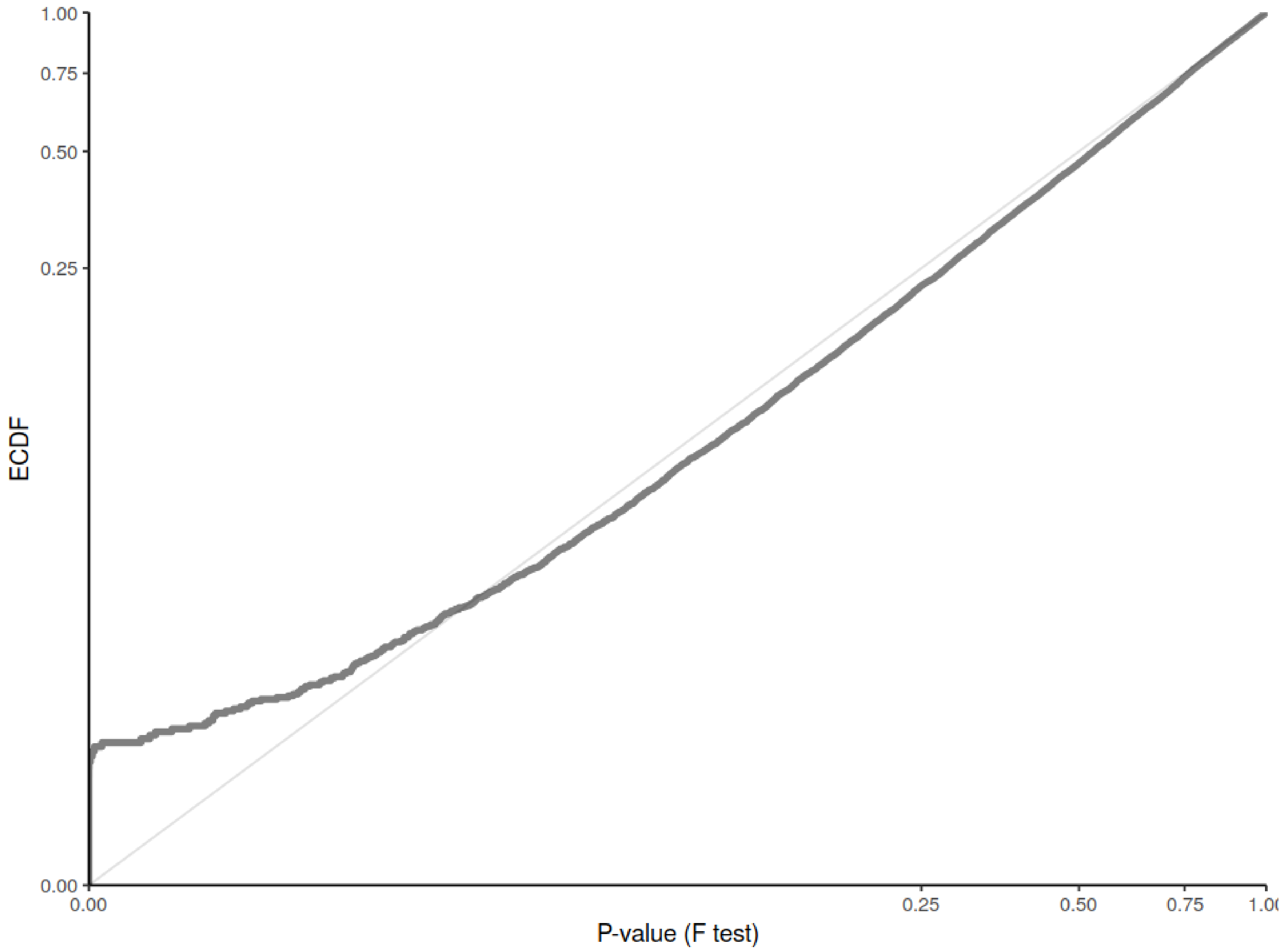
In [11]:
```
ruv_hist(fit.summary)
```



```
In [12]: ruv_ecdf(fit.summary)
```

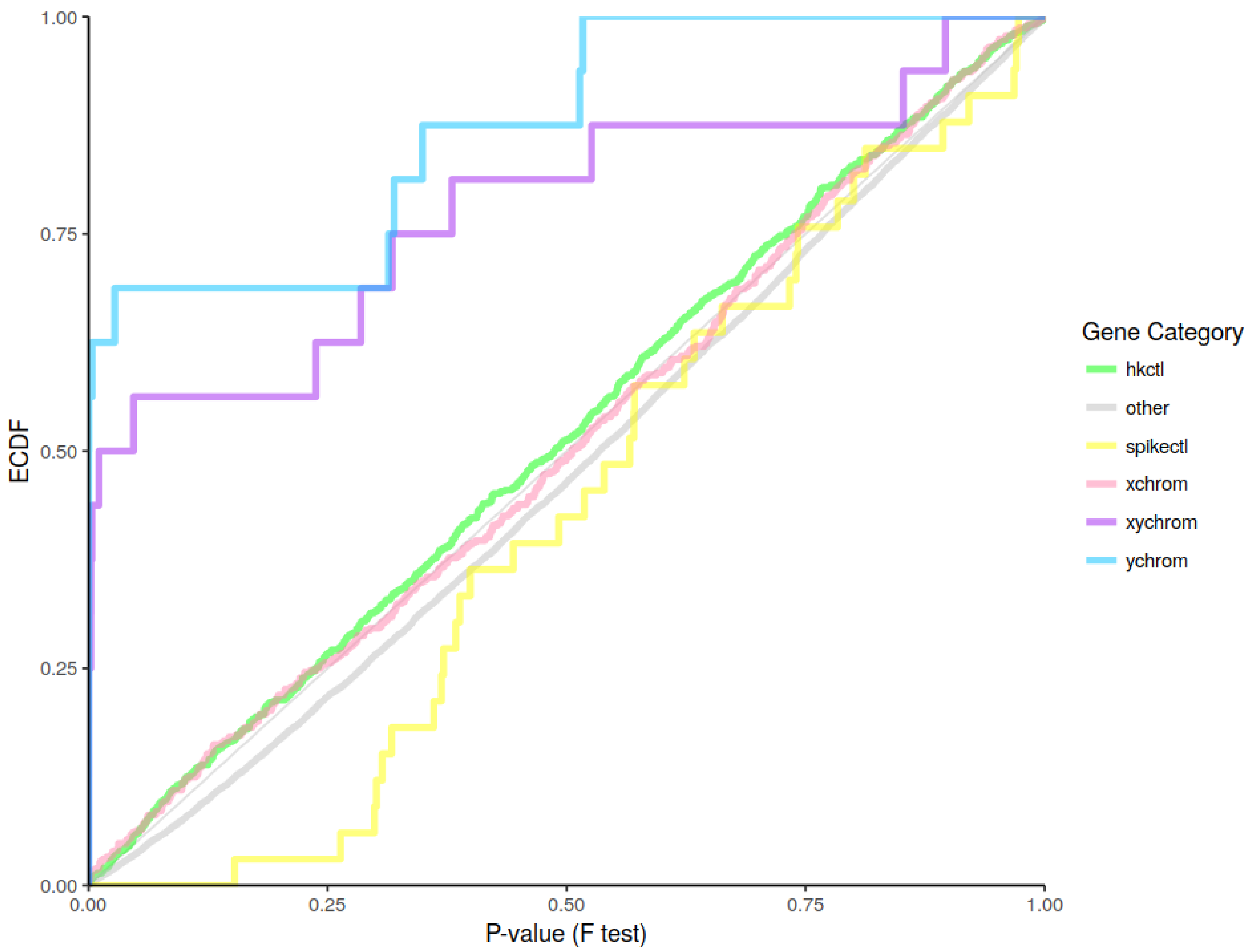


In [13]:
```
ruv_ecdf(fit.summary, power=1/4)
```

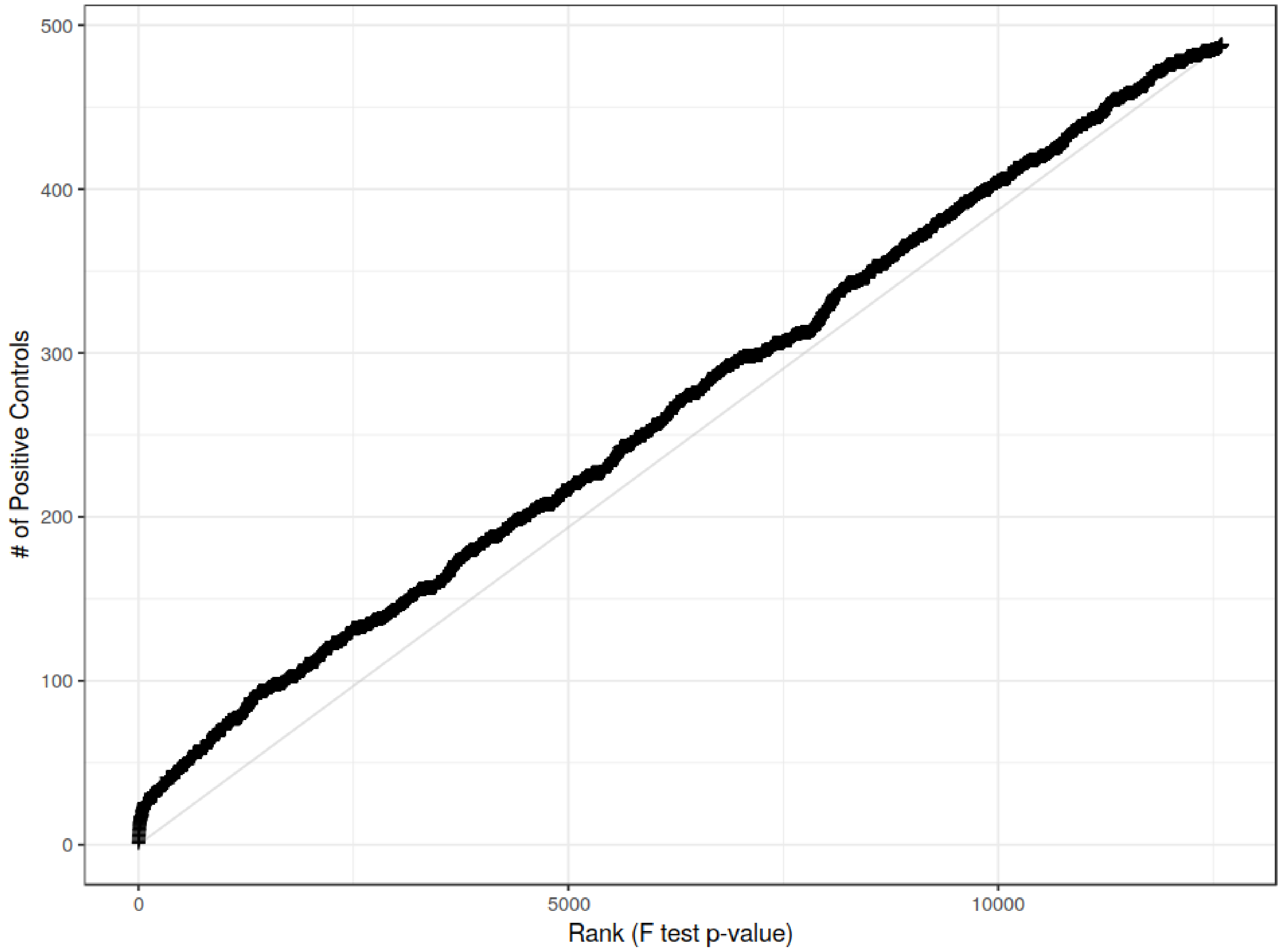


In [14]:
```
genecoloring = list(
aes(color=genetype),
scale_color_manual(name="Gene Category",
                   values=alpha(c("green", "gray", "yellow", "palevioletred1",
"purple", "deepskyblue"),
                                c(     .2,    .15,        1,                1,
       1,             1)))
)
```

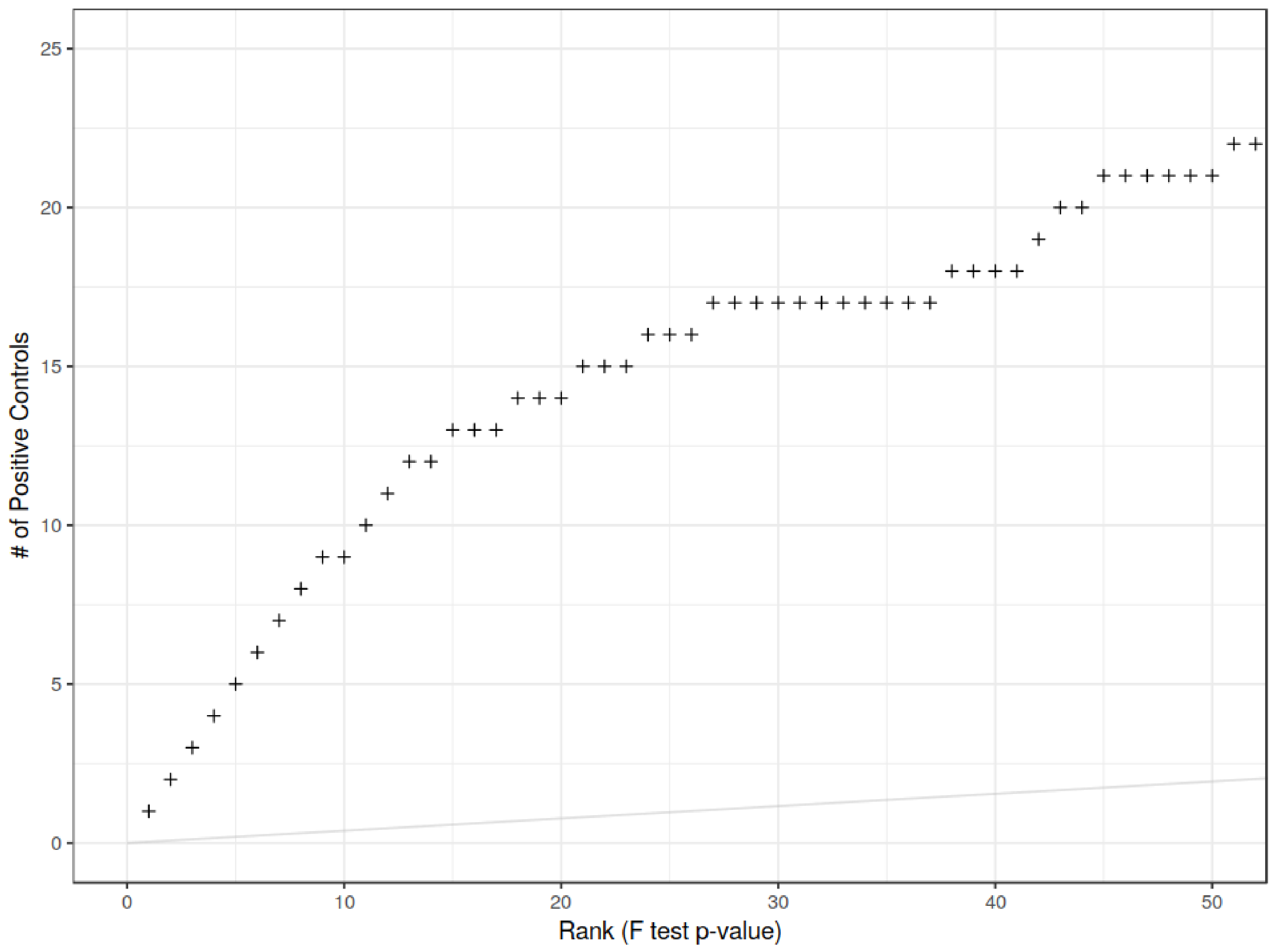
```
In [15]: ruv_ecdf(fit.summary) + genecoloring
```



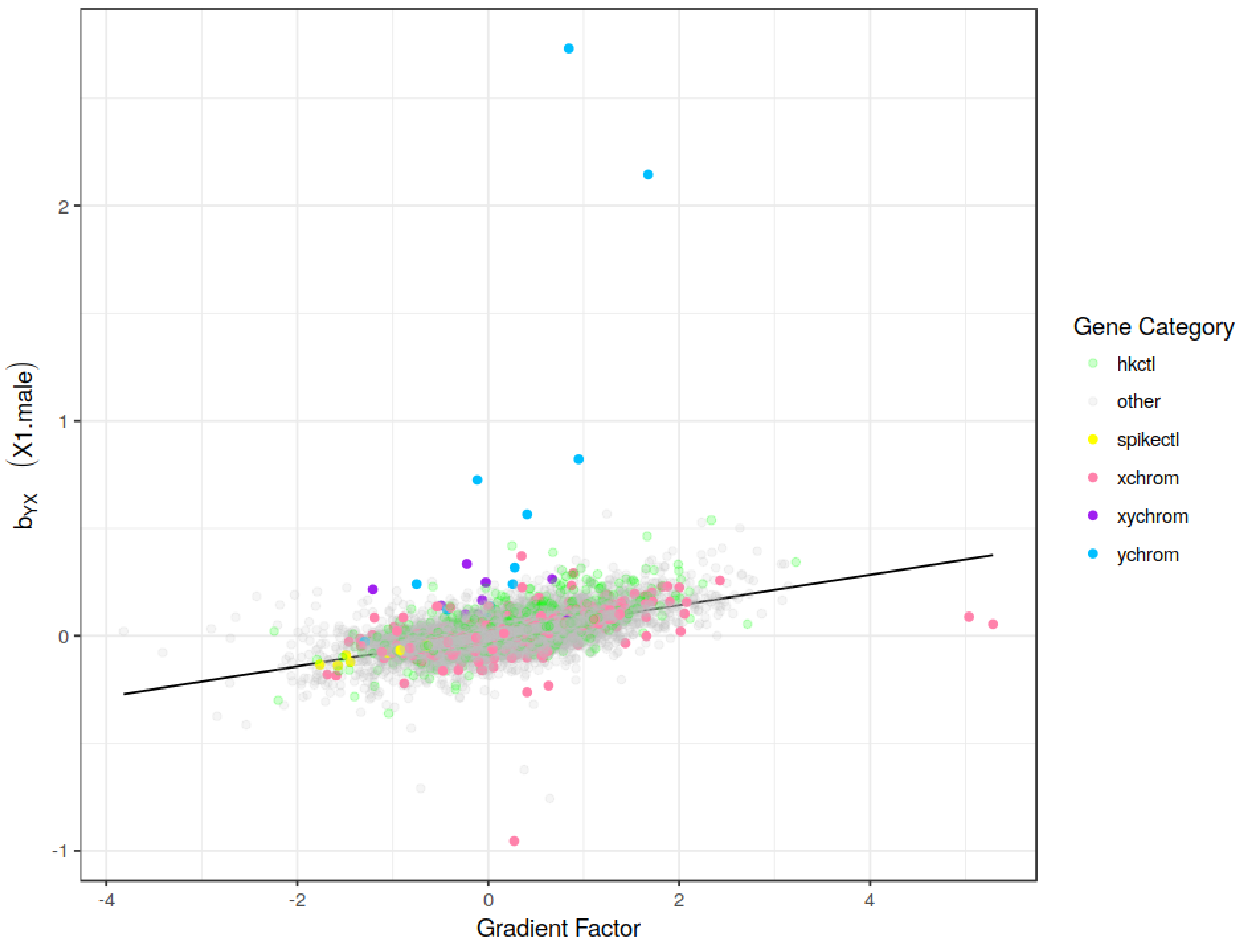
In [16]:
```
ruv_rankplot(fit.summary, "pctl")  # "pctl" is a column in "geneinfo".  Genes fr
om X/Y chrom.
```



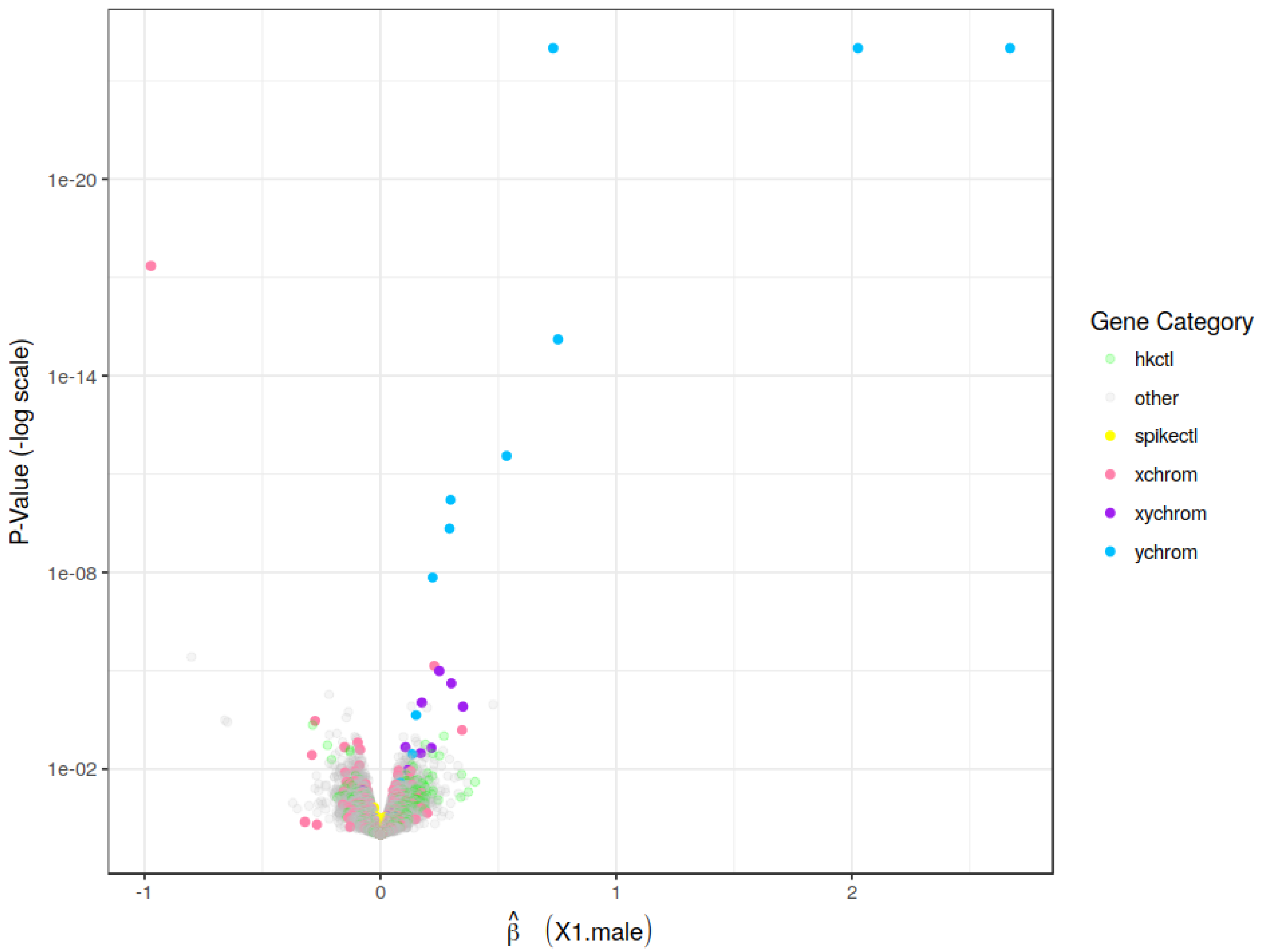
```
In [17]: ruv_rankplot(fit.summary, "pctl") + coord_cartesian(xlim=c(0,50), ylim=c(0,25))
```



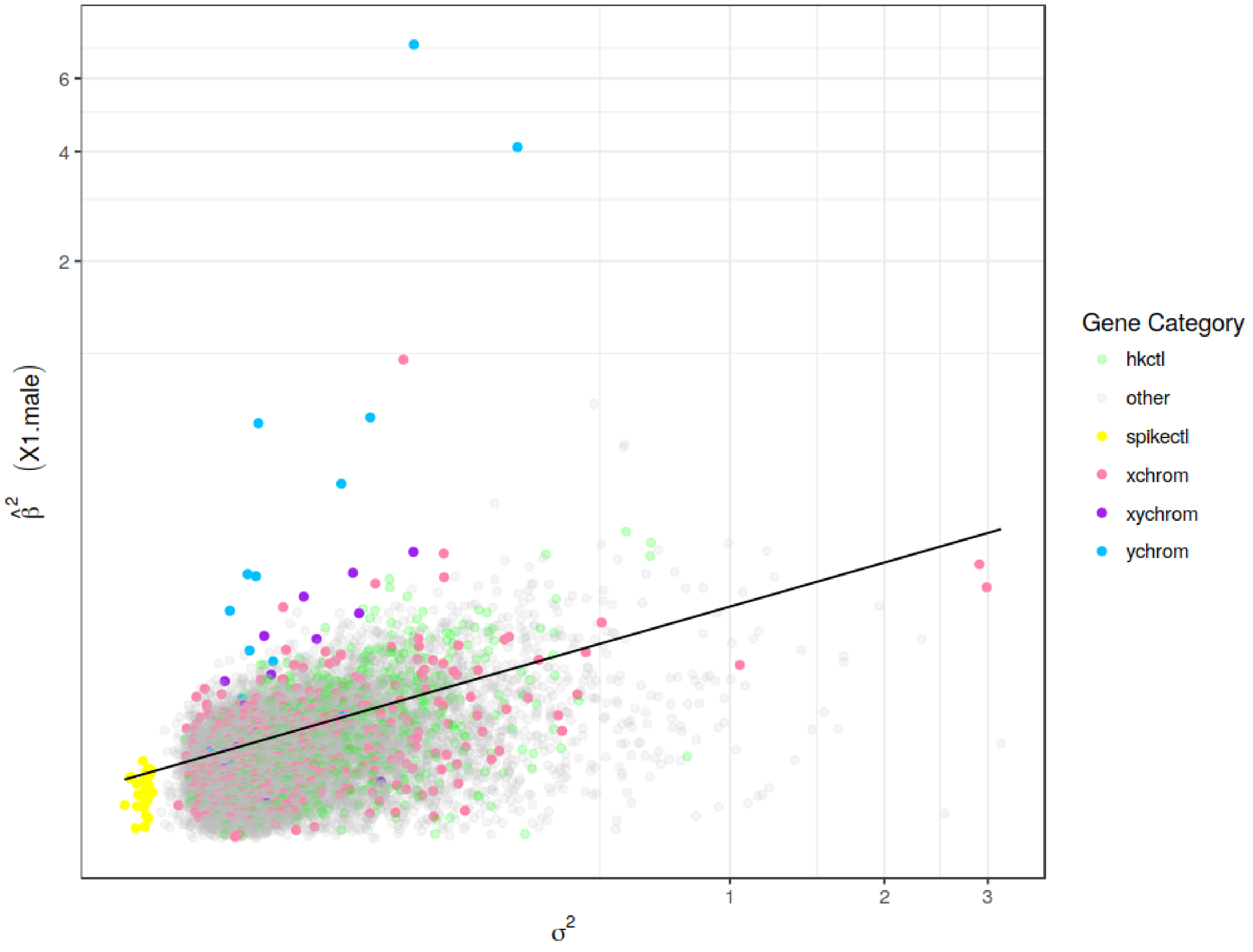
```
In [18]: ruv_projectionplot(fit.summary) + genecoloring
```



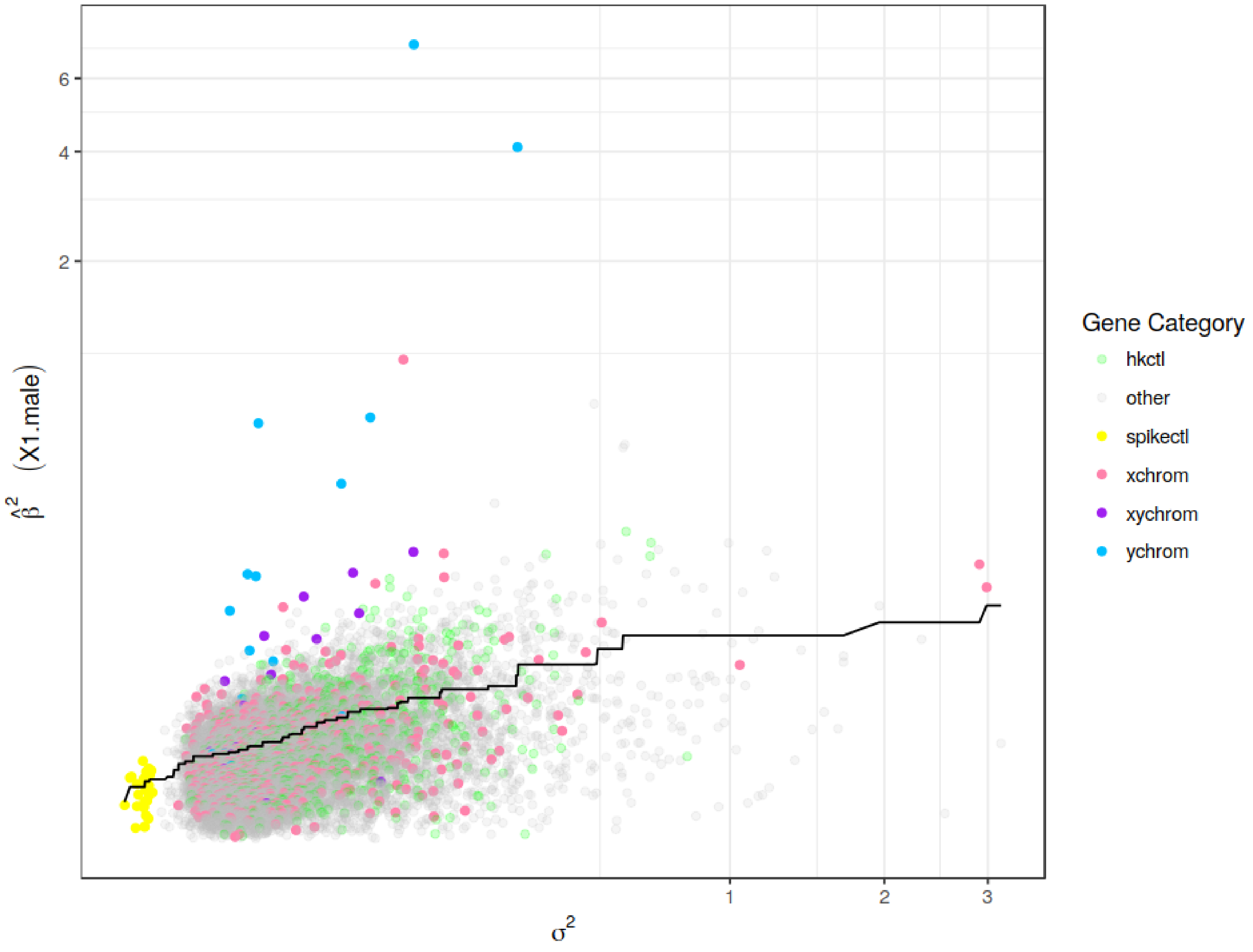
In [19]:
```
ruv_volcano(fit.summary) + genecoloring
```



```
In [20]: ruv_varianceplot(fit.summary) + genecoloring
```



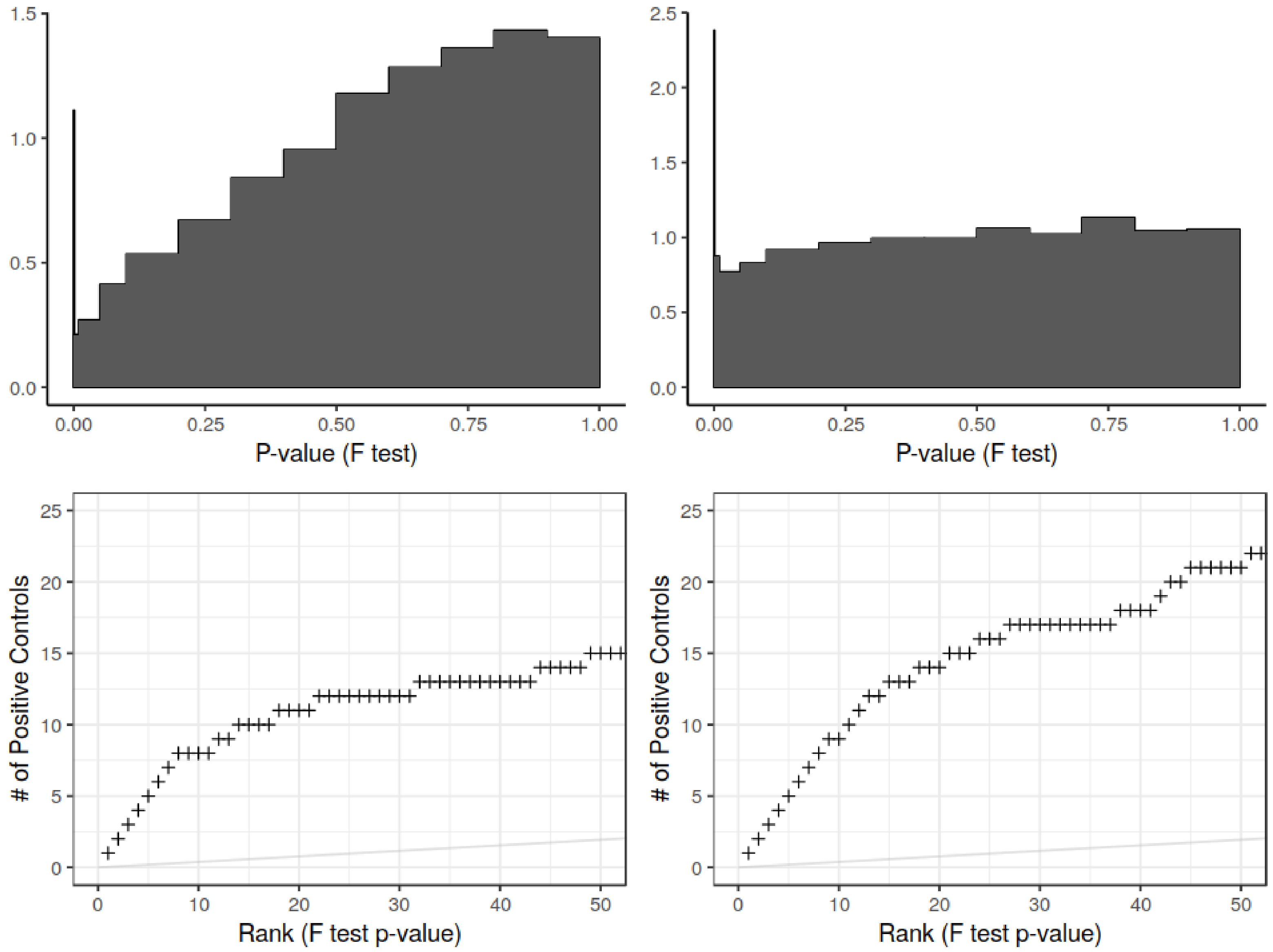
```
In [21]: fit.summary.evar = ruv_summary(Y, fit, sampleinfo, geneinfo, p.type="evar")
         ruv_varianceplot(fit.summary.evar) + genecoloring
```



Did we help?

```
In [22]:
# RUV4 with k = 0 for no adjustment
# Equivalent to a Limma Analysis
fit.unadj = RUV4(Y, sampleinfo$gender, geneinfo$spikectl, 0)
fit.summary.unadj = ruv_summary(Y, fit.unadj, sampleinfo, geneinfo)
# Make a list of plots to compare side-by-side
plots = list(
  ruv_hist(fit.summary.unadj),
  ruv_hist(fit.summary),
  ruv_rankplot(fit.summary.unadj, "pctl") +
    coord_cartesian(xlim=c(0,50), ylim=c(0,25)),
  ruv_rankplot(fit.summary, "pctl") +
    coord_cartesian(xlim=c(0,50), ylim=c(0,25))
)
```

```
In [23]: grid.arrange(grobs=plots)
```



Global Adjustments

Global Adjustments

Not similar:

```
RUVI   (Y, eta, ctl)
RUVIII (Y, M, ctl, k = NULL, eta = NULL, average = FALSE)
```

RUVI

```
RUVI(Y, eta, ctl)
```

- Requires gene-wise covariates associated with unwanted factors (GC content, etc.)
- Note: Integrated into RUV2/4/inv/rinv for convenience
- `eta = 1` corresponds to centering by the mean of the negative controls

RUVIII

```
RUVIII(Y, M, ctl, k = NULL, eta = NULL, average = FALSE)
```

- Requires **replicates**
- M is the *replicate mapping matrix*
- "`average`": Average the replicates (after the adjustment)

The Mapping Matrix

- Maps observations (rows of Y) to replicate sets
- $M_{ij} = 1$ if observation i is in replicate set j
  $M_{ij} = 0$ otherwise
- A replicate set may contain only one observation (a "singleton").
  But at least one replicate set must contain multiple observations.
- ***Any variation within a replicate set is assumed to be unwanted.***

`replicate.matrix`

Generates a mapping matrx.

```
replicate.matrix(a, burst = NULL, return.factor = FALSE)
```

Arguments:

- "`a`": An object that describes the replicate structure.
  - Converted to a dataframe
  - Observations with identical rows are taken to be replicates
- "`burst`": Replicate sets to be "burst" into singletons.
- "`return.factor`": Return a factor instead of a mapping matrix.

`collapse.replicates`

```
collapse.replicates(df, M)
```

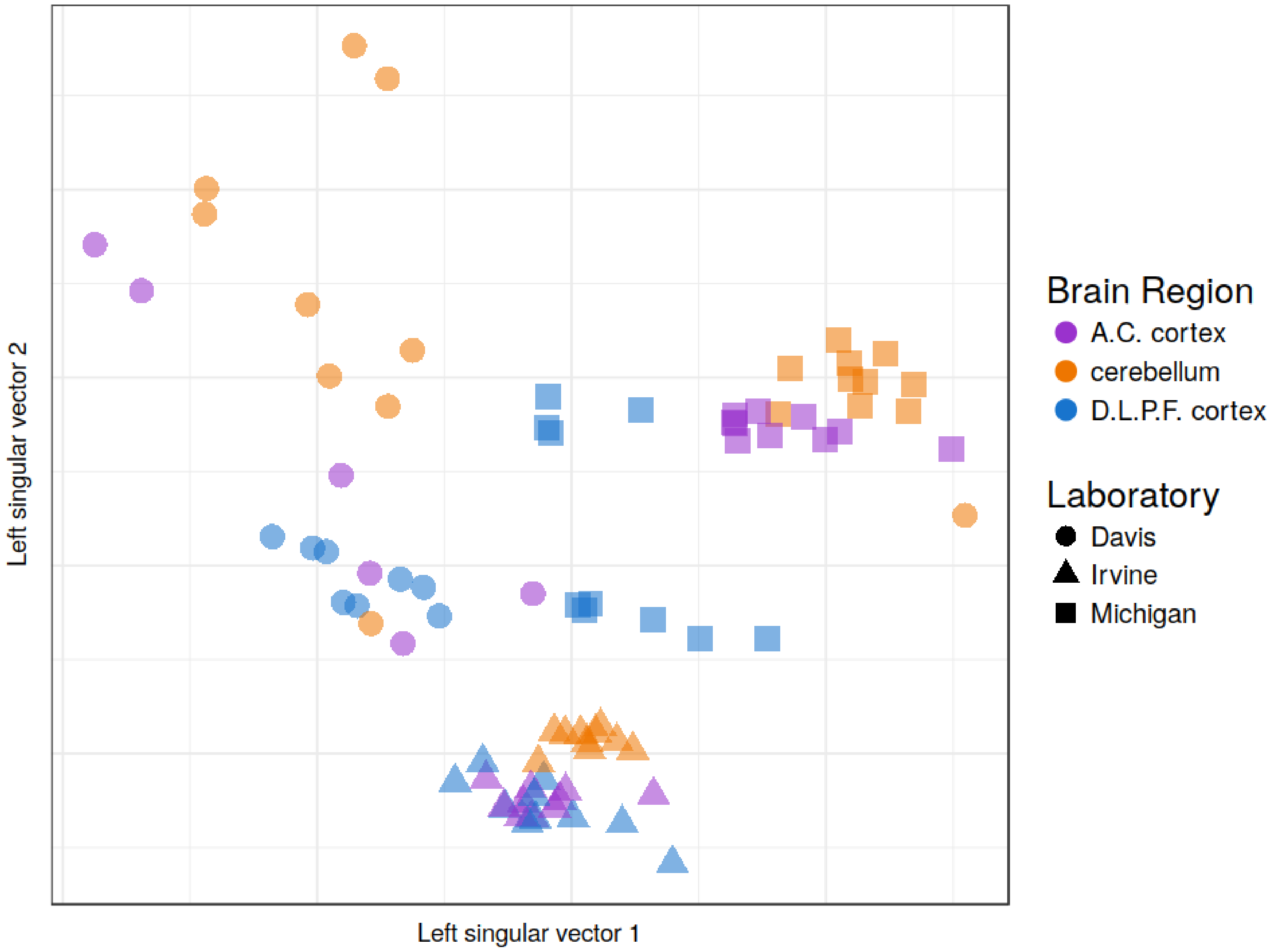
- For use with `RUVIII` when `average=TRUE`
- Input: A dataframe containing information about the observations (rows of Y)
- Output: A dataframe containing information about the replicate sets (rows of "new Y")

Example Analyses

Example 1

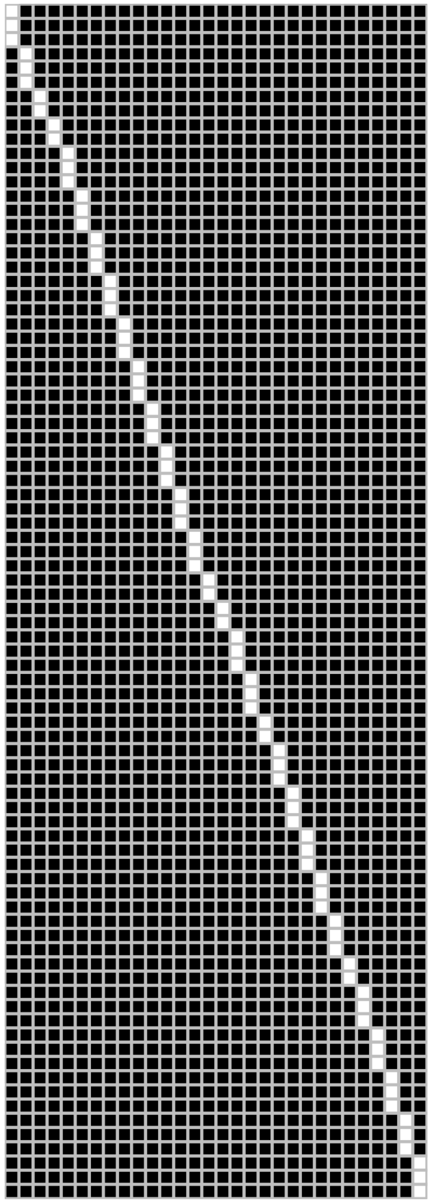
Spike-in Negative Controls and Technical Replicates

```
In [24]: ruv_svdplot(Y) + gg_additions
```

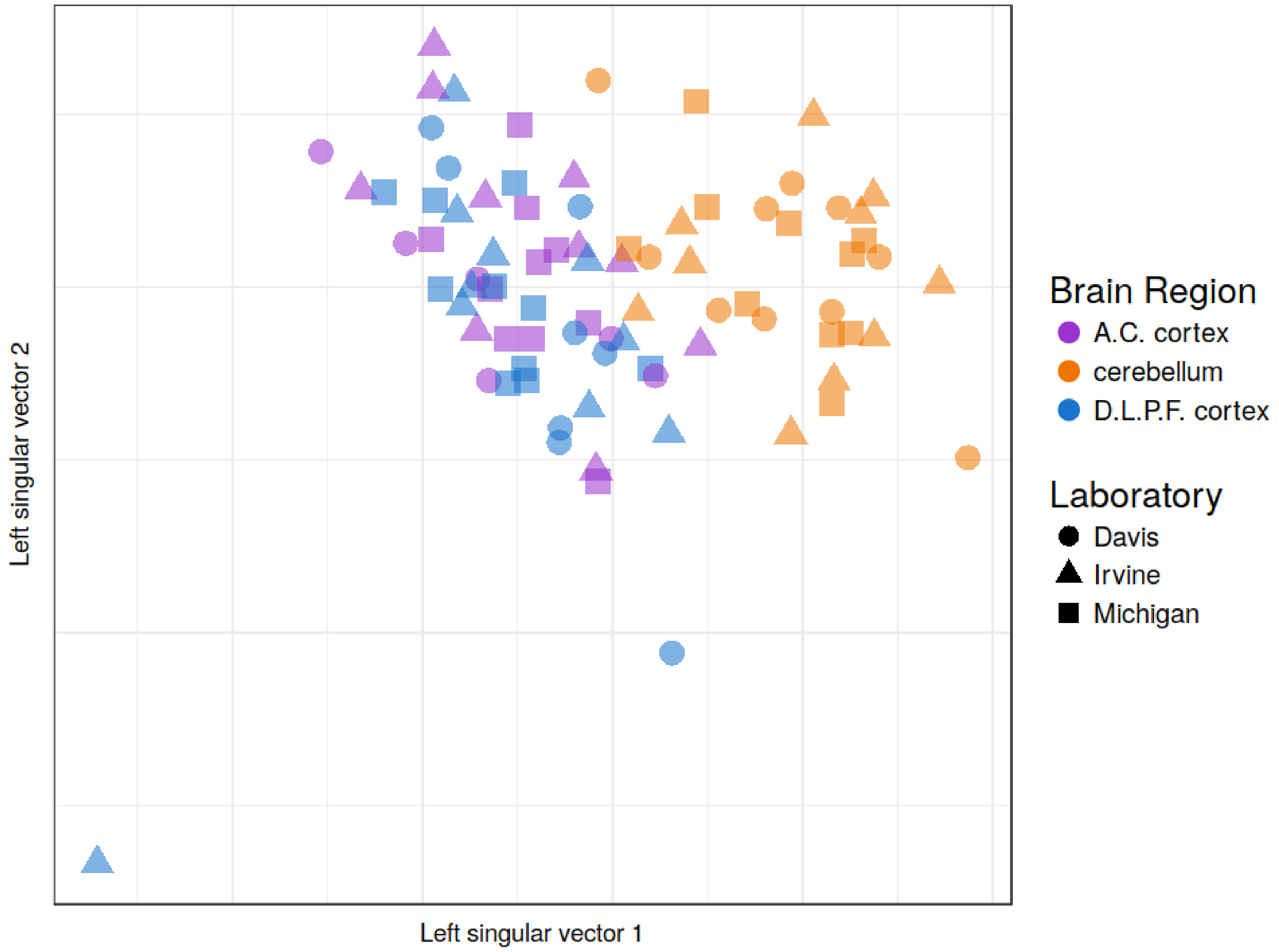


In [25]:

```
M = replicate.matrix(sampleinfo[,c("patient", "region")])
YIII.spike.tech = RUVIII(Y, M, geneinfo$spikectl, k=10)
```

$M =$ 

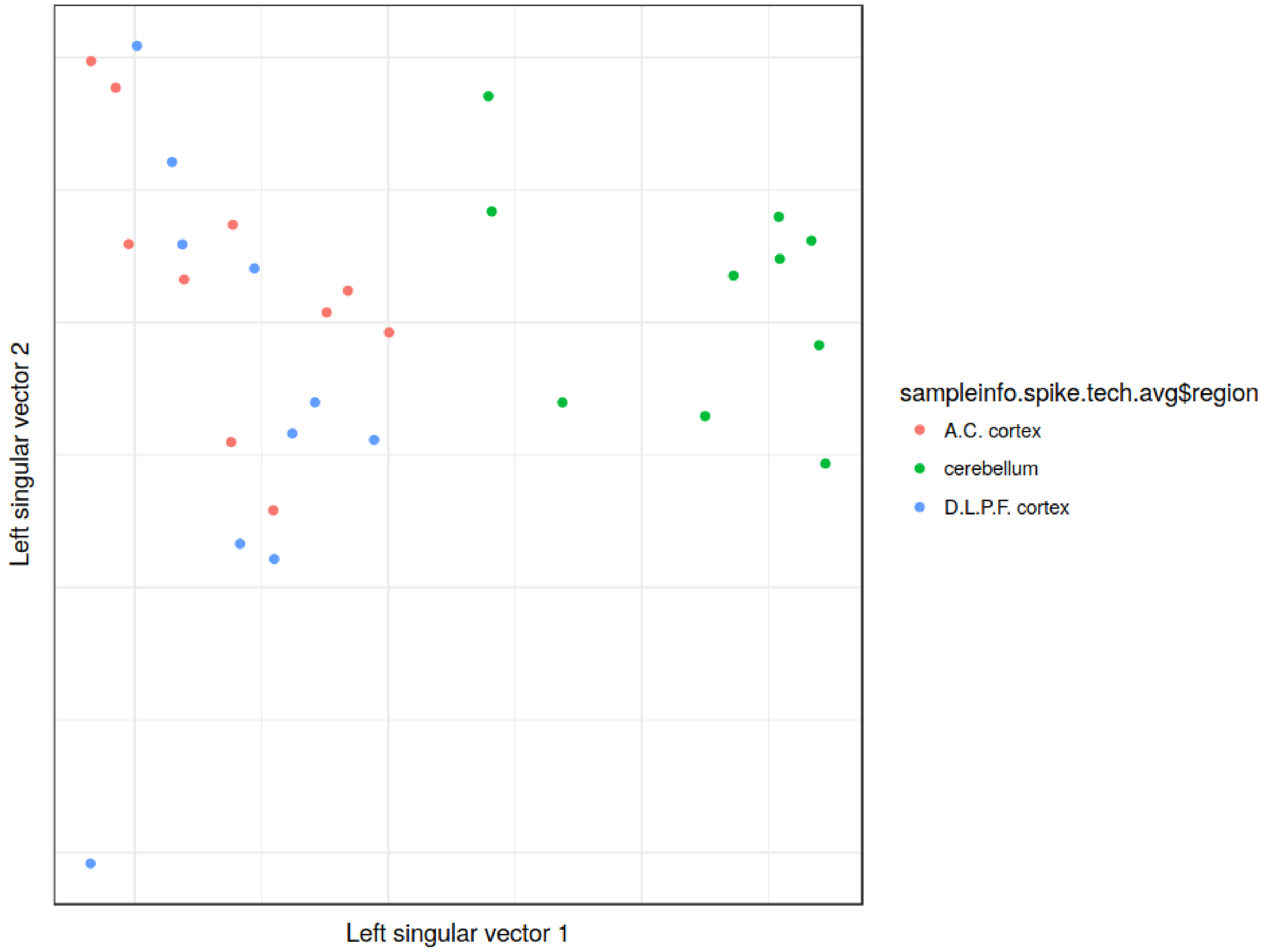
```
In [26]: ruv_svdplot(YIII.spike.tech) + gg_additions
```



In [27]:
```
# This time, set average=TRUE
YIII.spike.tech.avg = RUVIII(Y, M, geneinfo$spikectl, k=10, average=TRUE)
# Create "metadata" for the rows of YIII.spike.tech.avg
sampleinfo.spike.tech.avg = collapse.replicates(sampleinfo, M)
head(sampleinfo.spike.tech.avg)
```

	patient	**gender**	**region**
patient_01_A.C..cortex	patient_01	female	A.C. cortex
patient_01_D.L.P.F..cortex	patient_01	female	D.L.P.F. cortex
patient_01_cerebellum	patient_01	female	cerebellum
patient_02_A.C..cortex	patient_02	male	A.C. cortex
patient_02_D.L.P.F..cortex	patient_02	male	D.L.P.F. cortex
patient_02_cerebellum	patient_02	male	cerebellum

In [28]:
```
ruv_svdplot(YIII.spike.tech.avg) +
  aes(color=sampleinfo.spike.tech.avg$region)
```

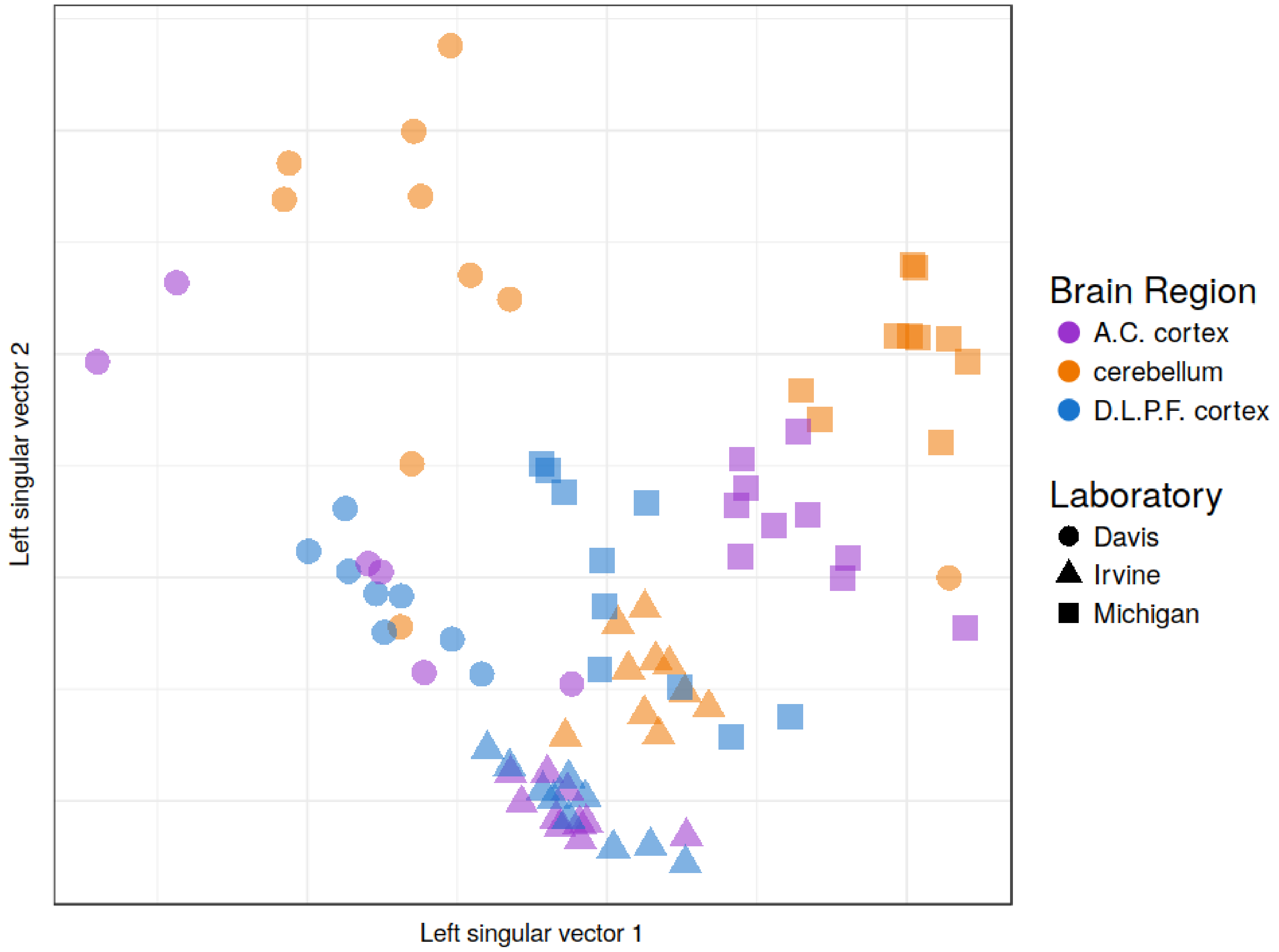


Example 2

Plotting just the X/Y genes

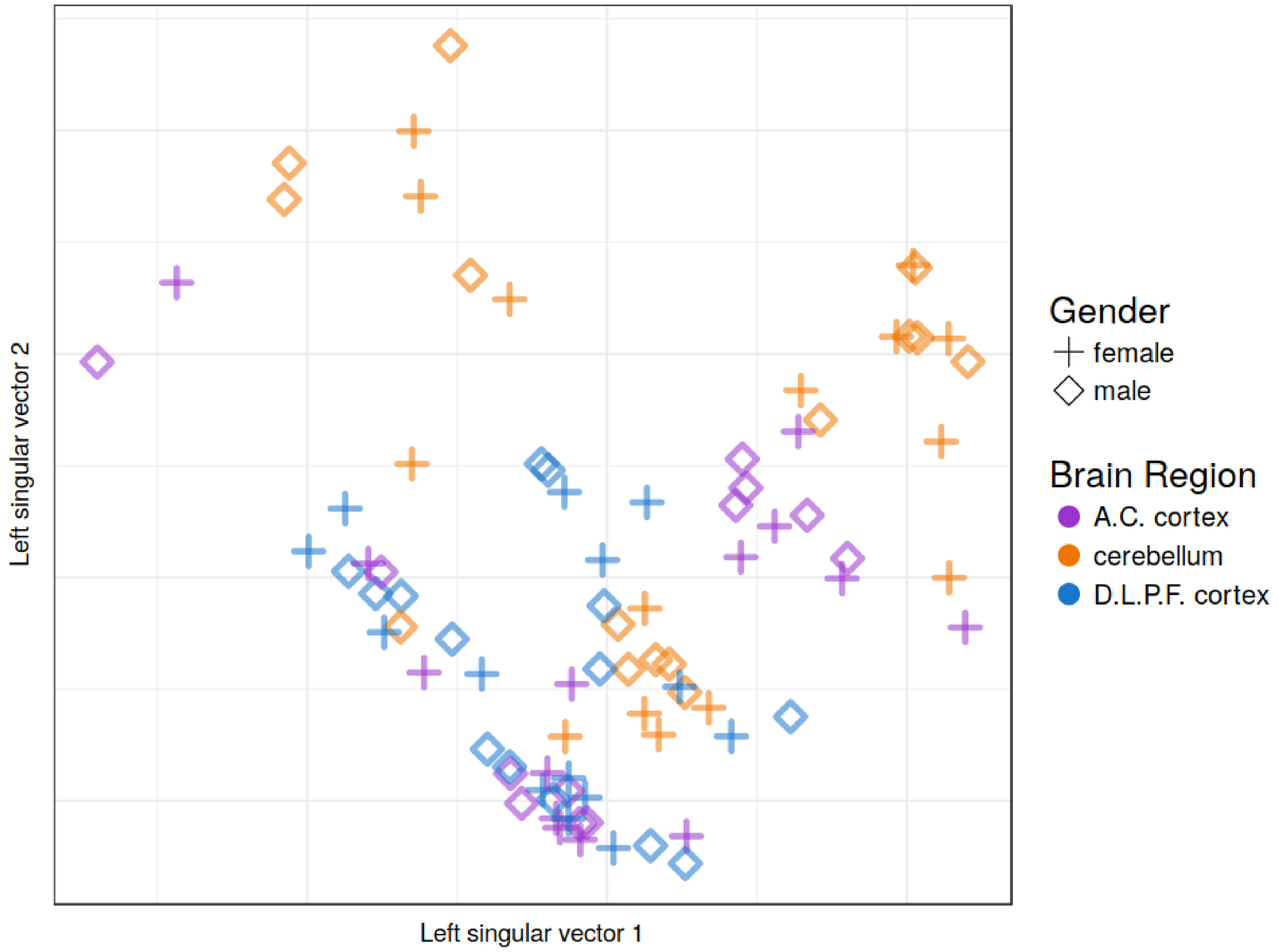
- Same analysis, but plot the PCs of just the X/Y genes
- What will we see...
  - ...before adjustment?
  - ...after adjustment?

In [29]:
```
ruv_svdplot(Y[,geneinfo$pctl]) + gg_additions
```

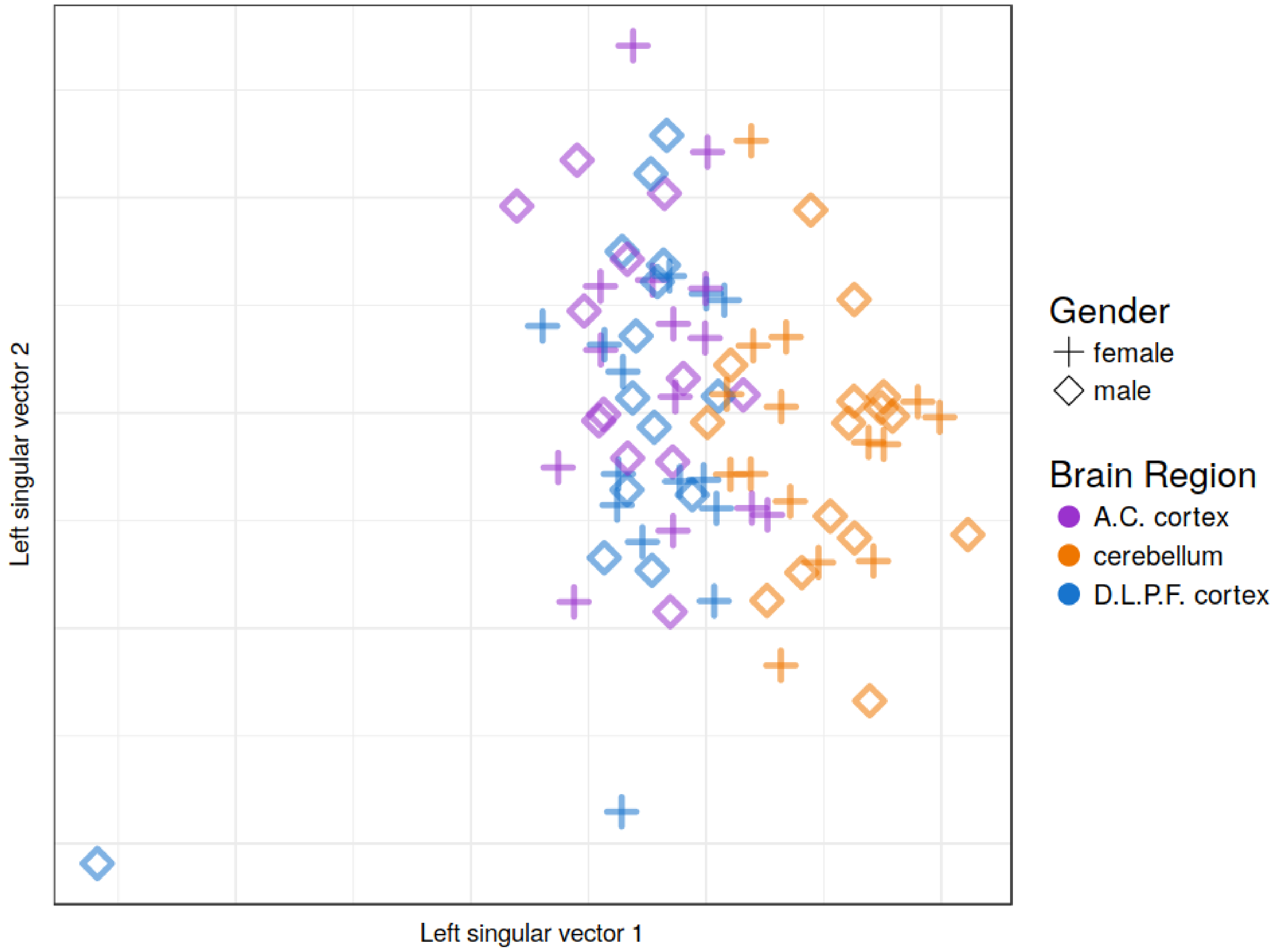


```
In [30]: gg_gender_region = list(aes(color=sampleinfo$region,
                                  shape=sampleinfo$gender,
                                  size=3, alpha=1, stroke=2),
                              labs(color="Brain Region",
                                   shape="Gender"),
                              scale_size_identity(guide="none"),
                              scale_alpha(guide="none"),
                              scale_shape_manual(values = c("male" = 5, "female" = 3
         )),
                              theme(legend.text=element_text(size=12),
                                    legend.title=element_text(size=16)),
                              guides(color = guide_legend(override.aes = list(size =
         4)),
                                     shape = guide_legend(override.aes = list(size =
         4))),
                              scale_color_manual(values=c("darkorchid3", "darkorange
         2", "dodgerblue3"))
                             )
```

In [31]:
```
ruv_svdplot(Y[,geneinfo$pctl]) + gg_gender_region
```



```
In [32]: ruv_svdplot(YIII.spike.tech[,geneinfo$pctl]) + gg_gender_region
```

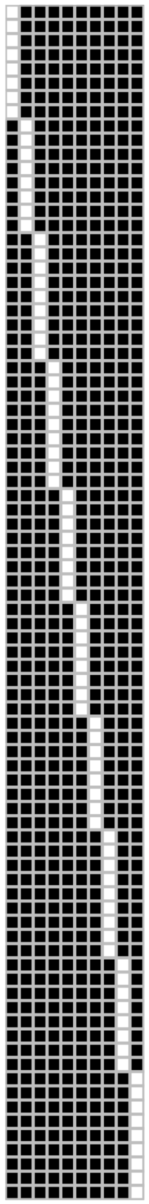


Example 3

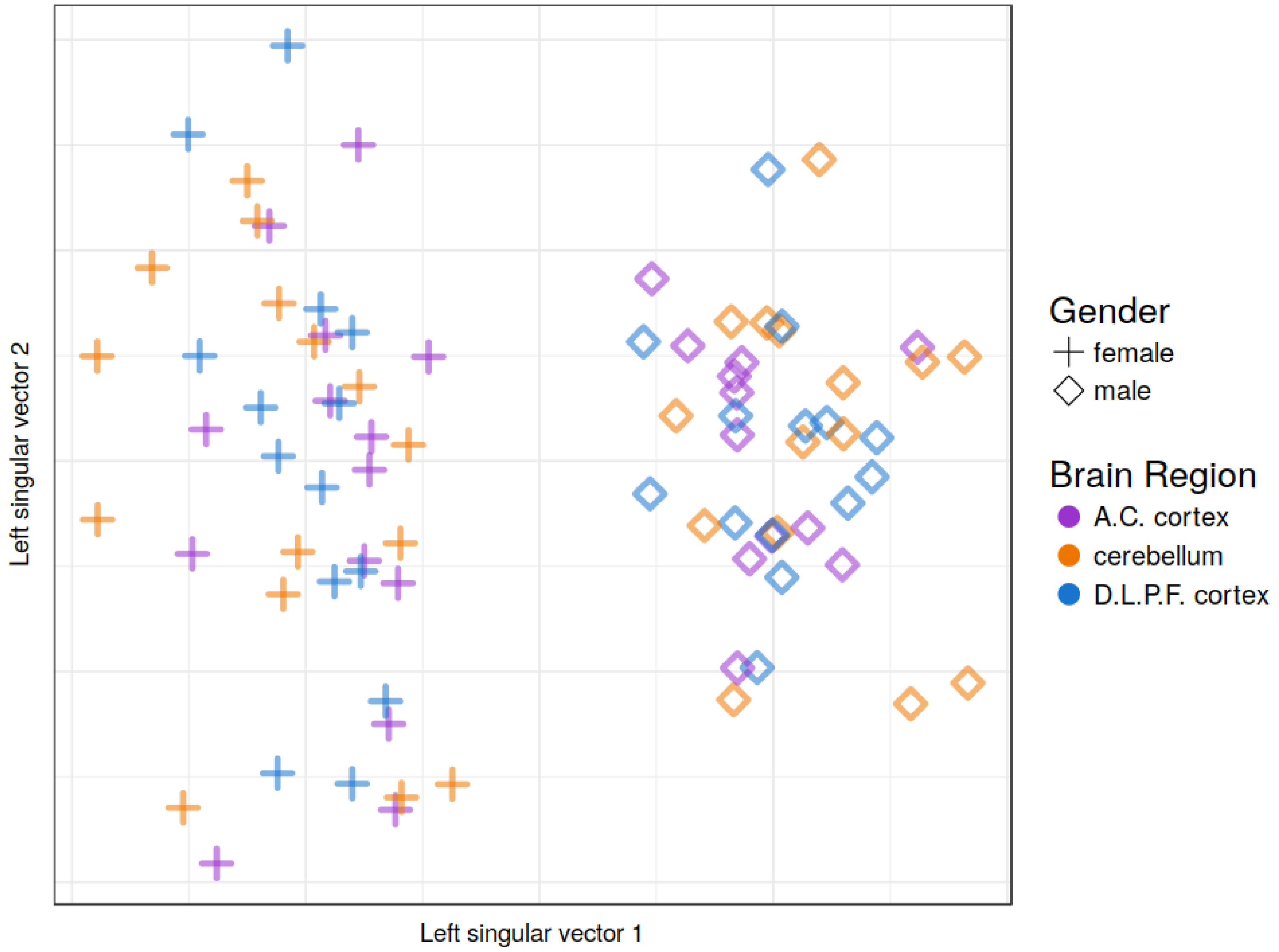
- Negative controls: Housekeeping genes
- Replicates: All observations from a single patient

In [33]:

```
M = replicate.matrix(sampleinfo[,c("patient")])
YIII.hk.bio = RUVIII(Y, M, geneinfo$hkctl, k=10)
```

$M =$ 

In [34]:
```
ruv_svdplot(YIII.hk.bio[,geneinfo$pctl]) + gg_gender_region
```



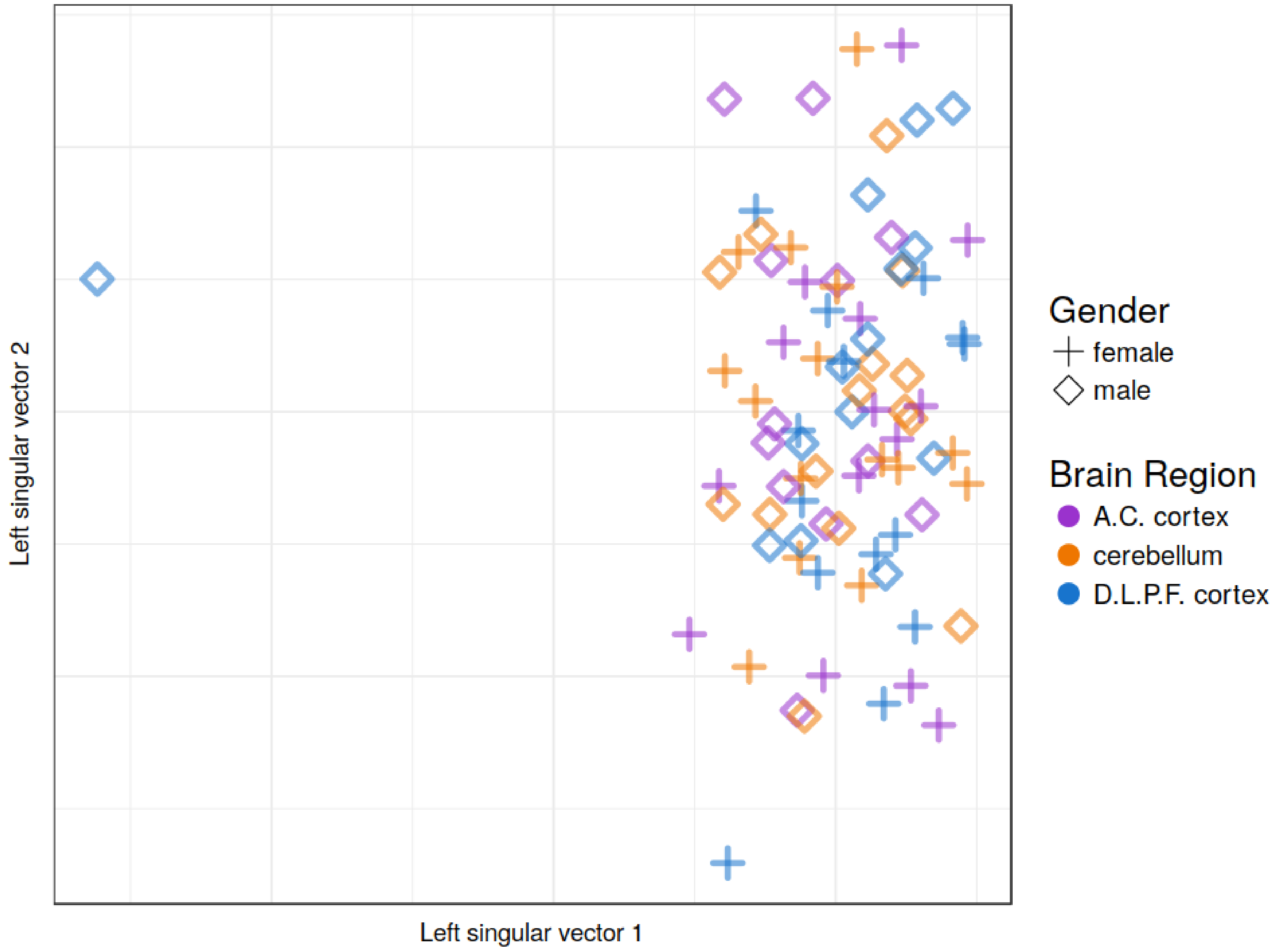
Comment

There is more going on here than simply "regressing out brain region."

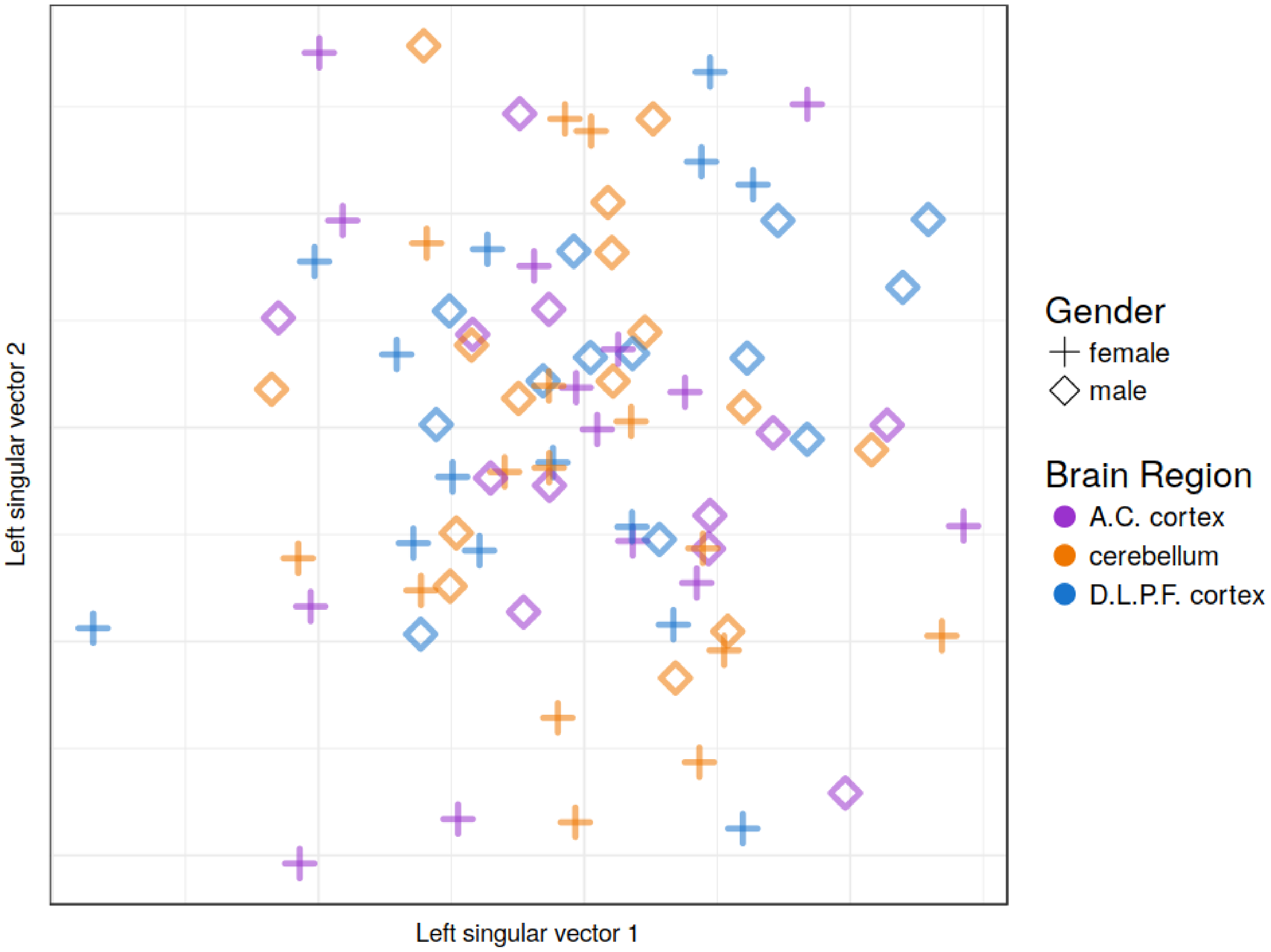
Compare:

```
In [35]:  # Create a design matrix for brain region:
          region_mat = design.matrix(sampleinfo$region)
          # Regress it out from the "technical-adjusted" dataset
          YIII.spike.tech.region_regression = residop(YIII.spike.tech, region_mat)
```

```
In [36]: ruv_svdplot(YIII.spike.tech.region_regression[,geneinfo$pctl]) + gg_gender_regi
         on
```

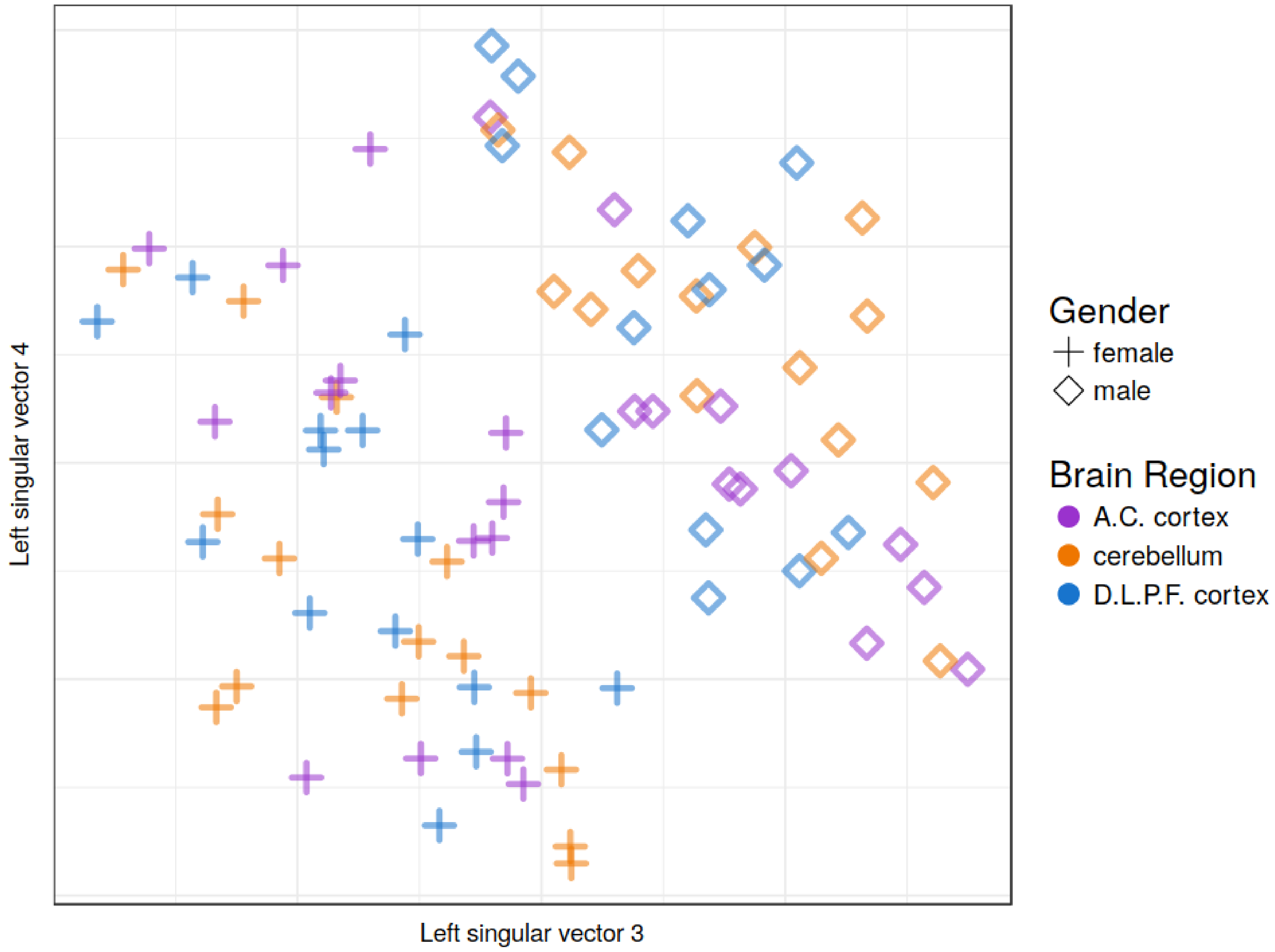


```
In [38]: ruv_svdplot(YIII.spike.tech.region_regression[-15,geneinfo$pctl]) + gg_gender_r
         egion_nooutlier
```



In [39]:

```
ruv_svdplot(YIII.spike.tech.region_regression[-15,geneinfo$pctl], k=3:4) + gg_g
ender_region_nooutlier
```



Two Important Differences

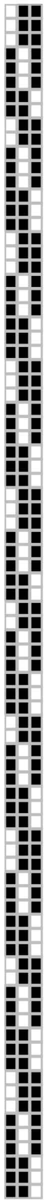
- "HK genes + bio replicates" offers a stronger adjustment
- Regressing out brain region problematic if it's correlated with other biology (not relevant in this example)

Final Example

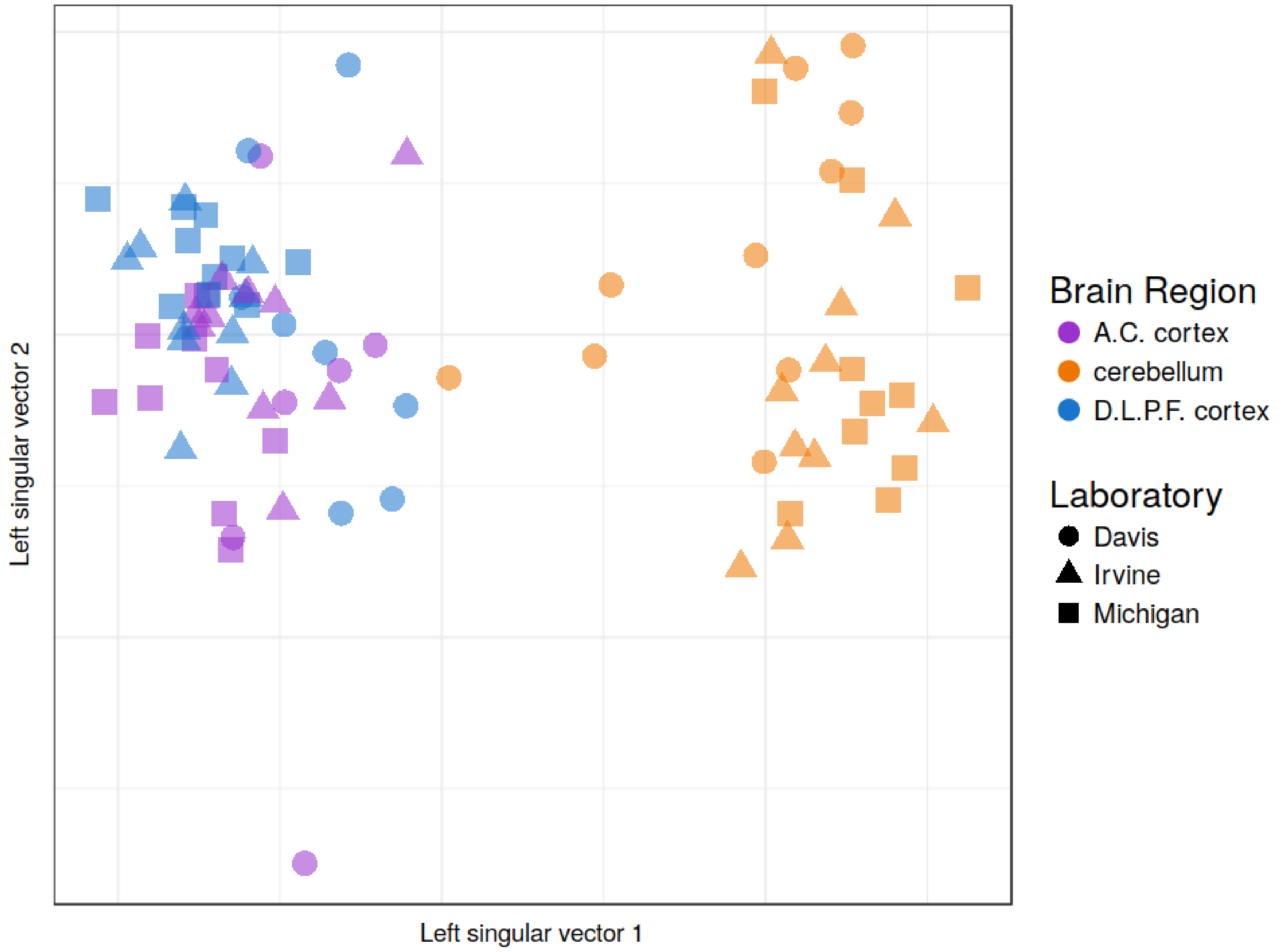
Use brain region to define replicates:

In [40]:

```
M = replicate.matrix(sampleinfo[,c("region")])
newY3 = RUVIII(Y, M, geneinfo$hkctl, k=10)
```

$M =$ 

```
In [41]: ruv_svdplot(newY3) + gg_additions
```

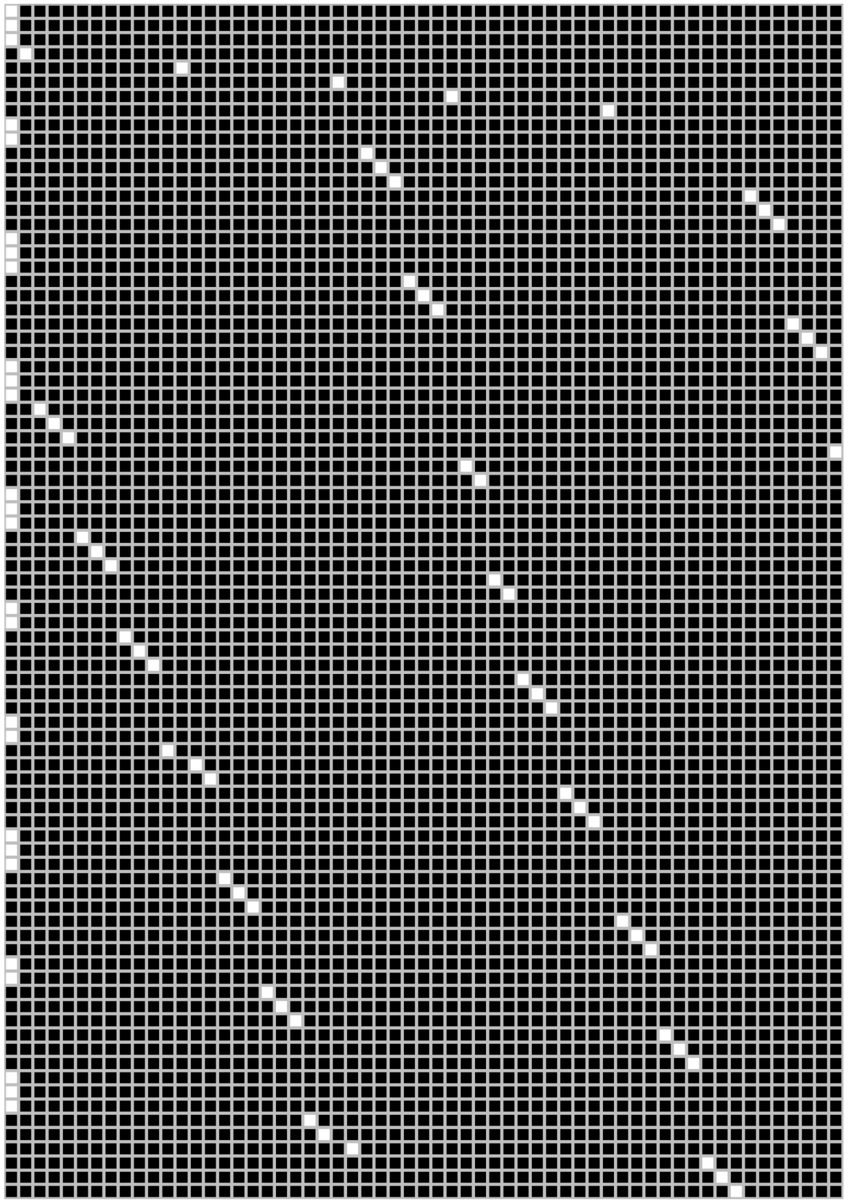


Bursting

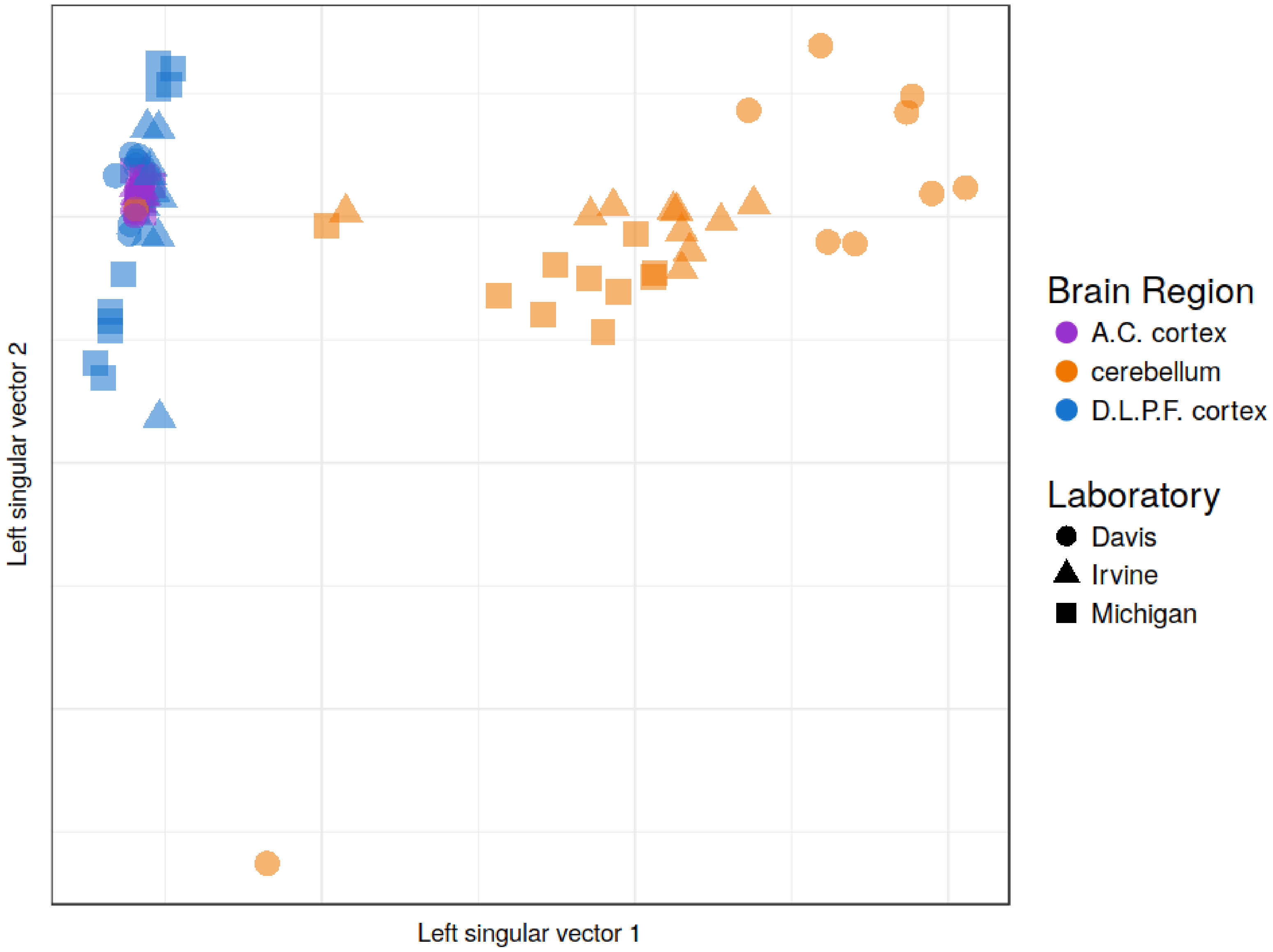
- Now "burst" Cerebellum and D.L.P.F. Cortex.
- Only A.C. Cortex samples are treated as replicates.

In [42]:

```
M = replicate.matrix(sampleinfo[,c("region")], burst=c("cerebellum", "D.L.P.F..
cortex"))
newY3 = RUVIII(Y, M, geneinfo$hkctl, k=10)
```

$M =$ 

```
In [43]: ruv_svdplot(newY3) + gg_additions
```



Comments

- Bursting is useful for validating an adjustment.

- The previous example highlights an interesting possibility for cluster analyses to discover disease sub-types:
    - Use healthy controls to define "ordinary" biological variation
    - After adjustment, only "disease-related" variation remains

Examples with Shiny

1. Gender: A Balanced Design
2. Gender: An Imbalanced Design
3. Brain region

Balanced Design

```
In [ ]: library(ruv)
        library(shiny)
        library(colourpicker)
        load("gender.rda")
        Y = Y.norm
        ruv_shiny(Y,sampleinfo,geneinfo,options=list(port=3840,host="0.0.0.0"))
```

Imbalanced Design

```
In [ ]:
keep = rep(T,nrow(Y))
keep[sampleinfo$lab=="Davis" & sampleinfo$gender=="male"] = FALSE
keep[sampleinfo$lab=="Michigan" & sampleinfo$gender=="female"] = FALSE
Y.imb = Y[keep,]
sampleinfo.imb = sampleinfo[keep,]
ruv_shiny(Y.imb,sampleinfo.imb,geneinfo,options=list(port=3840,host="0.0.0.0"))
```

In [ ]:

```
keep = rep(T,nrow(Y))
keep[sampleinfo$lab=="Davis" & sampleinfo$gender=="male"] = FALSE
keep[sampleinfo$lab=="Michigan" & sampleinfo$gender=="female"] = FALSE
Y.imb = Y.raw[keep,]
sampleinfo.imb = sampleinfo[keep,]
ruv_shiny(Y.imb,sampleinfo.imb,geneinfo,options=list(port=3840,host="0.0.0.0"))
```

Brain Region

```
In [ ]: ruv_shiny(Y.raw,sampleinfo,geneinfo,options=list(port=3840,host="0.0.0.0"))
In [ ]: newY = RUVI(Y.raw, 1, geneinfo$spikectl)
        M = replicate.matrix(sampleinfo[,c("patient", "region")])
        newY = RUVIII(newY, M, geneinfo$spikectl, k=4, average=TRUE)
        newsampleinfo = collapse.replicates(sampleinfo, M)
        fit = RUV4(newY, newsampleinfo$cortex, rep(TRUE,ncol(newY)), k=1)
        fit = ruv_summary(newY, fit, newsampleinfo, geneinfo)
```

```
In [ ]: ruv_ecdf(fit, uniform.lines=seq(0,1,by=.1))
```

In [ ]:

```
mean(fit$C$F.p > .25)
mean(fit$C$F.p.BH > .5)
```

```
In [ ]: ectl = colnames(newY) %in% rownames(fit$C)[fit$C$F.p.BH > .5]
        geneinfo = cbind(geneinfo, neg.cer=ectl)
```

```
In [ ]: ruv_shiny(Y.raw, sampleinfo, geneinfo)
```